Range-Based Localization for Swarms of Micro Air Vehicles

Pfeiffer, S.U.

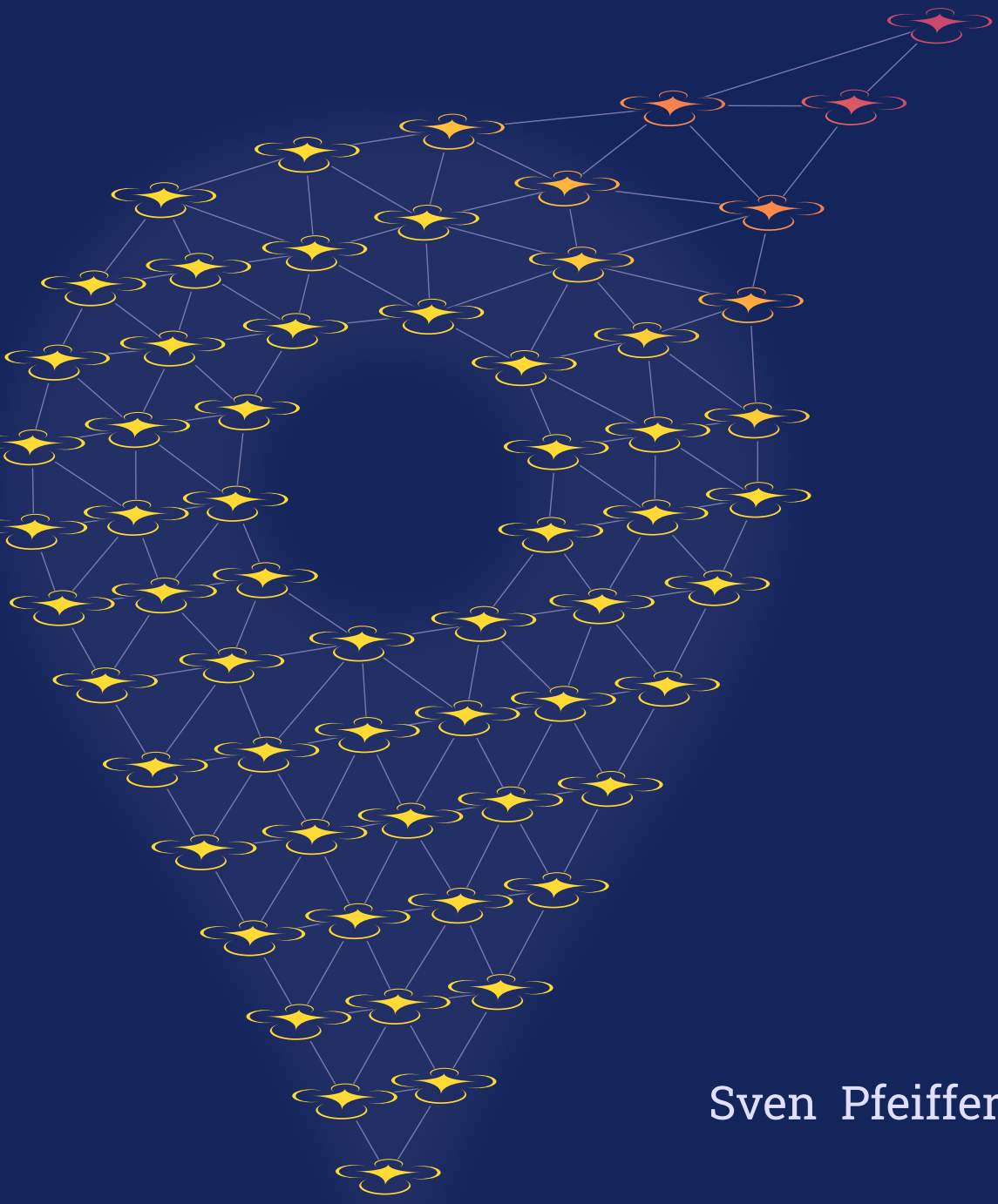
DOI
10.4233/uuid:56ba185c-9b84-460a-835d-16bf828ee9fc

Publication date
2024

Document Version
Final published version

Citation (APA)
Pfeiffer, S. U. (2024). *Range-Based Localization for Swarms of Micro Air Vehicles*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:56ba185c-9b84-460a-835d-16bf828ee9fc

Range-Based Localization for Swarms of Micro Air Vehicles



Sven Pfeiffer

RANGE-BASED LOCALIZATION FOR SWARMS OF MICRO AIR VEHICLES

RANGE-BASED LOCALIZATION FOR SWARMS OF MICRO AIR VEHICLES

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on Friday, 13 December, 2024 at 12:30

by

Sven Udo PFEIFFER

Master of Science en Génie mécanique,
L'Ecole polytechnique fédérale de Lausanne, Switzerland,
born in Munich, Germany.

This dissertation has been approved by the promotors

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. G.C.H.E. de Croon	Delft University of Technology, promotor
Dr.ir. C. De Wagter	Delft University of Technology, copromotor
Independent members:	
Prof.dr. E.K.A. Gill	Delft University of Technology
Prof.dr. M. Wisse	Delft University of Technology
Dr.ir. M. Mazo Espinosa	Delft University of Technology
Prof.dr. W. Hoenig	Technische Universität Berlin
Dr.ir. E. Ferrante	Vrije Universiteit Amsterdam

Keywords: Ultra-wideband, Micro aerial vehicles, State estimation, Localization, Swarm robotics

Printing: Ipskamp | www.ipskampprinting.nl

Front & Back: Sven Pfeiffer

ISBN 978-94-6384-696-7

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

CONTENTS

SUMMARY

With their ability to access hard-to-reach spaces and to provide a birds-eye view over large areas, Micro Air Vehicles (MAV) are already taking over a wide variety of monitoring and inspection tasks. As the continuing miniaturization of electronic components further drives down cost, recent advancements in the design of control algorithms promise a future, where these small drones will be able to operate fully autonomously and with minimal human input.

Spatial awareness is a key aspect in developing fully autonomous MAVs and in this dissertation, we develop new algorithms for on-board localization using Ultra-Wideband (UWB) ranging. UWB is a low-power technology that enables data communication and time-of-flight range measurements. The large bandwidth of UWB results in high timing accuracy as well as an improved resistance to shadowing and multipathing, which makes UWB a very useful technology for ranging in indoor environments.

In the first part of this dissertation, we investigate the use of UWB ranging for absolute localization. We take a look at Moving Horizon Estimation (MHE) as a better way to cope with the non-linear drone dynamics, and the non-Gaussian noise observed on UWB range measurements. Since MHE uses a computationally expensive, optimization-based approach, we mainly focus on reducing the computational complexity of the algorithm. By simplifying the system to an output error MHE, we can analytically calculate the gradient of the cost function and subsequently use a single-step gradient descent method for optimization. Since UWB is known to occasionally produce range measurements with larger errors, we introduce switching variables as additional decision variables for outlier rejection. By modifying these switching variables during the optimization step, individual measurements can be disabled if they are highly unlikely to be true. We find that the resulting algorithm can outperform a standard EKF, and runs in real-time on small drones with limited computational power.

In the second part, we look at range-based relative localization in small swarms of drones. After formulating a three-dimensional system model, we perform an observability analysis and identity constellations, in which relative localization might not be possible. We propose a simple EKF for state estimation, which we test in simulation and on-board of both rotary and flapping wing drones. When using the estimation results as inputs to a controller that performs a leader-follower behaviour, we notice that closing the loop offers a stabilizing effect in unobservable configurations. As a result, we show that such a behaviour works in practice, despite being theoretically non-observable.

In the third part, we modify this algorithm to be scalable to arbitrarily large swarms. The scalability of swarming algorithms is a key aspect in their utility as swarms grow larger and larger. To achieve scalability, we use a priority system to limit the maximum number of agents that are tracked at the same time. Not tracking all agents at the same time means that the list of tracked agents changes frequently and success depends on quick initialization of newly tracked agents. Through the use of secondary range mea-

surements, we improve the convergence of new estimates, thus making sure that we can quickly swap between different agents to track. While the use of secondary range measurements offers large advantages in terms of stabilizing the estimates and improving convergence, it also brings some disadvantages. Specifically, using secondary range measurements can cause estimates to get stuck at a stable local optimum. To solve this problem, we introduce multiple methods that counter the different types of local optima. Most importantly, these are improvements to the initialization procedure, and the use of covariance inflation during measurement updates. Our results show, that the resulting algorithm offers many advantages over our previous one-on-one relative localization.

This dissertation is of course not the end for the research on localization for swarming drones. In the conclusion, we highlight several additional topics we investigated together with our Master students, and finally give an outlook on the future of the field.

1

INTRODUCTION

In February 2023, a magnitude 7.8 earthquake struck the border region between Turkey and Syria [1]. The tragedy, which ended up claiming more than 50'000 lives, started a race against the time for the search and rescue teams arriving on the ground shortly after. With every passing hour, the chances of finding survivors in the collapsed buildings were decreasing.

Searching through the ruins to find and rescue the people trapped inside is a difficult and dangerous task. A wrong step or an aftershock could easily cause further parts of the buildings to collapse, putting the lives of the rescuers at risk and potentially turning them into victims themselves.

While there are currently few alternatives to scouring the ruins with human search teams, there could be one in the future. Robotic vehicles such as drones have become more and more powerful, evolving from simple toys to valuable tools that can perform a wide variety of tasks. Using a small, autonomous drone to search through the rubble, would mean that rescue teams only have to put themselves in danger when a victim is found. And if we were to use coordinating swarms of drones, we might be able to considerably speed up the search process at the same time [2].

Search and rescue operations are in fact not the only scenario where swarms of autonomous drones could prove to be incredibly beneficial. Many use cases have been suggested over the past years, and although few seem to be as challenging as flying through collapsed buildings, we are still far from seeing any of these ideas become reality.

So why don't we see swarms of drones soar through a greenhouse to monitor the crops [3], whiz through a warehouse counting the items in stock [4], or, search through ruined buildings after natural disasters? Simply put, it is cheaper and easier to perform all of these tasks in the traditional way, as long as drones don't operate fully autonomously. When drones require a human operator, their use only becomes interesting when the task involves accessing hard-to-reach spaces with enough room to maneuver, such as the inspection of wind turbines or industrial furnaces [5].

Making drones operate autonomously is not easy and involves addressing many different challenges. The first problem that needs to be solved, however, is giving the drone

knowledge about its location, and in the case of swarms, the location of its neighbors. This knowledge enables the drone to move to the right location, coordinate with its fellow drones, and find its way back to the base station. While the use of Global Navigation Satellite Systems (GNSS) has largely solved this problem in open areas, indoor environments still present a particular challenge. Since satellite signals can get blocked or deflected, the performance of GNSS is usually insufficient for operating in small, enclosed spaces [6].

In the research underlying this dissertation, we set out to explore new ways of giving our drones a sense of place. Specifically, we are looking at different ways to use Ultra-Wideband ranging for absolute and relative localization of small, swarming drones.

1.1. THE CHALLENGES OF MICRO AIR VEHICLES

Before diving head-first into numbers, algorithms and technical details, let us first have a look at the vehicles we are working with. "Micro Air Vehicles" or MAVs, are small, versatile drones that can typically fly for up to an hour. There are no official definitions on what "small" really means in this context, but there are good reasons for keeping MAVs as small as possible [7]. First of all, small MAVs are safer. Getting hit in the head by 50 grams of plastic is not nice, but bring this up to 1 kilo and one might already be in need of an ambulance. Of course we try to make our drones as reliable as possible, but experience has shown, that things can occasionally go wrong. Second, with indoor applications in mind, a smaller platform allows for more maneuverability in tight spaces, enabling the drone for example to navigate through doors and windows [8].

Finally, especially when working with swarms, the unit cost of a drone becomes important. Small drones are cheaper, because they require less material and smaller batteries. Furthermore, they are easier to transport and handle in large numbers.

These advantages do however come at a price. Making drones smaller and lighter means sacrificing computational power. High performance processors that draw large amounts of power and possibly even require active cooling are not only heavier by themselves, but also require the use of larger, heavier batteries. This limitation adds an additional challenge when designing localization algorithms for MAVs. As a result, some accuracy has to be sacrificed in favor of reducing the computational cost to keep the drones running smoothly.

1.2. ROBOTIC SWARMS

When multiple drones are coordinating to work together, we start talking of a swarm. Inspired by nature, swarms offer the potential of completing complex tasks while keeping the individual members, also called agents, small and simple. This offers additional advantages, such as operational flexibility and robustness: Since swarms rely on large numbers of agents, they can quickly react to changing circumstances, such as new tasks appearing or the failure of another drone [9].

To realize the full potential of a swarm, the individual agents will have to work together. Algorithms for task allocation and path planning can help a swarm collaborate, but usually require some form of estimate of the individual drones positions in space, and/or with respect to each other. There are two important classifications that can be

used to distinguish localization algorithms for swarming [10].

First, regarding the swarm architecture, we can distinguish between centralized and distributed swarms. Centralized swarms rely on a computational hub to collect information on all swarm members, execute the swarming algorithms and send back commands to all members of the swarm. This hub could be one of the drones in the swarm, but more commonly is a ground-based terminal with more extensive computational power. This means, that centralized algorithms can be computationally costly without the need for a lot of computational power on the individual drones. Furthermore, they can work with complete information and therefore usually outperform distributed architectures.

The biggest downsides of centralized architectures are latency, communication bandwidth and robustness, and this is where the appeal of distributed approaches comes in [11]. The requirement for stable communication links between all agents and the base station becomes particularly difficult to achieve in indoor environments (where signals can be blocked), over large distances (where signals are weak) or in fast changing environments (where agents react too slowly). Furthermore, the base station is a single point of failure for the system. In these scenarios, distributed algorithms can offer a distinctive advantage, as every agent executes its own version of the algorithm based on the information available. While the results might be less optimal, these algorithms still work when communication with some of the other members is lost and swarms can continue operation even if multiple members fail.

Second, looking at the source of positioning information, swarms may rely on external positioning or only on local sensing. External positioning such as GNSS or visual tracking provides high accuracy in a global context and reduces on-board computational requirements for the agents. It does however introduce similar risks to a centralized architecture, in that it requires stable communication links with the source of positioning information, and potentially modifications to the environment. Local sensing on the other hand, while often being less accurate and putting additional load on the agents processor, can still work under adverse circumstances such as loss of communication [10].

Looking at aerial swarms developed around the world over the past few years, it becomes clear that it is currently much easier to build large swarms based on a centralized architecture and external positioning information. Decentralized systems with local sensing on the other hand are still in their infancy and these swarms are usually limited to less than 20 robots [10].

While centralized swarms with external positioning are currently encountered more frequently, it is clear that they are limited in terms of scalability and flexibility, as they require an ever increasing amount of computational power to back them up from the ground. We therefore believe that it is only a matter of time until distributed systems with external positioning will take over. When it comes to swarms with local sensing, it is possible that they will never reach the same sizes. After all, GNSS can provide fast and reliable external positions for an unlimited number of receivers. However, for a variety of scenarios where GNSS is not available (indoors or on another planet), local sensing might be the only information available.

Especially for a search and rescue scenario in collapsed buildings as presented in the beginning, distributed swarms with local sensing are the most promising solution, as we

cannot guarantee a stable communication link to the outside.

1.3. INDOOR LOCALIZATION

If an autonomous vehicle is to conduct a complex mission, it needs some form of awareness of its position in the environment. A robot without localization system may be able to autonomously explore an environment and find locations of interest, but it would be incapable of sharing that location with others [12]. Furthermore, without at least a homing beacon, such a robot could not actively travel back to a charging station [13].

Two things are needed to give a robot information about its location: Sensors, with which to extract information from the environment, and filter algorithms, with which to convert these measurements into a position estimate.

1.3.1. SENSORS

Roboticists distinguish between proprioceptive and exteroceptive sensors. Proprioceptive sensors measure internal states of the robot, such as battery voltage or angular velocities, while exteroceptive sensors like cameras or a compass are used to observe the environment [14].

One of the most important proprioceptive sensor for localization that can be found on any drone is the inertial measurement unit (IMU). IMUs measure linear accelerations and angular velocities and have become incredibly small and energy efficient. Unfortunately, while being vital to keeping a drone stable in the air, IMU measurements by themselves are unsuitable for long-term navigation. This stems from the fact, that converting IMU measurements into position estimates requires integration of these measurements, which will lead to an accumulating drift. However, combining high-frequency IMU measurements with lower frequency exteroceptive measurements is a great way to achieve stable and drift-free position estimates [15].

For us humans, vision is probably the most intuitive form of exteroceptive localization, given that most of us use it every single day to find our way through the world. It also comes with the advantage, that it can be used to detect and avoid obstacles [16]. What our brains do with ease is however much more difficult for robots to achieve. Simultaneous localization and mapping (SLAM) for example requires a lot of memory and computational power [17]. An easier way of using visual information, visual-inertial odometry (VIO), consists of integrating movement between consecutive images. This saves memory, but is vulnerable to drift, just like IMU integration [18] [19]. Finally, there is the option of detecting markers at known positions in the area, which does however require previous preparation of the environment [20].

As we noted earlier, GNSS is not a viable solution for most indoor environments [6]. There are however other range-based techniques with similar working principles, that we can use if we allow modifications of the environment. These work by placing beacons at known positions in the room with which robots can then exchange messages. Depending on the ranging-protocols used, the robot can then calculate its distance from a specific beacon, or the difference in distance to two specific beacons. Based on these range measurements, a position can be calculated.

Technologies that fall into this category are Bluetooth [21], Wi-Fi [22] and Ultra-

Wideband (UWB) [23]. For all of these technologies, the Received Signal Strength (RSS) gives a good indication of the distance between the communicating devices. However, this method is not very accurate since walls and other objects can reflect or block part of the signal. A better option can be the use of transmission and reception timestamps that allow for the calculation of the time of flight for a signal. Since the signals travel at the speed of light, the distance between devices can be calculated easily. Time-based protocols that are commonly used are Two-Way Ranging (TWR) and Time-Difference of Arrival (TDoA) [24]. These protocols work best for signals with a large bandwidth, since they offer a better time-resolution.

In this research, we are using Ultra-Wideband as the technology of choice. Ultra-Wideband is a communication technology that uses frequencies in the 3-10 GHz range to exchange messages. Due to the large bandwidth of the channels, UWB can achieve high timing accuracy. This makes it well suited for time-based ranging protocols such as TWR or TDoA, with which it can achieve centimeter level accuracy [23]. In addition to leading to high accuracy in ranging, UWB can be used to communicate information at data-rates of several Mbps. This is of special interest in swarming applications, where agents might need to communicate anyway.

1.3.2. STATE ESTIMATION ALGORITHMS

Typically, a drone will carry more than just one sensor that can give information on its position. For example, our drones might carry an IMU and at the same time perform ranging with Ultra-Wideband. State estimation algorithms are designed to use a system model with which they combine measurements from different sources. This allows them to return a better state estimate than what we could get from one single type of measurement alone.

The most commonly used algorithm is the Kalman Filter [25][15]. The Kalman Filter returns the best linear combination of measurements that minimizes a weighted two-norm of the estimation error [26]. The Kalman Filter works best for linear systems that exhibit uncorrelated, white Gaussian noise, but various adaptations were devised to better deal with systems where this is not the case [27][28]. Due to the iterative calculation of estimates at each time step, the Kalman Filter is extremely efficient in terms of computational cost.

For highly non-linear systems and systems that exhibit non-Gaussian or unknown noise, alternative filters exist. In chapters 2 and 3, we will specifically look at an optimization based approach, known as Moving Horizon Estimation (MHE)[29]. The advantage of this approach is, that it is well suited to deal with non-linear systems (such as drones) and non-Gaussian noise (as the noise present in UWB measurements). The main disadvantage is, that as an optimization based method, it comes at a much higher computational cost.

1.4. RESEARCH OBJECTIVE

After this initial overview regarding the current state of swarms of micro-air vehicles, it is possible to formulate the objective for this research. Having identified UWB as a promising technology for these swarms, we also recall that scalability is a major concern

in swarming applications. The following research objective is therefore formulated:

Research Objective

Provide robotic swarms with scalable means to determine their position in the environment, and with respect to each other, using Ultra-Wideband ranging.

To approach this problem step by step, we split this large objective into several more precise research questions, that will each address a part of the general research objective.

Indoor localization of individual drones using Ultra-Wideband has already been investigated by several other authors and therefore lends itself as a convenient entry point into this research [15][30]. Most of the previous research focuses on the use of Kalman Filters, which have known limitations when it comes to non-linear dynamics and non-Gaussian noise [26]. Given that drones are highly non-linear platforms and UWB ranging exhibits non-Gaussian noise, we want to start by investigating an alternative filter for this purpose, specifically the Moving Horizon Estimator[29]. Due to the computational complexity of this estimator, we place a specific focus on reducing the computational load on the drone.

Research Question 1

How can Moving Horizon Estimation be used to improve localization with Ultra-Wideband on platforms with limited computational resources?

The second aspect of the research objective is related to relative localization with respect to other agents in a swarm. We will first approach this problem in a setting with few drones to avoid the problem of scalability. Instead we will investigate how these small groups of drones can determine their relative positions, and use that information to perform coordinated maneuvers.

Research Question 2

How can Ultra-Wideband be used to achieve spatial coordination in small groups of aerial robots?

Finally, we will tackle the problem of extending these methods to larger, potentially infinite swarms. While infinite swarms of course don't exist, they force us to think of solutions that work in swarms, where you can always add an additional agent without breaking the system.

Research Question 3

How can relative localization be achieved in potentially infinite swarms, despite physical limitations of the individual agents and their communication channel?

1.5. THESIS OUTLINE

Following this introduction to the topic and the research objective, Chapter 2 and Chapter 3 will address Research Question 1 by exploring the use of Moving Horizon Estimation for localization with Ultra-Wideband ranging. In a first step, Chapter 2 will focus on the computationally efficient formulation of the Moving Horizon estimator. Subsequently, Chapter 3 will refine this estimator by improving robustness and further reducing the computational requirements.

Research Question 2 will be addressed in Chapter 4, which presents a relative position estimator for agent-to-agent localization in three dimensions. In addition to an observability analysis and the design of the estimator, the chapter also includes physical experiments, using the relative position estimates to implement a leader-follower behavior.

Chapter 5 will then cover the design and evaluation of a relative position estimator for large, dynamic swarms. Furthermore, the chapter addresses the localization ambiguities that appear due to the use of secondary range measurements and can lead to stable but wrong estimates.

Finally, 6 will conclude this dissertation by looking back at the main results and placing them into a larger context. It will also outline several related investigations that were conducted by Master students under our supervision.

REFERENCES

[1] Wikipedia, *2023 Turkey–Syria earthquakes — Wikipedia, the free encyclopedia,* `http://en.wikipedia.org/w/index.php?title=2023%20Turkey%E2%80%93Syria%20earthquakes&oldid=1178217024` (2023), [Online; accessed 02-October-2023].

[2] J. Q. Cui, S. K. Phang, K. Z. Y. Ang, F. Wang, X. Dong, Y. Ke, S. Lai, K. Li, X. Li, F. Lin, J. Lin, P. Liu, T. Pang, B. Wang, K. Wang, Z. Yang, and B. M. Chen, *Drones for cooperative search and rescue in post-disaster situation,* in *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)* (2015) pp. 167–174.

[3] J. Robbins and J. M. Maja, *Drones in a GREENHOUSE ?* Big Grower , 10 (2018).

[4] L. Wawrla, O. Maghazei, and P. D. T. Netland, *Applications of drones in warehouse operations, Whitepaper,* (2019).

[5] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, *State-of-the-art technologies for UAV inspections,* IET Radar, Sonar and Navigation **12**, 151 (2018).

[6] L. Mainetti, L. Patrono, and I. Sergi, *A survey on indoor positioning systems,* 2014 22nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2014 , 111 (2014).

[7] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, *Towards a swarm of agile micro quadrotors,* Autonomous Robots **35**, 287 (2013).

[8] D. Guo and K. K. Leang, *Image-based estimation, planning, and control for high-speed flying through multiple openings,* The International Journal of Robotics Research **39**, 1122 (2020), https://doi.org/10.1177/0278364920921943 .

[9] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, *Swarm robotics: A review from the swarm engineering perspective,* Swarm Intelligence **7**, 1 (2013).

[10] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, *A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints,* Frontiers in Robotics and AI **7** (2020), 10.3389/frobt.2020.00018.

[11] E. Şahin, *Swarm robotics: From sources of inspiration to domains of application,* in *Swarm Robotics*, edited by E. Şahin and W. M. Spears (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005) pp. 10–20.

[12] C. Dimidov, G. Oriolo, and V. Trianni, *Random walks in swarm robotics: An experiment with kilobots,* in *Swarm Intelligence*, edited by M. Dorigo, M. Birattari, X. Li, M. López-Ibáñez, K. Ohkura, C. Pinciroli, and T. Stützle (Springer International Publishing, Cham, 2016) pp. 185–196.

[13] K. N. McGuire, C. D. Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, *Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment,* Science Robotics **4**, eaaw9710 (2019), https://www.science.org/doi/pdf/10.1126/scirobotics.aaw9710 .

[14] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots* (MIT press, 2011).

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

[16] K. McGuire, G. C. H. E. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, *Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone,* IEEE Robotics and Automation Letters **2**, 1070 (2017).

[17] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, *Vision-based autonomous mapping and exploration using a quadrotor mav,* in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012) pp. 4557–4564.

[18] G. Huang, *Visual-inertial navigation: A concise review,* in *2019 International Conference on Robotics and Automation (ICRA)* (2019) pp. 9572–9582.

[19] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, *Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors,* IEEE Robotics and Automation Letters **3**, 1801 (2018).

[20] A. Babinec, L. Jurišica, P. Hubinský, and F. Duchoň, *Visual localization of mobile robot using artificial markers,* Procedia Engineering **96**, 1 (2014), modelling of Mechanical and Mechatronic Systems.

[21] Y. Zhuang, C. Zhang, J. Huai, Y. Li, L. Chen, and R. Chen, *Bluetooth localization technology: Principles, applications, and future trends,* IEEE Internet of Things Journal **9**, 23506 (2022).

[22] J. Biswas and M. Veloso, *Wifi localization and navigation for autonomous indoor mobile robots,* in *2010 IEEE International Conference on Robotics and Automation* (2010) pp. 4379–4384.

[23] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa, *Ultra wideband indoor positioning technologies: Analysis and recent advances,* Sensors (Switzerland) **16**, 1 (2016).

[24] F. Zafari, A. Gkelias, and K. K. Leung, *A Survey of Indoor Localization Systems and Technologies,* IEEE Communications Surveys and Tutorials **21**, 2568 (2019), arXiv:1709.01015 .

[25] R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems,* Journal of Basic Engineering **82**, 35 (1960).

[26] D. Simon, *Optimal State Estimation* (John Wiley & Sons, Hoboken, NJ, 2006).

[27] R. Mehra, *A comparison of several nonlinear filters for reentry vehicle tracking,* IEEE Transactions on Automatic Control **16**, 307 (1971).

[28] S. Julier and J. Uhlmann, *Unscented filtering and nonlinear estimation,* Proceedings of the IEEE **92**, 401 (2004).

[29] D. A. Allan and J. B. Rawlings, *Moving horizon estimation,* in *Handbook of Model Predictive Control*, edited by S. V. Raković and W. S. Levine (Springer International Publishing, Cham, 2019) pp. 99–124.

[30] Y. Xu, Y. S. Shmaliy, X. Chen, Y. Li, and W. Ma, *Robust inertial navigation system / ultra wide band integrated indoor quadrotor localization employing adaptive interacting multiple model-unbiased finite impulse response / Kalman filter estimator,* Aerospace Science and Technology **98**, 105683 (2020).

2

A Computationally Efficient Moving Horizon Estimator for Ultra-Wideband Localization on Small Quadrotors

We present a computationally efficient moving horizon estimator that allows for real-time localization using Ultra-Wideband measurements on small quadrotors. The estimator uses a single iteration gradient descent method to optimize the state estimate based on past measurements, while using random sample consensus to reject outliers. We compare our algorithm to a state-of-the-art Extended Kalman Filter and show its advantages when dealing with heavy-tailed noise, which is frequently encountered in Ultra-Wideband ranging. Furthermore, we analyze the algorithm's performance when reducing the number of beacons for measurements and we implement the code on a 30 g Crazyflie drone, to show its ability to run on computationally limited devices.

Parts of this chapter have been published in IEEE Robotics and Automation Letters **6**, 4 (2021) [1].

2.1. INTRODUCTION

Over the past few years, the continuing miniaturization of computational hardware has advanced the development of small, agile quadrotors. This new generation of flying robots performs tasks in environments, that were previously inaccessible to them due to size and safety constraints. Most notably, this concerns indoor environments, such as warehouses, greenhouses or industrial facilities, which present numerous new potential use-cases.

A key component required for autonomous operation in these environments is a robust and accurate localization system. Ultra-Wideband (UWB) technology has been gaining a lot of attention for indoor positioning, as it offers high accuracy at an affordable cost. UWB systems allow for the localization of a mobile tag, by exchanging messages with a set of static beacons at known locations in the environment. Although a position can be calculated from a sufficient number of independent UWB measurements through multilateration, better results can be achieved when using additional filtering. Commonly, a state estimator that can fuse the UWB measurements with data collected by the on-board inertial measurement unit (IMU) is employed.

For linear systems with Gaussian noise, the Kalman Filter yields an optimal estimate in the sense that it minimizes a weighted two-norm of the expected value of the estimation error [2]. Different formulations have been developed to approximate this result for more general systems, while retaining the computationally efficient, iterative structure of the Kalman Filter. The Extended Kalman Filter (EKF) [2] or Unscented Kalman Filter (UKF) [3] are often used to address non-linearities, but may fail to address the issues that can arise from uncertain system models and unknown or non-Gaussian noise. Since UWB measurements often exhibit heavy-tailed noise due to multi-pathing and non-line-of-sight (NLOS) conditions [4, 5], this can be a significant shortcoming.

For highly non-linear problems and systems with non-Gaussian noise, Moving Horizon Estimation (MHE) yields better results at the cost of requiring more computational power. MHE aims at finding a sequence of states, that minimizes the noise and disturbances required to explain the observed measurements over a moving time horizon [6]. A multitude of options exist for the formulation of the cost-function to this minimization problem, but most commonly, a batch least-squares formulation is used [7]. In addition, past data can be included in the cost-function through an "arrival cost". Unfortunately, the requirement of solving an optimization problem at every time step currently make MHE impractical for real-time applications on computationally limited devices or for systems with fast dynamics.

This chapter presents a computationally efficient Moving Horizon Estimator with RANSAC [8] outlier rejection that can perform real-time localization with UWB on a small quadrotor with limited computational power. We compare it with a state-of-the-art EKF in simulated noise environments and show its advantages when dealing with heavy-tailed noise, which is observed on UWB measurements when multi-pathing and NLOS effects occur. We also show that the MHE is still able to perform state estimation when the number of UWB beacons in the environment is drastically reduced.

In Section 2.2 we will present an overview of the related work. Section 2.3 outlines the design of our computationally efficient moving horizon estimator, while Section 2.4 will cover its implementation for position estimation with UWB. We will then compare our

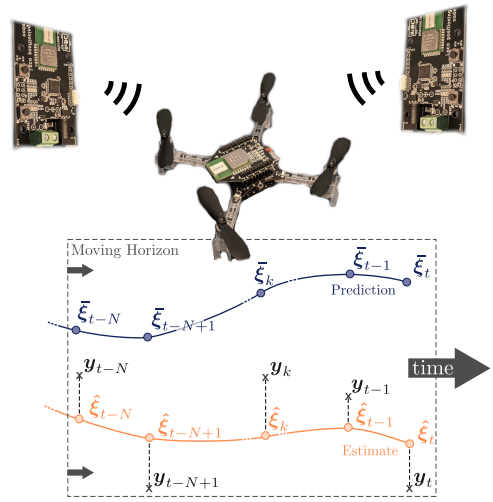


Figure 2.1: Moving horizon estimators can optimize the evolution of a drone's trajectory based on a limited number of previous UWB measurements. We propose a novel formulation for UWB localization that is efficient enough to run on a 30 g Crazyflie drone and performs robustly in scenarios with very noisy or limited measurements. In the plot, $\boldsymbol{\xi}$ stands for the state and $\boldsymbol{y}$ for a measurement of the state.

MHE with a state-of-the-art EKF in Section 2.5. Finally we will conclude on our findings in Section 2.6.

2.2. RELATED WORK

Shortly after the original paper on the Kalman Filter appeared in 1960 [9], the concept of limited memory filters was introduced by Andrew Jazwinski to address cases in which the Kalman Filter diverges due to modelling errors [10]. Several strategies have since been proposed to reduce the high computational cost of such limited memory filters.

The Unbiased Finite Impulse Response (UFIR) Filter was developed with a computationally efficient iterative structure like the Kalman Filter. It offers robustness with respect to unknown noise and initial position, but still relies on linearization when dealing with non-linear system models. Furthermore, the UFIR filter is only unbiased and not optimal [11].

Moving Horizon Estimation (MHE) on the other hand solves an online optimization problem and can therefore accomodate non-linear system models. However, the solution of these optimization problems can quickly become expensive, which has mostly limited the use of MHE to slow processes, e.g. in the chemical industry. There are however approaches to improve the computational complexity of moving horizon estimation. Zavala, Laird and Biegler [12] have used sensitivity analysis to formulate and solve a reference optimization problem based on previous measurements, while waiting for a new measurement to arrive. Once the new measurement is known, the solution to the real problem can be calculated more quickly. Kühl et al. [13] have demonstrated a scheme that only relies on a single iteration of a sequential quadratic programming

method per time step. Both papers demonstrate performance on examples from the chemical industry, which deal with more states but involve significantly slower dynamics than quadrotors.

For this work, the use of computationally more efficient gradient descent methods was investigated, which where proven to be stable by Alessandri and Gaggero [14]. Similar to Kühl et al. they were able to show, that also with simple gradient descent minimization, a single iteration at every time step is sufficient to get a stable state estimate.

On autonomous flying robots, MHE is rarely used due to the before mentioned computational complexity. Girrbach et al. have investigated the influence of horizon length when fusing GNSS and IMU with MHE, but only perform comparisons offline and without addressing the challenges of computational cost [15]. Shuo et al. formulate the MHE problem with position measurements and a simplified model in such a way, that there is an analytical solution to the optimization problem. This allows them to use their MHE on a 72 g quadrotor in a drone racing context [16].

Instead of using MHE, state estimation with UWB measurements on quadrotors is usually based on variations of the Kalman Filter. The Crazyflie drone which we use as our testing platform in this work for example uses an EKF for fusing data from a wide variety of on-board sensors [17]. As an alternative to Kalman Filters, Xu et al. [18, 19] recently started to investigate different forms of UFIR filters for the purpose of fusing UWB and IMU on drones.

In our work, we are approaching the problem of sensor fusion on quadrotors by using a moving horizon estimator, which uses a single iteration gradient descent algorithm to optimize its state-estimate. We will show that this reduces the computational load of MHE sufficiently, while providing accurate position estimates on computationally limited drones.

2.3. MOVING HORIZON ESTIMATOR

2.3.1. PROBLEM FORMULATION

Let us consider a non-linear, discrete-time dynamic system as described by (2.1):

$$\begin{cases} \boldsymbol{\xi}_{t+1} &= \boldsymbol{f}(\boldsymbol{\xi}_t, \boldsymbol{u}_t) + \boldsymbol{w}_t \\ \boldsymbol{y}_t &= \boldsymbol{h}(\boldsymbol{\xi}_t, \boldsymbol{u}_t) + \boldsymbol{v}_t \end{cases} \tag{2.1}$$

where $t = 0, 1, \ldots$ is the time step, $\boldsymbol{\xi}_t \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}_t \in \mathbb{R}^p$ is the (control) input vector, and $\boldsymbol{y}_t \in \mathbb{R}^m$ is the measurement vector. Finally $\boldsymbol{w}_t \in \mathbb{R}^n$ and $\boldsymbol{v}_t \in \mathbb{R}^m$ are the process and measurement noise vectors, all at time step t respectively. The evolution of the state is described by the non-linear prediction equation $\boldsymbol{f}$, while the measurements are described by the non-linear measurement equation $\boldsymbol{h}$.

The classical MHE problem can be formulated, as finding the state $\hat{\boldsymbol{\xi}}_{t-N|t}$, that minimizes the cost function $J(\hat{\boldsymbol{\xi}}_{t-N|t})$, where $\hat{\boldsymbol{\xi}}_{t-N|t}$ is the estimated state at time step $t-N$, based on information available at time step t, N being the length of the moving horizon. The cost function typically includes an arrival cost, which takes into account a prior estimate of the state, $\bar{\boldsymbol{\xi}}_{t-N|t}$, the sum of squares of the process noise and the sum of squares of the measurement noise. In this work, we simplify the problem to an output noise MHE, which neglects the process noise terms. This simplification is necessary to reduce

the size of the optimization problem sufficiently for it to be solved on an embedded system. It removes the ability of the estimator to adjust inputs to better fit the measurements and is therefore likely to reduce the accuracy of the estimate. Since it does however not impact the convergence of the estimate, this can be an acceptable trade-off for many systems.

Since the measurement frequency of UWB varies unpredictably, some measurements in $\boldsymbol{y}_k$ might be outdated. We can accommodate for this fact, by writing the cost function in terms of individual measurements y_i, $i = 0,\ldots,m$, and including the binary switching sequences $\theta_{i,k}$, which are equal to 1 if a new measurement arrives and 0 otherwise [20].

$$\theta_{i,k} = \begin{cases} 1 & \text{if } y_{i,k} \neq y_{i,k-1} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

To express the cost function in term of the state at the beginning of the horizon, $\hat{\boldsymbol{\xi}}_{t-N|t}$, it is helpful to define the composite prediction and measurement equations at time step k:

$$\mathscr{F}_k(\bar{\boldsymbol{\xi}}_{t-N|t}) = \boldsymbol{f}^{u_k} \circ \ldots \circ \boldsymbol{f}^{u_{t-N}}(\bar{\boldsymbol{\xi}}_{t-N|t}) \tag{2.3}$$

$$\mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N|t}) = h_i^{u_k} \circ \mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N|t}) \tag{2.4}$$

For compactness, we omit the input vectors $\boldsymbol{u}_{t-N}$ to $\boldsymbol{u}_k$ in the composite functions, as the output error structure of the MHE does not allow them to change anyway. For the composing functions we note the corresponding input vector in the superscript. This results in the final cost function shown in (2.5).

$$J_t(\hat{\boldsymbol{\xi}}_{t-N|t}) = \mu\|\hat{\boldsymbol{\xi}}_{t-N|t} - \bar{\boldsymbol{\xi}}_{t-N|t}\|^2 + \sum_{k=t-N}^{t}\sum_{i=1}^{m}\left(\theta_{i,k}\cdot\|y_{i,k} - \mathscr{H}_{i,k}(\hat{\boldsymbol{\xi}}_{t-N|t})\|^2\right) \tag{2.5}$$

2.3.2. COMPUTATIONALLY EFFICIENT MHE

To solve the MHE problem efficiently, we use a single iteration of the gradient method at every time step [14]. The prior estimate $\bar{\boldsymbol{\xi}}_{t-N|t}$ is obtained by applying the prediction equations on the last available state estimate.

$$\hat{\boldsymbol{\xi}}_{t-N|t} = \bar{\boldsymbol{\xi}}_{t-N|t} - \alpha\nabla J_t(\bar{\boldsymbol{\xi}}_{t-N|t}) \tag{2.6}$$

$$\bar{\boldsymbol{\xi}}_{t-N|t} = \boldsymbol{f}(\hat{\boldsymbol{\xi}}_{t-N-1|t-1}, \boldsymbol{u}_{t-N-1}) \tag{2.7}$$

If the number of states n becomes large, the numerical calculation of $\nabla J_t(\bar{\boldsymbol{\xi}}_{t-N|t})$ becomes very expensive. This is due to the fact, that every evaluation of $J_t(\bar{\boldsymbol{\xi}}_{t-N|t})$ requires the prediction of the state vector's evolution over the complete horizon. We are therefore using an iterative, analytical solution based on the chain rule, which only requires a single prediction cycle over the full horizon.

In the analytical expression of the gradient of the cost function, the first term (arrival cost) disappears since we evaluate the gradient at $\bar{\boldsymbol{\xi}}_{t-N|t}$. What remains is a sum of terms that stem from the individual measurements:

$$\nabla J_t(\bar{\boldsymbol{\xi}}_{t-N|t}) = -2\sum_{k=t-N}^{t}\sum_{i=1}^{m}\left[\theta_{i,k}\left(y_{i,k} - \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N|t})\right)\nabla\mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N|t})\right] \tag{2.8}$$

By stepping through the full horizon, we can calculate the sequence of prior estimates $\bar{\boldsymbol{\xi}}_{k|t}$, $k = t-N,\dots,t$ in an iterative manner, which makes calculating the predicted measurement $\mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N|t}) = h_i(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k)$ trivial.

To calculate the gradient of $\mathscr{H}_{i,k}$, we use the chain rule and split the equation into two factors: The gradient based on the current predicted state and input, $\nabla h_i(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k)$, and the Jacobian of the composite prediction equation $D\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N|t})$:

$$\nabla \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N|t}) = \left(D\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N|t})\right)^T \cdot \nabla h_i(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k) \tag{2.9}$$

$$D\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N|t}) = D\mathbf{f}\left(\bar{\boldsymbol{\xi}}_{k-1|t}, \boldsymbol{u}_{k-1}\right) \cdot D\mathbf{f}\left(\bar{\boldsymbol{\xi}}_{k-2|t}, \boldsymbol{u}_{k-2}\right)\right) \cdot \ldots \cdot D\mathbf{f}\left(\bar{\boldsymbol{\xi}}_{t-N|t}, \boldsymbol{u}_{t-N}\right) \tag{2.10}$$

Similar to the predicted state, the Jacobians of the prediction equation can be calculated in an iterative manner by premultiplying the previous value with the Jacobian evaluated at the current prediction. In the end, calculating the Jacobian of the cost function therefore requires the following computations for each time step k in the horizon (i.e. N times):

- Forward prediction of the prior estimate

$$\bar{\boldsymbol{\xi}}_{k|t} = \boldsymbol{f}(\hat{\boldsymbol{\xi}}_{k-1|t}, \boldsymbol{u}_{k-1}) \tag{2.11}$$

- Calculation of the $n \times n$ Jacobian of the prediction function f at $t = k-1$

$$D\mathbf{f}(\bar{\boldsymbol{\xi}}_{k-1|t}, \boldsymbol{u}_{k-1}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\xi}}(\bar{\boldsymbol{\xi}}_{k-1|t}, \boldsymbol{u}_{k-1}) \tag{2.12}$$

- Calculation of the composite Jacobian of $\mathscr{F}_{k-1}$ by multiplication of two $n \times n$ matrices (chain rule)

$$D\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N|t}) = D\mathbf{f}(\bar{\boldsymbol{\xi}}_{k-1|t}, \boldsymbol{u}_{k-1}) \cdot D\mathscr{F}_{k-2}(\bar{\boldsymbol{\xi}}_{t-N|t}) \tag{2.13}$$

- Up to m measurement predictions for measurements that arrived at $t = k$,

$$h_i(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k) \tag{2.14}$$

- Calculation of up to m measurement gradients for measurements that arrived at $t = k$,

$$\nabla h_i(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k) = \frac{\partial h_i}{\partial \boldsymbol{\xi}}(\bar{\boldsymbol{\xi}}_{k|t}, \boldsymbol{u}_k) \tag{2.15}$$

- Calculation of up to m composite measurement gradients by multiplication of an $n \times n$ matrix with a size n vector, (2.9)

- Up to m subtractions, $m \cdot n$ multiplications and $m \cdot n$ additions to form the full measurement sum at $t = k$, (2.8)

To complete the estimation process, the gradient descent step needs to be performed (2.6), which requires n multiplications and n subtractions. Finally, a forward prediction of $\hat{\boldsymbol{\xi}}_{t-N|t}$ must be performed to actually know the current state $\hat{\boldsymbol{\xi}}_{t|t}$, which requires k evaluations of the prediction equation $\boldsymbol{f}$.

From these steps, the most expensive calculation in terms of computational complexity is the multiplication of two $n \times n$ matrices, which behaves as $\mathcal{O}(n^3)$ without using specialized algorithms. Neglecting lower order terms, our MHE therefore ends up with a computational complexity of $\mathcal{O}(N \cdot n^3)$. With respect to the number of states, our MHE therefore scales just as well as the EKF and the increase in complexity only comes from the number of time steps in the moving horizon.

2.3.3. RANSAC OUTLIER REJECTION

The occurrence of outliers in UWB measurements is a problem for any state estimator, as they can have a big impact on the estimated state. To deal with outliers, other authors have used the Mahalanobis distance [17, 18], which weighs the measurement error with its variance. To eliminate any reliance on the knowledge of error statistics, we have decided to use a random sample consensus (RANSAC) scheme instead.

To do this in an efficient manner, we split the measurement terms in (2.8) in two sums: A common sum and a RANSAC sum. The common sum includes measurements that are not subject to RANSAC outlier rejection (e.g. altitude measurements) while the RANSAC sum includes only a fraction of the terms from UWB measurements. By adding to multiple RANSAC sums at the same time, multiple gradients of the cost function can be computed after going through only one prediction cycle for the full horizon.

Since this removes the need to step through the full horizon several times, the only real contribution of the outlier rejection to the computational cost is from the evaluation of the different RANSAC estimates. The evaluation of every RANSAC estimate still requires an independent prediction cycle over the full horizon, to choose the best estimate.

2.4. IMPLEMENTATION

In our implementation we limit ourselves to the estimation of the drone's position and velocity, while the attitude is estimated using Mahony's AHRS algorithm [21].

2.4.1. PREDICTION EQUATIONS

We use a slightly simplified version of the model in [17], where we represent the attitude with quaternions and neglect the dependence of the drag on the propellers' angular velocities.

$$\dot{\boldsymbol{x}} = \left[\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\rho} \end{bmatrix} \otimes \boldsymbol{q}^{-1} \right]_v \tag{2.16}$$

$$\dot{\boldsymbol{\rho}} = \frac{f}{m} \boldsymbol{e_3} + (\mathbf{K}_a - [[\boldsymbol{\omega}_\times]])\, \boldsymbol{\rho} - \left[\boldsymbol{q}^{-1} \otimes \begin{bmatrix} 0 \\ \boldsymbol{g} \end{bmatrix} \otimes \boldsymbol{q} \right]_v \tag{2.17}$$

In this model, our six dimensional state $\boldsymbol{\xi} = [\boldsymbol{x}^T, \boldsymbol{\rho}^T]^T$ encompasses the position $\boldsymbol{x} \in \mathbb{R}^3$ in the global frame and the local velocities $\boldsymbol{\rho} \in \mathbb{R}^3$ in the body frame. The inputs to the

Algorithm 1: Computationally efficient MHE

input : u_t, y_t
output: $\hat{\xi}_t$

$N_t \leftarrow 0$
$\bar{u} \leftarrow [\,], \bar{y} \leftarrow [\,]$
$\bar{\xi}_{\text{prior}} \leftarrow \xi_0$
while *true* **do**
  Update the moving window
  if $N_t < N_{max}$ **then**
    $\bar{u}[N_t] \leftarrow u_t$
    $\bar{y}[N_t] \leftarrow y_t$
    $N_t \leftarrow N_t + 1$
  else
    $\bar{\xi}_{\text{prior}} \leftarrow f(\hat{\xi}_{t-N}, \bar{u}[0])$
    `shift_left(`$\bar{u}$`)`
    `shift_left(`$\bar{y}$`)`
    $\bar{u}[N_t] \leftarrow u_t$
    $\bar{y}[N_t] \leftarrow y_t$
  Calculate the gradient of the cost function
  $\xi \leftarrow \bar{\xi}_{\text{prior}}$
  $D\text{F} \leftarrow \mathscr{I}_{n\times n}$
  $M_\Sigma \leftarrow 0$
  for $k \in [0, \ldots, N_t]$ **do**
    if $k \neq 0$ **then**
      $D\text{F} \leftarrow D\text{f}(\xi, \bar{u}[k-1]) \cdot D\text{F}$
      $\xi \leftarrow f(\xi, \bar{u}[k-1])$
    for $i \in [0, \ldots, m-1]$ **do**
      $\mathscr{H}_{i,k} = h_i(\xi, \bar{u}[k])$
      $\nabla\mathscr{H}_{i,k} = D\text{F}^T \cdot \nabla h_i(\xi, \bar{u}[k])$
      $M_\Sigma \leftarrow M_\Sigma - \theta_{i,k}\left(\bar{y}[k][i] - \mathscr{H}_{i,k}\right) \cdot \nabla\mathscr{H}_{i,k}$
  Calculate the new estimate
  $\nabla J \leftarrow -2 \cdot M_\Sigma$
  $\hat{\xi}_{t-N} \leftarrow \bar{\xi}_{\text{prior}} - \alpha\nabla J$
  $\hat{\xi}_t \leftarrow \hat{\xi}_{t-N}$
  for $k \in [0, \ldots, N_t]$ **do**
    $\hat{\xi}_t \leftarrow f(\hat{\xi}_t, \bar{u}[k])$
  `yield(`$\hat{\xi}_t$`)`

system are the total thrust f, the attitude quaternion $\boldsymbol{q} \in \mathbb{R}^4$, and the angular velocities as measured by the gyroscopes, $\boldsymbol{\omega} \in \mathbb{R}^3$. The drone's mass m and drag coefficients $\mathbf{K}_\mathrm{a} = \mathrm{diag}(k_\perp, k_\perp, k_\parallel)$ enter as parameters. $\otimes$ is the quaternion product and the subscript 0 and v refer to the scalar and vector portion of the quaternion respectively. Finally, $\boldsymbol{g}$ is the directed acceleration of gravity and $[[\boldsymbol{\omega}_\times]]$ is the skew symmetric matrix defined as

$$[[\boldsymbol{\omega}_\times]] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.18}$$

Since inputs are usually available at higher rates than the estimator's update rate, we accumulate these inputs between two estimation cycles and average them to reduce input noise. For the thrust and each of the gyroscope's axes, this is done by simply calculating the mean, while for the attitude quaternion, the average is found by calculating the dominant eigenvector of the matrix Q [22]:

$$Q = \sum_i \boldsymbol{q}_i \boldsymbol{q}_i^T \tag{2.19}$$

2.4.2. MEASUREMENT EQUATIONS

We consider two different types of UWB measurements that can be used with the estimator. In two-way ranging (TWR), the UWB system returns the distance between the tag at its current position, and the (known) location of an anchor, $\boldsymbol{p}_{\mathrm{uwb},i}$. The measurement equation, including measurement noise η_twr is then given by

$$y_\mathrm{twr} = \|\boldsymbol{p}_{\mathrm{uwb},i} - \boldsymbol{x}\| + \eta_\mathrm{twr} \tag{2.20}$$

Alternatively, time-difference of arrival (TdoA) measurements can be used. In TdoA schemes, the measurement yields the difference between the distances to two different anchors:

$$y_{\mathrm{tdoa},ij} = \|\boldsymbol{p}_{\mathrm{uwb},i} - \boldsymbol{x}\| - \|\boldsymbol{p}_{\mathrm{uwb},j} - \boldsymbol{x}\| + \eta_\mathrm{tdoa} \tag{2.21}$$

Since in most indoor environments, the distribution of anchors is mostly in the xy-plane with comparatively small variations in height, altitude estimation based on UWB alone is often difficult and noisy. We therefore also include the measurement equations for dedicated altitude measurements from a barometer or from laser time-of-flight (ToF) measurements:

$$y_\mathrm{baro} = x_3 + z_\mathrm{ref} + \eta_\mathrm{baro} \tag{2.22}$$

$$y_\mathrm{tof} = x_3 \cdot \cos\theta \cos\phi + \eta_\mathrm{tof} \tag{2.23}$$

Note that the ToF measurement equation must take into account the current pitch θ and current roll ϕ since the laser is not measuring the shortest distance to the ground, but the distance on the body frame's z-axis.

2.5. RESULTS

2.5.1. SETUP

Figure 2.2: Flight data was collected in the "Cyberzoo" at TU Delft using a Crazyflie drone and the loco positioning system by bitcraze. Groundtruth is provided by an Optitrack motion capture system.

For testing and evaluation, we implemented our own algorithm and the EKF from [17] (which is used on the Crazyflie) in Python[1]. We compared their real-time position estimates at each timestep t, $\hat{\boldsymbol{\xi}}_{t|t}$, with ground-truth from an optitrack motion capture system. The evaluation was performed on real IMU and UWB data, which was collected on a Crazyflie drone at 100 Hz, and optitrack data collected in the "Cyberzoo" testing area at the Aerospace Faculty of TU Delft[2](seen in Fig. 2.2). In total, 24 data sets were gathered on 6 different trajectories, containing either two-way ranging (TWR) or time-difference of arrival (TdoA) data from the UWB system. The different trajectories used were a square, a triangle, an octagon, an hourglass, a five-pointed star and a random sequence of points. On each trajectory, we recorded four runs, two for TWR data and two for TdoA data. The UWB setup included eight beacons, positioned roughly in the eight corners of a cube, enclosing our testing area (cf. Table 2.1). An example for a run of the Octagon trajectory where we test the estimator performance offline but on real TWR data is shown in Fig. 2.3.

2.5.2. HEAVY-TAILED MEASUREMENT NOISE

While the noise on TWR measurements can be assumed to be Gaussian in an ideal scenario, it is more realistically modelled as being heavy-tailed. This is due to multi-pathing and non-line-of-sight (NLOS) effects, which cause UWB signals to arrive with a slight delay compared to the shortest path [4, 5]. To simulate heavy-tailed noise for TWR, we

[1] https://github.com/SUPfeiffer/uwb-simulator
[2] Data available at https://doi.org/10.4121/14827680

ID	x [m]	y [m]	z [m]
0	-4.48	-4.87	1.16
1	-4.78	4.52	0.67
2	4.81	4.57	0.81
3	4.91	-4.77	0.96
4	-4.34	-5.05	2.60
5	-4.91	4.41	2.65
6	4.56	4.77	2.59
7	4.91	-4.77	2.53

Table 2.1: UWB Beacon Positions

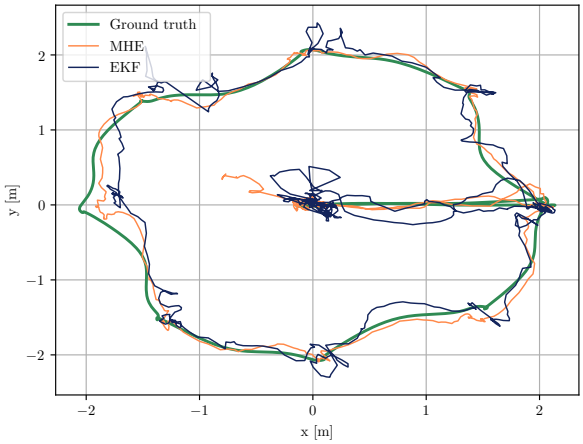


Figure 2.3: Example of one of the collected trajectories (Octagon) with offline performance of MHE and EKF on real TWR data from 4 beacons.

adapt the noise model by Jimenez and Seco [4], which models the heavy-tailed noise as the sum of a Gaussian distribution for the LOS component, a Gamma distribution for the NLOS component, and a constant for outliers. To avoid dealing with a probability density function (PDF) that does not integrate to 1, we remove the constant, and rely on getting outliers from the Gamma distribution alone. To show the effect of an increasingly non-Gaussian distribution, we multiply the Gamma distribution with a scale factor s_{ht}, which we vary between 0 and 1.25. Since a reduction in the NLOS component should also cause the mean to go towards zero, we multiply μ by the same factor, which causes the noise distribution to represent the ideal LOS condition (i.e. a pure, zero-mean Gaussian) for $s_{\text{ht}} = 0$. Finally, we divide the PDF by $(1 + s_{\text{ht}})$, which causes the integral to be

equal to 1. The resulting model is given in (2.24).

$$f(x) = \frac{1}{1+s_{\text{ht}}} \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-s_{\text{ht}}\mu)^2}{2\sigma^2}\right) + \frac{s_{\text{ht}}}{1+s_{\text{ht}}} \cdot \frac{\lambda^k}{(k-1)!} x^{k-1} \exp(-\lambda x) \tag{2.24}$$

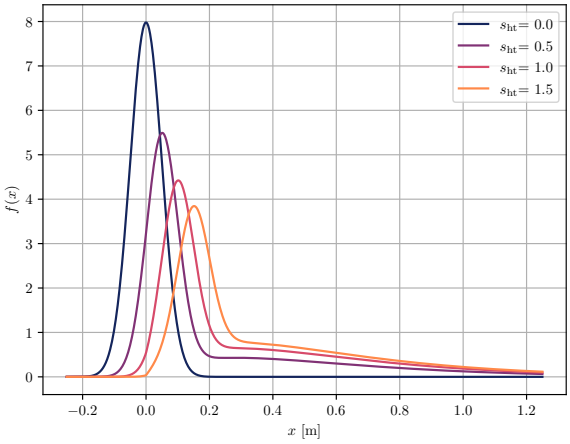


Figure 2.4: Probability Density Function (PDF) of the heavy-tailed noise distribution used for generating UWB measurements. The scale factor s_{ht} is used to change the influence of the heavy-tailed Gamma distribution added to the underlying Gaussian.

Even though there are significant differences between individual anchors, we find that using the parameters suggested by Jimenez and Seco ($\mu = 0.1$, $\lambda = 3.5$, $k = 2$) and a standard deviation of $\sigma = 0.1$ (determined from our own data), our TWR data is best fit with a heavy-tail scale factor of $s_{\text{ht}} = 0.2$. The main differences between anchors are the need for some higher values for λ and k, but we also noted some anchors with shifted or multiple peaks in the noise distribution. These odd behaviours might originate from localized biases on different trajectories and were not modeled in this research.

Since TdoA measurements can be seen as resulting from two TWR measurements to different beacons, the heavier tail appears on both sides and is better modeled by using a Cauchy distribution. To vary the intensity of the heavy-tail, we model the noise distribution for TdoA as the weighted sum of a Cauchy distribution ($\gamma = 0.3$) and a Gaussian distribution ($\sigma = 0.3$), both centered around $\mu = 0$. The resulting noise model, shown in (2.25) best matches the recorded TdoA data for a relative weight of the Cauchy distribution of $r_{\text{ht}} = 0.5$.

$$f(x) = (1-r_{\text{ht}}) \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) + r_{\text{ht}} \cdot \left[\pi\gamma\left(1+\left(\frac{x}{\gamma}\right)^2\right)\right]^{-1} \tag{2.25}$$

Using these noise models, we compare the performance of MHE and EKF on a mix of real IMU data combined with simulated UWB measurements from 4 anchors (IDs

1,3,4,6) that provide measurements on average every 0.05 s (TWR) or 0.1 s (TdoA) each. We run the simulation 5 times on every set of IMU data collected, for a total of 120 runs per value of s_{ht} and r_{ht}. Fig. 2.5 and Fig. 2.6 show the average RMSE of these runs for TWR and TdoA measurements respectively, with the shaded regions representing the standard error of the mean.

In the case of TWR, both estimators perform similarly well, with the EKF having a slight edge for Gaussian noise and the MHE showing an advantage for noise with a stronger heavy tail. In the case of TdoA, the MHE performs better over the full range of simulated noise, with the difference increasing at higher values of r_{ht}. The cause for this difference between TWR and TdoA (also for the Gaussian noise) most likely stems from the larger standard deviation for TdoA, which also leads to more outliers.

2.5.3. REMOVING UWB ANCHORS

Of particular interest in the field of UWB localization is the reduction of required fixed infrastructure, namely the number of beacons that need to be placed in the environment. To assess the performance of our estimator in applications with respect to the number of beacons, we use our full data sets (IMU and UWB) and remove the measurements from randomly selected beacons. Specifically, we run the filters with real UWB data with the number of enabled beacons ranging from 1-8 for TWR and 2-8 for TdoA (TdoA measurements require two beacons). We perform 10 runs for every number of beacons on each of the 12 corresponding (TWR or TdoA) data sets, for a total of 120 runs per number of beacons. For each individual run, we randomly select the beacons used, since the placement of beacons with respect to the trajectory was found to have a significant effect on performance for low numbers of beacons. Due to beacons losing connection on three TdoA trajectories, we only use 9 of the 12 data sets for the TdoA comparison. Results of these runs are shown in Fig. 2.7 and Fig. 2.8 respectively.

One of the downsides of the proposed gradient-descent method is that the magnitude of the cost function's gradient $\nabla J(\hat{\boldsymbol{\xi}}_{t-N|t})$ varies with the number of measurements. This means that the value of the step size α had to be slightly adjusted depending on the number of beacons.

As was to be expected, the lower number of beacons causes a significant increase in the estimation error. While the MHE still shows an advantage, the observed errors are higher than with simulated measurements. This is likely due to inaccuracies in measuring out the anchor positions, and spatially varying biases that have also been mentioned by [17]. In fact, we found that measurements from some beacons exhibit far more challenging noise than what is modeled by a heavy-tailed distribution, such as multiple peaks. For TDOA measurements, both estimators are failing quite often. Indeed, there seems to be a much larger discrepancy between simulated and real TDOA data than for TWR. This is probably due to the fact, that with a large number of beacons, the TDOA algorithm requires a lot of communication between beacons, causing the frequency of the individual measurements to go down as more beacons are present. When removing beacons after the measurements were taken, this leads to much lower measurement frequencies than otherwise expected. It is however still visible, that the reduced measurement frequency seems to have a smaller effect on the MHE's performance.

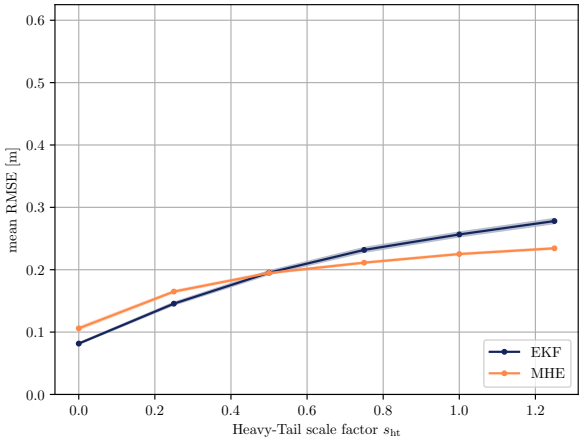


Figure 2.5: Estimator performance on TWR measurements with increasingly heavy-tailed noise. Shaded regions represent the standard error of the mean.

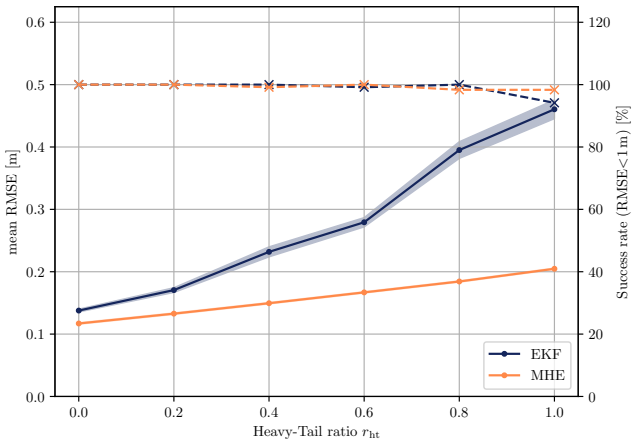


Figure 2.6: Estimator performance on TdoA measurements with increasingly heavy-tailed noise. Dashed lines show success rate of the estimator (RMSE<1 m). Unsuccessful runs are excluded from the calculation of the mean RMSE.

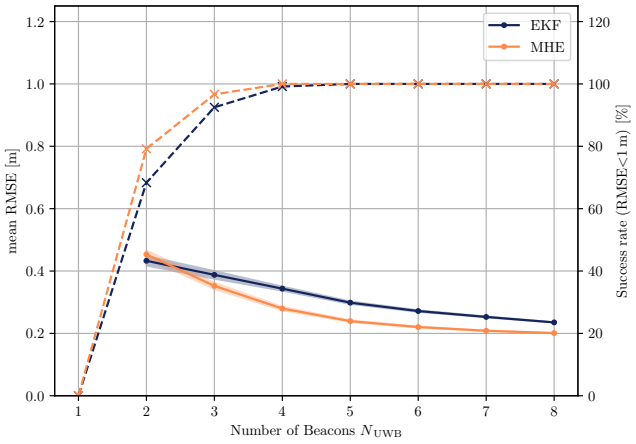


Figure 2.7: Estimator performance on TWR measurements with varying number of beacons. Dashed lines show the success rate (RMSE<1 m), shaded regions represent the standard error of the mean.

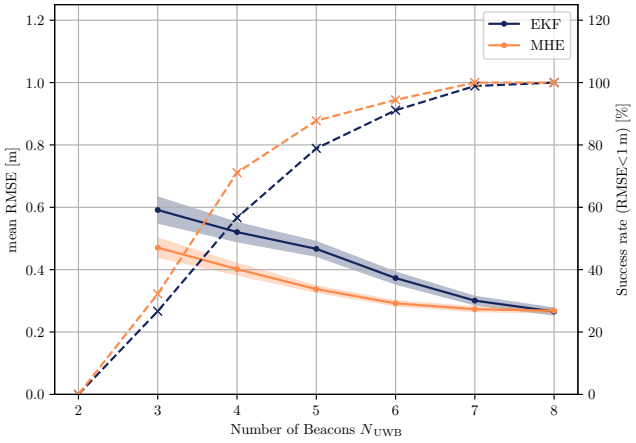


Figure 2.8: Estimator performance on TdoA measurements with varying number of beacons. Dashed lines show the success rate (RMSE<1 m), shaded regions represent the standard error of the mean.

2.5.4. COMPUTATIONAL COMPLEXITY

To compare the computational complexity, we implement our algorithm as a separate task on a Crazyflie drone which performs these computations on an STM32F405 microcontroller. This task replaces the EKF task in the original Crazyflie firmware, so that only one estimator is running at the same time.

By analyzing task dumps, we can identify how much of the available computational resources a task uses. When using the same settings as in simulation, the MHE task accounted for about 15% of the CPU load, while the EKF only accounted for about 11%. This shows the potential of the algorithm to indeed be used on computationally limited devices. With more potent microprocessors being released continuously, we believe that computationally efficient MHE algorithms can become useful for many applications, for which they were previously considered too expensive.

2.6. CONCLUSION

A computationally efficient MHE algorithm for quadrotor localization with UWB was developed and tested. We showed that by simplifying the problem to an output noise MHE and using a single-iteration gradient descent method, MHE can yield good results on computationally limited devices. While these simplifications helped us in reducing the computational cost of MHE, they also pose some limitations on the applications of the algorithm.

The simplification to an output noise MHE assumes that the process model is perfectly known and does not exhibit any process noise, which is rarely the case. Even though the algorithm performed well, there is potential for improvement, especially when only few measurements are available.

Using a simple single-iteration gradient descent algorithm causes the optimization for the correct state to happen over several time steps, and requires the step-size α to be well chosen to keep the algorithm stable. Especially when the number of measurements is varying, this can become difficult, due to the varying magnitude of the gradient of the cost function. We found that conservative values of α can yield stable results for large variations in the number of measurements, but the performance will be worse when fewer measurements are available.

We plan to address these issues in future research to further improve the accuracy of the proposed method and make it useful for a wider variety of tasks.

REFERENCES

[1] S. Pfeiffer, C. De Wagter, and G. C. H. E. de Croon, *A computationally efficient moving horizon estimator for ultra-wideband localization on small quadrotors,* IEEE Robotics and Automation Letters **6**, 6725 (2021).

[2] D. Simon, *Optimal State Estimation* (John Wiley & Sons, Hoboken, NJ, 2006).

[3] S. J. Julier and J. K. Uhlmann, *New extension of the Kalman filter to nonlinear systems,* in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Vol. 3068, edited by I. Kadar, International Society for Optics and Photonics (SPIE, 1997) pp. 182 – 193.

[4] A. R. Jiménez and F. Seco, *Comparing Ubisense, BeSpoon, and DecaWave UWB Location Systems: Indoor Performance Analysis,* IEEE Transactions on instrumentation and Measurement **66**, 2106 (2017).

[5] M. Kok, J. D. Hol, and T. B. Schon, *Indoor positioning using ultrawideband and inertial measurements,* IEEE Transactions on Vehicular Technology **64**, 1293 (2015).

[6] D. A. Allan and J. B. Rawlings, *Moving horizon estimation,* in *Handbook of Model Predictive Control*, edited by S. V. Raković and W. S. Levine (Springer Nature, Cham, Switzerland, 2019).

[7] D. G. Robertson, J. H. Lee, and J. B. Rawlings, *A Moving Horizon-Based Approach for Least-Squares Estimation,* AIChE Journal **42**, 2209 (1996).

[8] M. A. Fischler and R. C. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,* Commun. ACM **24**, 381–395 (1981).

[9] R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems,* Journal of Basic Engineering **82**, 35 (1960).

[10] A. H. Jazwinski, *Limited Memory Optimal Filtering,* IEEE Transactions on Automatic Control **13**, 558 (1968).

[11] Y. S. Shmaliy, S. Zhao, and C. K. Ahn, *Unbiased finite impluse response filtering: An iterative alternative to kalman filtering ignoring noise and initial conditions,* IEEE Control Systems Magazine **37**, 70 (2017).

[12] V. M. Zavala, C. D. Laird, and L. T. Biegler, *A fast moving horizon estimation algorithm based on nonlinear programming sensitivity,* Journal of Process Control **18**, 876 (2008).

[13] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock, *A real-time algorithm for moving horizon state and parameter estimation,* Computers and Chemical Engineering **35**, 71 (2011).

[14] A. Alessandri and M. Gaggero, *Fast moving horizon state estimation for discrete-time systems using single and multi iteration descent methods,* IEEE Transactions on Automatic Control **62**, 4499 (2017).

[15] F. Girrbach, J. D. Hol, G. Bellusci, and M. Diehl, *Optimization-Based Sensor Fusion of GNSS and IMU Using a Moving Horizon Approach,* Sensors (Basel, Switzerland) **17**, 1 (2017).

[16] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. H. E. de Croon, *Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone,* Journal of Field Robotics **37**, 667 (2020).

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

[18] Y. Xu, C. K. Ahn, Y. S. Shmaliy, X. Chen, and Y. Li, *Adaptive robust INS/UWB-integrated human tracking using UFIR filter bank,* Measurement: Journal of the International Measurement Confederation **123**, 1 (2018).

[19] Y. Xu, Y. S. Shmaliy, T. Shen, D. Chen, M. Sun, and Y. Zhuang, *INS / UWB-Based Quadrotor Localization Under Colored Measurement Noise,* IEEE Sensors Journal **21**, 6384 (2021).

[20] A. Liu, W. A. Zhang, M. Z. Chen, and L. Yu, *Moving Horizon Estimation for Mobile Robots with Multirate Sampling,* IEEE Transactions on Industrial Electronics **64**, 1457 (2017).

[21] R. Mahony, T. Hamel, and J. Pflimlin, *Nonlinear complementary filters on the special orthogonal group,* IEEE Transactions on Automatic Control **53**, 1203 (2008).

[22] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, *Averaging quaternions,* Journal of Guidance, Control, and Dynamics **30**, 1193 (2007).

3

Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones

Moving Horizon Estimation (MHE) offers multiple advantages over Kalman Filters when it comes to the localization of drones. However, due to the high computational cost, they can not be used on Micro Air Vehicles (MAVs) with limited computational power. In Chapter 2 we have shown, that with a few assumptions and simplifications, MHE can be made more efficient while retaining good localization performance. In this chapter, we present two additional improvements: the introduction of dynamic step sizes to the gradient descent algorithm, which leads to a significant increase in robustness, and the use of switching variables for outlier rejection, which further reduces the computational load. Both improvements are implemented and assessed in simulation and experiments. Using dynamic step sizes makes it possible to reliably use the estimator on board of a real drone, and the use of Newton's method specifically opens the option to add different types of measurements. The new outlier rejection method on the other hand is shown to reduce the computational load significantly while having no big impact on accuracy.

Parts of this chapter have been published at the International Micro Air Vehicles Conference 2023.

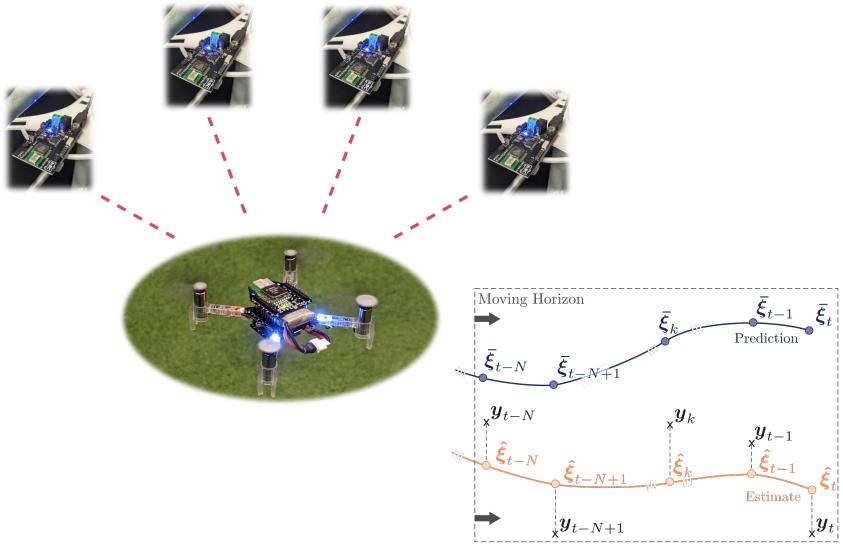


Figure 3.1: Moving Horizon Estimators (MHEs) usually requires a lot of computational power. Improvements to our previous work on computationally efficient MHEs allow for on-board execution of the algorithm to localize a nano-copter using Ultra-Wideband measurements.

3.1. INTRODUCTION

Self-localization is an important task for autonomous drones, especially when operating in indoor environments. Since navigation using Global Navigation Satellite Systems (GNSS) exhibits a significant decrease in performance when used inside of buildings, an alternative solution is needed. Ultra-Wideband (UWB) technology has been investigated as a promising alternative in scenarios where the environment can be prepared in advance. UWB localization works in a similar fashion as GNSS, but instead of ranging with respect to satellites, it uses fixed beacons at known locations in the environment [1].

The UWB range measurements can be used for triangulation [2], but are more commonly fused with measurements from an inertial measurement unit (IMU) for more accurate results [3]. Kalman filter variants such as the Extended Kalman Filter (EKF) are often used due to their speed and efficiency. However, they are not ideal when working with highly non-linear dynamics or non-Gaussian noise, both of which are present in the system in question.

Moving Horizon Estimators (MHEs) are well suited for state estimation in non-linear systems with non-Gaussian noise, but are computationally expensive, which makes them impractical to use on small drones with limited computational power. In Chapter 2, we have investigated several simplifications that reduce the load for localization of small drones using an MHE. While the resulting estimator performed well in offline evaluations on real measurement data, its robustness was found to be lacking. Specifically in

situations where the number of incoming measurements varies a lot, the tuning had to be quite conservative to avoid unstable behaviour when many measurements were coming in. Furthermore, while we were happy with the performance of Random Sample Consensus (RANSAC) for outlier rejection, its computational cost was quite high for an algorithm aimed at computationally extremely limited systems, like the microcontrollers on board of nano-drones.

In this chapter we will present the following two improvements to efficient moving horizon estimation for localization with UWB:

- Using a dynamic step size in the gradient-descent step to improve robustness
- Outlier rejection based on switching variables to further improve the computational efficiency

3.2. PRELIMINARIES

3.2.1. COMPUTATIONALLY EFFICIENT MHE

Let us quickly review the computationally efficient MHE presented in Chapter 2 to highlight some of its shortfalls. We again consider the discrete-time non-linear dynamic system given by (3.1), with prediction equation f and measurement equation h, expressed in terms of the system state $\boldsymbol{\xi}_t$, input vector $\boldsymbol{u}_t$, measurement vector $\boldsymbol{y}_t$, process noise $\boldsymbol{w}_t$ and measurement noise $\boldsymbol{v}_t$, all at timestep t respectively.

$$\begin{cases} \boldsymbol{\xi}_{t+1} &= \boldsymbol{f}(\boldsymbol{\xi}_t, \boldsymbol{u}_t) + \boldsymbol{w}_t \\ \boldsymbol{y}_t &= \boldsymbol{h}(\boldsymbol{\xi}_t, \boldsymbol{u}_t) + \boldsymbol{v}_t \end{cases} \tag{3.1}$$

The full MHE problem for a horizon of length $N+1$ is given by the optimization problem (3.2), formulated in terms of the decision variables $\boldsymbol{\xi}$, $\boldsymbol{w}$ and $\boldsymbol{v}$. Put into words, the MHE tries to minimize the squared observed process and measurement noise, as well as changes to the prior estimate (also called the arrival cost). The prior $\bar{\boldsymbol{\xi}}_{t-N}$ is calculated from the previous estimate using the prediction equation, $\bar{\boldsymbol{\xi}}_{t-N} = \boldsymbol{f}(\hat{\boldsymbol{\xi}}_{t-N-1})$.

$$\begin{aligned} \min_{\boldsymbol{\xi},\boldsymbol{w},\boldsymbol{v}} \quad & \mu \|\boldsymbol{\xi}_{t-N} - \bar{\boldsymbol{\xi}}_{t-N}\|^2 + \sum_{k=t-N}^{t-1} \boldsymbol{w}_k^2 + \sum_{k=t-N}^{t} \boldsymbol{v}_k^2 \\ \text{s.t.} \quad & \boldsymbol{\xi}_{k+1} = \boldsymbol{f}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \\ & \boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \end{aligned} \tag{3.2}$$

By assuming negligible process noise ($\boldsymbol{w} = \boldsymbol{0}$), the problem can be reduced to an output error MHE. Furthermore, by defining the composite prediction and measurement equations $\mathscr{F}_k(\boldsymbol{\xi}_{t-N})$ and $\mathscr{H}_{i,k}(\boldsymbol{\xi}_{t-N})$, we can merge the constraints into the cost-function J, which now only depends on the decision variable $\boldsymbol{\xi}_{t-N}$. Note that the number of measurements m_t can differ between timesteps.

$$\mathscr{F}_k(\boldsymbol{\xi}_{t-N}) = \boldsymbol{f}^{u_k} \circ \ldots \circ \boldsymbol{f}^{u_{t-N}}(\boldsymbol{\xi}_{t-N}) \tag{3.3}$$

$$\mathscr{H}_{i,k}(\boldsymbol{\xi}_{t-N}) = h_i^{u_k} \circ \mathscr{F}_{k-1}(\boldsymbol{\xi}_{t-N}) \tag{3.4}$$

$$J(\boldsymbol{\xi}_{t-N}) = \mu\|\boldsymbol{\xi}_{t-N} - \bar{\boldsymbol{\xi}}_{t-N}\|^2 + \sum_{k=t-N}^{t}\sum_{i=1}^{m_t}\left(\|y_{i,k} - \mathscr{H}_{i,k}(\boldsymbol{\xi}_{t-N})\|^2\right) \tag{3.5}$$

To save computation time, instead of fully solving the optimization problem at every timestep, only a single gradient descent step is performed as suggested by [4].

$$\hat{\boldsymbol{\xi}}_{t-N} = \bar{\boldsymbol{\xi}}_{t-N} - \alpha\nabla J(\bar{\boldsymbol{\xi}}_{t-N}) \tag{3.6}$$

Using the chain rule, the gradient of the cost function is easily calculated analytically. When using only a single iteration, the gradient is calculated with the prior best estimate $\boldsymbol{\xi}_{t-N} = \bar{\boldsymbol{\xi}}_{t-N}$, thus the arrival cost term is zero.

$$\nabla J(\bar{\boldsymbol{\xi}}_{t-N}) = -2\sum_{k=t-N}^{t}\sum_{i=1}^{m_t}\left[\left(y_{i,k} - \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N})\right)\nabla\mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N})\right] \tag{3.7}$$

To save computational power, the gradient of the composite measurement equation can be simplified by iterative computation of the Jacobian of the composite prediction equation (chain rule):

$$\nabla\mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N}) = \left(\mathrm{D}\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N})\right)^T \cdot \nabla h_i(\bar{\boldsymbol{\xi}}_k, \boldsymbol{u}_k) \tag{3.8}$$

$$\mathrm{D}\mathscr{F}_{k-1}(\bar{\boldsymbol{\xi}}_{t-N}) = \mathrm{D}\boldsymbol{f}(\bar{\boldsymbol{\xi}}_{k-1}, \boldsymbol{u}_{k-1}) \cdot \mathrm{D}\mathscr{F}_{k-2}(\bar{\boldsymbol{\xi}}_{t-N}) \tag{3.9}$$

For outlier rejection, a RANSAC scheme was used, which can identify outliers very well but requires a significant amount of computational power.

3.2.2. DRONE AND MEASUREMENT MODEL

We use the drone model in [3], but represent the attitude in quaternions and simplify the drag forces. For the use in our estimator, we discretize the model using the forward Euler method.

$$\dot{\boldsymbol{x}} = \left[\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\rho} \end{bmatrix} \otimes \boldsymbol{q}^{-1}\right]_v \tag{3.10}$$

$$\dot{\boldsymbol{\rho}} = \frac{f}{m}\boldsymbol{e_3} + (\mathbf{K}_a - [[\boldsymbol{\omega}_\times]])\,\boldsymbol{\rho} - \left[\boldsymbol{q}^{-1} \otimes \begin{bmatrix} 0 \\ \boldsymbol{g} \end{bmatrix} \otimes \boldsymbol{q}\right]_v \tag{3.11}$$

The six dimensional state vector $\boldsymbol{\xi} = [\boldsymbol{x}^T, \boldsymbol{\rho}^T]^T$ contains the position $\boldsymbol{x} \in \mathbb{R}^3$ in the global frame and the drone's velocities $\boldsymbol{\rho} \in \mathbb{R}^3$ in the body frame. The system inputs are the total thrust f, the attitude quaternion $\boldsymbol{q} \in \mathbb{R}^4$, and the angular velocities measured by the IMU, $\boldsymbol{\omega} \in \mathbb{R}^3$. m is the drone's mass and $\mathbf{K}_\mathrm{a} = \mathrm{diag}(k_\perp, k_\perp, k_\parallel)$ are the linear drag coefficients. The quaternion product is noted as $\otimes$, while the subscript 0 and v refer to the scalar and vector portion of the quaternion respectively. Finally, $\boldsymbol{g}$ is the acceleration due to gravity and $[[\boldsymbol{\omega}_\times]]$ is the skew-symmetric matrix defined as

$$[[\boldsymbol{\omega}_\times]] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.12}$$

We consider two different types of UWB measurements. Two-way ranging (TWR) can be used to determine the distance between the drone and a fixed beacon located at $\boldsymbol{p}_A$. Time-difference of Arrival (TDOA) measurements on the other hand represent the difference in the distance of the drone to two fixed beacons at positions $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$. The measurement noise is represented by η_{twr} and η_{TDOA} respectively.

$$y_{\text{twr}} = \|\boldsymbol{p}_A - \boldsymbol{x}\| + \eta_{\text{twr}} \tag{3.13}$$

$$y_{\text{TDOA}} = \|\boldsymbol{p}_A - \boldsymbol{x}\| - \|\boldsymbol{p}_B - \boldsymbol{x}\| + \eta_{\text{TDOA}} \tag{3.14}$$

3.3. DYNAMIC STEP SIZE

A serious problem with the approach in Chapter 2 is the fixed step size used in the gradient descent algorithm. Specifically in situations where the numbers of measurements m_t varies between steps, the fact that the gradient of the cost function is the sum of m_t individual measurement terms can lead to large variations in the gradient's magnitude. As a result, a value for α that is optimal with 4 UWB beacons, might cause the estimator to become unstable for 8 beacons. A solution to this problem is to dynamically adjust the step size of the gradient descent algorithm. We investigated two options for choosing the step size dynamically: linesearch and Newton's method. While robustness is of course an important property of any state estimation algorithm, it should be noted that we are sacrificing computational efficiency to achieve it.

3.3.1. LINESEARCH

With the linesearch method, different values for the step size α are tested at each time step. The step size resulting in the estimate with the lowest cost is then retained for the next step. The algorithm is laid out in Algorithm 2.

Algorithm 2: Linesearch Method

$t \leftarrow 0 \quad \boldsymbol{\xi} \leftarrow \boldsymbol{\xi}_0 \quad \alpha \leftarrow \alpha_0$
$s \leftarrow \{0.5, 0.8, 1.0, 1.25, 2.0\}$
while *true* **do**
  $t \leftarrow t+1$
  calculate $\nabla J(\boldsymbol{\xi}_{t-N})$
  for $s_i \in s$ **do**
    $\boldsymbol{\xi}_i = \boldsymbol{\xi}_{t-N} - s_i \cdot \alpha \nabla J(\boldsymbol{\xi}_{t-N})$
    calculate $J_i(\boldsymbol{\xi}_i)$
  $i_{\text{opt}} \leftarrow \text{argmin}_i(J_i)$
  $\boldsymbol{\xi}_{t-N} \leftarrow \boldsymbol{\xi}_{i_{\text{opt}}}$
  $\alpha \leftarrow s_{i_{\text{opt}}} \cdot \alpha$

The linesearch method is easy to implement in the existing algorithm, but has the downside of scaling the step size for all state variables at once. If different types of measurements are available, this can lead to problems. As an example, if a lot of velocity

estimates are available but only few position estimates, it would be good to choose different step sizes for the different states. An approach that allows for this flexibility is the use of Newton's method.

3.3.2. NEWTON'S METHOD

Newton's method is more complex to implement than linesearch, but offers two additional advantages. First, there is no need to calculate the cost of multiple estimates (which requires several predictions through the complete horizon) and second, Newton's method allows to adjust the step size along different axes. Specifically, instead of using a scalar step size α, the gradient of the cost function is multiplied by the inverse Hessian matrix of the cost function:

$$\hat{\bar{\boldsymbol{\xi}}}_{t-N} = \bar{\boldsymbol{\xi}}_{t-N} - \left(\mathrm{D}^2 J(\bar{\boldsymbol{\xi}}_{t-N})\right)^{-1} \nabla J(\bar{\boldsymbol{\xi}}_{t-N}) \tag{3.15}$$

The Hessian matrix of the cost function can be obtained by calculating the Jacobian of the cost function gradient.

$$\begin{aligned} \mathrm{D}^2 J(\bar{\boldsymbol{\xi}}_{t-N}) = 2\mu I - 2 \sum_{k=t-N}^{t} \sum_{i=0}^{m_t} \big(& y_{i,k} \, \mathrm{D}^2 \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N}) \\ & - \nabla \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N}) \nabla \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N})^T \\ & - \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N}) \, \mathrm{D}^2 \mathscr{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N}) \big) \end{aligned} \tag{3.16}$$

As with the gradients, the Hessian matrices of the cost and measurement functions can be calculated algebraically in an iterative manner while stepping through the horizon. Making use of the chain rule, we find the following expression [5]:

$$\begin{aligned} \mathrm{D}^2 \mathscr{H}_{i,k}(\boldsymbol{\xi}) = & \left(\mathrm{D}\mathscr{F}_{k-1}(\boldsymbol{\xi})\right)^T \cdot \mathrm{D}^2 h\left(\mathscr{F}_{k-1}(\boldsymbol{\xi})\right) \cdot \mathrm{D}\mathscr{F}_{k-1}(\boldsymbol{\xi}) \\ & + \sum_{j=1}^{n} \frac{\partial h}{\partial \xi_j} \left(\mathscr{F}_{k-1}(\boldsymbol{\xi})\right) \cdot \mathrm{D}^2 \mathscr{F}^j_{k-1}(\boldsymbol{\xi}) \end{aligned} \tag{3.17}$$

Luckily, many of the terms appearing in this equation are already needed when calculating the cost function gradient. Furthermore, since the prediction model we use is linear in terms of the states, the Hessian matrix of the composite prediction equations turns out to be zero. The simplified expression for the Hessian matrix of the cost function thus only depends on the Hessian matrix of the measurement function and the Jacobian matrix of the composite prediction equation:

$$\mathrm{D}^2 \mathscr{H}_{i,k}(\boldsymbol{\xi}) = \left(\mathrm{D}\mathscr{F}_{k-1}(\boldsymbol{\xi})\right)^T \cdot \mathrm{D}^2 h(\mathscr{F}_{k-1}(\boldsymbol{\xi})) \cdot \mathrm{D}\mathscr{F}_{k-1}(\boldsymbol{\xi}) \tag{3.18}$$

3.4. SWITCHING VARIABLE OUTLIER REJECTION

Since using RANSAC for outlier rejection has the disadvantage of being computationally heavy, we decided to look for a more efficient method. The need for outlier rejection in optimization problems also arises in Simultaneous Localization and Mapping (SLAM),

where incorrectly identified loop-closures can cause large errors in the resulting map. A possible approach to deal with this issue is the addition of switching variables that multiply the constraints in question [6]. Applying this concept to our MHE, we introduce for each measurement $y_{i,k}$ an associated switching variable $s_{i,k}$. By giving the optimization algorithm the ability to change these switching variables, the algorithm can enable and disable measurements based on how they impact the value of the cost function.

To use the switching variables as decision variables, they must be continuous. At the same time, for their function to enable and disable measurements, binary values would be more useful. Following [6], we use a sigmoid function to convert the continuous switching variables into values between 0 and 1, which then multiply the measurement equation. A simple sigmoid function to use is the logistic function (3.19).

$$\text{sig}(s_{i,k}) = \frac{1}{1+e^{-s_{i,k}}} \tag{3.19}$$

In the context of calculating gradients, it is also helpful to note the derivative of the logistic function (3.20).

$$\frac{\partial}{\partial s_{i,k}} \text{sig}(s_{i,k}) = \text{sig}(s_{i,k}) \left(1 - \text{sig}(s_{i,k})\right) \tag{3.20}$$

It is important to penalize the estimator for ignoring a measurement. We do this by adding a term similar to the arrival cost, that penalizes switching variables that deviate from a prior value $\bar{s}$, which enables the measurement (e.g. $\bar{s} = 5$). To simplify notation, we collect all switching variables in a single column vector $\boldsymbol{s} = [s_{i,j}]$. The resulting cost function including switching variables then looks like this:

$$\begin{aligned} J(\boldsymbol{\xi}_{t-N}, \boldsymbol{s}) = \mu \| \boldsymbol{\xi}_{t-N} - \bar{\boldsymbol{\xi}}_{t-N} \|^2 \\ + \mu_s \cdot \sum_{k=t-N}^{t} \sum_{i=1}^{m} \left(s_{i,k} - \bar{s}\right)^2 \\ + \sum_{k=t-N}^{t} \sum_{i=1}^{m} \left(\text{sig}(s_{i,k}) \cdot \| y_{i,k} - \mathcal{H}_{i,k}(\boldsymbol{\xi}_{t-N}) \|^2\right) \end{aligned} \tag{3.21}$$

When calculating the gradient of the cost function, we now also need to consider the derivative with respect to the individual switching variables. Luckily, each switching variable only appears once in each of the two sums and the corresponding derivative is easily calculated:

$$\frac{\partial J}{\partial s_{i,k}}(\boldsymbol{\xi}_{t-N}, \boldsymbol{s}) = 2\mu_s \left(s_{i,k} - \bar{s}\right) + \text{sig}(s_{i,k}) \left(1 - \text{sig}(s_{i,k})\right) \cdot \| y_{i,k} - \mathcal{H}_{i,k}(\boldsymbol{\xi}_{t-N}) \|^2 \tag{3.22}$$

The changes to the derivatives with respect to $\boldsymbol{\xi}_{t-N}$ are also small, one only has to add the sigmoid multiplication to each of the terms of the sums:

$$\frac{\partial J}{\partial \boldsymbol{\xi}_{t-N}}(\boldsymbol{\xi}_{t-N}, \boldsymbol{s}) = -2 \sum_{k=t-N}^{t} \sum_{i=1}^{m} \left[\text{sig}(s_{i,k}) \left(y_{i,k} - \mathcal{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N})\right) \nabla \mathcal{H}_{i,k}(\bar{\boldsymbol{\xi}}_{t-N})\right] \tag{3.23}$$

The full gradient could now be formed from the partial derivatives with respect to the state and the individual switching variables, but since the switching variables are decoupled from the rest of the system, it makes sense to update them individually using Newton's method:

$$\hat{s}_{i,k} = \hat{s}_{i,k-1} - \left(\frac{\partial^2}{\partial s_{i,k}^2} J(\bar{\hat{\boldsymbol{\xi}}}_{t-N}, \hat{\boldsymbol{s}}) \right)^{-1} \frac{\partial}{\partial s_{i,k}} J(\bar{\hat{\boldsymbol{\xi}}}_{t-N}, \hat{\boldsymbol{s}}) \tag{3.24}$$

For large outliers, even with the use of a switching penalty, the switching variable can take very large negative values, resulting in a variable overflow for the intermediate result of $e^{-s_{i,k}}$. To avoid this issue, it makes sense to place a lower limit on the value of $s_{i,k}$.

There is one final consideration regarding switching outlier rejection, which is the initial value of the switching variable. For simplicity, we decided to use the Mahalanobis distance based on only the measurement noise (the MHE doesn't estimate state covariance) of the associated measurement.

3.5. OFFLINE EVALUATION

The presented improvements were evaluated offline, using real UWB measurements gathered in the "Cyberzoo" test arena at TU Delft [7]. We simulated the performance on 12 different flights for TWR and 9 different flights for TDOA. The simulations were performed 10 times with different seeds for the random number generator, for a total of 120 simulations for TWR and 90 simulations for TDOA. For every simulation, all estimators were executed in parallel and saw therefore the same noise. For comparison, we also simulated the initial version of the MHE using the simple gradient (SG) method with a fixed step size and RANSAC for outlier rejection from Chapter 2, as well as the EKF from [3].

We compared the different estimators based on three metrics, which are presented in Figure 3.2 for TWR and in Figure 3.3 for TDOA.

- **Success rate:** We defined a run to be successful if the root-mean-square error (RMSE) on the position stays below 0.5 m. This definition is somewhat arbitrary but is meant to reflect a value at which the estimator's accuracy is good enough to allow navigation in spacious indoor environments.

- **Accuracy:** The accuracy of the estimators is assessed using the position RMSE. Unsuccessful runs were removed from the data prior to plotting the accuracy statistics, since unsuccessful runs for only a few beacons can easily reach errors of tens of meters and therefore have a tendency to distort the plots.

- **Normalized computation time:** To assess the computational efficiency, we calculated the average computation time per step. For every run, we normalize the average by dividing by the average computation time of the EKF. This was done to reduce the effect of the hardware used and the system load from other sources.

At first glance we can see, that TWR yields more stable results. The success rate of all estimators is near identical and reaches 100% at 5 or 6 beacons. Looking at the accuracy,

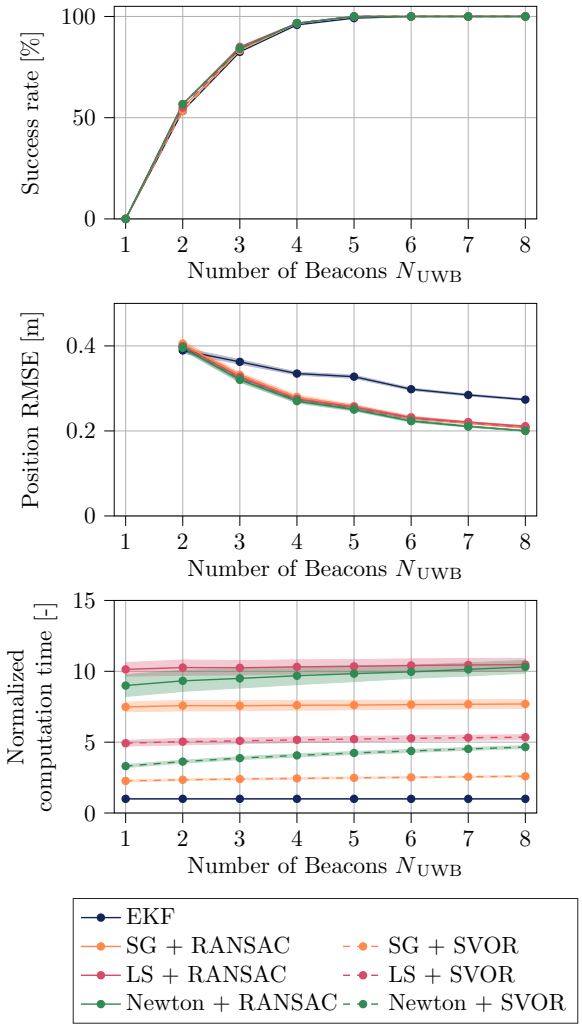


Figure 3.2: Performance of MHE configurations in simulation using TWR measurements: Simple gradient (SG), linesearch (LS) and Newton's method to solve the minimization problem, random sample consensus (RANSAC) and switching variable outlier rejection (SVOR) for outlier rejection. Shaded regions represent the standard error of the mean.

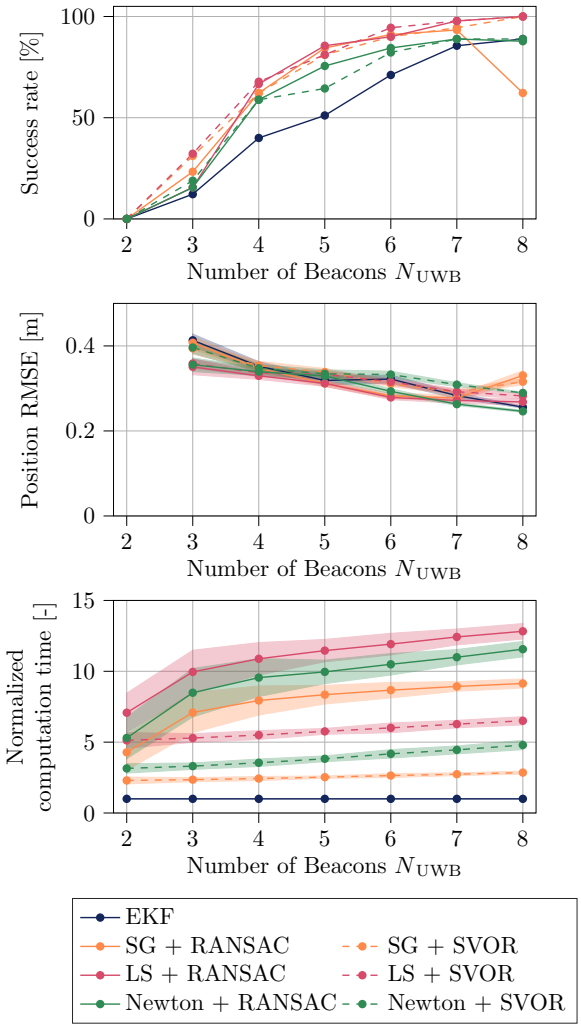


Figure 3.3: Performance of MHE configurations in simulation using TDOA measurements: Simple gradient (SG), linesearch (LS) and Newton's method to solve the minimization problem, random sample consensus (RANSAC) and switching variable outlier rejection (SVOR) for outlier rejection. Shaded regions represent the standard error of the mean.

we can see that all MHE versions outperform the EKF slightly, but no difference can be seen between the different versions.

The TDOA results are a bit messier. The success rate is quite different between different versions and for many estimators never reaches 100%. It is therefore more interesting to compare the quality of the estimates based on the success rate, rather than based on the accuracy of the successful runs. All MHE variants surpass the EKF in terms of success rate up to 7 beacons. The effect of a constant step size for varying numbers of measurements can be seen in the decreased performance of the simple gradient MHE when the number of beacons increases to 8. Interestingly, using switching variables for outlier rejection (SVOR) seems to be another suitable method for reducing this problem, as seen in the higher success rate. This is likely due to the fact, that SVOR can turn off measurements to keep the magnitude of the gradient closer to constant. This does however mean, that the estimator does not profit from the increased number of measurements, which becomes apparent when looking at the accuracy plots. The most successful approach for dealing with variable gradient magnitudes seems to be the linesearch (LS) approach. We suspect however, that in scenarios that incorporate other measurements (e.g. velocity from optic flow), using Newton's method will be advantageous.

Let's finally have a look at the computational efficiency. The EKF clearly still outperforms all versions of the MHE, but as expected, SVOR clearly reduces the computation time compared to RANSAC. Looking at the methods for choosing dynamic step sizes, Newton's method is more efficient than Linesearch, but still adds some additional cost compared to the simple gradient method.

3.6. ON-BOARD EXPERIMENTS

To verify our simulation results and demonstrate that the MHE is indeed capable of running on computationally limited devices, we implemented the code on a Crazyflie 2.1. The Crazyflie 2.1. is a small (38 g), versatile platform that can be equipped with a variety of expansion boards, such as the UWB Deck for UWB ranging and the Flow Deck, for altitude and flow measurements. To make full use of the Flow Deck and challenge the versatility of our estimator, we not only used the time-of-flight measurements for altitude, but also incorporated the measurement equations for flow and adjusted the MHE formulation to enable the use of measurements with different units (the flow measurements are in pixels per second). Specifically, we use Newton's method with SVOR and introduce the inverse covariance matrix of the prior, $\boldsymbol{P}^{-1}$, and the inverse measurement variance, σ^2 as weights in the cost function [8]:

$$\begin{aligned} J(\boldsymbol{\xi}_{t-N}, \boldsymbol{s}) = \mu \left(\boldsymbol{\xi}_{t-N} - \bar{\boldsymbol{\xi}}_{t-N}\right)^T \boldsymbol{P}^{-1} \left(\boldsymbol{\xi}_{t-N} - \bar{\boldsymbol{\xi}}_{t-N}\right) \\ + \mu_s \cdot \sum_{k=t-N}^{t} \sum_{i=1}^{m} \left(s_{i,k} - \bar{s}\right)^2 \\ + \sum_{k=t-N}^{t} \sum_{i=1}^{m} \left(\mathrm{sig}(s_{i,k}) \cdot \frac{\| y_{i,k} - \mathscr{H}_{i,k}(\boldsymbol{\xi}_{t-N}) \|^2}{\sigma^2} \right) \end{aligned} \tag{3.25}$$

This way, the units of the measurement errors are removed and the measurements are immediately weighted according to their reliability. Each estimator was flown at least

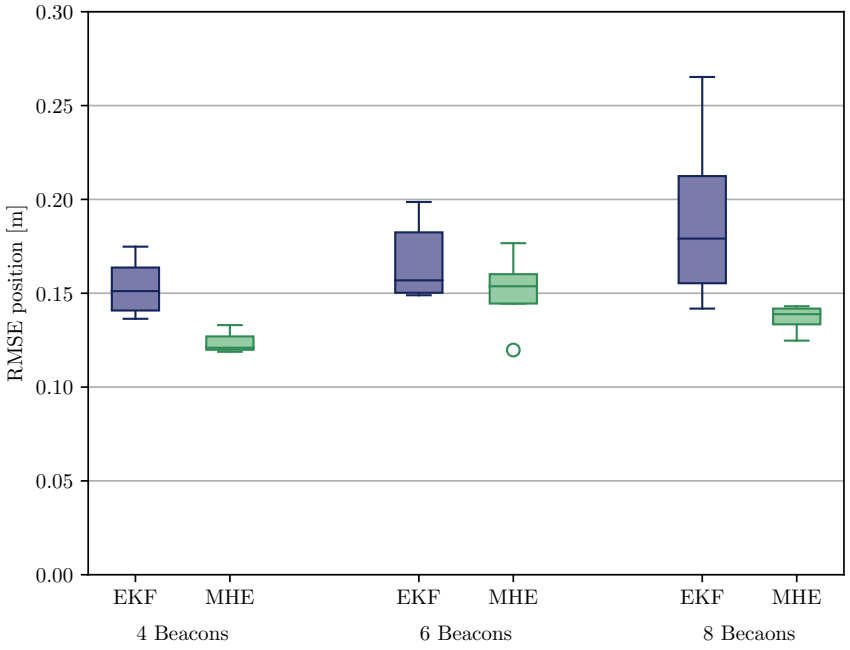


Figure 3.4: Comparison of MHE and EKF using Two-Way Ranging (5 flights per setting)

five times for a given number of beacons (4, 6 and 8) with the UWB ranging set to TWR. The results in Figure 3.4 show, that the MHE is indeed able to compete with the EKF in terms of performance for the given scenarios. In terms of computational requirements, we limited the number of measurements the MHE keeps in memory to 80 and used a quality measure based on the switching variable to remove bad measurements before they fall out of the window. With these changes, the MHE runs on-board, using about 25% of the available computational resources, whereas the EKF operates with only about one-fifth of this number. This actually matches quite well with the observations from the offline evaluation.

3.7. CONCLUSION

Moving Horizon Estimation holds a lot of potential, even for computationally limited platforms. By simplifying the problem to an output error MHE and approximating the optimization problem with a single-step gradient descent algorithm, the computational requirements of MHE can be reduced significantly, albeit at the cost of a bit of accuracy. In the end, the MHE slightly outperforms the extended Kalman Filter, which is a popular option for any non-linear state estimation problem.

The use of dynamic step sizes in a single-step gradient descent algorithm helps alleviate problems that occur when the number of incoming measurements varies greatly. Both a linesearch approach and Newton's method are suitable, but Newton's method of-

fers more flexibility when dealing with different kinds of measurements.

To further reduce the computational cost, we employed outlier rejection based on switching variables, which offers advantages over RANSAC in terms of computation time.

The improved MHE shows promise in simulation and experiments, where it outperforms a classical EKF and previous iterations of our computationally efficient MHE. The main limitations of the approach are the more complex formulation and the still non-negligible increase in computational cost.

In its current formulation, the MHE only estimates position and velocity, while attitude estimates are provided by the complementary filter. This approach is suitable for pitch and roll, since the gravitational pull can be used as reference. The yaw estimates on the other hand are subject to drift and currently require proper initialization. This problem should be addressed in future work by including yaw (error) as an additional state in the model.

REFERENCES

[1] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa, *Ultra wideband indoor positioning technologies: Analysis and recent advances,* Sensors (Switzerland) **16**, 1 (2016).

[2] A. Norrdine, *An Algebraic Solution to the Multilateration Problem,* in *International Conference on Indoor Positioning and Indoor Navigation* (2012).

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

[4] A. Alessandri and M. Gaggero, *Fast moving horizon state estimation for discrete-time systems using single and multi iteration descent methods,* IEEE Transactions on Automatic Control **62**, 4499 (2017).

[5] M. Skorski, *Chain rules for hessian and higher derivatives made easy by tensor calculus,* (2019), arXiv:1911.13292 [cs.SC] .

[6] N. Sünderhauf and P. Protzel, *Towards a robust back-end for pose graph SLAM,* Proceedings - IEEE International Conference on Robotics and Automation , 1254 (2012).

[7] S. Pfeiffer, C. De Wagter, and G. C. H. E. de Croon, *Data underlying the publication: "A Computationally Efficient Moving Horizon Estimator for UWB Localization on Small Quadrotors",* (2021), 10.4121/14827680.v1.

[8] C. V. Rao, *Moving horizon strategies for the constrained monitoring and control of nonlinear discrete-time systems* (The University of Wisconsin-Madison, 2000).

4

RELATIVE LOCALIZATION AND SYNCHRONIZED MOVEMENT IN SMALL SWARMS

Relative localization is a key capability for autonomous robot swarms and it is a substantial challenge especially for small flying robots, as they are extremely restricted in terms of sensors and processing while other robots may be located anywhere around them in three-dimensional space. In this chapter, we generalize wireless-ranging-based relative localization to three dimensions. In particular, we show that robots can localize others in three dimensions by ranging to each other and only exchanging body velocities and yaw rates. We perform a nonlinear observability analysis, investigating the observability of relative locations for different cases. Furthermore, we show both in simulation and with real-world experiments that the proposed method can be used for successfully achieving various swarm behaviors. In order to demonstrate the method's generality, we demonstrate it both on tiny quadrotors and light-weight flapping wing robots.

Parts of this chapter have been published in Swarm Intelligence **17**, 1 (1906) [1].

4.1. INTRODUCTION

Following recent technological developments, autonomous micro-air vehicles (MAVs) have been getting increasing attention for a wide variety of tasks. These range from monitoring crops [2] and inspecting power-plants [3] to acting as communication relays in areas with damaged infrastructures [4]. For many of these tasks, using multiple collaborating agents has advantages in terms of performance, robustness and flexibility. These robotic swarms are inspired by self-organizing behaviour in animals and have the potential to complete complex tasks by exploiting cooperation among team members [5], [6]. It is important here to emphasize that we do not equate 'swarming' with having 1000s of robots in a tight space. For example, in exploration tasks, swarm members should spread out as much as possible over the area of interest, hence there may well be only small groups in the order of five or ten robots on a floor of a building.

Relative localization of other agents is a key component of MAVs operating in swarms [7]. It is required for a variety of tasks, from avoiding collisions between swarm members to enabling complex swarming behaviours such as flocking, and a multitude of options are available when it comes to sensors being used [8]. In outdoor areas and in adequately equipped indoor environments, agents can communicate absolute positioning information, for example from GNSS (Global Navigation Satellite System) [9], motion-tracking cameras [10] or beacon based systems [11][12]. In GNSS denied environments that cannot be prepared in advance, cameras [13][14], infrared sensors [15] and microphones [16] have been used for relative localization with varied success.

Based on knowledge of the distances between all swarm members, the morphology of a rigid swarm can be determined with the exception of rotation and translation. If the position of 4 nodes (in 3D) is known in a common reference frame, the network localization problem can be solved completely, yielding the position of each node in the same reference frame [17][18].

In our work, we are interested in building swarms based on small and lightweight agents. This is not only advantageous for monetary and logistic reasons, but it also makes the system inherently safe. Simply put, a tiny drone can't build enough momentum to cause serious damage to people, infrastructure or other assets in the operational area. The drones we are working with in this research weigh ≤ 100 g (including battery) and do most of their processing on a STM32F4 series microcontroller (192 kB memory, 168 MHz processor). As a result, most of the previously mentioned strategies are difficult to implement due to the weight of the sensors or the computational requirements of the processing. A relative localization approach based on odometry and ranging between agents, while being less accurate, can be implemented with lightweight and energy-efficient hardware such as Ultra-Wideband (UWB) communication chips.

Some relevant work in this direction has been carried out by Trawny in [19] and [20], where the authors aim at finding a solution for the 6-degree of freedom relative-pose estimation problem in 3D based on odometry and inter-agent ranging. In their work, they also include a nonlinear local weak observability analysis to find sufficient conditions for the 3D relative pose to become locally weakly observable. However, their theoretical approach relies on a large set of distance measurements (10), which results into a computationally heavy polynomial system of equations.

More recently, several authors have worked on performing relative localization more

efficiently, so that it can be used on smaller MAVs. For example, Coppola *et al.* implemented a relative localization algorithm for collision avoidance based on signal strength measurements with Bluetooth [21]. The use of received signal strength (RSSI) and Bluetooth does however impose significant limitations on the range and accuracy of the system. In [22], Guo *et al.* present a 2D localization system based on UWB that does not suffer from these limitations, but requires the agents to determine their displacement in a common frame of reference. Li *et al.*, on the other hand, use an Extended Kalman Filter (EKF) and UWB ranging to estimate relative position and heading in 2D, removing the need for a common orientation reference [23]. While the above-mentioned, real-world studies on flying robots have focused on 2D localization, where the robots measured and controlled their height to a flat ground surface, some recent efforts have been made to perform relative localization in 3D as well. Cossette *et al.* present a ranging-based 3D relative localization scheme, which does however require a common orientation reference [24]. The same goes for Shalaby *et al.*, which additionally makes use of multiple UWB tags per agent. Using a common orientation reference requires exteroceptive sensors which often require proper initialization and can be subject to external disturbances. Specifically, the magnetometers used in the aforementioned papers are affected when operating in proximity of hard-iron objects or electrical transmission lines. For higher robustness in GPS-denied, unprepared environments, the 3D relative localization without common orientation reference has been studied, using multiple consecutive distance measurements and computationally expensive optimization algorithms [25][26][27].

In this work, we perform wireless-ranging-based relative localization for drones in three dimensions, without the need for a common orientation reference or extensive computational resources. With Lie derivatives [28][29], we study the system observability and find favorable trajectories with high localization observability. Our contributions are thus:

- A formulation for a wireless-ranging-based 3D relative localization system which does not require a common orientation reference and can be solved efficiently on computationally limited devices.
- An observability analysis of the system
- Experimental validation and comparison of open- and closed-loop performance
- The first experimental results of relative localization on flapping-wing drones

This chapter will be structured as follows: In Sec. 4.2 we introduce the relative localization problem, the sensory information available to the drones, the continuous-time model and the EKF that performs the estimation. Sec. 4.3 presents the observability analysis carried out to evaluate the quality of the information provided by the filter and to detect conditions that may cause a degradation of the degree of observability of the system. The theoretical analysis from Sec. 4.3 is verified by means of simulation in Sec. 4.4 and by means of real-world robotic experiments in Sec. 4.5 and 4.6. Finally we draw conclusions in Sec. 4.7.

(a) Crazyflie (b) Flapper Drone

Figure 4.1: Illustration of the coordinate system conventions shown on the drones used in this chapter

4.2. RANGE-BASED RELATIVE LOCALIZATION IN 3D

4.2.1. NOTATION AND CONVENTIONS

Let us first introduce some notation and conventions. Throughout this chapter, we use right-handed coordinate systems with the x-axis pointing forward, the z-axis pointing up and the y-axis completing the right-handed system as shown in Figure 4.1. When expressing the orientation of the MAV in Tait-Bryan angles (roll ϕ, pitch θ, yaw ψ) we follow the sequence ZYX from global to body frame, leading to the following expression for the rotation matrix:

$$\boldsymbol{u}^G = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\boldsymbol{u}^B = \mathbf{R}_{gb}\boldsymbol{u}^B \tag{4.1}$$

$$\mathbf{R}_{gb} = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi - \mathrm{s}\psi\mathrm{c}\phi & \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\psi\mathrm{s}\phi \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi & \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{c}\psi\mathrm{s}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{4.2}$$

$\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_z(\psi)$ are the rotation matrices for the elemental rotations around the local x, y and z axis, respectively, and we use shorthands for sines and cosines ($\mathrm{s}x = \sin x, \mathrm{c}x = \cos x$).

Since there is no common inertial reference frame, each agent i must be able to determine the relative position of the other robots in a body centered frame. However, instead of working in the drone's body frame B_i, we mainly work in a drone-centered horizontal frame H_i. In this horizontal frame, the xy-plane is parallel to the ground and the z-axis points upwards. Since roll and pitch can be accurately estimated from IMU data thanks to the gravitational force [30], any vector $\boldsymbol{u}^B$ expressed in B_i can be expressed in H_i by multiplication with the following rotation matrix:

$$\mathbf{R}_{hb} = \begin{bmatrix} \mathrm{c}\theta & \mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\theta\mathrm{c}\phi \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{4.3}$$

4.2.2. SYSTEM MODEL

The 3D relative localization problem can be described as determining the 3D relative position of a tracked drone D_j in the body frame of a tracking drone D_i, $\{j | j \in \mathbb{N}, j \neq i\}$ where $\mathbb{N} = \{1, 2, ..., N\}$ and N is the number of robots in the flock within sight of D_i. We define the relative state vector for the tracking drone as $\mathbf{X}_{ij} = \left[x_{ij}, y_{ij}, z_{ij}, \psi_{ij}\right]^T$, where $\mathbf{p}_{ij} = \left[x_{ij}, y_{ij}, z_{ij}\right]^T$ is the relative position along the three axes of the tracking drone's horizontal frame H_i and ψ_{ij} is the relative yaw.

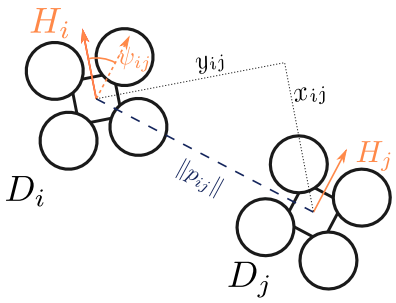


Figure 4.2: Relative localization problem, seen from above

The continuous-time kinematic model $f(\mathbf{X}_{ij})$ describing the relative motion of the drones in the three dimensions can be derived using the Newton formula [23]. The full model, including the measurement equation is then:

$$\dot{\mathbf{X}}_{ij} = f(\mathbf{X}_{ij}) = \begin{bmatrix} \mathbf{R}_z(\psi_{ij})\boldsymbol{v}_j^{H_j} - \boldsymbol{v}_i^{H_i} - \mathbf{S}\dot{\psi}_i^G \boldsymbol{p}_{ij} \\ \dot{\psi}_j^G - \dot{\psi}_i^G \end{bmatrix} \tag{4.4}$$

$$h(\mathbf{X}_{ij}) = \|\boldsymbol{p}_{ij}\| = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2} \tag{4.5}$$

$$\mathbf{S} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

The inputs to this model are the horizontal velocities $\boldsymbol{v}_i^{H_i}$ and $\boldsymbol{v}_j^{H_i}$ (expressed in H_i and H_j, respectively) as well as the yaw rates $\dot{\psi}_i^G$ and $\dot{\psi}_j^G$ in the global frame. Each drone can estimate its own horizontal velocity and yaw rate from on-board measurements and communicates them to the other drones by including them in the same UWB messages that are used for ranging. Furthermore, the ranging messages are used to calculate a measurement of the distance between the drones, d_{ij}. Estimation of the horizontal velocity can vary depending on the drone platform and available on-board sensors. Two examples will be presented in section 4.5.2. The flow of data on-board and between the drones is shown in Figure 4.3.

Note that the yaw rate $\dot{\psi}$ is defined around an intermediate axis of rotation and does therefore not directly correspond to the angular velocity around the z axis, ω_z, mea-

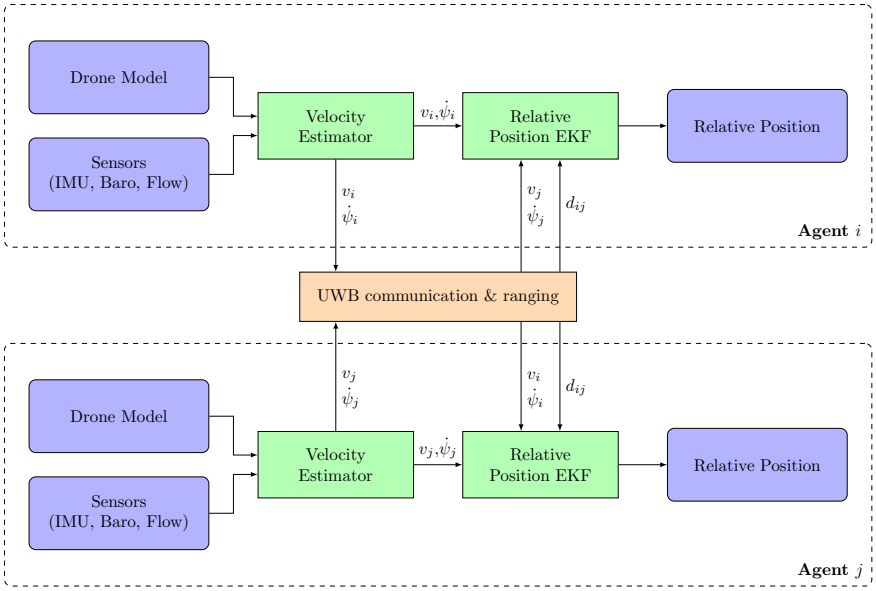


Figure 4.3: Data flow on-board and between drones

sured on-board the drone. Specifically, the relation between the angular rates around the body-frame axes, $\boldsymbol{\omega}$, and the Euler angle rates (ZYX sequence) are given by:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_x^{-1}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_x^{-1}(\phi)\mathbf{R}_y^{-1}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{4.7}$$

This relation can be expressed using a single matrix, which is easily inverted to find the expression for the Euler angle rates in terms of the gyro measurements:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\mathrm{s}\theta \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi\mathrm{c}\theta \\ 0 & -\mathrm{s}\phi & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \leftrightarrow \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\mathrm{s}\theta\mathrm{s}\phi}{\mathrm{c}\theta} & \frac{\mathrm{s}\theta\mathrm{c}\phi}{\mathrm{c}\theta} \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \frac{\mathrm{s}\phi}{\mathrm{c}\theta} & \frac{\mathrm{c}\phi}{\mathrm{c}\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.8}$$

4.2.3. EKF FOR RELATIVE LOCALIZATION

To estimate the relative state vector $\boldsymbol{X}_{ij}$ using the system model (4.4) and the measurement equation (4.5), an EKF is used [31]. For the prediction step, we discretize (4.4) using the forward Euler method and linearize the system for the covariance update. In what follows, T_s is the sampling time, $\mathbf{P}$ is the state covariance matrix and $\mathbf{Q}$ is the process noise covariance matrix. The input vector $\mathbf{U}$ is defined as $\mathbf{U} = [\boldsymbol{v}_i^T, \boldsymbol{v}_j^T, \dot{\psi}_i, \dot{\psi}_j]^T$. To simplify notation, we drop the indices ij for the state vector $\boldsymbol{X}$.

$$\begin{aligned} \hat{\mathbf{X}}_{k+1|k} &= F(\hat{\mathbf{X}}_k, \mathbf{U}_k) = \hat{\mathbf{X}}_k + \dot{\mathbf{X}}_k T_s \\ \mathbf{P}_{k+1|k} &= \mathbf{A}_k \mathbf{P}_{k|k} \mathbf{A}_k^T + \mathbf{B}_k \mathbf{Q} \mathbf{B}_k^T \end{aligned} \tag{4.9}$$

with

$$\mathbf{A} = \frac{\partial F}{\partial \mathbf{X}} = \begin{bmatrix} 1 & \dot{\psi}_i T_s & 0 & T_s(-\mathrm{s}\psi_{ij}\, v_j^x - \mathrm{c}\psi_{ij}\, v_j^y) \\ -\dot{\psi}_i T_s & 1 & 0 & T_s(\mathrm{c}\psi_{ij}\, v_j^x - \mathrm{s}\psi_{ij}\, v_j^y) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$\mathbf{B} = \frac{\partial F}{\partial \mathbf{U}} = \begin{bmatrix} -T_s & 0 & 0 & T_s\mathrm{c}\psi_{ij} & -T_s\mathrm{s}\psi_{ij} & 0 & T_s y_{ij} & 0 \\ 0 & -T_s & 0 & T_s\mathrm{s}\psi_{ij} & T_s\mathrm{c}\psi_{ij} & 0 & -T_s x_{ij} & 0 \\ 0 & 0 & -T_s & 0 & 0 & T_s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -T_s & T_s \end{bmatrix} \tag{4.11}$$

The UWB measurement of the distance between the drones is the only measurement considered for the correction step of the EKF. The Jacobian matrix of the corresponding measurement equation (4.5) is given by

$$\mathbf{H} = \frac{\partial h}{\partial \mathbf{X}} = \frac{\boldsymbol{p}_{ij}^T}{\|\boldsymbol{p}_{ij}\|} = \frac{1}{\sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2}} \cdot [x_{ij}, y_{ij}, z_{ij}] \tag{4.12}$$

Given a noisy distance measurement d_{ij} with variance σ_d^2, it is then possible to calculate the Kalman gain $\boldsymbol{K}$ and apply the measurement update to the state vector and the state covariance matrix. Note that since we work with a scalar measurement, no matrix inversion is necessary to compute the Kalman gain.

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \sigma_d^2\right)^{-1} \tag{4.13}$$

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_{k|k-1} + \mathbf{K}_k \left(d_{ij} - h\left(\hat{\mathbf{X}}_{k|k-1}\right)\right) \tag{4.14}$$

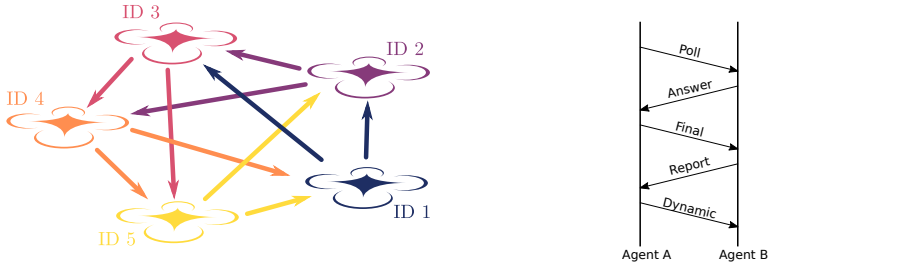
To use this approach with multiple drones, a filter bank of several EKFs that run in parallel can be used. This approach was demonstrated successfully for a swarm of five drones in 2D [23]. While this approach does not leverage additional information from distances between other drones, it is highly flexible and thus suited for dynamic swarms. Furthermore, this approach is computationally efficient as there is no need to track covariances between the relative states, preventing large matrix operations.

4.2.4. ULTRA-WIDEBAND COMMUNICATION PROTOCOL

In large part, the UWB communication protocol used is the same as described in [23].

On a high level, access to the communication channel is managed using a token ring protocol. Every agent has its own ID and ranges to a set of predetermined other agents. After ranging and exchanging information with those agents, control of the communication channel is passed on to the next agent, which then performs its own ranging and communication. By sharing ranging information with the agent that is being ranged to, each link only needs to be initiated from one side and the control can be passed around in a circle.

The lower level ranging protocol is an extension to the standard symmetrical double-sided two-way ranging (SDS-TWR). The "Report" message now also includes the reporting drone's velocity and yaw rate estimates. Furthermore, an additional "dynamic" message is returned by the agent that initiated the ranging which also transmits velocity estimates and yaw rate alongside the calculated range. Finally, the "Dynamic" message is also used to pass control of the communication channel to the next agent.



(a) Token-Ring Protocol

(b) SDS-TWR with additional "Dynamic" Message

Figure 4.4: Ultra-Wideband communication protocols

4.3. OBSERVABILITY ANALYSIS

It is important to be aware of the potential dangers resulting from rapidly degrading relative position estimates in a robotic swarm. This can result from a lack of observability of the system, that is, when the combination of inputs and measurements is not sufficient to infer the full state of the system. To identify these situations, we proceed with an observability analysis of the system.

Based on [32] and [33], we study the *local weak observability* of the system described by (4.4) and (4.5) in Lie derivatives, by computing the observability matrix $\mathbf{O}$. The system is termed observable if $\mathbf{O}$ is full rank. To simplify the expressions in this chapter, we consider the measurement equation (4.5) in power form, i.e.

$$h(\mathbf{X}) = \frac{\boldsymbol{p}_{ij}^T \boldsymbol{p}_{ij}}{2} = \frac{1}{2}\left(x_{ij}^2 + y_{ij}^2 + z_{ij}^2\right) \tag{4.15}$$

hence

$$\mathbf{O} = \begin{bmatrix} \nabla \mathscr{L}_f^0 h \\ \nabla \mathscr{L}_f^1 h \\ \nabla \mathscr{L}_f^2 h \\ \nabla \mathscr{L}_f^3 h \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{ij} & 0 \\ (\mathbf{R}_z \boldsymbol{v}_j - \boldsymbol{v}_i)^T & \mathbf{p}_{ij}^T \mathbf{R}_z \mathbf{S} \boldsymbol{v}_j \\ (\dot{\psi}_j \mathbf{R}_z \mathbf{S} \boldsymbol{v}_j - \dot{\psi}_i \mathbf{S} \boldsymbol{v}_i)^T & -2\boldsymbol{v}_i^T \mathbf{R}_z \mathbf{S} \boldsymbol{v}_j + \dot{\psi}_j \boldsymbol{p}_{ij}^T \mathbf{R}_z \mathbf{S}^2 \boldsymbol{v}_j \\ (\dot{\psi}_j^2 \mathbf{R}_z \mathbf{S}^2 \boldsymbol{v}_j - \dot{\psi}_i^2 \mathbf{S}^2 \boldsymbol{v}_i)^T & -3(\dot{\psi}_j - \dot{\psi}_i)\boldsymbol{v}_i^T \mathbf{R}_z \mathbf{S}^2 \boldsymbol{v}_j - \dot{\psi}_j^2 \boldsymbol{p}_{ij}^T \mathbf{R}_z \mathbf{S} \boldsymbol{v}_j \end{bmatrix} \tag{4.16}$$

Without delving into an exhaustive analysis of all the states that would cause matrix [4.16] to lose its rank, some unobservable conditions can be highlighted by observing its structure and computing its determinant $|\mathbf{O}|$. While more conditions exist, they involve more complicated combinations of states and inputs that are unlikely to occur for longer time spans anyway. Specifically, the following conditions will result in a full row or

column of zeros which in turn will result in $|\mathbf{O}| = 0$. This can easily be seen by performing a cofactor expansion following a particular row or column:

- **Identical velocities:** $\boldsymbol{v}_i = \mathbf{R}_z \boldsymbol{v}_j$
  This condition could potentially be problematic for missions that require leader-follower or flocking behaviour, where the drones are expected to move at identical velocities.

- **Same altitude and no relative motion along z:** $z_{ij} = 0$ and $v_i^z = v_j^z$
  Similar to the condition of identical velocities, this situation could occur in multiple scenarios.

- **Tracked drone not moving:** $\boldsymbol{v}_j = 0$
  This is a very relevant condition, as it means that the agents should be constantly moving in order to be able to determine their relative positions.

- **Identical positions:** $\boldsymbol{p}_{ij} = \mathbf{0}$
  While this condition will in theory cause the system to become unobservable, it is not relevant in practical applications, where it is impossible for two agents to be located in the same position in space.

It is also interesting to evaluate not only *if*, but *how well* the system is observable. To this end it is possible to employ the *local estimation condition number* C, defined as the ratio between the largest singular value $\sigma_{\max}$ and the smallest singular value $\sigma_{\min}$ of $\mathbf{O}$ ([34]). To avoid division by zero if $\mathbf{O}$ loses rank, we will actually consider the inverse of the local estimation condition number, C^{-1}:

$$C^{-1} = \frac{\sigma_{\min}(\mathbf{O})}{\sigma_{\max}(\mathbf{O})} \tag{4.17}$$

Favourable conditions on the system will result in a small C^{-1}, while an ill-conditioned estimation problem will entail a rather large C^{-1}. The inverse local estimation condition number can be used to understand how the addition of the third dimension affects the observability of the system. We do this by varying one parameter of matrix $\mathbf{O}$ at a time and comparing the values of C^{-1} when estimating relative position in x and y, as well as the relative yaw. Fig. 4.5 shows an example of the study. It can be noticed that the maximum value of C^{-1} is significantly lower for the 3D case compared to the 2D one, which means that adding the third dimension decreases the observability of the system in general. This can be explained by the higher ambiguity when determining the true states by relying on the same measurement: In 3D, any relative position on a sphere around the agent can result in a given measurement, while in 2D, this uncertainty is limited to a circle.

4.4. SIMULATION RESULTS

In this section we discuss results obtained by simulating the multi-agent system in Python to validate the relative localization method. The EKF performance is assessed in terms of quality of estimation and consistency. For comparison, we also implemented the

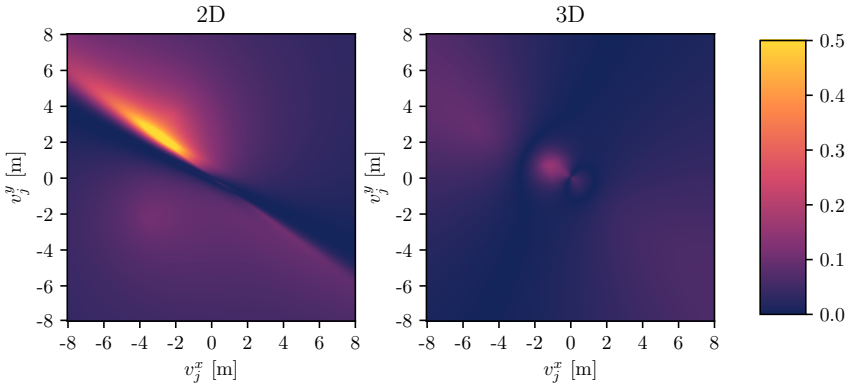


Figure 4.5: Effect of v_j on the observability index. Analysis parameters $\boldsymbol{p}_{ij} = [2,2,3]^T$ m, $\boldsymbol{v}_i = [2,2,5]^T$ m/s, $v_{ij}^z = 5$ m/s $\psi_{ij} = 160°$, $\dot{\psi}_i = 10°$/s and $\dot{\psi}_j = -20°$/s.

sliding-window estimator (SWE) presented in [24]. The parameters for the EKF are computed as follows:

$$\mathbf{Q} = \text{diag}([\sigma_v^2, \sigma_v^2, \sigma_v^2, \sigma_v^2, \sigma_v^2, \sigma_w^2, \sigma_w^2]) \tag{4.18}$$

$$\mathbf{R} = \sigma_r^2 \tag{4.19}$$

where the input noise deviation of velocity and yaw rate are respectively $\sigma_v = 0.15$ m/s and $\sigma_w = 0.1$ rad/s, the measurement noise deviation is $\sigma_r = 0.1$ m and the sampling time is $T_s = 0.01$ s, which is similar to the values observed in logs of the real systems.

The state vector $\mathbf{X}$ is initialized to a random initial relative position $(x_{ij}, y_{ij}, z_{ij}, \psi_{ij}$ in the vicinity of the true relative position. The covariance matrix $\mathbf{P}$, is set according to the random noise added to the initial position value. The horizontal velocity of the drones which serves as input to the EKF can stem from different sources, that can affect the consistency of the estimator. The most general approach is to use a velocity that is provided from an on-board estimator. While this approach has the advantage of being applicable to almost any drone imaginable, it results in a cascaded setup of filters, which is known to lead to problems regarding the estimator's consistency [35]. To avoid this problem, we compare the performance of the naive (simple) approach with the two variations. The Sigma-Point Covariance-Intersection (SPCI) variant uses covariance intersection with a sigma-point approach to avoid having to estimate cross-covariances [35][7]. Furthermore, a direct approach can be used on platforms where direct velocity measurements are available, e.g. in form of optic flow measurements.

For the first test, the robots are assigned a random trajectory. The test is repeated ten times with different seeds for the random number generator that is used to add noise to the inputs and measurements. Fig. 4.6 shows the estimated relative positions of both estimators compared to the groundtruth. Furthermore, we compare the absolute relative position error e_{ij} and the normalized estimation error squared $NEES_{ij}$ to assess the performance and consistency of the estimators, as well as the inverse of the local

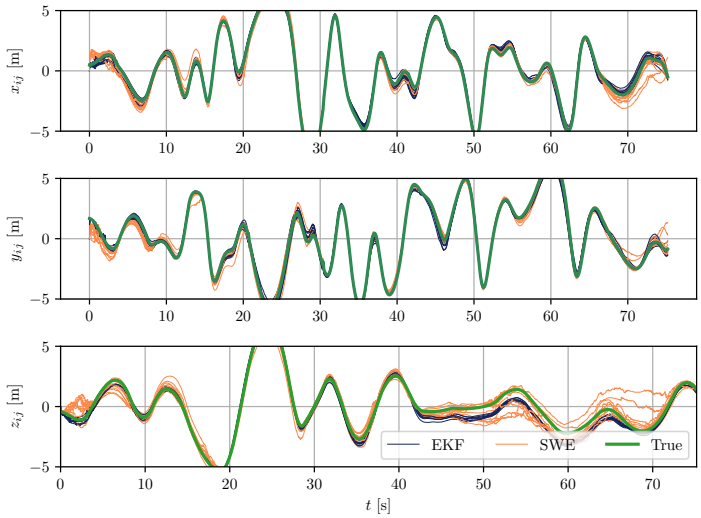


Figure 4.6: Simulated performance of simple EKF and SWE for the three states x_{ij}, y_{ij}, z_{ij} for 10 runs on a random trajectory.

estimation condition number C^{-1} (Figure 4.7). Note that only the error in position is considered to allow a comparison between the different estimators.

As can be seen, the EKF getting direct velocity measurements performs best, both in terms of error and consistency. The simple EKF using estimated velocities is highly inconsistent and does not provide the same quality of estimation. Using the SPCI approach improves the consistency in a cascaded setup, but does increase the error on this trajectory. We therefore conclude that where possible, direct velocities should be used.

It is interesting to verify the link between observability index and the filter performance. In the first test, we can already see that all filters perform better in the first half of the simulation, where the observability is better (larger values of C^{-1}. In a second test, the system is simulated with trajectories designed to maximize the observability of the system. The favourable trajectories are realized by making MAV i fly along a clockwise circular trajectory of radius ρ_i at an angular velocity ω at a height z_i, which oscillates in a sinusoidal way. MAV j, on the other hand, follows a circular path concentric to MAV i, counterclockwise, with a radius $\rho_j > \rho_i$, an angular velocity 2ω and a height $z_j \neq z_i$. With this setup, parallel velocities are avoided, as well as possible hits between the MAVs. Moreover, the difference in the velocity magnitude allows to excite the system with sufficient dynamics. The favorable trajectories are shown in Figure 4.8.

The results of the second test are shown in Fig. 4.9 and Fig. 4.10. As can be seen, avoiding unobservable conditions leads to frequent high values of the inverse local estimation condition number, due to the better conditioning of matrix $\mathbf{O}$. We continue however to observe worse performance during times of lower observability, especially for the SPCI EKF between 40s and 65s, with a drop around 55s. In addition to the improved performance we also observe a higher consistency for all EKF variants.

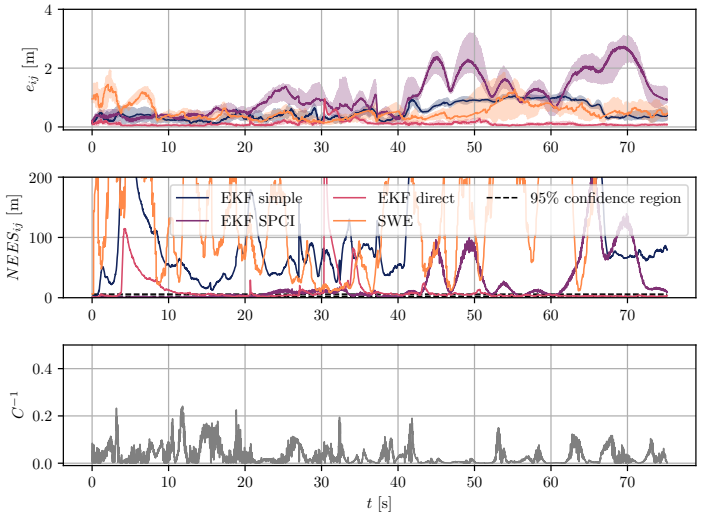


Figure 4.7: Median absolute position error (e_{ij}) and Normalized-Estimation-Error-Squared ($NEES_{ij}$) over 10 runs, alongside the inverse local estimation condition number C^{-1} for random trajectories. Shaded regions indicate the lower and upper quartiles.

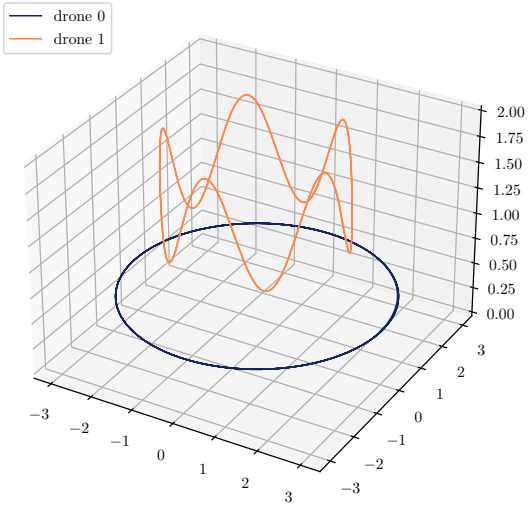


Figure 4.8: Trajectories designed to maximize observability

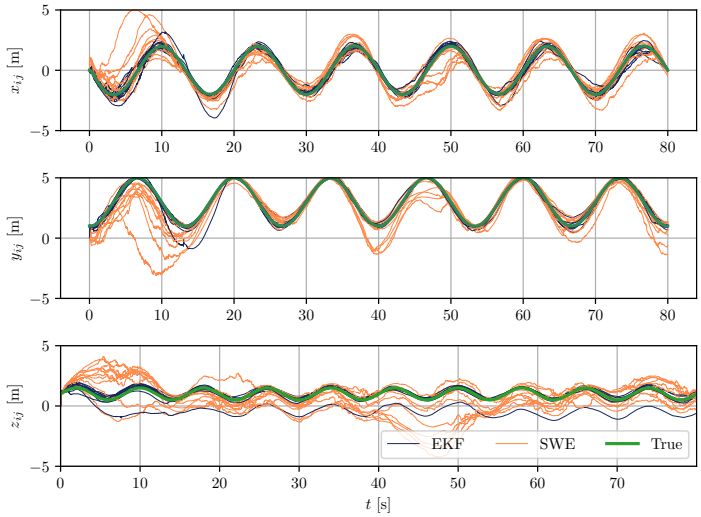


Figure 4.9: Simulated performance of simple EKF and SWE for the three states x_{ij}, y_{ij}, z_{ij} for 10 runs on favorable trajectories.

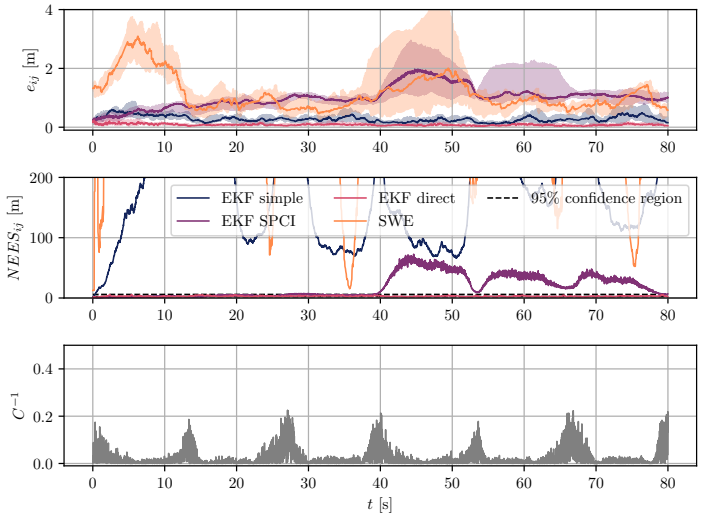
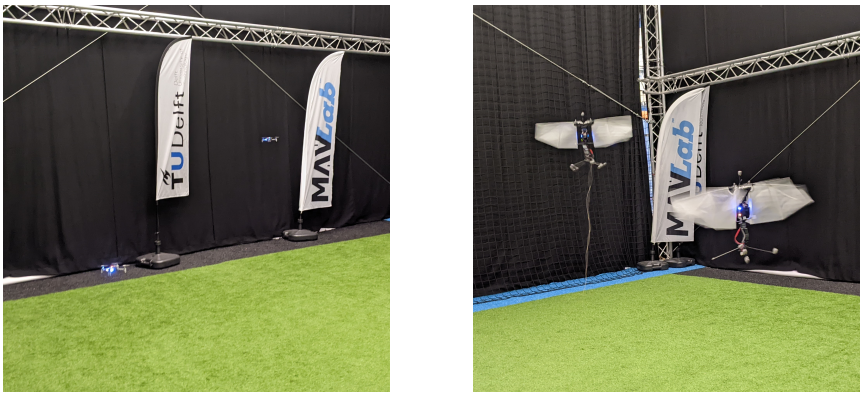


Figure 4.10: Median absolute position error (e_{ij}) and Normalized-Estimation-Error-Squared ($NEES_{ij}$) over 10 runs, alongside the inverse local estimation condition number C^{-1} for favorable trajectories. Shaded regions indicate the lower and upper quartiles.

The mean NEES after convergence is used as a measure of consistency. For a consistent, approximately linear-Gaussian filter, the mean NEES is expected to tend towards the number of states of the filter, which in our case is three [36]. In our simulation, all estimators turn out to be overconfident to various degrees, with exception for the direct EKF on the favorable trajectory. For the simple and SPCI EKF there is also an improved consistency on the favorable trajectory, but the NEES still stays outside of the 95% confidence interval most of the time. The advantage that the SPCI EKF offers over the simple EKF in terms of consistency does unfortunately not lead to an improved performance. Combined with the larger computational cost and the need to exchange covariance matrices between drones, this means that in its current form, the SPCI EKF is not appropriate for this problem.

4.5. EXPERIMENTAL SETUP

To test the proposed algorithm, we implemented our 3D relative localization on two different types of MAVs: Quadrotors (Crazyflie 2.1[1]) and flapping wing drones (Flapper Drones[2]), shown in Figure 4.11. While flapping wing drones offer many potential advantages such as energy efficiency, agility and robustness to collisions, they have so far not received the same level of attention [37]. This might be due to additional challenges arising from the use of a flapping wing platform. Specifically, the vibrations created from the flapping of the wings results in more noise on IMU measurements. Furthermore, a large horizontal drag leads to increased pitch and roll angles. By performing tests on these vastly different platforms, we want to demonstrate the versatility and robustness of our proposed method.



(a) Crazyflies (b) Flapper Drones

Figure 4.11: Drones used during experimental testing

[1] https://www.bitcraze.io/
[2] https://flapper-drones.com/wp/

4.5.1. HARDWARE

The Crazyflie is a versatile, lightweight platform with open-source hardware and software, whose capabilities can be extended by making use of expansion decks. In addition to the STM32F405 MCU which runs the firmware, it also employs a nRF51822 MCU for radio and power management. Since the Flapper Drones are based on the same Crazyflie hardware, they run almost the same firmware and are compatible with the Crazyflie expansion decks. Both types of drones are equipped with an Inertial Measurement Unit (IMU), containing a three-axis accelerometer and gyroscope (BMI088), and a barometric pressure sensor (BMP388). The Loco-Positioning Deck, which is used for UWB ranging between the drones, is based on the Decawave DWM1000 module. On the original Crazyflie, the Flow Deck V2 is used for optic flow measurements (PMW3901 optical flow sensor) and height measurements with a time-of-flight distance sensor (VL53L1x). Unfortunately, due to the orientation of the autopilot on the Flapper Drones and the large pitch and roll angles present during its flight, the Flow Deck V2 can't be employed on the Flapper Drones. As a result, it is more difficult to estimate the Flapper Drone's body velocity and altitude. To the best of our knowledge, this study is the first attempt to have flapping wing robots perform relative localization with onboard means.

4.5.2. RELATIVE LOCALIZATION ALGORITHM

To estimate the relative position of two drones, the EKF presented in Section 4.2 requires the yaw rate and horizontal velocities of both drones as input. Using (4.8), the yaw rate can be easily obtained from gyro measurements. Especially on the flapping wing drones, however, these measurements are extremely noisy and cause significant noise on the relative position estimates. We therefore filter yaw rate measurements using a second order Butterworth filter with a cutoff frequency of 5 Hz (lower than the ~ 12 Hz flapping frequency).

On the Crazyflie 2.1, we make use of the built-in EKF for on-board state estimation. The EKF relies on measurements from the IMU and the Flow Deck to estimate the drones attitude and velocity, which can then be used as inputs for the relative localization EKF after rotating them into the horizontal frame.

On the Flapper Drones on the other hand, using the Flow Deck is not possible due to the large vibrations and the drones architecture. Without optic flow and laser altimeter measurements, the on-board EKF cannot be used to calculate the inputs required for relative localization. Instead, we use a complementary filter for attitude estimation and predict the horizontal velocity as a function of attitude alone. For this model we consider the drone as a rigid body subject to three principal forces acting on its center of mass: thrust, drag and gravity. We further assume small pitch and roll angles and neglect changes in altitude to arrive at a rather simple expression for the evolution of the drone's horizontal velocity.

$$v_{x,k+1} = v_{x,k} + \mathrm{d}t\left(\tan\theta_k \cdot |g| - \cos^2\theta_k \cdot b_x v_{x,k}\right) \tag{4.20}$$

$$v_{y,k+1} = v_{y,k} + \mathrm{d}t\left(-\frac{\tan\phi_k}{\cos\theta_k}|g| - \cos^2\phi_k \cdot b_y v_{y,k}\right) \tag{4.21}$$

The drag coefficients b_x and b_y are found empirically by solving a least-squares minimization on a data set with ground-truth. More details on the derivation of the model

can be found in Appendix A.

Since this model assumes no changes in altitude during horizontal motion, it can not be used to estimate the vertical velocity. Instead, we fuse barometer and accelerometer data in a complementary filter following the approach of Wei *et al.* [38]. The resulting filter equation is given by (4.22), where h_{acc} is the accelerometer and h_{baro} the barometer measurement. For the implementation on the drone, we discretize the filter using the bilinear transform and a sampling period of $T_s = 0.01\,\text{s}$. Note that the accelerometer measurement must of course be rotated first to correct for the pitch and roll of the drone.

$$\hat{V}_z(s) = \frac{s^2}{s^2 + k_1 s + k_2} \cdot \frac{h_{\text{acc}}(s)}{s} + \frac{k_1 s + k_2}{s^2 + k_1 s + k_2} \cdot s \cdot h_{\text{baro}}(s) \tag{4.22}$$

Since the estimation of v_z relies on different sensor inputs, it makes sense to assume a different standard deviation of the error, when compared to v_x and v_y. Additionally, due to the different dynamics of the flapper in x and y direction, it makes sense to use individual process noise parameters for the relative localization EKF. Specifically, the process noise covariance matrix (4.18) is rewritten as (4.23), and is chosen independently for the quadrotor and for the flapping wing drone respectively.

$$\mathbf{Q} = \text{diag}([\sigma_{vx}^2, \sigma_{vy}^2, \sigma_{vz}^2, \sigma_{vx}^2, \sigma_{vy}^2, \sigma_{vz}^2, \sigma_w^2, \sigma_w^2]) \tag{4.23}$$

To measure the distance between two MAVs and exchange their horizontal velocity and yaw rate, we use UWB signals, employing the Crazyflie Loco-Positioning deck. The ranging and communication protocol used is the same as in [23], with the sole addition of also exchanging vertical velocity estimates.

The complete code used in the experiments is available on github for the crazyflies[3] and the Flapper Drones[4].

4.5.3. TEST ENVIRONMENT

The flight tests were performed in the "Cyberzoo" test arena at the TU Delft Faculty of Aerospace Engineering. The "Cyberzoo" is a $10\,\text{m} \times 10\,\text{m} \times 5\,\text{m}$ indoor arena equipped with an optitrack motion capture system, that provides groundtruth with millimeter level accuracy. Data was logged from the optitrack system and from the drone using the Crazyflie Suite[5].

4.6. RESULTS

In this section, we will present the results of our flight tests. The complete data underlying the plots and analysis has been published on 4TU Research Data[6]. Videos of the experiments can be found on Youtube[7].

[3] https://github.com/tudelft/crazyflie-firmware/tree/cf_swarm3d
[4] https://github.com/tudelft/crazyflie-firmware/tree/flapper_swarm3d
[5] https://github.com/tudelft/crazyflie-suite
[6] https://doi.org/10.4121/17372348
[7] https://www.youtube.com/playlist?list=PL_KSX9GOn2P-dzSwBvYYRZuKOiYvxOpIS

4.6.1. HORIZONTAL VELOCITY ESTIMATION

Since the horizontal velocity of the drones is an important input to the relative localization EKF, it makes sense to analyze the performance of the simplified velocity model developed for the flapper. The flapper model was tuned offline on a dataset consisting of 3 trajectories, flown manually at almost constant altitude, resulting in drag coefficients $b_x = 4.2$ and $b_y = 1.8$. Using these parameters, the root mean squared error (RMSE) was 0.13 m/s for v_x and 0.22 m/s for v_y on the training set. The RMSE for the vertical velocity estimate using the manually tuned complementary filter was 0.20 m/s on the same dataset. During the derivation of the model, a constant altitude was assumed, which is often not the case. We therefore also validated the model on more aggressive trajectories that included major variations in altitude. An example for the performance in such a flight is shown in Figure 4.12. While the performance is slightly reduced in x (RMSE=0.18 m/s), y (RMSE=0.31 m/s) and z (RMSE=0.26 m/s), the results still look good enough to justify the use of the model on-board the drones.

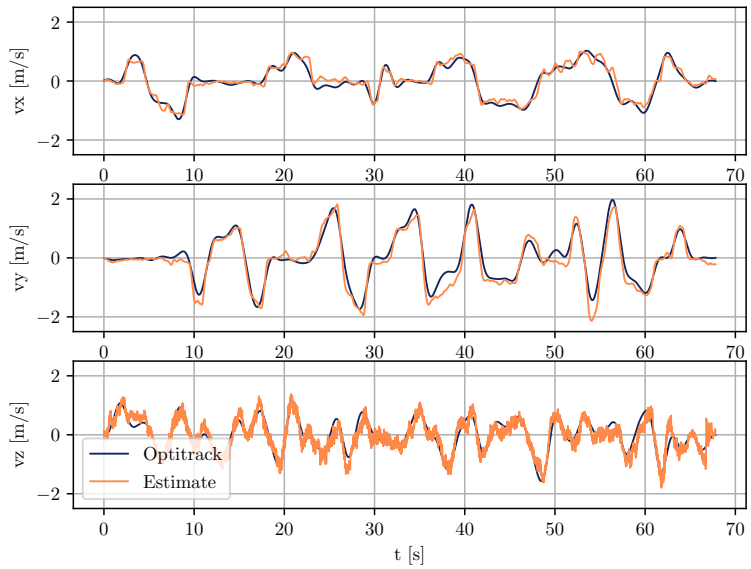


Figure 4.12: Offline performance of the horizontal velocity model for the Flapper Drone when flying with varying altitude.

We can now also compare the on-board performance of the flapper model with the EKF of the Crazyflie 2.1. Figure 4.13 shows the aggregated performance data over 10 manual flights for the Crazyflie, Flapper Drones flying at almost constant altitude and Flapper Drones flying with large changes in altitude. Due to range limitations of the Flow Deck, it is not possible to fly large altitude variations with the Crazyflie while still estimating its velocities.

The on-board performance of the Flapper Drone's velocity estimation matches the

observation from the offline testing. When flying aggressive trajectories with frequent altitude changes, the performance of the model degrades particularly along the y and z axis. It is however striking that, for similar conditions (level flight), the model performs only slightly worse in x and y than the EKF on the Crazyflie, despite having no access to optic flow.

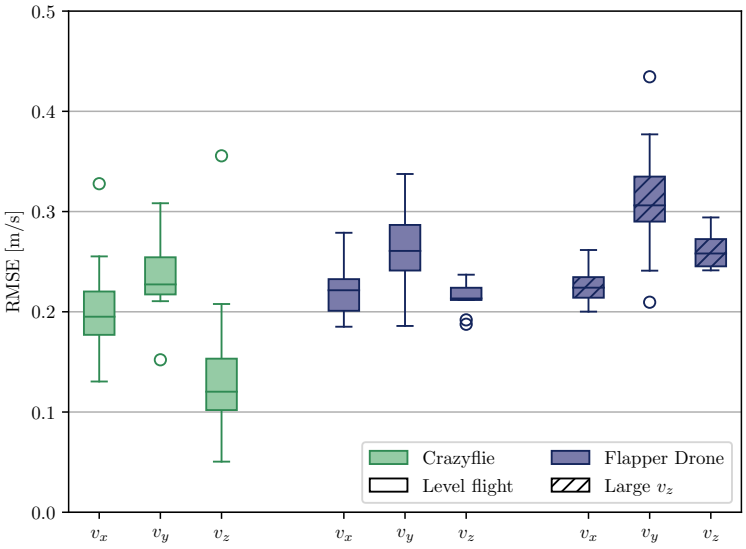


Figure 4.13: RMSE of the horizontal and vertical velocity estimation on the Crazyflie 2.1 and on the Flapper Drones aggregated over 10 trajectories

4.6.2. RELATIVE LOCALIZATION

To test the 3D relative localization, we performed manual flights with two MAVs without following any particular patterns. Looking at the results for two individual flights using Crazyflies (Figure 4.14) and Flapper Drones (Figure 4.15), we can observe that the relative position estimate stabilizes within several seconds, which is similar to what we saw in simulation.

Looking at the aggregated data over 5 flights in Figure 4.16, it can be seen that the performance along different axes varies in a pattern similar to what could be observed for the velocity estimation (figure 4.13). This makes sense, given that the velocity estimates serve as the main input to determine the relative motion between the MAVs.

4.6.3. LEADER-FOLLOWER

Finally, to show the applicability of the proposed relative localization method to real swarming scenarios, we performed leader-follower flights with two MAVs. In this sce-

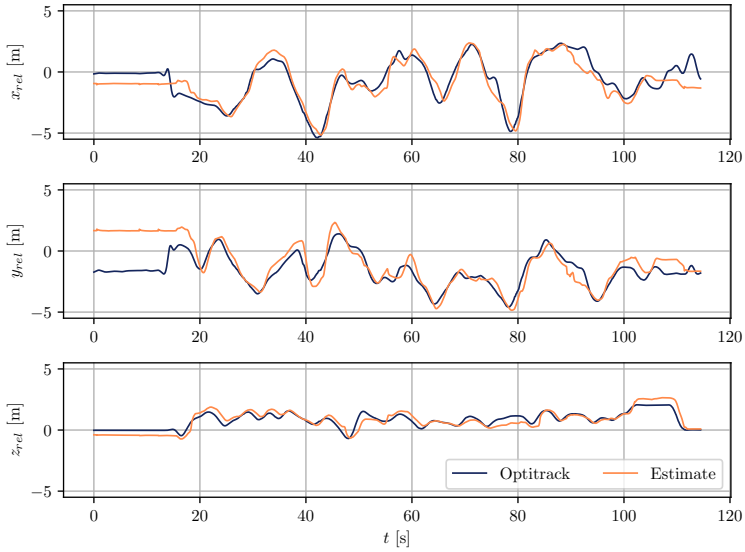


Figure 4.14: Relative Localization during free flight with two Crazyflies

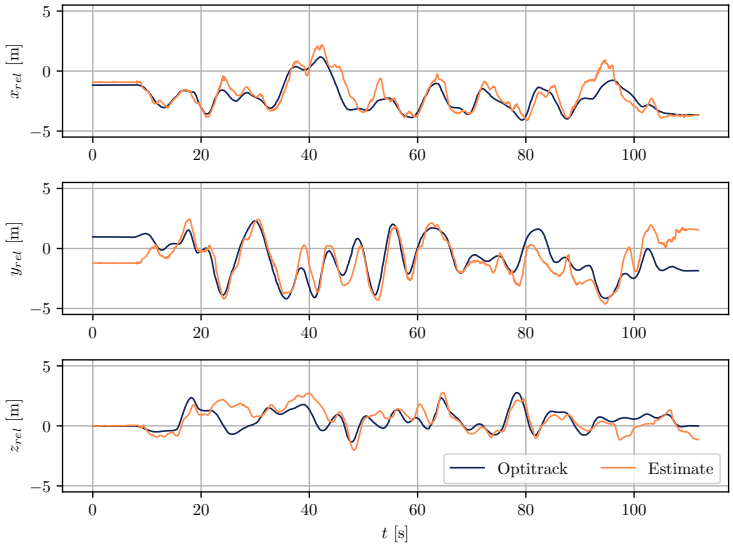


Figure 4.15: Relative Localization during free flight with two Flapper Drones

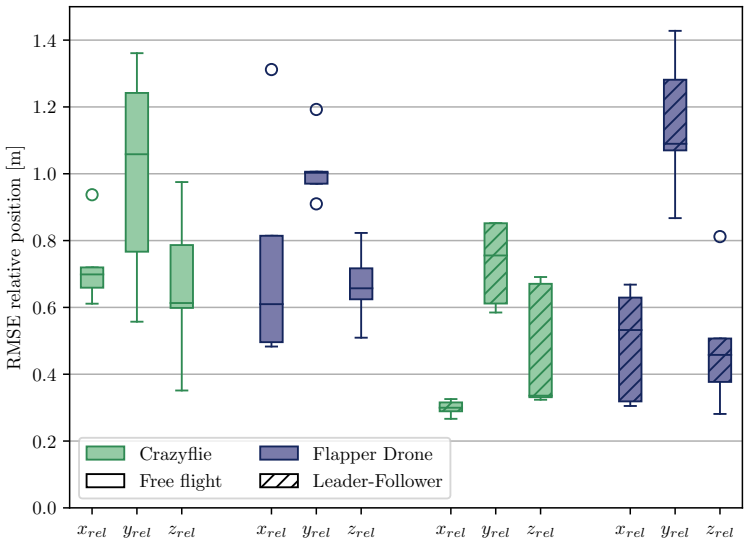


Figure 4.16: Comparison of relative localization on Crazyflie and Flapper Drones for free flight and leader-follower experiment aggregated over 5 flights. RMSE calculation on starts 30 s into the flight and stops 5 s before landing.

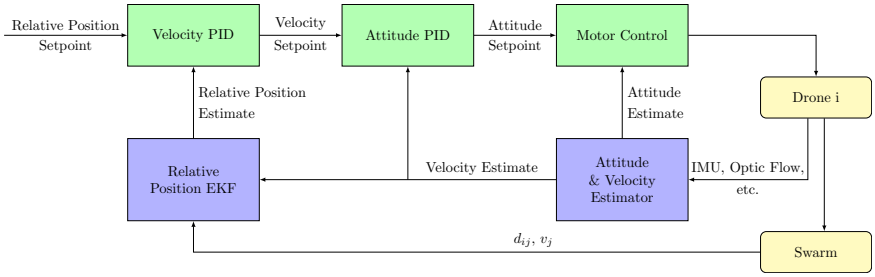


Figure 4.17: Leader-Follower control layout. Blocks in yellow represent the physical systems.

nario, the first MAV is controlled manually while the second MAV tries to keep a constant relative position. When the first MAV takes off manually, the second MAV receives a command to start its flight via UWB. The second MAV then takes off as well and performs a 20 s initialization sequence, during which it flies in random patterns based on odometry alone. After the initialization sequence is complete, the second MAV switches to follow mode, where it tries to stay in its desired relative position. This is done by calculating velocity commands directly from the relative position error using a simple PID controller along each individual axis. These velocity commands are then passed to the inner control loop of the drones that control attitude as shown in Figure 4.17. Different gains were used for the two types of drones as reported in Table 4.1. Note that for the Flapper drone only a PD controller was used along the xy axes.

	k_P	k_D	k_I
Crazyflie	2.0	0.01	0.0001
Flapper Drone (xy)	3.5	0.06	0
Flapper Drone (z)	3.5	0.06	0.001

Table 4.1: PID gains for relative position control

During the experiments, it was found that large deviations from the target position at the end of the initialization sequence could cause unstable behaviour, which is why during the initialization sequence, the leader MAV was flown close to the desired relative position. As can be seen in Figure 4.18, the Crazyflie follower can track the leader very well after initialization, with errors rarely exceeding 1.0 in y and 0.5 m in x and z directions. The Flapper Drone on the other hand performs a bit worse in the sense that, while the general shape of the trajectory is tracked, the offset from the desired position is larger (figure 4.19). This could be due to the differences in relative localization accuracy, but is also likely to result from sub-optimal controller tuning for this vehicle with more complex dynamics. The large difference between the error in x and y direction, also seen in Figure 4.16, can be explained by the desired position offset of the follower, causing movement in x direction to be more observable than in y (movement perpendicular to the UWB ranging).

Surprisingly, Figure 4.16 shows that the relative localization mostly performs better (lower RMSE) in the leader-follower scenario than in free flight. Particularly for the ex-

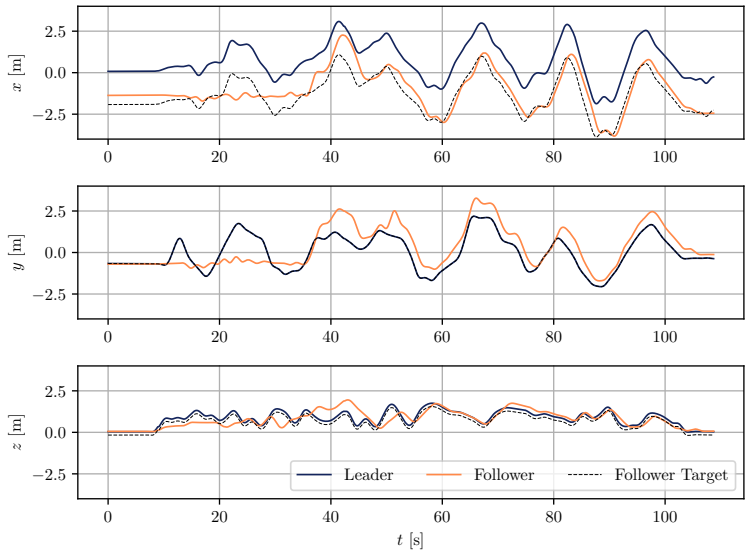


Figure 4.18: Optitrack leader and follower position during leader-follower experiment with Crazyflies (target Δx=-2 m, Δy=0 m, Δz=-0.2 m)

periments on the crazyflie, improvements of around 0.3 m can be observed along all directions. For the flapper drone the difference is not as clear and is most pronounced along the z axis. This improvement is likely due to a self-stabilizing effect in the closed loop system. In fact, if the relative position along a given axis becomes less observable, its estimate will degrade. This results in a control input that excites that particular axis, rendering it observable again. A more detailed look at this behaviour is given by Li *et al.* [23]. In free flight on the other hand, the MAVs might fly in patterns that in which an axis remains unobservable for longer periods of times.

4.6.4. MULTIPLE FOLLOWERS

One major concern in the development of algorithms for groups of robots is the scalability. For this relative localization method, the communication frequency is considered the main bottleneck. Specifically, the more drones are part of the swarm, the more messages need to be exchanged. This will result in a reduced frequency of ranging between two given drones. A naive, fully connected communication graph will drop below 5 Hz ranging frequency for more than 12 drones [23]. Furthermore, small drones don't carry a lot of computational power, adding another limitation to the number of drones to which any single drone is localizing. Advances in communication setup and flexible localization in local neighbourhoods will be necessary to scale up the UWB-based relative localization to much larger groups of robots. Here, to reduce the load on both communication and computation, we tested the implementation of a filter bank with a group of 4 drones. This limits the information that has to be exchanged through UWB and keeps the filter matrices small, resulting in a better computational efficiency. For our experiments, we

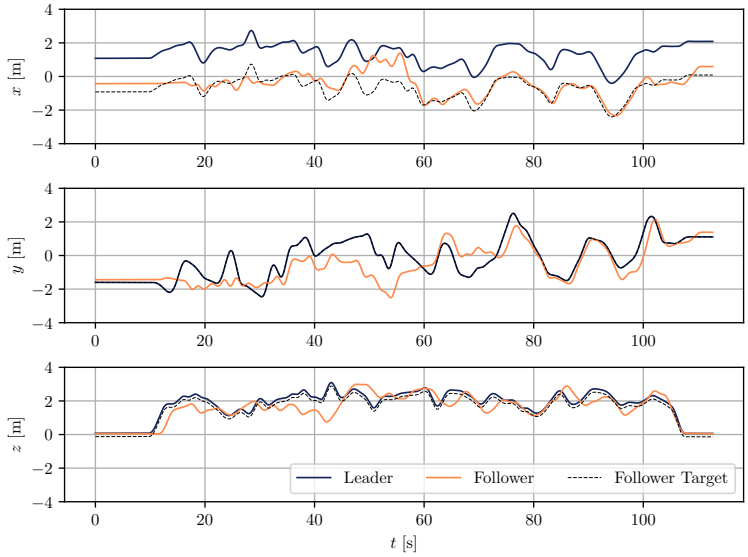


Figure 4.19: Optitrack leader and follower position during leader-follower experiment with Flapper Drones (target Δx=-2 m, Δy=0 m, Δz=-0.2 m)

let the followers fly behind the leader in a diamond formation with the following offsets: $(-1.0, -1.0)$ m, $(0.0, -2.0)$ m and $(-1.0, 1.0)$ m. As can be seen in Figures 4.20 and 4.21, the group is still capable of performing the leader-follower behaviour with a similar accuracy as for the two drone case. Note that towards the end of the experiment, Follower 1 is running out of battery.

4.7. CONCLUSION

Relative localization is a key aspect of agents in robotic swarms. On a lower level, it helps the robots to avoid collisions with their peers, while on a higher level, it is a prerequisite for most complex swarming behaviours. In this chapter, we presented a 3D relative localization algorithm based only on inter-agent ranging and body-velocity estimates. Due to its computational efficiency and the use of lightweight sensors, this algorithm is particularly promising for swarms of MAVs, where individual agents have a very limited payload capacity of only several grams and perform most of their computations on a single core microcontroller.

To understand the limitations of the algorithm, we performed an observability analysis of the system and determined conditions in which observability is no longer guaranteed. We then validated the algorithm in simulation and investigated the effect of choosing trajectories that maximize observability.

Finally, the algorithm was tested on two different types of drones: quadrotors and flapping wing drones. These experiments not only demonstrated the platform independence of the proposed algorithm, but also showed that purely IMU based velocity

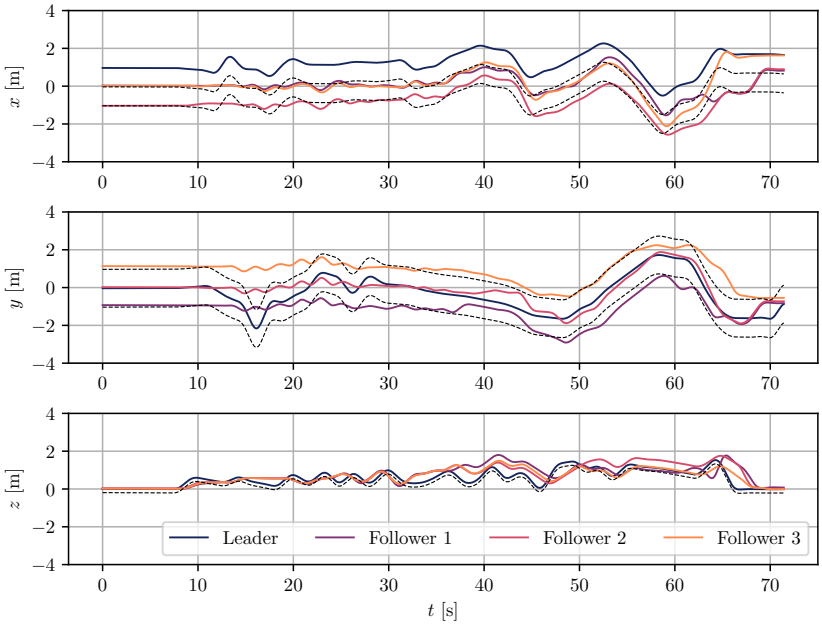


Figure 4.20: Absolute position of the leader and the 3 followers. The dashed lines represent the target trajectory for the followers.

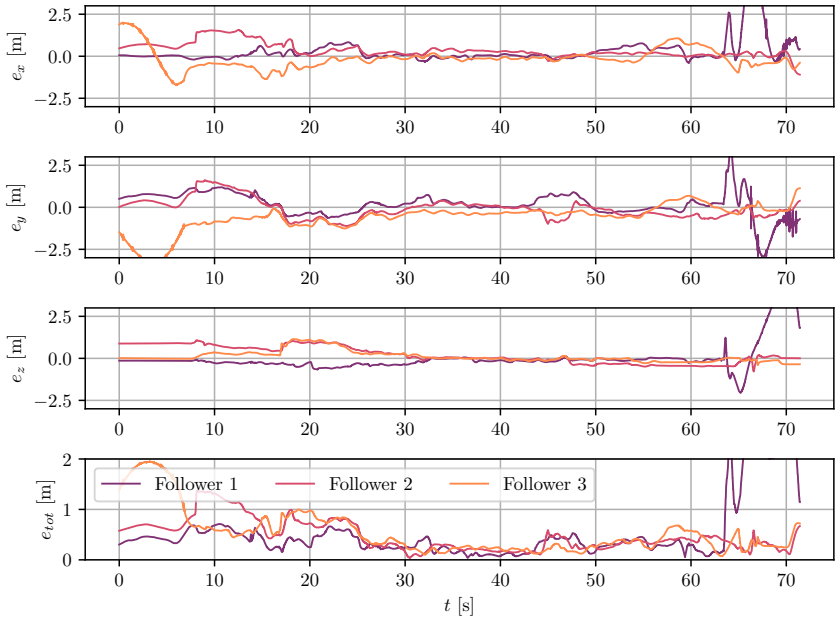


Figure 4.21: Error of the relative position estimate on the leader drone

estimation is sufficiently accurate for relative localization on flapping wing drones. It is however important to note, that the reliance on velocity estimates is a limitation of our algorithm. In situations, where the body velocities of the drones are difficult to estimate or are very noisy, the performance of the algorithm will start to deteriorate. While an analysis of the consistency of our method shows overconfidence of the estimator, it's performance in simulation and practical experiments is in line with other algorithms developed for relative localization. It is however an aspect that would profit from additional improvements.

A noteworthy observation is the improved performance of the relative localization in a leader-follower scenario. While one would intuitively expect the algorithm to perform worse in an unobservable configuration (identical velocities), the closed-loop system exhibits a self-stabilizing behaviour that actually improves the localization accuracy. In fact, the unobservability of the relative position leads to small excitations along the unobservable axes. This in turn quickly makes the axes observable again, while for uncoordinated flights, these unobservable configurations might persist for longer.

Since this research mainly focused on relative localization between two agents, it makes sense to place this work in the larger swarming context. While our method can be easily extended to small swarms by running multiple EKFs in parallel on each drone, this would quickly strain memory and computation capacity if too many drones are added. Furthermore, this method does not take advantage of additional measurements between other drones like rigidity based methods do. However, it does allow drones to continue tracking each other's relative position, even in cases where communication links are interrupted and the swarm loses it's rigidity. As a result, real-world robotic swarms would benefit from a combination of these two methods to improve the robustness of relative localization in a wide variety of scenarios.

In addition to investigating mixed relative localization methods, the scalability of these methods should further be improved. A critical bottleneck in wireless-ranging-based relative localization is the communication channel which can quickly saturate as more agents are added to the swarm. To resolve this issue, a reduction in the number of required communication links within the swarm and in the amount of data exchanged is required.

REFERENCES

[1] S. Pfeiffer, V. Munaro, S. Li, A. Rizzo, and G. C. H. E. de Croon, *Three-dimensional relative localization and synchronized movement with wireless ranging,* Swarm Intelligence **17**, 147 (2023).

[2] W. H. Maes and K. Steppe, *Perspectives for Remote Sensing with Unmanned Aerial Vehicles in Precision Agriculture,* Trends in Plant Science **24**, 152 (2019).

[3] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, *State-of-the-art technologies for UAV inspections,* IET Radar, Sonar and Navigation **12**, 151 (2018).

[4] J. Kwon and S. Hailes, *Scheduling uavs to bridge communications in delay-tolerant*

networks using real-time scheduling analysis techniques, in *2014 IEEE/SICE International Symposium on System Integration* (2014) pp. 363–369.

[5] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, *Swarm robotics: A review from the swarm engineering perspective,* Swarm Intelligence **7**, 1 (2013).

[6] E. Şahin, *Swarm robotics: From sources of inspiration to domains of application,* in *Swarm Robotics*, edited by E. Şahin and W. M. Spears (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005) pp. 10–20.

[7] J. Zhu and S. S. Kia, *Cooperative localization under limited connectivity,* IEEE Transactions on Robotics **35**, 1523 (2019).

[8] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, *A survey on swarming with micro air vehicles: Fundamental challenges and constraints,* Frontiers in Robotics and AI **7**, 18 (2020).

[9] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, *Optimized flocking of autonomous drones in confined environments,* Science Robotics **3**, 1 (2018).

[10] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, *Crazyswarm : A Large Nano-Quadcopter Swarm,* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017) pp. 3299–3304.

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

[12] B. Vedder, H. Eriksson, D. Skarin, J. Vinter, and M. Jonsson, *Towards collision avoidance for commodity hardware quadcopters with ultrasound localization,* in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (2015) pp. 193–203.

[13] J. Faigl, T. Krajník, J. Chudoba, L. Přeučil, and M. Saska, *Low-cost embedded system for relative localization in robotic swarms,* in *2013 IEEE International Conference on Robotics and Automation* (2013) pp. 993–998.

[14] A. Carrio, H. Bavle, and P. Campoy, *Attitude estimation using horizon detection in thermal images,* International Journal of Micro Air Vehicles **10**, 352 (2018), https://doi.org/10.1177/1756829318804761 .

[15] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, *3-D relative positioning sensor for indoor flying robots,* Autonomous Robots **33**, 5 (2012).

[16] E. Tijs, G. C. H. E. de Croon, J. Wind, B. Remes, C. De Wagter, H. de Bree, and R. Ruijsink, *Hear-and-avoid for micro air vehicles,* in *Proceedings of the International Micro Air Vehicle Conference and Competitions (IMAV), Braunschweig, Germany*, Vol. 69 (2010).

[17] T. Eren, O. Goldenberg, W. Whiteley, Y. Yang, A. Morse, B. Anderson, and P. Belhumeur, *Rigidity, computation, and randomization in network localization,* in *IEEE INFOCOM 2004*, Vol. 4 (2004) pp. 2673–2684 vol.4.

[18] Y. Shang and W. Ruml, *Improved mds-based localization,* in *IEEE INFOCOM 2004*, Vol. 4 (2004) pp. 2640–2651 vol.4.

[19] N. Trawny, X. S. Zhou, K. Zhou, and S. I. Roumeliotis, *Interrobot transformations in 3-d,* IEEE Transactions on Robotics **26**, 226 (2010).

[20] N. Trawny, X. S. Zhou, K. X. Zhou, and S. I. Roumeliotis, *3d relative pose estimation from distance-only measurements,* in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2007) pp. 1071–1078.

[21] M. Coppola, K. N. McGuire, K. Y. W. Scheper, and G. C. H. E. de Croon, *On-board communication-based relative localization for collision avoidance in micro air vehicle teams,* Autonomous Robots **42**, 1787 (2018).

[22] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, *Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments,* International Journal of Micro Air Vehicles **9**, 169 (2017), https://doi.org/10.1177/1756829317695564 .

[23] S. Li, M. Coppola, C. D. Wagter, and G. C. H. E. de Croon, *An autonomous swarm of micro flying robots with range-based relative localization,* (2021), arXiv:2003.05853 [cs.RO] .

[24] C. C. Cossette, M. Shalaby, D. Saussié, J. R. Forbes, and J. Le Ny, *Relative position estimation between two uwb devices with imus,* IEEE Robotics and Automation Letters **6**, 4313 (2021).

[25] B. Jiang, B. D. O. Anderson, and H. Hmam, *3-d relative localization of mobile systems using distance-only measurements via semidefinite optimization,* IEEE Transactions on Aerospace and Electronic Systems **56**, 1903 (2020).

[26] T. Ziegler, M. Karrer, P. Schmuck, and M. Chli, *Distributed formation estimation via pairwise distance measurements,* IEEE Robotics and Automation Letters **6**, 3017 (2021).

[27] T. H. Nguyen and L. Xie, *Relative transformation estimation based on fusion of odometry and uwb ranging data,* (2022), arXiv:2202.00279 .

[28] R. K. Williams and G. S. Sukhatme, *Observability in topology-constrained multi-robot target tracking,* in *2015 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2015) pp. 1795–1801.

[29] L. Heintzman and R. K. Williams, *Nonlinear observability of unicycle multi-robot teams subject to nonuniform environmental disturbances,* Autonomous Robots **44**, 1149 (2020).

[30] R. Mahony, T. Hamel, and J.-M. Pflimlin, *Nonlinear Complementary Filters on the Special Orthogonal Group,* IEEE Transactions on Automatic Control **53**, 1203 (2008).

[31] D. Simon, *Optimal State Estimation* (John Wiley & Sons, Hoboken NJ, 2006).

[32] F. Arrichiello, G. Antonelli, A. P. Aguiar, and A. Pascoal, *An observability metric for underwater vehicle localization using range measurements,* Sensors **13**, 16191 (2013).

[33] R. Hermann and A. Krener, *Nonlinear controllability and observability,* IEEE Transactions on Automatic Control **22**, 728 (1977).

[34] A. J. Krener and K. Ide, *Measures of unobservability,* in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference* (2009) pp. 6401–6406.

[35] M. Shalaby, C. C. Cossette, J. R. Forbes, and J. Le Ny, *Relative position estimation in multi-agent systems using attitude-coupled range measurements,* IEEE Robotics and Automation Letters **6**, 4955 (2021).

[36] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, *Consistency of the ekf-slam algorithm,* in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006) pp. 3562–3568.

[37] G. C. H. E. de Croon, *Flapping wing drones show off their skills,* Science Robotics **5**, eabd0233 (2020), https://www.science.org/doi/pdf/10.1126/scirobotics.abd0233 .

[38] S. Wei, G. Dan, and H. Chen, *Altitude data fusion utilising differential measurement and complementary filter,* IET Science, Measurement and Technology **10**, 874 (2016).

5

RANGE-BASED RELATIVE LOCALIZATION FOR LARGE AND DYNAMIC ROBOTIC SWARMS

While robotic swarms have become more potent over recent years, scalability has remained an issue for many types of swarming algorithms. Since the robotic agents making up the swarms have limited computational power and communication bandwidth, decentralized algorithms for large swarms must be efficient and should be bounded in terms of computational power, no matter the size of the swarm. This is especially true for relative localization algorithms, which are required to improve coordination and avoid collisions between agents. In this chapter, we present a decentralized algorithm for relative localization in large, dynamic swarms, where "dynamic" refers to changing connectivity between agents in the swarm. This "Dynamic EKF" selects a subset of agents to localize in real-time, according to the current neighborhood of the localizing agent. While designing the algorithm, we also investigate and address localization ambiguities in the system that can cause estimates to converge to wrong solutions. We evaluate the localization algorithm in simulations with tens of agents and show that it can achieve comparable accuracy to estimators for the full swarm. At the same time, we use the simulation to verify that the new estimator requires less computational resources and is inherently able to adjust to changing swarm connectivity.

5.1. INTRODUCTION

Inspecting industrial facilities [1], monitoring crops [2] or searching for victims in disaster zones [3]: the potential applications for small, autonomous drones are manifold. To increase flexibility and improve robustness in these kinds of tasks, it is desirable to use multiple cooperating drones at the same time, forming a robotic swarm [4]. For these swarms to perform optimally, each individual agent needs to know the location of the neighboring agents. On the one hand, this allows for better coordination and distribution of tasks, while on the other hand, it helps agents not to crash into each other [5].

For aerial swarms over open fields, this topic does not receive much attention, since satellite-based positioning like GPS can give each agent an accurate, global position estimate that can be broadcast to the other agents [6]. Inside or near buildings, however, the performance of GPS suffers due to blocked or reflected signals, while at the same time, the margins for error in an enclosed space are much smaller [7].

A simple approach that results in great localization in GPS-denied environments is the use of external means of localization such as visual motion capture systems. These systems have been used successfully to control swarms of up to 49 small drones [8][9], but require prior modification of the environment and a centralized system to calculate positions. For tasks in unknown environments, where a stable communication link to a central base station cannot be guaranteed, the swarm should only rely on local sensing and use decentralized localization algorithms.

On-board visual methods for example can be implemented completely decentralized and without the need for any communication between agents [10][11]. These methods use on-board cameras to detect other agents in images and calculate the relative position from image coordinates. While allowing accurate relative localization in small swarms, the main downside of these methods is that they struggle in bad lighting conditions and are limited by the camera's field of view.

Range-based methods, on the other hand, rely on knowledge of the distances between agents, which can, for example, be determined using infrared [12][13] (which can also provide angle-of-arrival information but suffers from field-of-view limitations), Bluetooth [14] or Ultra-Wideband [15][16][17]. There are multiple methods to determine the relative position of another agent using range measurements, all with their advantages and shortcomings. Using consecutive range measurements with knowledge of an agent's displacement, individual agents can be localized by minimizing an appropriate cost function [15]. When having access to all range measurements in the system, the connectivity of the swarm can even be used in an optimization-based approach to determine the relative positions of all agents at once [18][17]. Since solving optimization problems is often computationally expensive, the relative position of individual agents can also be estimated using Kalman filtering [14], like we did in Chapter 4.

An important shortcoming of these current relative localization algorithms is scalability. Most of the surveyed research papers on distributed algorithms with local sensing only look at very small swarms, usually with a single-digit number of agents. Furthermore, little thought is usually put into the scalability of these methods, which is a key aspect of their usefulness in robotic swarms [5]. One of the few investigations of scalability was conducted by Li [19] for a UWB-based relative ranging and localization approach which could achieve the desired ranging frequency of 5 Hz with up to 12 agents

in the swarm. In a very different context, Matsuka *et al.* [20] investigated scalable and distributed relative pose estimation for swarms of spacecraft, using pose measurements from a monocular camera.

Until recently, the same was also true for ranging protocols. The use of Master-Slave topologies [15], multiple tag-anchor pairs [17] or token-ring protocols (Chapter 4) does only really work for small and fully connected swarms. However, the recent work of Shan et al. on the scalability of UWB ranging [21] has opened the door for ranging in large and not fully connected swarms. With a scalable ranging protocol, it should now also be possible to design scalable relative localization algorithms for large, dynamic swarms.

Based on our previous work on agent-to-agent relative localization in Chapter 4, we will thus address the problem of range-based relative localization in large, *dynamic* swarms. Different from a *static* swarm, a *dynamic* swarm is characterized by an ever-changing connectivity matrix. Dynamic swarms may split and merge, while an individual agent might:

1. encounter an agent it has never seen before
2. see an agent for the last time ever
3. not be in contact with any agent at all for a while

As the size of any swarm grows, it becomes more likely that it will show some dynamic behavior. This dynamic behavior can result from drones encountering obstacles that block communication, or the swarm diameter exceeding the communication range. Our key contributions are:

1. The design of a dynamic, relative localization EKF, that incorporates secondary range measurements, reduces the computational requirements and enables easy swapping of the tracked agents.
2. An analysis of localization ambiguities that can appear in the system, as well as the presentation and evaluation of methods designed to resolve them.

This chapter is organized as follows: We will first formulate the relative localization problem in Section 5.2, where we will also present the mathematical basis for the relative localization EKF as well as the associated problem of localization ambiguities. In Section 5.3, we will present various methods to address these ambiguities, while in Section 5.4, we show how we implemented scalable and dynamic relative localization in our simulator. Finally, the results of the simulation experiments will be presented in Section 5.5.

5.2. PROBLEM FORMULATION

5.2.1. NOTATION

The state estimators presented in this chapter are designed for an agent s, ("self") that simultaneously estimates the relative position of n other agents in its neighborhood. For a swarm with N agents, $n \leq N-1$. For generality, we use the subscripts i, j when referring to variables associated with other agents in the swarm.

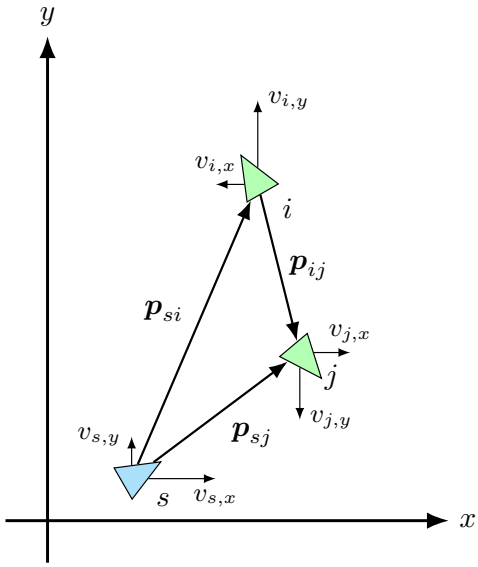


Figure 5.1: The relative localization problem shown with $N = 3$ agents. The goal is for agent s to reliably estimate the relative positions of agents i and j, $\boldsymbol{p}_{si} = [x_{si}, y_{si}]$ and $\boldsymbol{p}_{sj} = \begin{bmatrix} x_{sj}, y_{sj} \end{bmatrix}$, based on direct ($\|\boldsymbol{p}_{si}\|, \|\boldsymbol{p}_{sj}\|$) and secondary ($\|\boldsymbol{p}_{ij}\|$) range measurements.

To simplify calculations, we break up vectors and matrices into blocks that correspond to the different drones in the system. Full matrices and vectors will be written in bold, while blocks will use regular capital letters and indices in the subscript. For example, the covariance matrix $\boldsymbol{P}$ is broken down as follows:

$$\boldsymbol{P} = \begin{bmatrix} P_{00} & \dots & P_{0j} & \dots \\ \vdots & \ddots & \vdots & \\ P_{i0} & \dots & P_{ij} & \dots \\ \vdots & & \vdots & \ddots \end{bmatrix} \tag{5.1}$$

Blocks that only contain zeros will be denoted by $\mathbf{0}$ and $\mathcal{I}_N$ is the $N \times N$ identity matrix. Finally, dt is the time between two consecutive predictions of the EKF. To keep equations clean, we will omit the use of the subscript k for the current time when simplifying terms.

5.2.2. RELATIVE LOCALIZATION IN STATIC SWARMS

Let us first formulate the relative localization problem for a *static swarm* with N agents. Since the main topic of this work is the scalability of relative localization, we consider the simplified case of drones moving in two dimensions with access to a compass to determine a common reference orientation. Using the reference orientation, each drone can estimate its global velocity from optic flow or IMU measurements [22]. Furthermore, range measurements between drones can be made using Ultra-Wideband, which can also be used for communicating global velocities. An illustration of the relative localiza-

tion problem with all relevant variables is shown in Figure 5.1.

The full state vector on agent s is $\boldsymbol{X}$ and contains the relative positions of $n \leq N-1$ tracked agents.

$$\boldsymbol{X} = [X_0, \cdots, X_{s-1}, X_{s+1}, \cdots, X_{n-1}] \tag{5.2}$$
$$X_i = \begin{bmatrix} x_{si}, y_{si} \end{bmatrix} \tag{5.3}$$

The input vector $\boldsymbol{U}$ contains the global velocities measured by all of the drones. Note that the velocities of drone s itself are appended at the end of the vector as they will impact all of the relative positions.

$$\boldsymbol{U} = [U_0, \cdots, U_{s-1}, U_{s+1}, \cdots, U_{n-1}, U_s] \tag{5.4}$$
$$U_i = \begin{bmatrix} v_{i,x}, v_{i,y} \end{bmatrix} \tag{5.5}$$

The prediction equation for our EKF is then quite simple:

$$\boldsymbol{X}_{k+1}^{-} = \boldsymbol{A}\boldsymbol{X}_k^{+} + \boldsymbol{B}\boldsymbol{U}_k \tag{5.6}$$
$$\boldsymbol{A} = \mathscr{I}_{2n} \tag{5.7}$$
$$\boldsymbol{B} = \begin{bmatrix} \ddots & \mathbf{0} & \mathbf{0} & \vdots \\ \mathbf{0} & B_i & \mathbf{0} & B_s \\ \mathbf{0} & \mathbf{0} & \ddots & \vdots \end{bmatrix} \tag{5.8}$$
$$B_i = dt \cdot \mathscr{I}_2 \qquad B_s = -dt \cdot \mathscr{I}_2 \tag{5.9}$$

Even when using more complex system models (such as in Chapter 4), $\boldsymbol{A}$ and $\boldsymbol{B}$ will remain sparse, since the relative positions of two agents are only linked through U_s. That means, that the computational cost of the prediction step can be decreased significantly by propagating each block independently. Let's therefore consider the more general case where $\boldsymbol{A}$ is block-diagonal, $\boldsymbol{B}$ has the form shown in (5.8) and all drones experience the same process noise, i.e. $Q_i = Q_s = Q$.

$$\boldsymbol{A} = \begin{bmatrix} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{bmatrix} \tag{5.10}$$

$$\boldsymbol{Q} = \begin{bmatrix} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{bmatrix} \tag{5.11}$$

The covariance propagation for the prediction step is then

$$\boldsymbol{P}_{k+1}^{-} = \boldsymbol{A}\boldsymbol{P}_k^{+}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^T \tag{5.12}$$

$$\left[\boldsymbol{P}_{k+1}^{-}\right]_{ij} = A_i \left[\boldsymbol{P}_k^{+}\right]_{ij} A_j^T + \delta_{ij} \cdot B_i Q B_i^T + B_s Q B_s^T \tag{5.13}$$

where δ_{ij} is the Kronecker delta. In our simplified 2D model with global velocities, this further reduced to

$$\left[\boldsymbol{P}_{k+1}^{-}\right]_{ij} = \left[\boldsymbol{P}_k^{+}\right]_{ij} + \left(\delta_{ij} + 1\right) \cdot dt^2 \cdot Q \tag{5.14}$$

In the correction step, we can make use of direct range measurements between agents s and i to refine the estimate. The measurement equation $h_D(\boldsymbol{X})$ and its Jacobian $\boldsymbol{H}_D(\boldsymbol{X})$ are given by

$$h_D(\boldsymbol{X}) = \|\boldsymbol{p}_{si}\| = \sqrt{x_{si}^2 + y_{si}^2} \tag{5.15}$$

$$\boldsymbol{H}_D(\boldsymbol{X}) = \left[\frac{\partial h_D(\boldsymbol{X})}{\partial \boldsymbol{X}} + \right] = \left[\ldots, \mathbf{0}, H_{D,i}, \mathbf{0}, \ldots\right] \tag{5.16}$$

$$H_{D,i} = \left[\frac{x_{si}}{\|\boldsymbol{p}_{si}\|}, \frac{y_{si}}{\|\boldsymbol{p}_{si}\|}\right] \tag{5.17}$$

5

In this work, we further want to take advantage of range measurements between two other agents i and j ($\|\boldsymbol{p}_{ij}\|$ in Figure 5.1)to improve the accuracy and convergence rate of the estimator. We call these measurements *secondary range measurements*, which are described by the following measurement equation and Jacobian:

$$h_S(\boldsymbol{X}) = \|\boldsymbol{p}_{ij}\| = \|\boldsymbol{p}_{sj} - \boldsymbol{p}_{si}\| = \sqrt{\left(x_{sj} - x_{si}\right)^2 + \left(y_{sj} - y_{si}\right)^2} \tag{5.18}$$

$$\boldsymbol{H}_S(\boldsymbol{X}) = \left[\frac{\partial h_S(\boldsymbol{X})}{\partial \boldsymbol{X}}\right] = \left[\ldots, \mathbf{0}, H_{S,i}, \mathbf{0}, \ldots, \mathbf{0}, H_{S,j}, \mathbf{0}, \ldots\right] \tag{5.19}$$

$$H_{S,i} = \left[\frac{-\left(x_{sj} - x_{si}\right)}{\|\boldsymbol{p}_{ij}\|}, \frac{-\left(y_{sj} - y_{si}\right)}{\|\boldsymbol{p}_{ij}\|}\right] \tag{5.20}$$

$$H_{S,j} = \left[\frac{\left(x_{sj} - x_{si}\right)}{\|\boldsymbol{p}_{ij}\|}, \frac{\left(y_{sj} - y_{si}\right)}{\|\boldsymbol{p}_{ij}\|}\right] \tag{5.21}$$

Since measurements arrive asynchronously, they have to be applied consecutively. This has the benefit of reducing the inversion of the innovation covariance matrix $\boldsymbol{S}$ to the inversion of a scalar. The downside of this approach however is, that it neglects correlations between measurements and can lead to an overconfident estimator. We will partly address this problem when dealing with localization ambiguities in Section 5.3.

$$\boldsymbol{K} = \boldsymbol{P}_k^{-} \boldsymbol{H}^T \boldsymbol{S}^{-1} \tag{5.22}$$

$$\boldsymbol{X}_k^{+} = \boldsymbol{X}_k^{-} + \boldsymbol{K}\nu \tag{5.23}$$

$$\begin{aligned} \boldsymbol{P}_k^{+} &= \boldsymbol{P}_k^{-} - \boldsymbol{K}\boldsymbol{H}\boldsymbol{P}_k^{-} \\ &= \boldsymbol{P}_k^{-} - \boldsymbol{P}_k^{-} \boldsymbol{H}^T \boldsymbol{H} \boldsymbol{P}_k^{-} \boldsymbol{S}^{-1} \end{aligned} \tag{5.24}$$

Where the innovation ν and the innovation covariance $\boldsymbol{S}$ are given by

$$\nu = y - h\left(\boldsymbol{X}_k^{-}\right) \qquad \boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}_k^{-}\boldsymbol{H}^T + R \tag{5.25}$$

Given the sparsity of the measurement equations, it can again be useful to simplify expressions. Since the covariance matrix is symmetric, we can reduce the number of necessary matrix multiplications by writing

$$\boldsymbol{P}^{-}\boldsymbol{H}^{T}\boldsymbol{H}\boldsymbol{P}^{-} = \left(\boldsymbol{P}^{-}\boldsymbol{H}^{T}\right)\left(\boldsymbol{P}^{-}\boldsymbol{H}^{T}\right)^{T} \tag{5.26}$$

For direct measurements, we then have

$$\boldsymbol{P}^{-}\boldsymbol{H}_{D}^{T} = \begin{bmatrix} P_{0i}^{-}H_{D,i}^{T} \\ P_{1i}^{-}H_{D,i}^{T} \\ \vdots \end{bmatrix} \tag{5.27}$$

$$\boldsymbol{H}_{D}\boldsymbol{P}^{-}\boldsymbol{H}_{D}^{T} = H_{D,i}P_{ii}^{-}H_{D,i}^{T} \tag{5.28}$$

And for secondary measurements,

$$\boldsymbol{P}^{-}\boldsymbol{H}_{S}^{T} = \begin{bmatrix} P_{0i}^{-}H_{S,i}^{T} + P_{0j}^{-}H_{S,j}^{T} \\ P_{1i}^{-}H_{S,i}^{T} + P_{1j}^{-}H_{S,j}^{T} \\ \vdots \end{bmatrix} \tag{5.29}$$

$$\boldsymbol{H}_{S}\boldsymbol{P}^{-}\boldsymbol{H}_{S}^{T} = H_{S,i}P_{ii}^{-}H_{S,i}^{T} + 2H_{S,i}P_{ij}^{-}H_{S,j}^{T} + H_{S,j}P_{jj}^{-}H_{S,j}^{T} \tag{5.30}$$

These simplified expressions can then be used in the measurement update equations (5.22) to (5.24).

5.2.3. LOCALIZATION AMBIGUITIES

Recent work like the swarm in the wild utilize known initial positions to ensure a stable swarm geometry [23]. However, this is not always desirable as it takes a longer time to deploy the swarm in a new environment. Furthermore, this approach breaks down when the swarm is not fully connected at start-up and new agents are encountered throughout the mission.

The initialization time of a local agent-to-agent UWB-based localization scheme was studied in [24] - showing that initialization can take on the order of half a minute. As soon as the swarm reaches a size of three or more drones, secondary range measurements can help to reduce the time needed for a new agent's position estimate to converge. However, in our preliminary experiments we noticed that the use of secondary range measurements also adds new problems in the form of rotation and flip ambiguities. These can occur when a large number of secondary range measurements "overpower" crucial information from direct range measurements and relative velocities. This can lead to stable but wrong relative position estimates.

Let us have a quick look at how these ambiguities manifest. For this purpose, we will consider the swarm a graph, where each agent represents a node, and each communication link represents an edge. In a fully connected static swarm, the shape of this graph can be uniquely determined with the exception of translations, rotations and reflections. This even holds true for not fully connected swarms, as long as their graphs are rigid [25].

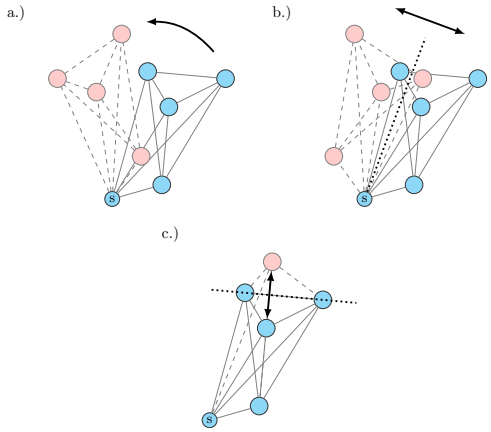


Figure 5.2: Ambiguities encountered in relative localization EKF with secondary range measurements: a.) Rotation Ambiguity, b.) Global Flip Ambiguity, c.) Local Flip Ambiguity. Correct positions are in blue, and erroneous estimates are in red.

5

To fully localize a network of interconnected nodes without ambiguities, a rigidity-based approach can be used in combination with knowledge of the positions of at least three nodes (in 2D)[26]. Since we can define the position of the localizing drone as the origin, we can at least remove translation ambiguity in our swarm. However, rotations around the origin, as well as global and local flips (c.f. Figure 5.2), were regularly encountered when testing our initial estimators.

Rotation ambiguities appear when after initialization or due to the accumulation of small errors, the relative position estimate of all agents moves on a circle around the localizing agent. With a growing number of secondary range measurements we observe that the estimator tends to become overconfident and the impact of direct range measurements (which are the only measurements linking the swarm to the localizing drone) becomes very small. Over long periods of time, rotation ambiguities can resolve by themselves, but they usually stick around for several minutes (in simulation).

Flip ambiguities generally appear immediately after initialization, as soon as secondary range measurements are used. Global flips result in the state estimates of all agents in the swarm being mirrored on an axis passing through the origin. These estimates are very stable since there is no continuous sequence of states to move to the correct solution. They only resolve to the correct solution by themselves, when all agents move onto the mirror axis, at which point the real and mirrored states coincide and the estimate can transition seamlessly.

Finally, local flips only concern a single agent and are again the result of a critical range measurement being “overpowered” by several other measurements. Due to the incremental change of the position estimate in the EKF, an update to the state of the drone pushing it out of the flip is then counteracted by multiple other range estimates pushing back as illustrated in Figure 5.3 (the same principle applies to the stability of the other ambiguities). Since fewer agents are involved in local flips, these tend to be less stable than the other ambiguities however.

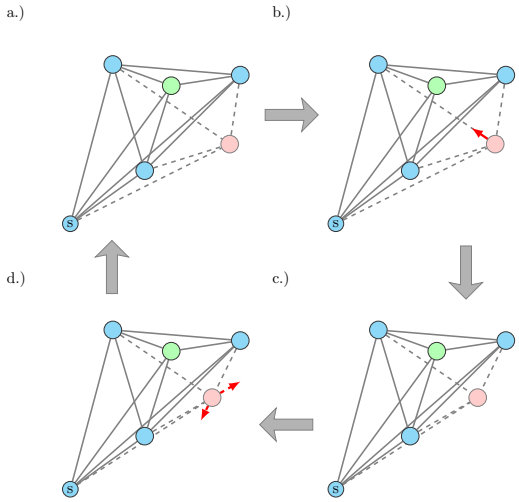


Figure 5.3: Local flips can be stable due to the incremental corrections applied by the EKF: a.) Wrong estimate (red) due to a local flip (correct location in green). b.) Beneficial correction is applied based on a large prediction error. c.) The improved estimate after the beneficial correction. d.) Multiple counteracting corrections are applied in the direction of observed prediction errors. These updates are correct, but cause the estimate to return to the previous, wrong position.

5.3. RESOLVING AMBIGUITIES

To detect cases where the estimator has converged to the correct swarm geometry, but incorrect relative positions, we can make use of the Normalized Innovation Squared (NIS). The NIS is a measure of the consistency of the estimate and can be used to detect if a new measurement y_m is at odds with the current estimate [27][28]. It is computed by normalizing the measurement innovation ν_m with the innovation covariance $\boldsymbol{S}_m$:

$$NIS_m = \nu_m^T \boldsymbol{S}_m^{-1} \nu_m \tag{5.31}$$

For a consistent estimator, the NIS should follow a χ^2 distribution with d degrees of freedom (d being the dimension of the measurement vector, in our case $d = 1$). To improve robustness, the NIS can be summed up over a window of length l, in which case that sum follows a χ^2 distribution with $d \cdot l$ degrees of freedom. A large NIS indicates that the estimator is overconfident which can be caused by rotated or flipped estimates. To avoid reacting to very noisy measurements and outliers, we track the NIS for each agent individually over a window of 20 consecutive direct-range measurements.

In the following, we present several methods to avoid ending up with these stable errors. Some of the methods aim at reducing the risk of stable errors appearing, while others aim at resolving these errors when they appear nonetheless.

5.3.1. IMPROVED INITIALIZATION

To increase the rate of convergence and reduce the risk of ending with flipped estimates, it is important to initialize each agent as close as possible to its real location. When working with static swarms where all agents are kept in the state-space at all times, it is possible to initialize agents correctly at start-up, by placing the swarm in a predefined pattern. For dynamic swarms, this approach is not sufficient. Instead, we have identified two initialization methods that use the same information that is also used by the EKF. Furthermore, we added a short stabilization period after an agent is initialized, during which its estimate is only refined using direct measurements.

MULTILATERATION BASED INITIALIZATION

If there are already several agents whose relative location is known with sufficient accuracy (based on covariance and NIS), secondary range measurements can be used for a multilateration approach [29]. In fact, two secondary range measurements and one direct range measurement are enough to calculate an initial position estimate in 2D, while in 3D a third secondary range measurement would be needed. Due to the better initialization data, we can also reduce the stabilization time with this sort of initialization.

TRAJECTORY-BASED INITIALIZATION

If there are not enough other agents localized yet, we can resort to an initialization based on the agent's relative trajectory. More specifically, we can integrate the relative movement between two direct range measurements, which leaves us with a triangle, in which the lengths of all three sides are known. Due to knowledge of the orientation of the side representing the relative movement, the agent's relative position can be narrowed down to at most two points in 2D, or a circle in 3D. This approach resembles a simplified version of the relative localization in [15], and is presented in more detail in Appendix B. This method is less accurate, more susceptible to measurement errors and rarely returns only a single point. In this case, we choose to average between two equally likely points or a circle in 3D. It is however much better than initializing the agent without any knowledge at all, especially when the agent is not too far away.

5.3.2. COVARIANCE INFLATION

Given that the consecutive application of measurement updates can move an estimate back and forth during the same update cycle (see Figure 5.3, our EKF tends to be overconfident even for correct estimates. This reduces the estimator's ability, to independently correct wrong estimates with a larger Kalman gain. For example, note that in Figure 5.3.b, a larger correction could have resolved the flip entirely. The same holds for rotation ambiguities that are often not corrected because the estimator is overconfident. To address this issue, we suggest the use of Covariance Inflation, a method used to prevent filter divergence in ensemble Kalman Filtering [30] and for state decorrelation in Simultaneous Localization and Mapping (SLAM) [31]. We follow a similar approach as Ghobadi et al. [32], but inflate the state covariance instead of the measurement covariance. In practice, we are looking to inflate the covariance of the prediction $h(\boldsymbol{X}_k)$ so that it includes the measurement y. The prediction covariance U that includes the

measurement y is calculated using the Generalized Covariance Union [33] as

$$U = \beta\left(R + vv^T\right) \tag{5.32}$$

$$\beta = \begin{cases} 2 & \text{if} \quad NIS > 1 \\ \frac{(1+\sqrt{NIS})^2}{1+NIS} & \text{if} \quad NIS \leq 1 \end{cases} \tag{5.33}$$

With one-dimensional measurement updates, both U and the original prediction covariance are scalars. We can therefore also express the covariance inflation as a multiplication with a scalar α:

$$U = \alpha \cdot \boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^T \tag{5.34}$$

Since we are only interested in inflating the state covariance while the Jacobian matrix of the measurement equation remains constant, we apply this multiplicative inflation directly to the relevant blocks of $\boldsymbol{P}^{-}$. For direct measurements,

$$P^{-}_{i,i} \rightarrow \alpha \cdot P^{-}_{i,i} \tag{5.35}$$

while for secondary range measurements

$$\begin{matrix} P^{-}_{i,i} \rightarrow \alpha \cdot P^{-}_{i,i} & P^{-}_{i,j} \rightarrow \alpha \cdot P^{-}_{i,j} \\ P^{-}_{j,i} \rightarrow \alpha \cdot P^{-}_{i,j} & P^{-}_{j,j} \rightarrow \alpha \cdot P^{-}_{j,j} \end{matrix} \tag{5.36}$$

The calculation of α itself can be performed from the original and inflated prediction covariance as

$$\alpha = \frac{U}{\boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^T} \tag{5.37}$$

Note that for secondary range measurements where the update to j is disabled, only $P^{-}_{i,i}$ is inflated (c.f. Section 5.3.3).

To improve the stability and smoothness of the estimates, we added two additional modifications: First, we only inflate the covariance when a measurement has a high NIS. Second, instead of inflating the covariance to the point where the measurement is included within a single standard deviation σ, we reduce alpha to only include the measurement within 3σ, by dividing α by 9 (the covariance is the square of standard deviation).

$$\alpha_{3\sigma} = \frac{\alpha}{9} \tag{5.38}$$

This of course only makes sense if $\alpha > 9$, which is always the case if the NIS threshold is set high enough. If $\alpha > 9$, then $\alpha_{3\sigma} < 1$, and we would actually deflate the covariance.

5.3.3. SELECTIVE USE OF SECONDARY RANGES

When an agent i exhibits a high NIS over several consecutive direct range measurements, it is likely that its position estimate is incorrect. To avoid propagation of these errors to other estimates, we should disable secondary range updates involving this agent. However, since these measurements can help us fix the wrong estimate of agent i, we choose to only partially disable these updates. Specifically, if we receive a new secondary range measurement between agent i (high NIS) and agent j (low NIS), we disable updates to j by setting $H_{S,j} = \mathbf{0}$.

5.3.4. WITHHOLD MEASUREMENTS

To directly address errors from rotation ambiguity, we propose to periodically withhold secondary range measurements. This can be done continuously independent of the observed NIS. In fact, when the estimates have converged to the correct graph, secondary range measurements only serve to deal with unobservable situations and reduce small fluctuations due to measurement errors. Withholding these measurements for several seconds (in our case for 10 s every 30 s) does not noticeably degrade a correct estimate. However, in combination with covariance inflation, direct measurements can be used during this time to correct for rotations of the graph in most position estimates. Once secondary ranges are added again, these changes are propagated even to estimates that could not be directly improved.

5.3.5. RESET AGENTS

This is the final resort for agents that continue to display extremely high values for the NIS over the window. By resetting the agent in question, the re-initialization can very likely make use of the triangulation method, improving the initial estimate, and reducing the time needed for the estimate to converge towards its true value.

5.4. IMPLEMENTATION

5.4.1. SIMULATION ENVIRONMENT

We implemented our algorithms in the "Swarmulator" simulation environment[1], which allows for the simulation of robotic swarms in 2D. We extended the swarmulator with the possibility to simulate UWB communication and implemented a relative localization EKF base class, on which all our EKF variations are based.

As of yet, we have not implemented any complex behaviors, instead, each agent simply chooses a random point in the arena as its new goal, every time its current goal is reached. Between two points, agents accelerate linearly until they reach their maximum velocity, and decelerate linearly to stop at their goal. This behavior is implemented with ground truth knowledge of the absolute position in the arena.

The complete code used for the simulations in this chapter is available on Github[2].

5.4.2. UWB RANGING

The scalability of the Ultra-Wideband ranging protocol is just as important as the scalability of the relative localization estimator itself. In this work, we base ourselves on the swarm ranging protocol developed by Shan et al.[21]. Our main modification lies in adjusting the packet format to include the additional information required in our estimator. For seamless integration with our primary test platform, the crazyflie[3], we consider the same implementation of the IEEE 802.15.4 MAC header, which is 21 Bytes long. When keeping 2 Bytes for the MAC Footer, this leaves 104 Bytes for the payload. This is respecting the 127 Byte data limit implemented on the DW1000 UWB transceiver in accordance with the IEEE 802.15.4 UWB physical layer specification [34] [35].

[1] https://github.com/coppolam/swarmulator
[2] https://github.com/tudelft/swarmulator/tree/uwb_swarm_with_compass
[3] https://www.bitcraze.io/products/crazyflie-2-1/

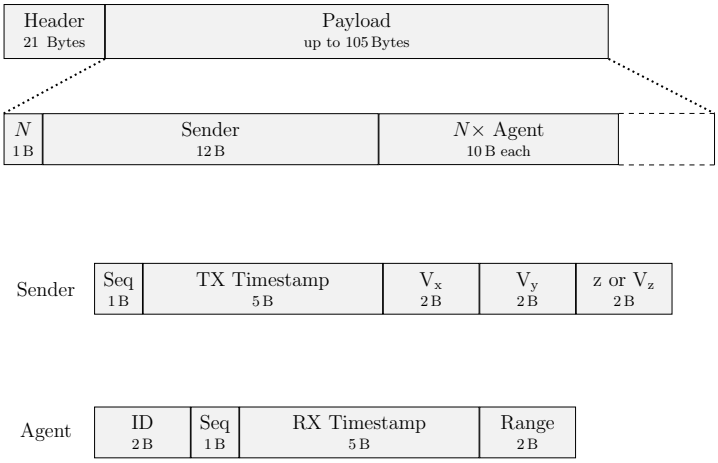


Figure 5.4: Packet structure and block sizes for the adapted swarm ranging protocol (1 B = 1 Byte)

The available payload is split into three parts as shown in Figure 5.4: One byte to communicate the number of "Agent" blocks in the message, then a block with information from the sender, and finally up to 9 "Agent" blocks. The "Sender" Block contains information from the sender s: An 8-bit sequence number, the transmission timestamp of its previous message, and the sender's global velocity estimates (2 Bytes per dimension). For a 2D implementation such as in our case, the z dimension is not necessary, however, when working with drones in 3D, it would be sensible to at least include an estimate of the current altitude in a real-world implementation. The sender does not need to include its ID in the "Sender" block since it is already communicated in the header. Subsequently, each "Agent" Block contains information that the sender collected about another agent i, which is identified by its 16-bit ID. The block then includes the sequence number and reception timestamp of the last message that s received from i. Finally, if s was able to calculate a range measurement to i since its last communication, this measurement is also included to serve as a secondary range measurement for other drones. Both range and velocity measurements can be communicated with 16-bit integers in mm or mm/s, as the uncertainty of these numbers does not require more accuracy.

In our current implementation, swarm-ranging pings are transmitted at regular intervals, where each drone randomizes its transmission interval around a base value of 0.05 s. Since the DW1000 chip implements the ALOHA protocol, no additional care needs to be taken about medium access control. By tracking the number and size of the messages sent and received by an individual agent, we can estimate the air utilization a in the agent's vicinity during a period T to ensure it stays below the 18% required for ALOHA to work reliably [35].

$$a_T = \frac{1}{T} \sum_{i=0}^{N_T} \left(t_{PHY} + \frac{b_i}{R} \right) \tag{5.39}$$

Here, t_{PHY} is the duration of the physical layer synchronization header (Preamble, start of frame delimiter and physical header) in seconds, b_i is the payload size of message i in

bits and R is the data rate in bits per second.

5.4.3. DYNAMIC EKF

In dynamic swarms, we can't know in advance how many and which drones will be in our communication range at any given moment. In principle, there are however only two events that we need to be able to deal with: New agents entering the communication range, and known agents leaving it. To limit computational cost in very large swarms, it furthermore makes sense to put a limit on the agents that can be tracked at the same time. This means that we have to implement a flexible state space, which allows adding, removing and swapping agents.

By breaking up the state and input vector, as well as the covariance matrix into blocks, we already created the flexibility to easily access and manipulate the data for individual agents. Furthermore, if there are not enough agents in range to fill all slots, the blocks corresponding to empty slots can simply be skipped in the prediction and correction steps.

When messages are received from a new agent, the new agent can be assigned one of a limited number of slots in the EKF memory. By saving the agent's unique communication ID in that same slot, incoming inputs and measurements can be easily matched to the slot index i, which is used to identify the agent in the EKF's state space. The maximum number of slots can be set according to the processing power of the system, as it determines the maximum dimension of the state space. Other relevant information such as the last time an agent was seen, or the last recorded range to the target, can be added to the same slot.

An important consideration is how to deal with new agents if all slots are already used. To add new agents, we use the simple priority system, also shown in Fig. 5.5:

1. *Empty Slot:* Use the first empty slot encountered

2. *Timeout:* Use the slot of the agent that has been timed out for the longest. Agents time out after no new message has been received for a specified timeout period.

3. *Range:* Use the slot of the furthest away agent. Only swap in the new agent if it is sufficiently closer compared to the agent that is swapped out (Hysteresis).

When an agent leaves the communication range without another agent entering to replace it, it can make sense to keep a loose estimate of that agent's position in case it enters the communication range again. This can be achieved by setting the agent's input velocities to zero while increasing the input uncertainty to the maximum velocity the agent can reach. The uncertainty of this agent's position estimate will degrade quickly, but the estimate can still be updated with secondary range measurements to other agents that are still in communication range.

5.4.4. REFERENCE ESTIMATORS

For comparison with the final version of our dynamic EKF, we implemented three other similar estimators. When comparing the estimators, all estimators are run at the same time on each agent, i.e. they receive exactly the same inputs at the same time (with the

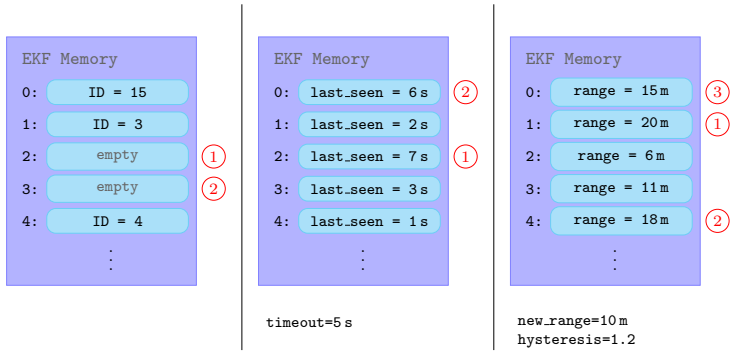


Figure 5.5: Slot priority for new agents (red). Empty slots are always prioritized (left) before timeouts are considered (middle). If no empty slots are available and no slots have timed out, the last available range is considered after being multiplied by a hysteresis factor, to avoid constant switching (right)

exception of the centralized EKF, which receives additional inputs from agents that are out of communication range).

5

CENTRALIZED EKF

While still running on each drone individually, this estimator represents a centralized approach as it has access to all information collected by any agent in the simulation. It uses a full state vector to track all other agents at all times while receiving all velocities and range measurements calculated, independent of communication range limitations. Furthermore, agents are initialized at their true relative positions at startup, meaning that no time is needed for the filter to converge. The centralized EKF represents a best-case scenario in terms of accuracy, but a worst-case scenario in terms of computational cost (each additional agent adds up to N additional range measurements and increases the size of the state vector and covariance matrix).

DECOUPLED EKF

The decoupled EKF does not use secondary range measurements and neglects any coupling between different agents' relative positions. It is therefore functionally equivalent to running a filter bank with an independent EKF for each individual agent, such as we did in Chapter 4. As a result, the number of states in each EKF is constant and the computational cost scales much slower due to the smaller matrices being multiplied. Furthermore, only one additional range measurement is added by any new agent, and agents can leave the communication range. Since the decoupled EKF does not suffer from local minima, we also stick with the "lazy" initialization used in Chapter 4 (all agents initialized at the origin).

FULL-STATE EKF

The full-state EKF implements all the improvements we developed for the dynamic EKF, but still uses a full-state vector (including all other agents). This means using trajectory- or multilateration-based initialization for new agents, as well as covariance inflation, NIS-based resets and temporarily disabling secondary range measurement to deal with

localization ambiguities. It can therefore be seen as a variation of the centralized EKF that only has access to limited information but is still suffering from the same scalability limitations. It is however expected to perform better than our proposed dynamic EKF, since it can track all other agents at the same time.

5.5. RESULTS

To evaluate the different estimators, we run each simulation five times for 300 s each, while logging the relevant performance indicators.

For a fair comparison of the full state estimators with the dynamic EKF, which only tracks a subset of agents, we compare the average performance for the 1, 3 and 5 closest agents (according to ground truth). We call these C1, C3 and C5 errors. We also look at the performance of all tracked agents in the communication range (ICR). For the ICR error, the number of agents that contribute to the average depends on the estimator and the current state of the simulation. While ICR does not take into account agents for which there is no current estimate, the other errors will be marked as 'No estimate' if there is no valid estimate for all required agents (e.g. if there is no estimate for the 4^{th} closest agent, C5 will be invalid but C1 and C3 will still be computed correctly). This situation can appear either when an agent is not tracked, or if the estimate is still in its initial stabilization period.

When presenting the error distribution, we aggregate the data of all drones from all relevant simulations, ignoring the first 60 s to allow the estimates to converge. This duration was chosen from several preliminary experiments to ensure that all estimators had time to converge to reach an initial stable estimate. We show an example of the differences in initial convergence in Section 5.5.3, Figure 5.12. Since we found that errors depend on the distance between agents (which can vary a lot, even for the closest agents), we calculate the relative error by normalizing the absolute error with said distance.

$$e_{si,\text{rel}} = \frac{e_{si,\text{abs}}}{\|\boldsymbol{p}_{si}\|} \tag{5.40}$$

5.5.1. RESOLVING AMBIGUITIES

In Section 5.3, we introduced a multitude of methods to detect and resolve cases in which the estimator converges to a rotated or flipped estimate. We performed an ablation study to determine the effect of each of those methods on the final results in both the full-state EKF (Figure 5.6) and the proposed dynamic EKF (Figure 5.7) in a static swarm. The figures show the distribution of the relative errors for all tracked agents with a valid estimate. We find that when applying each method individually, only the improved initialization (A) and covariance inflation (B) methods lead to a better result in at least one of the estimators. Selective use of secondary ranges (C), withholding measurements (D) and resetting agents (E) rely on these first two methods to function properly and were thus only evaluated in combination.

It can be seen that the combined use of all our suggested methods results in a large improvement of localization performance. In the Full-state EKF, the fraction of errors above 10% is reduced by about 40 percentage points moving from using none of the methods to using all methods, while in the dynamic case, the difference is even bigger at

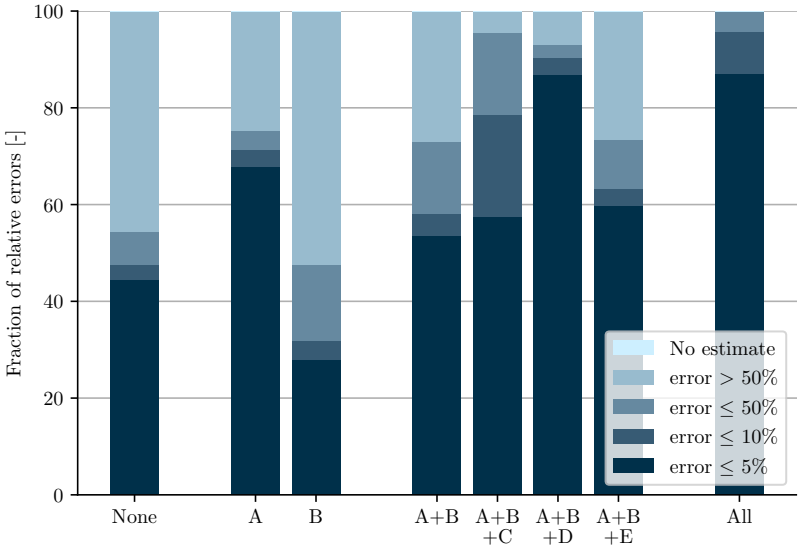


Figure 5.6: Performance of different methods for resolving ambiguities in the *Full State EKF* when tracking 10 drones in a static swarm (10 drones total): (A) Improved initialization, (B) Covariance inflation, (C) Selective use of secondary ranges, (D) Withhold measurements, (E) Reset agents

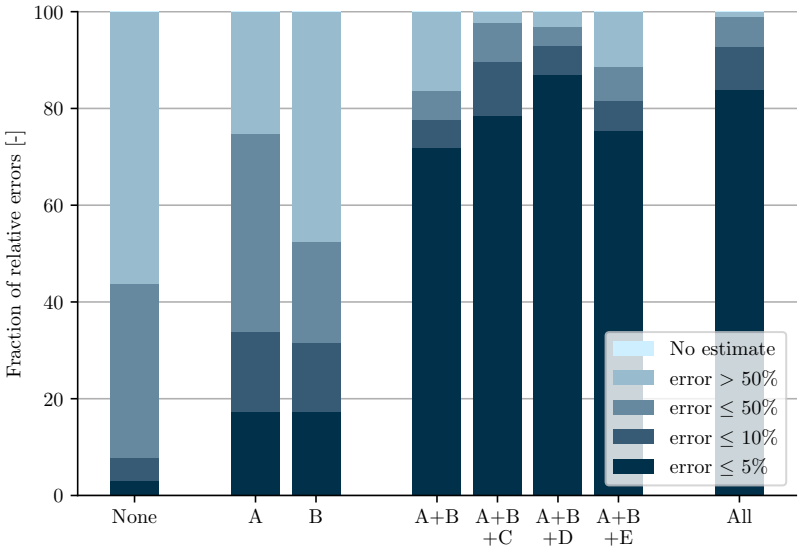


Figure 5.7: Performance of different methods for resolving ambiguities in the *Dynamic EKF* when tracking 10 out of 20 other agents (21 drones total): (A) Improved initialization, (B) Covariance inflation, (C) Selective use of secondary ranges, (D) Withhold measurements, (E) Reset agents

almost 90 percentage points.

There are significant differences in the effect of individual methods between the Full State and Dynamic EKF. Specifically, the effect of individual methods seems much higher in the Dynamic EKF than in the Full State EKF. This difference likely comes from the fact that new agents are added to the Dynamic EKF all the time. In the Full State EKF, it is very important to avoid any problems in the short initialization period at the start, while in the Dynamic EKF, every new agent is another opportunity to resolve issues in the estimates.

While one could argue that some methods are not needed in Dynamic EKF, it is important to remember that the Full-state EKF represents a special case of the Dynamic EKF, in which the same 10 agents are in communication range for long periods of time. To maximize robustness, we therefore argue that all methods should be active at all times.

5.5.2. COMPUTATIONAL COST AND AIR UTILIZATION

Let's now look at the scalability of the relative localization estimators. In the first step, we evaluate how the computational cost of the different estimators varies for different swarm sizes. We only evaluate the time each estimator takes to execute a single estimation step consisting of prediction and measurement updates. This evaluation is more qualitative than quantitative since the exact numbers will not only depend on the hardware used but also on the computational load from other processes. In our case, we performed the evaluation on a Dell XPS-15-7590 with 6 Intel Core i7-9750H processors (12 threads). For this simulation setup, we limited the number of drones being tracked in the dynamic EKF to $n = 10$. In this evaluation, we look at a static swarming setup by setting the communication range to a value larger than the arena size. As could be expected, the computational load of the centralized and full-state EKF initially grows exponentially, making them impossible to use for large swarms. It flattens out for larger swarms, where the limiting factor becomes the number of range measurements that can be performed by the UWB protocol (only 9 "Agent" blocks can be included at a time). The computation time for the decoupled EKF on the other hand grows only linearly, and the dynamic EKF reaches its peak around its number of tracked agents n. The drop for higher numbers is the result of fewer measurements being processed since more estimates are in their stabilization period at the same time (meaning that their secondary range measurements are not processed).

This simulation setup also lends itself well to an investigation into the load of the UWB channel. Although we do not simulate the physical UWB transmission process, we can make some assumptions on the UWB settings to estimate the air utilization. Table 5.1 shows the settings that were chosen for good ranging performance with short transmission ranges [35].

The transmission time of the physical header t_{PHY} is then calculated as

$$t_{\text{PHY}} = (L_{\text{PMB}} + L_{\text{SFD}}) \cdot t_s + \frac{b_{\text{PHR}}}{R} = 141.2\,\mu\text{s} \tag{5.41}$$

where $t_s = 1.1017\,\mu$s is the preamble symbol duration (PSD) for a PRF of 64 MHz. Using this value, each agent in simulation can keep track of the air utilization at its location,

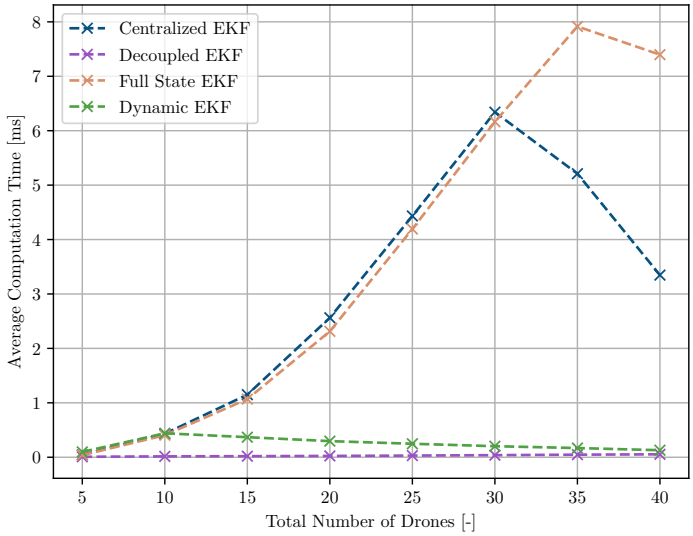


Figure 5.8: Average computation cost of a single EKF step for drones in a static swarm on our Dell XPS-15-7590 with 6 Intel Core i7-9750H Processors

Pulse Repetition Frequency (PRF)	64 MHz
Data Rate R	6.8 Mbps
Preamble Length L_{PMB}	128 Symbols
Start of Frame Delimiter (SFD) L_{SFD}	8 Symbols
Physical Header Length b_{PHR}	19 bits

Table 5.1: Ultra-Wideband Settings for calculating air utilization

using the size of transmitted and received messages with equation 5.39. Based on the observed values, we conclude that the ALOHA protocol should work well in settings with up to 35 drones in the communication range using the current transmission interval of 0.05 s. Since the communication range can be easily varied between 0 m and (approximately) 60 m by adjusting the transmission power [35], this allows for the design of extremely dense dynamic swarms.

5.5.3. LOCALIZATION ACCURACY

Having shown that the dynamic EKF offers significant advantages in terms of computational cost compared to a centralized or full-state approach, it is now time to also compare the accuracy of their localization accuracy. Comparing the performance in static (Figure 5.10) and dynamic (Figure 5.11) swarms, we can see that the Dynamic EKF is the only estimator that is not impacted by agents entering or leaving the swarm. Furthermore, we see that the Dynamic EKF performs at a comparable level for the closest

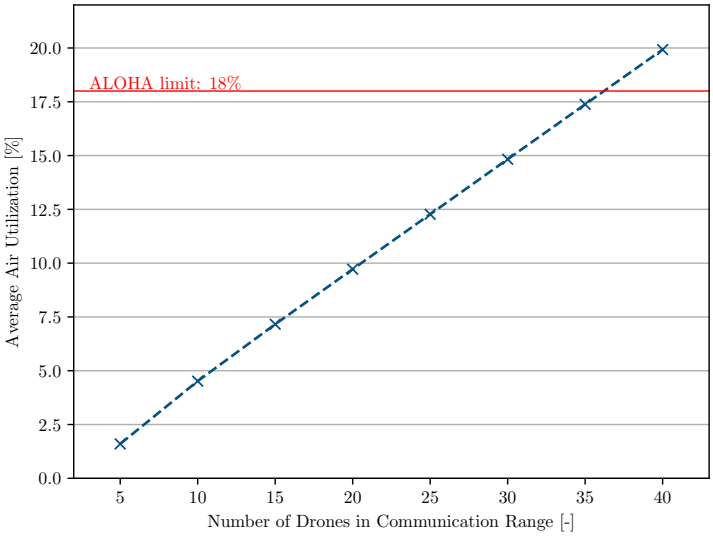


Figure 5.9: Air utilization of the modified swarm ranging protocol compared to the recommended limit of 18% when using ALOHA for media access control

drones, even in static swarms.

Especially noteworthy is the comparison between the Dynamic EKF and the Decoupled EKF, which are the only practical options in large swarms due to computational power requirements. It is clear that our proposed Dynamic EKF performs substantially better in terms of localization errors. Furthermore, although the Decoupled EKF has access to all estimates at all times, we argue that not having an estimate is not worse than having a high error. On the contrary, a missing estimate can be used to command a more careful behaviour, while wrong estimates could lead to dangerous maneuvers.

Looking at the effect of reducing the state-space from tracking all agents (Full-state EKF) to only tracking a select few (Dynamic EKF), we note that for the closest drone (C1) the difference is barely noticeable. Looking at the dynamic swarm, there is even a similar C3 performance. Only as we look at even more agents (C5) do the differences become clear. This is a trade-off that has been expected, given that agents are constantly being swapped in an out of the estimator.

Finally, looking at the evolution of the state estimates on a single drone throughout a simulation in Figure 5.12, we can verify that the use of secondary range measurements indeed results in a significant reduction of the convergence time for the estimates. In the same simulation run, we can also have a look at the qualitative evolution of the relative position estimate shown in Figure 5.13. For clarity, we isolate the data from a 30 s interval between 150 s and 180 s. During this period, we observe very good tracking by all estimators, with exception for the decoupled estimate for agent 6, which is still converging at this point. We also see, that the dynamic EKF is correctly tracking the five closest agents,

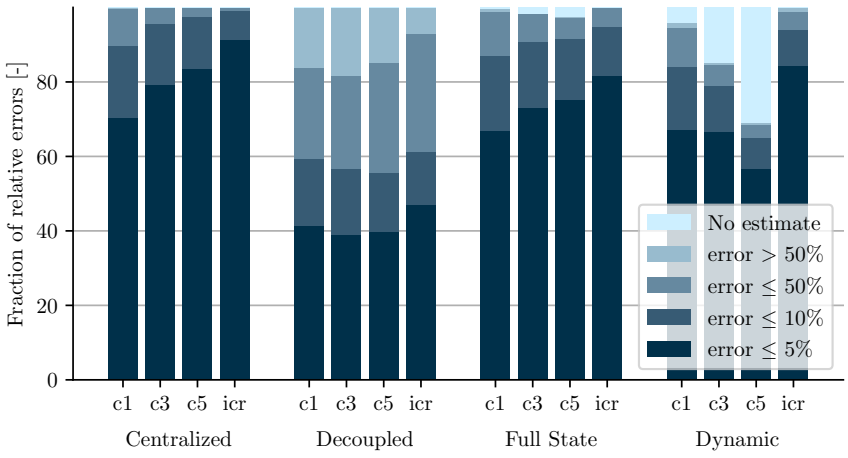


Figure 5.10: Relative error distribution for different EKF variants in a 21-agent swarm with *infinite communication range*

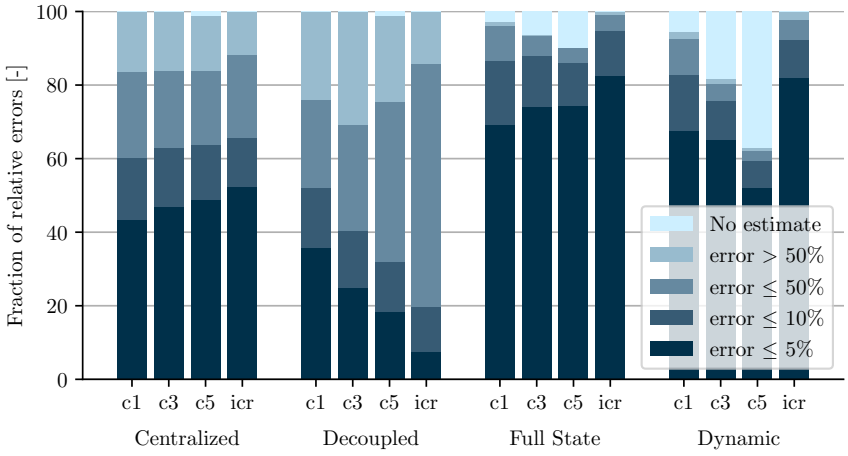


Figure 5.11: Relative error distribution for different EKF variants in a 21-agent swarm with *limited communication range*

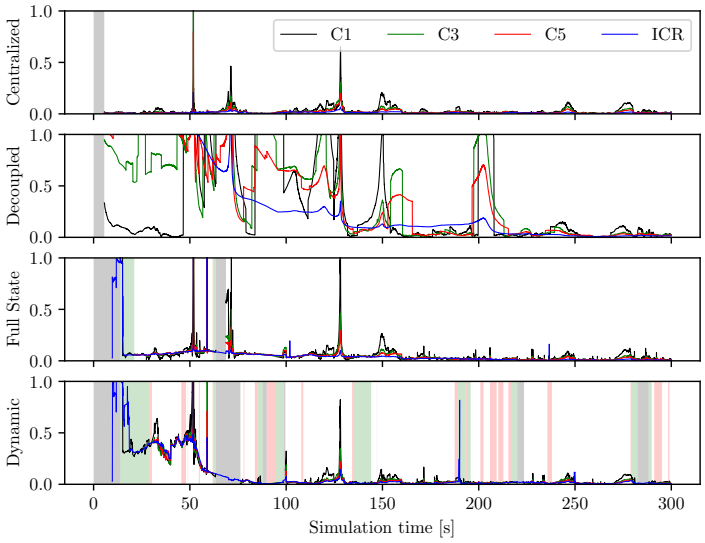


Figure 5.12: Relative error evolution through time for different EKF variants in a 20-agent swarm with infinite communication range. Shaded regions represent times without valid estimates for the given error metric.

while the tracking of agent 8 was terminated as it moved further away. At this point, we can expect the initialization of agent 3 to be well underway, as it is approaching the tracking agent.

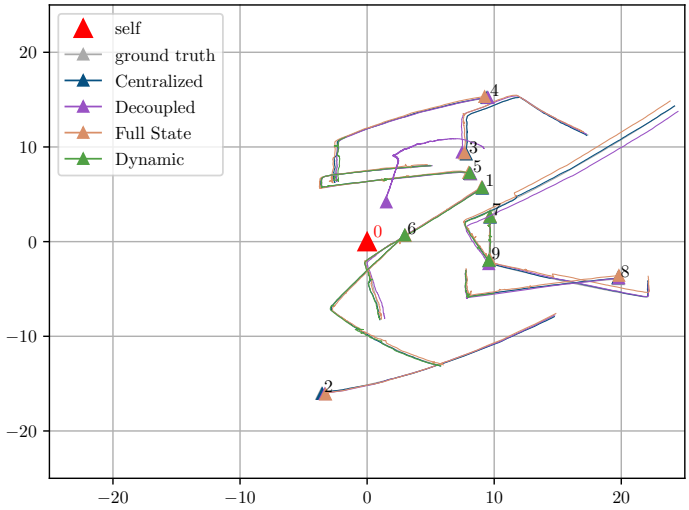


Figure 5.13: Relative motion of the agents in 2D for the time between 150 s and 180 s in the same simulation run as Figure 5.12

5.6. CONCLUSION

In this chapter, we presented and evaluated a relative localization algorithm for large, dynamic swarms. By adding secondary range measurements, we managed to significantly reduce the convergence rate and improve accuracy when compared to simple agent-to-agent schemes. Furthermore, our Dynamic EKF offers clear advantages in terms of computational cost, while still providing a high level of accuracy when compared to a centralized estimator tracking all agents in the swarm.

During the design of our estimator, we observed how the use of secondary range measurement could lead to the manifestation of localization ambiguities. We proposed an approach based on tracking the Normalized Innovation Squared (NIS) of direct range measurement updates to identify and correct situations in which localization ambiguities led to stable but wrong position estimates. We specifically introduced improved initialization methods and covariance inflation, while also restricting the use of secondary range measurements. We showed that these methods help significantly improve the performance of the estimators using secondary range measurements.

While only evaluating the estimators in simulation, we tried to be as close to the physical system during implementation to facilitate future tests on actual drones. The only addition we would recommend is tracking the air utilization and adjusting the communication range (transmission power) and transmission frequency in real-time, to optimize the use of the communication channel. Also, the effective maximum number of agents that are tracked at a time should of course be adapted to the capabilities of the physical system.

REFERENCES

[1] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, *State-of-the-art technologies for UAV inspections,* IET Radar, Sonar and Navigation **12**, 151 (2018).

[2] U. R. Mogili and B. B. V. L. Deepak, *Review on application of drone systems in precision agriculture,* Procedia Computer Science **133**, 502 (2018), international Conference on Robotics and Smart Manufacturing (RoSMa2018).

[3] S. M. S. Mohd Daud, M. Y. P. Mohd Yusof, C. C. Heo, L. S. Khoo, M. K. Chainchel Singh, M. S. Mahmood, and H. Nawawi, *Applications of drone in disaster management: A scoping review,* Science & Justice **62**, 30 (2022).

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, *Swarm robotics: a review from the swarm engineering perspective,* Swarm Intelligence **7**, 1 (2013).

[5] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, *A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints,* Frontiers in Robotics and AI **7** (2020), 10.3389/frobt.2020.00018.

[6] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, *Optimized flocking of autonomous drones in confined environments,* Science Robotics **3**, 1 (2018).

[7] L. Mainetti, L. Patrono, and I. Sergi, *A survey on indoor positioning systems,* 2014 22nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2014 , 111 (2014).

[8] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, *Towards a swarm of agile micro quadrotors,* Autonomous Robots **35**, 287 (2013).

[9] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, *Crazyswarm: A large nanoquadcopter swarm,* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017) pp. 3299–3304.

[10] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, *System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization,* Autonomous Robots **41**, 919 (2017).

[11] S. Li, C. De Wagter, and G. C. H. E. de Croon, *Self-supervised monocular multirobot relative localization with efficient deep neural networks,* in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022) pp. 9689–9695.

[12] T. Stirling, J. Roberts, J.-C. Zufferey, and D. Floreano, *Indoor navigation with a swarm of flying robots,* in *2012 IEEE International Conference on Robotics and Automation* (2012) pp. 4641–4647.

[13] J. Pugh and A. Martinoli, *Relative localization and communication module for small-scale multi-robot systems,* in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (2006) pp. 188–193.

[14] M. Coppola, K. N. McGuire, K. Y. Scheper, and G. C. H. E. de Croon, *On-board communication-based relative localization for collision avoidance in micro air vehicle teams,* Autonomous robots **42**, 1787 (2018).

[15] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, *Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments,* International Journal of Micro Air Vehicles **9**, 169 (2017).

[16] S. Van Der Helm, M. Coppola, K. N. McGuire, and G. C. H. E. de Croon, *On-board range-based relative localization for micro air vehicles in indoor leader–follower flight,* Autonomous Robots **44**, 415 (2020).

[17] V. Brunacci, A. De Angelis, G. Costante, and P. Carbone, *Development and analysis of a uwb relative localization system,* IEEE Transactions on Instrumentation and Measurement **72**, 1 (2023).

[18] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz, *Localization from mere connectivity,* in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing* (2003) pp. 201–212.

[19] S. Li, *Autonomous Swarms of Tiny Flying Robots*, Ph.D. thesis, Technische Universiteit Delft (2021).

[20] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, *Decentralized formation pose estimation for spacecraft swarms,* Advances in Space Research **67**, 3527 (2021).

[21] F. Shan, H. Huo, J. Zeng, Z. Li, W. Wu, and J. Luo, *Ultra-wideband swarm ranging protocol for dynamic and dense networks,* IEEE/ACM Transactions on Networking **30**, 2834 (2022).

[22] S. Pfeiffer, V. Munaro, S. Li, A. Rizzo, and G. C. H. E. de Croon, *Three-dimensional relative localization and synchronized movement with wireless ranging,* Swarm Intelligence **17**, 147 (2023).

[23] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, *Swarm of micro flying robots in the wild,* Science Robotics **7**, eabm5954 (2022).

[24] S. Li, M. Coppola, C. De Wagter, and G. C. H. E. de Croon, *An autonomous swarm of micro flying robots with range-based relative localization,* arXiv preprint arXiv:2003.05853 (2020).

[25] B. Hendrickson, *Conditions for unique graph realizations,* SIAM Journal on Computing **21**, 65 (1992).

[26] T. Eren, O. Goldenberg, W. Whiteley, Y. Yang, A. Morse, B. Anderson, and P. Belhumeur, *Rigidity, computation, and randomization in network localization,* in *IEEE INFOCOM 2004*, Vol. 4 (2004) pp. 2673–2684 vol.4.

[27] S. E. Houts, S. G. Dektor, and S. M. Rock, *A robust framework for failure detection and recovery for terrain-relative navigation,* in *Proceedings of the Unmanned Untethered Submersible Technology Conference* (2013).

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

[29] A. Norrdine, *An Algebraic Solution to the Multilateration Problem,* in *International Conference on Indoor Positioning and Indoor Navigation* (2012).

[30] H. Li, E. Kalnay, and T. Miyoshi, *Simultaneous estimation of covariance inflation and observation errors within an ensemble kalman filter,* Quarterly Journal of the Royal Meteorological Society **135**, 523 (2009).

[31] J. Guivant and E. Nebot, *Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms,* IEEE Transactions on Robotics and Automation **19**, 749 (2003).

[32] M. Ghobadi, P. Singla, and E. T. Esfahani, *Robust attitude estimation from uncertain observations of inertial sensors using covariance inflated multiplicative extended kalman filter,* IEEE Transactions on Instrumentation and Measurement **67**, 209 (2018).

[33] S. Reece and S. Roberts, *Generalised covariance union: A unified approach to hypothesis merging in tracking,* IEEE Transactions on Aerospace and Electronic Systems **46**, 207 (2010).

[34] *IEEE Standard for Low-Rate Wireless Networks,* IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015) , 1 (2020).

[35] *DW1000 User Manual*, Decawave Ltd (2017), version 2.18.

6

CONCLUSION

6.1. ANSWERING THE RESEARCH QUESTIONS

So after four years of research and four chapters filled with results, where do we stand? While our dream of autonomous drones searching for victims in a collapsed building still seems out of reach, it is not a bad moment to look back to our initial objectives and summarize our achievements. After all, autonomous search and rescue is a very challenging task, and we never set out to solve all aspects of it. Our primary research objective was much more specific and focused on localization with Ultra-Wideband ranging. Can we say that we achieved this objective?

Research Objective

Provide robotic swarms with scalable means to determine their position in the environment, and with respect to each other, using Ultra-Wideband ranging.

Our first research question pertained to the use of Ultra-Wideband ranging for absolute localization. Specifically, we were interested in the compatibility with Moving Horizon Estimation and its use on computationally limited platforms:

Research Question 1

How can Moving Horizon Estimation be used to improve localization with Ultra-Wideband on platforms with limited computational resources?

In Chapters 2 and 3, we described computationally efficient approaches to Moving Horizon Estimation, that can be used to fuse UWB and IMU measurements for accurate position estimation on computationally limited platforms. Given the need to install UWB anchors in the area of operation before deployment of the drones, this method for absolute localization is mainly suited to applications in industrial environments (and

not disaster zones). One key observation however, the better resilience of Moving Horizon Estimation to unknown or non-Gaussian noise, could be of great use in complex and unstructured environments too.

With our second research question, we wanted to guide our investigation into the use of UWB ranging for swarming applications. Of specific interest at this stage of our research was not so much the size of the swarm, but the mechanisms that allowed relative localization in three dimensions and coordination between individual agents.

Research Question 2

How can Ultra-Wideband be used to achieve spatial coordination in small groups of aerial robots?

With our work in Chapter 4, we managed to demonstrate a working, three-dimensional relative localization scheme in the real world. The results showed us, that UWB was well-suited for this type of application, but that there were several severe shortcomings of the current approach, mainly related to the scalability of the method. This inspired us to set out onto our final research question, with the intent of developing a scalable scheme for relative localization that is not impacted by the size of the swarm.

Research Question 3

How can relative localization be achieved in potentially infinite swarms, despite physical limitations of the individual agents and their communication channel?

Our investigation led us to the development of a dynamic relative localization EKF, that we tested extensively in simulation on large, dynamic swarms. By limiting the number of tracked drones and ensuring seamless switching between them, our previous results can be easily extended to potentially infinite swarms. With this work, we covered the final aspects of our research objective, albeit for this last step, for now, only in simulation.

6.2. PUSHING FURTHER

While I think it is fair to say that we accomplished our research objective, this is of course not the end of the road. The algorithms we developed and presented in the scope of this research still leave room for improvement and there are many alternative approaches to these problems that might warrant a closer look. During the course of my research at TU Delft, I have had the pleasure to work with several bright and enthusiastic Master students on some of these topics, that should not go unmentioned.

Starting with the research on absolute localization, it is important to realize that with the miniaturization of cameras, vision has become a very promising technology to use for this purpose. While the computational complexity of Simultaneous Localization and Mapping (SLAM) algorithms is too large for use on smaller platforms, there is potential for Visual-Inertial Odometry (VIO). Looking at the advantages and disadvantages of VIO over UWB-based methods, there is an interesting case to be made for combining

these two methods: VIO does not require any modification of the environment, but experiences drift. This drift could be corrected using strategically placed UWB beacons, or conversely, using VIO could help in reducing the number of beacons needed for successful navigation.

Under my supervision, Stavrow Bahnam started investigations into improving state-of-the-art stereo-VIO methods, and achieved impressive results. Not only did he manage to significantly cut the computational cost, but he also improved accuracy in cases with a low number of features [1]. I am proud to say that after successful publication of his work, Stavrow has joined our Lab as a PhD candidate to continue his research into visual navigation.

Another strategy to reduce the required number of beacons relies on the idea of extracting more information from each UWB message. Specifically, angle-of-arrival information could provide extremely useful. Since specialized hardware that makes use of two antennas to determine the phase-difference of arrival in the UWB signal is expensive, inflexible and hard to come by, Bas van Beurden and I decided to look into an alternative option. By combining two spatially separated UWB receivers, Bas tried to use the two different position estimates to geometrically determine the angle of arrival from the signal. Unfortunately, despite his hard work, the noise on the UWB measurements required the two receivers to be too far apart for use on small drones. We therefore concluded, that this method might only find successful use on larger fixed-wing UAVs, where the beacons could be placed at least one meter apart.

Joining multiple drones to form a swarm offers new possibilities, also in terms of absolute localization. Together with Fréderic Dupon, we looked at how being in a swarm could improve individual agent's absolute position estimates. By ranging to the other drones, Fréderic managed to significantly improve absolute localization in scenarios where large parts of the operational area were not covered by the stationary UWB beacons. This was done by treating other drones as mobile beacons with uncertain positions.

On a more theoretical side, I worked with Changrui Liu, who delved deep into the foundations of Kalman Filtering to find other ways to deal with the non-Gaussian noise present in UWB ranging. Changrui looked at a modified Kalman filter that uses correntropy instead of the mean-squared error as an error metric. Using a newly developed kernel function, he designed a filter for relative localization that provides more robustness against non-Gaussian measurement noise [2].

Finally, it should be mentioned that these algorithms have for the most part only been tested in a scientific environment. The challenges that can arise from deploying such a system in a real-world application are numerous and should not be underestimated, as we have learned from our cooperation with Royal Brinkman. As a manufacturer and provider of greenhouse equipment, Royal Brinkman has been looking at better ways of detecting pests and diseases on crops growing inside of greenhouses. By equipping autonomous drones with cameras, they hope to identify affected plants early during an infection, giving them the ability to react more quickly. During this cooperation, we tested multiple types of drones, cameras and localization algorithms in Royal Brinkman's greenhouses. Together with the drone start-up Mapture, we supported the installation of a prototype system that can be controlled and monitored through a web interface. The

drone of this system was able to take off from its box, fly autonomously over flower fields in a greenhouse with the help of UWB, and land in the box again. Images made during the flight were then automatically uploaded to an online database.

6.3. WHAT COMES NEXT

Although I have already provided several suggestions for continuing the presented research in the individual chapters, I would like to take this final opportunity to put all of it into a bigger context. While pure UWB-based systems will likely find some use in structured indoor environments, I believe that in the future, UWB will mostly be used in combination with other technologies. These other technologies are often required for different mission aspects, and UWB offers a light-weight and cheap option to cover their shortcomings.

In many real-world scenario, drones must be able to navigate in an ever-changing environment that presents unknown and potentially even dynamic obstacles. For several years now, vision has been the method of choice for performing obstacle detection and avoidance. As cameras are getting smaller, micro-processors are becoming more powerful, allowing image processing even on small drones. It is therefore reasonable to assume, that almost all drones will carry the necessary hardware to perform visual navigation in the near future.

However, vision-based systems are severely impacted by adverse visual conditions such as darkness or fog, while the need for energy-efficient, low-interference communication is a great reason to still carry an UWB transceiver on any drone. Furthermore, there is also great potential in combining these two technologies outright for precise and efficient localization solutions: For absolute localization, VIO can help significantly reduce the need for static UWB beacons, while the remaining UWB beacons can be used to counter the drift that is inherent to odometry-based navigation. For relative localization, the range information received through UWB can be complemented with bearing information from a front-facing camera, allowing better accuracy in the cameras field-of-view, while still tracking all other agents using the range measurements.

Even for outdoor applications, UWB has the potential to complement GPS-based solutions: While GPS is very reliable over open fields, its estimates can degrade severely in built environments. While an accuracy of a few meters might still be sufficient for absolute navigation, UWB can be used to provide accurate, relative position estimates needed to maintain dense formations in swarms.

In this context, I think the main focus for future research on Ultra-Wideband localization should be on how it can best be integrated with other localization technologies. Furthermore, I believe that the results from Chapter 5 should be followed up upon, not just in the context of relative localization, but also in terms of how they can support absolute localization of swarms.

Finally, let's not forget that localization itself is not sufficient to make a drone truly autonomous. In Chapter 4 we have seen the impact that "closing the loop" (i.e. using the position estimate as an input for control) can have on the localization performance. There are many different real-world applications that might affect the accuracy of the presented algorithms in unexpected ways. The only way to find out how they perform in a fully autonomous system, is using them in a fully autonomous system.

REFERENCES

[1] S. Bahnam, S. Pfeiffer, and G. C. H. E. de Croon, *Stereo visual inertial odometry for robots with limited computational resources,* in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021) pp. 9154–9159.

[2] C. Liu, S. Pfeiffer, and G. C. H. E. de Croon, *Cooperative relative localization in mav swarms with ultra-wideband ranging,* arXiv preprint arXiv:2405.18234 (2024).

ACKNOWLEDGEMENTS

When I moved to Delft in late 2019, I was looking forward to a challenging, but exciting four years of research on autonomous, swarming drones. Five years later, I am now finally ready to defend my work and move towards the next chapter of my life. Writing a dissertation and working towards a PhD is always a long process with many ups and downs, but for me (and many other PhD candidates starting around the same time) there was an unforeseen, additional challenge. Less than six months into my research, the COVID-19 pandemic hit the world and turned the subsequent two years into an emotional and psychological roller-coaster. Getting through long periods of working from home without much social interaction, pushing forward when I was stuck on my research, and dealing with difficult personal situations - I'm not sure I could have handled it without the many people that were there to support me.

First of all, I want to thank Guido for his support and guidance throughout these five years. A few weeks after we were forced to start working from home, you gave me a call on my phone (which was unusual), and asked how it was going. When I proceeded to tell you what I had been working on, you stopped me and said that you weren't concerned about my work, but wanted to know how I was holding up personally. In that moment I realized how lucky I was to have you as my promotor.

Of course there was another person on my supervisory team, who was just as committed to helping me succeed and that was Christophe, a big thanks also to you. Your technical insights were in many cases what made the difference in turning a nice theoretical result into a working experiment.

A big thanks also the my parents, Udo and Barbara, without whom I most certainly wouldn't be where I am today. You have endured my scientific enthusiasm since I have been able to talk, and have encouraged me to pursue my dreams ever since. Talking to you about my problems, scientific or not, has always helped me to see things in a different light, and it has kept me going.

Nils, living in the Netherlands has not always made it easy to stay in touch with you, but I'm glad we are managing it well so far. You are the one person I can truly talk to about everything as well as having a blast and getting my mind off things. Thank you for being a fantastic brother.

I also want to thank Iris for dancing with me when Erin initially pushed me towards her, and for staying with me ever since. Your unconditional support and encouragement, have given me energy when I was feeling exhausted and desperate, and your texts and stickers got me through many difficult days.

Dancing was a big part of my social life here in Delft and I want to thank all current and past members of Blue Suede Shoes for being an awesome group of people that provided me with a place to socialize, let off some steam, and express myself. A special thanks goes to Remy, for hosting online technique lessons during COVID lock-downs, as well as to my dance partners Luna, Lucy, and of course, Iris.

Last but not least, a big thanks to the whole MAVLab team and all of my colleagues from the department of Control and Operations at TU Delft. You were always ready to help me with questions, problems, or to just have a chat over a cup of tea (or coffee). I was thinking of naming all of you individually, but I am terribly scared that I might miss a name. Therefore, I will leave it at a huge, collective thank you to everyone. You all know who you are.

A

DERIVATION OF A HORIZONTAL VELOCITY MODEL FOR THE FLAPPING WING DRONE

In this derivation we consider three main forces acting on the flapping wing drone. Since we are mainly interested in accelerations, we already divide out the mass in the following equations, i.e. $\boldsymbol{f} = \frac{\boldsymbol{F}}{m}$. First, the thrust along the drones central axis is given in the drones body frame as

$$\boldsymbol{f}_T^b = |T| \cdot \boldsymbol{e}_3 \tag{A.1}$$

Second, we consider a simple drag model with drag forces proportional and opposed to the drone's body velocity:

$$\boldsymbol{f}_d^b = \begin{bmatrix} -b_x v_x^b \\ -b_y v_y^b \\ -b_z v_z^b \end{bmatrix} \tag{A.2}$$

Finally, the force of gravity is easily expressed directly in the horizontal frame

$$\boldsymbol{f}_g^h = -\left|g\right| \cdot \boldsymbol{e}_3 \tag{A.3}$$

The sum of acceleration in the horizontal frame experienced by the flapper is then given by A.4.

$$\boldsymbol{a}^h = \sum \boldsymbol{f}^h = \mathbf{R}_{hb} \boldsymbol{f}_T^b + \mathbf{R}_{hb} \boldsymbol{f}_d^b + \boldsymbol{f}_g^h \tag{A.4}$$

We can now put the drag coefficients into a diagonal matrix $\mathbf{B} = \text{diag}(\boldsymbol{b})$ and expand the equation. Recall that rotation matrices are orthogonal, and therefore $\mathbf{R}_{bh} = \mathbf{R}_{hb}^{-1} = \mathbf{R}_{hb}^T$. To simplify notation, we will at this point drop the superscript h for vectors in the horizontal frame.

$$\boldsymbol{a} = |T| \cdot \mathbf{R}_{hb} \boldsymbol{e}_3 - \mathbf{R}_{hb} \mathbf{B} \mathbf{R}_{hb}^T \cdot \boldsymbol{v} - \left|g\right| \cdot \boldsymbol{e}_3 \tag{A.5}$$

A

We now write out the equation for each individual axis in the horizontal frame.

$$a_x = |T|\,\mathrm{s}\theta\mathrm{c}\phi - (b_x\mathrm{c}^2\theta + b_y\mathrm{s}^2\theta\mathrm{s}^2\phi + b_z\mathrm{s}^2\theta\mathrm{c}^2\phi)\,v_x - (b_y - b_z)\mathrm{s}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_y + (b_x - b_y\mathrm{s}^2\phi - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\mathrm{c}\theta\,v_z \quad \text{(A.6)}$$

$$a_y = -|T|\,\mathrm{s}\phi - (b_y - b_z)\mathrm{s}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_x - (b_y\mathrm{c}^2\phi + b_z\mathrm{s}^2\phi)\,v_y - (b_y - b_z)\mathrm{c}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_z \quad \text{(A.7)}$$

$$a_z = |T|\,\mathrm{c}\theta\mathrm{c}\phi + (b_x - b_y\mathrm{s}^2\phi - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\mathrm{c}\theta\,v_x - (b_y - b_z)\mathrm{c}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_y - (b_x\mathrm{s}^2\theta + b_y\mathrm{c}^2\theta\mathrm{s}^2\phi + b_z\mathrm{c}^2\theta\mathrm{c}^2\phi)\,v_z - |g| \quad \text{(A.8)}$$

Assuming small roll and pitch angles, we can neglect any terms that include the multiplication of two sines (i.e. $\mathrm{s}^2\theta \approx \mathrm{s}^2\phi \approx \mathrm{s}\theta\mathrm{s}\phi \approx 0$)

$$a_x = |T|\,\mathrm{s}\theta\mathrm{c}\phi - b_x\mathrm{c}^2\theta\,v_x + (b_x - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\mathrm{c}\theta\,v_z \quad \text{(A.9)}$$

$$a_y = -|T|\,\mathrm{s}\phi - b_y\mathrm{c}^2\phi\,v_y - (b_y - b_z)\mathrm{c}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_z \quad \text{(A.10)}$$

$$a_z = |T|\,\mathrm{c}\theta\mathrm{c}\phi + (b_x - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\mathrm{c}\theta\,v_x - (b_y - b_z)\mathrm{c}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_y - b_z\mathrm{c}^2\theta\mathrm{c}^2\phi\,v_z - |g| \quad \text{(A.11)}$$

Due to the difficulty of estimating the generated thrust of the flapper, we further simplify the model by decoupling horizontal and vertical movement of the flapper, i.e. we consider the vertical velocity to be negligible during horizontal motion. This allows us to find an expression for the thrust in terms of $|g|$ and the horizontal velocities.

$$a_z = |T|\,\mathrm{c}\theta\mathrm{c}\phi + (b_x - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\mathrm{c}\theta\,v_x - (b_y - b_z)\mathrm{c}\theta\mathrm{s}\phi\mathrm{c}\phi\,v_y - |g| = 0 \quad \text{(A.12)}$$

$$|T| = \frac{|g|}{\mathrm{c}\theta\mathrm{c}\phi} - (b_x - b_z\mathrm{c}^2\phi)\mathrm{s}\theta\,v_x + (b_y - b_z)\mathrm{s}\phi\,v_y \quad \text{(A.13)}$$

This expression for the thrust can now be inserted into A.9 and A.10, while setting $v_z = 0$. In both cases, the thrust is multiplied with a sine, which again allows us to neglect some terms due to our small angle approximation. As a result, only the gravity term remains in the final expressions for the acceleration alongside the deceleration caused by drag.

$$a_x = \frac{\mathrm{s}\theta}{\mathrm{c}\theta}|g| - b_x\mathrm{c}^2\theta\,v_x \quad \text{(A.14)}$$

$$a_y = -\frac{\mathrm{s}\phi}{\mathrm{c}\theta\mathrm{c}\phi}|g| - b_y\mathrm{c}^2\phi\,v_y \quad \text{(A.15)}$$

We can now integrate these accelerations using the forward Euler method to estimate our horizontal velocities.

$$v_{x,k+1} = v_{x,k} + \mathrm{d}t\left(\frac{\mathrm{s}\theta_k}{\mathrm{c}\theta_k}|g| - b_x\mathrm{c}^2\theta_k v_{x,k}\right) \quad \text{(A.16)}$$

$$v_{y,k+1} = v_{y,k} + \mathrm{d}t\left(-\frac{\mathrm{s}\phi_k}{\mathrm{c}\theta_k\mathrm{c}\phi_k}|g| - b_y\mathrm{c}^2\phi_k v_{y,k}\right) \quad \text{(A.17)}$$

The drag coefficients b_x and b_y can be estimated from a data set with ground truth using a least squares approach. Specifically, for an overdetermined system of equations $\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta}$ the least-squares estimate for the parameters $\boldsymbol{\beta}$ is given by $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}$. To determine the drag coefficients in our horizontal velocity model, we solve two independent least-squares problems with for b_x

$$\boldsymbol{\beta} = b_x \tag{A.18}$$

$$y_k = -\frac{v_{x,k+1} - v_{x,k}}{\mathrm{d}t} + \frac{\mathrm{s}\theta_k}{\mathrm{c}\theta_k}|g| \tag{A.19}$$

$$X_k = \mathrm{c}^2\theta_k \, v_{x,k} \tag{A.20}$$

and for b_y

$$\boldsymbol{\beta} = b_y \tag{A.21}$$

$$y_k = -\frac{v_{y,k+1} - v_{y,k}}{\mathrm{d}t} - \frac{\mathrm{s}\phi_k}{\mathrm{c}\theta_k \mathrm{c}\phi_k}|g| \tag{A.22}$$

$$X_k = \mathrm{c}^2\phi_k \, v_{y,k} \tag{A.23}$$

From our data, we find $b_x = 4.2$ and $b_y = 1.8$ for the Flapper Drones.

B

POSITION INITIALIZATION FROM TWO RANGE MEASUREMENTS AND INTEGRATED VELOCITIES

To better initialize a new agent in the relative position EKF, an initial relative position estimate can be calculated heuristically from two range measurements r_0 and r_1, and the difference between the corresponding relative positions (dx, dy). The difference in relative positions can for example be estimated by integrating the relative velocities. The geometric problem is shown in Figure B.1.

Our goal is to find x_1 and y_1, from the following system of equations describing the situation:

$$r_0^2 = x_0^2 + y_0^2 \tag{B.1}$$
$$r_1^2 = x_1^2 + y_1^2 \tag{B.2}$$
$$x_1 = x_0 + dx \tag{B.3}$$
$$y_1 = y_0 + dy \tag{B.4}$$

Using B.3 and B.4, we can express B.1 in terms of x_1 and y_1:

$$r_0^2 = (x_1 - dx)^2 + (y_1 - dy)^2 \tag{B.5}$$
$$= x_1^2 - 2x_1 dx + dx^2 + y_1^2 - 2y_1 dy + dy^2 \tag{B.6}$$

Using B.2, we can now get to an equation with only x_1 as unknown:

$$r_0^2 = x_1^2 - 2x_1 dx + dx^2 + r_1^2 - x_1^2 - 2dy\sqrt{r_1^2 - x_1^2} + dy^2 \tag{B.7}$$

Rearranging and multiplying by itself to remove the square root:

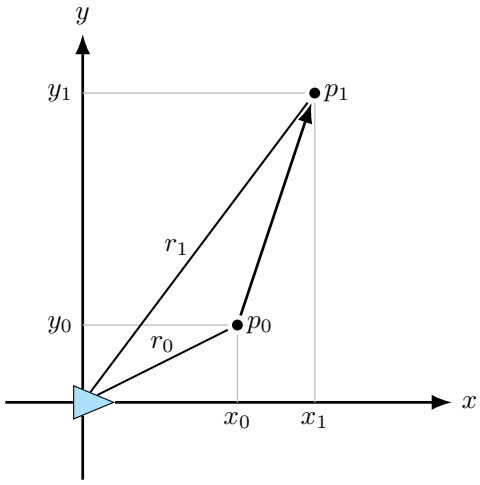


Figure B.1: Initialization from ranges and accumulated relative motion

$$r_0^2 - r_1^2 - dx^2 - dy^2 + 2x_1 dx = -2dy\sqrt{r_1^2 - x_1^2} \tag{B.8}$$

$$\left(r_0^2 - r_1^2 - dx^2 - dy^2 + 2x_1 dx\right)^2 = 4dy^2\left(r_1^2 - x_1^2\right) \tag{B.9}$$

Finally, we need to separate the terms including x_1 and x_1^2. For clarity, we'll substitute the known term

$$u = r_0^2 - r_1^2 - dx^2 - dy^2 \tag{B.10}$$

$$u^2 + 4udx\,x_1 + 4dx^2 x_1^2 = 4dy^2 r_1^2 - 4dy^2 x_1^2 \tag{B.11}$$

$$4\left(dx^2 + dy^2\right)x_1^2 + 4udx\,x_1 + u^2 - 4dy^2 r_1^2 = 0 \tag{B.12}$$

This results in a quadratic equation of the form

$$ax^2 + bx + c = 0 \tag{B.13}$$

with

$$a = 4\left(dx^2 + dy^2\right) \tag{B.14}$$

$$b = 4\left(r_0^2 - r_1^2 - dx^2 - dy^2\right)dx \tag{B.15}$$

$$c = \left(r_0^2 - r_1^2 - dx^2 - dy^2\right)^2 - 4dy^2 r_1^2 \tag{B.16}$$

This results in two solutions for x_1, calculated as

$$x_1 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{B.17}$$

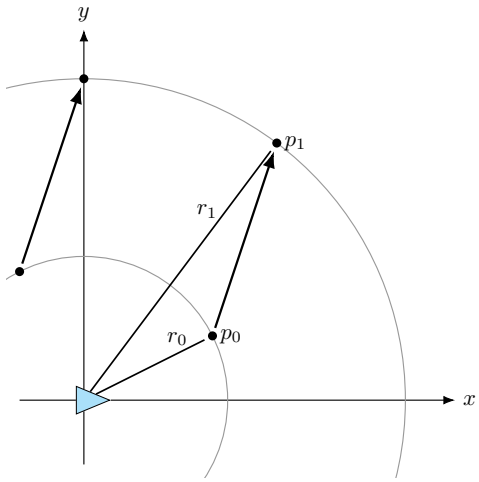


Figure B.2: Second solution to the initialization problem

Receiving two solutions for this problem makes sense when looking at the problem again, especially when we add circles with radius r_0 and r_1. Looking at Figure B.2 we see that there are two places where we can connect the circles with a line of a given length and direction. We can also see, that the distance between the two solutions d_{AB} can never be bigger than the diameter of the inner circle, i.e. $d_{AB} \leq \min(2r_0, 2r_1)$.

From Figure B.2, one can also expect the special case where only one solution exists when p_0, p_1 and the origin lie on a single line.

$$\left\{ \begin{array}{ll} \sqrt{dx^2+dy^2} = r_0 + r_1 & \text{(case 1)} \\ \sqrt{dx^2+dy^2} = r_0 - r_1 & \text{(case 2)} \\ \sqrt{dx^2+dy^2} = r_1 - r_0 & \text{(case 3)} \end{array} \right\} \tag{B.18}$$

Mathematically, a single solution to B.17 means that the discriminant is 0. Keeping our substituted term u from B.10, we can simplify the discriminant significantly:

$$b^2 - 4ac = 16u^2dx^2 - 16\left(dx^2+dy^2\right)\left(u^2 - 4dy^2r_1^2\right) \tag{B.19}$$

$$= 16\left(u^2dx^2 - u^2\left(dx^2+dy^2\right) + 4dy^2r_1^2\left(dx^2+dy^2\right)\right) \tag{B.20}$$

$$= -16dy^2\left(u^2 - 4r_1^2\left(dx^2+dy^2\right)\right) \tag{B.21}$$

For the discriminant to be zero, we can thus conclude that either dy or $u^2 - 4r_1^2(dx^2 + dy^2)$ must be zero. By substituting $dx^2 + dy^2$ from B.18 (case 2 and 3 being equivalent if squared), one can easily verify that the latter is indeed the case.

Furthermore, it makes geometrical sense that for $dy = 0$ only one solution exists for x and the second solution arises due to different and opposing values of y.

Having a solution for x_1, the associated value for x_0 is easily calculated from B.3. Using B.1 and B.2, there are then two possible solutions for y_0 and y_1, of which the correct one has to be determined using dy. As an example, if we find $y_0 = \pm 2$ and $y_1 = \pm 3$ given

$dy = 1$, B.4 dictates that the correct solution must be $y_0 = 2$ and $y_1 = 3$. In the special case where $dy = 0$, both the positive and negative values represent a valid solution. In all other cases, the second valid solution comes from the second solution for x_1.

Geometrically, this problem should always have at least one solution. Unfortunately, the existence of noise on both range measurements and the integrated relative velocity can cause the problem to become unsolvable. Specifically, this happens when the triangle inequality is violated by the measurements, causing the discriminant to be negative (in our example in Figure B.1, that would mean $|r_1| > |r_0| + |\overline{p_1 - p_0}|$). For such measurements to appear, the triangle is likely close to a line, i.e. we observe a radial relative movement. The best course of action is then to use the integral of relative velocity as direction, and the second range measurement as distance.

$$\hat{p}_1 = \frac{r_1}{\sqrt{dx^2 + dy^2}} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{B.22}$$

CURRICULUM VITÆ

Sven Udo PFEIFFER

08-06-1994 Born in Munich, Germany.

EDUCATION

2009–2013 Grammar School
Freies Gymnasium Bern (Switzerland)

2013–2016 BSc in Mechanical Engineering
École Polytechnique fédérale de Lausanne (Switzerland)

2016–2019 MSc in Mechanical Engineering
École Polytechnique fédérale de Lausanne (Switzerland)

2019–2024 PhD. Aerospace Engineering
Delft University of Technology (The Netherlands)
Thesis: Range-Based Localization for Swarms of Micro Air Vehicles
Promotor: Prof. dr. G.C.H.E. de Croon

AWARDS

2013 1st Place Swiss Physics Olympiad

2013 SPG Nachwuchsförderpreis

LIST OF PUBLICATIONS

4. **S. Pfeiffer**, C. De Wagter, and G.C.H.E. de Croon, *Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones*, in 14^{th} Annual International Micro Air Vehicle Competition and Conference 2023, Aachen, 85-91 (2023).

3. **S. Pfeiffer**, V. Munaro, S. Li, A. Rizzo, and G.C.H.E. de Croon, *Three-dimensional relative localization and synchronized movement with wireless ranging*, Swarm Intelligence **17**, 147-172 (2023).

2. S. Bahnam, **S. Pfeiffer**, and G.C.H.E. de Croon, *Stereo Visual Inertial Odometry for Robots with Limited Computational Resources*, in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 9154-9159 (2021).

1. **S. Pfeiffer**, C. De Wagter, and G.C.H.E. de Croon, *A Computationally Efficient Moving Horizon Estimator for Ultra-Wideband Localization on Small Quadrotors*, IEEE Robotics and Automation Letters **6**, 6725 (2021).