Autonomous Swarms of Tiny Flying Robots

Li, S.

DOI
10.4233/uuid:825f3a6b-3039-4a6e-8f3f-9f0871bd9ce5

Publication date
2021

Document Version
Final published version

Citation (APA)
Li, S. (2021). *Autonomous Swarms of Tiny Flying Robots*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:825f3a6b-3039-4a6e-8f3f-9f0871bd9ce5

AUTONOMOUS SWARMS OF TINY FLYING ROBOTS

AUTONOMOUS SWARMS OF TINY FLYING ROBOTS

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 Monday, 29 November, 2021 at 17:30 o'clock

by

Shushuai LI

Master of Science in Control Science and Engineering,
Hunan University, China,
born in Handan, China.

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
Prof.dr.ir. M. Mulder	Delft University of Technology, promotor
Independent members:	
Prof.dr. D. Floreano	Swiss Federal Institute of Technology Lausanne
Dr. M. Saska	Czech Technical University
Dr. W. Hönig	Technische Universität Berlin
Prof.dr.ir. M. Wisse	Delft University of Technology
Prof.dr. E.K.A. Gill	Delft University of Technology

Keywords: Swarm robotics, Micro aerial vehicles, Relative localization, Nonlinear predictive control, Visual deep learning

Printing: Ridderprint | www.ridderprint.nl

Front & Back: Shushuai Li

ISBN 978-94-6366-472-1

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

CONTENTS

SUMMARY

In the last decade, the research field of aerial swarms has grown at a rapid pace. These multi-robot systems possess desirable abilities including mobility in 3D spaces, efficient task execution in parallel, and redundant characteristics for fault tolerance. Many applications with multiple flying robots have already been demonstrated, such as light shows, search and rescue, area coverage, etc. Most studies for the above applications deal with position estimation, coordinated control, motion planning, or task assignments. However, the fundamental challenge remains to develop autonomous swarm systems that can work together and tackle real-world applications.

As a special case of aerial swarms, multiple tiny (pocket-size) flying robots are safer and thus promising for real-world applications. These robots are highly limited in computation power and sensor capability, which makes the system design more challenging. An essential capability required for swarm coordination is that the individual robots are able to localize themselves with respect to others, preferably without the help of external infrastructure. Even though some works address the problem of onboard relative localization, the relative estimation is not accurate or consistent enough for precise swarm behaviors. This thesis investigates how to build a fully autonomous swarm of tiny aerial robots, featuring accurate relative state estimation and distributed control for different multi-robot tasks in unknown 3D environments.

First, this dissertation studies a low-level estimation problem for aerial swarms, regarding the estimation autonomy of the individual micro aerial vehicle (MAV). Consider that each MAV typically carries various sensors, of which the measurements are fused by a state estimation filter. Such multi-sensor filters heavily rely on manually-tuned parameters for correct attitude and position estimation. Obviously, if a swarm consists of many individual robots, manual tuning of the filter parameters for each robot becomes a tedious and time-consuming process. To eliminate manual tuning, we propose a novel tuning method that enables the robot to learn optimal filter parameters automatically, by minimizing the discrepancy between expected and received signals from all sensors. Given the raw sensor data of a 10-second manual flight, the unsupervised optimization learns optimal filter parameters that are initialized to zero. The optimized filter parameters deliver more precise state estimation than preset parameters for a Crazyflie, and accurate attitude estimation for another test on the Euroc MAV dataset.

Second, this dissertation develops a fully autonomous swarm of tiny flying robots in GPS-denied environments. The system framework consists of fast inter-robot ranging, relative localization, distributed control, and visual navigation. The relative localization is performed by an Extended Kalman Filter (EKF) with onboard sensing of velocity, yaw rate, and height as inputs, in combination with ranging measurements from onboard ultra wide-band (UWB). This system design also involves an automatic initialization procedure, and proofs of consistent estimation convergence even under unobservable conditions. Real-world experiments are conducted with a team of five Crazyflie2.0 quadro-

tors, demonstrating an autonomous formation flight and leader-follower coordination through a window. This tiny aerial swarm system is precisely localized, supporting varied autonomous tasks such as collaborative gas-seeking. Since we consider that this relative localization is useful for many aerial swarm applications, we have made the code open source.

Third, this dissertation focuses on the visual relative localization of tiny flying robots, due to the intrinsic scalability and rich information of visual sensors. A deep neural network (DNN) is designed for monocular relative localization by predicting the center pixel position and depth of other robots. For training, the position and depth annotations come from the onboard range-based relative estimation developed earlier in the thesis, converted with the camera intrinsic parameters. After training from scratch in this self-supervised way, the DNN can predict the relative positions of peer robots by purely using the monocular image, which is scalable, distributed, and autonomous. A simulation pipeline is developed with Blender for rendering synthetic images of multiple drones, to facilitate the preliminary validation of the proposed network. This network is further refined with real-world datasets collected from two Crazyflie quadrotors flying in each other's vicinity, and it can be run on a tiny AI chip, called the AIdeck, for the onboard relative localization.

Finally, this dissertation exploits nonlinear optimal control for range-based aerial swarms. Wireless ranging measurements are capable of multi-MAV relative localization, but the high-dimensional states are weakly observable concerning the scalar measurement. Consequently, the MAVs have degraded localization and control performance under unobservable situations as can be deduced by the Lie derivatives. Therefore, we present a nonlinear model predictive control (NMPC) to maximize the determinant of the observability matrix and minimize the formation tracking error. In the meantime, the control commands satisfy all constraints from the input limitation and state bound. Simulation results validate the localization and control efficacy of the proposed MPC method for range-based multi-MAV systems with weak observability, proving its faster convergence time and more accurate localization compared to the previously proposed random motions.

Overall, this thesis contributes to a fully autonomous swarm of tiny flying robots featuring high-speed ranging, accurate relative localization, observability analysis, and cooperative navigation. It additionally introduces improvements to the aerial swarm, such as optimized individual state estimation, vision-based relative localization, and optimal control for maximizing observability.

SAMENVATTING

Het onderzoek naar zwermen van vliegende robots heeft het afgelopen decennium een hoge vlucht genomen. Zwermen van zulke vliegende robots, ook wel drones genoemd, hebben een aantal gewenste eigenschappen. Zo kunnen ze parallel en dus efficiënt taken uitvoeren, zijn ze robuust ten opzichte van fouten die op kunnen treden bij enkele drones, en hebben ze de mogelijkheid om in drie dimensies te bewegen. Zwermen drones hebben al meerdere toepassingen gevonden, zoals lichtshows, hulp bij calamiteiten, en het afzoeken van gebieden, bijvoorbeeld om zo snel mogelijk drenkelingen te vinden op zee. De meeste onderzoeken naar deze toepassingen richten zich op positieschatting, coördinatie tussen de drones, planning en besturing van de bewegingen, of het toewijzen van taken aan de verschillende individuele drones. Echter, de uitdaging om autonome zwermen te creëren die taken in de echte wereld kunnen uitvoeren is nog verre van opgelost.

Zwermen van kleine "mini-drones" zijn van extra belang voor toepassing in de echte wereld omdat ze vanwege hun kleine omvang en lichte gewicht veiliger zijn voor mensen. Hierdoor zijn ze wel ook erg gelimiteerd in termen van de sensoren en rekenkracht die ze mee kunnen dragen. Een essentiële vaardigheid voor zwermcoördinatie is dat de individuele drones in staat zijn om de relatieve positie van andere drones te bepalen, liefst zonder daarbij externe infrastructuur nodig te hebben. Alhoewel meerdere onderzoeken dit probleem van relatieve localisatie zonder externe hulp hebben bestudeerd, is er nog geen oplossing die precies en consistent genoeg is voor nauwkeurige zwermcoördinatie. Dit proefschrift onderzoekt hoe een volledig autonome zwerm van mini-drones ontwikkeld kan worden, waarbij de drones andere drones nauwkeurig kunnen lokaliseren en ze verchillende zwermtaken uit kunnen voeren in onbekende 3D-omgevingen.

Ten eerste bestudeert dit proefschrift het elementaire probleem van toestandschatting door een individuele drone. Elke mini-drone draagt altijd een variëteit aan sensoren mee, die gecombineerd worden in een toestandschattingfilter. Een dergelijk filter is normaliter afhankelijk van met de hand gekozen parameters om zo nauwkeurige schattingen te verkrijgen van positie en "houding" (oftewel de hoeken van de drone ten opzichte van het aardoppervlak). Aangezien zwermen uit grote aantallen drones kunnen bestaan, wordt zo'n manueel proces van het kiezen van parameters per drone een lastige en tijdrovende bezigheid. Om dit manuele proces te elimineren, stellen we een nieuwe methode voor die drones in staat stelt om zelf automatisch de beste parameters te leren. Dit doen ze door het verschil te minimaliseren tussen wat ze verwachten en wat ze echt binnenkrijgen aan sensormetingen. Met slechts 10 seconden data, verzameld in een korte vlucht, kan de voorgestelde optimalisatie de optimale parameterwaardes leren, startend vanaf nulwaardes. De voorgestelde methode vereist geen "toezicht", wat inhoudt dat de echte toestand van de drone niet gemeten hoeft te worden met een extra en zeer precies extern meetsysteem. De geoptimaliseerde filterparameters leveren een nauwkeurigere toestandschatting op dan de standaardparameters in de drone gebruikt in de experimenten, d.w.z. de CrazyFlie drone. Ook resulteert het in precieze schattingen bij toepassing op de openbare

"EuRoC dataset".

Ten tweede wordt er in dit proefschrift een zwerm van mini-drones ontwikkeld die volledig autonoom kan vliegen in omgevingen waarin GPS niet beschikbaar is. Het voorgestelde systeem bestaat uit snelle afstandsbepalingen tussen drones, relatieve lokalisatie, decentrale besturing, en visuele navigatie. De relatieve lokalisatie wordt uitgevoerd door een "Extended Kalman Filter" (EKF), dat sensormetingen van de snelheid, rotatiesnelheid, en hoogte meeneemt en combineert met afstandsbepalingen op basis van ultra wide-band (UWB). Het voorgestelde systeem bevat ook een automatische initialisatieprocedure en bewijzen van de convergentie van relatieve positieschattingen zelfs onder onobserveerbare omstandigheden. Experimenten zijn uitgevoerd met vijf CrazyFlie 2.0 drones waarin ze formatievluchten uitvoeren. Verder is er een leider-volger experiment gedaan waarin de drones zelfstandig door een raam vliegen. De geïntroduceerde relatieve lokalisatie is erg nauwkeurig en is in staat om een variëteit aan taken te ondersteunen, zoals bijvoorbeeld het zelfstandig zoeken naar een gasbron door een zwerm mini-drones. Omdat relatieve lokalisatie zo belangrijk is voor zwermtoepassingen, hebben we de broncode openbaar gemaakt.

Ten derde focust het proefschrift op visuele relatieve lokalisatie van mini-drones, omdat deze manier van lokalisatie goed opschaalt naar grotere aantallen drones en camerabeelden een rijke bron van informatie vormen. Een diep neuraal netwerk (DNN) wordt geïntroduceerd dat relatieve lokalisatie uitvoert door de centrale pixel en afstand tot een andere drone te voorspellen op basis van een enkel plaatje. Dit netwerk wordt getraind met behulp van de UWB-gebaseerde relatieve lokalisatie die eerder in het proefschrift ontwikkeld is. Op deze manier hoeft er geen mens aan te pas te komen om aan te geven waar andere drones zich in de plaatjes bevinden. Na het zelfstandig trainen van het netwerk kan de drone de relatieve posities van andere drones bepalen in een enkel plaatje, een oplossing die schaalbaar, gedistribueerd en autonoom is. Om het trainen en de verificatie van de methode verder te vergemakkelijken is er een simulatiepijplijn opgezet in de "Blender" software, waarin kunstmatig plaatjes gegenereerd worden van meerdere andere drones met verscheidene achtergronden. Het netwerk wordt verfijnd met behulp van een echte dataset waarin twee CrazyFlies dichtbij elkaar vliegen. Het diepe neurale netwerk is klein genoeg om aan boord van een zeer kleine CrazyFlie berekend te worden, op een kunstmatige intelligentiechip die het "AI deck" heet.

Tenslotte wordt in het proefschrift optimale besturing bestudeerd voor het verbeteren van de relatieve lokalisatie in vliegende zwermen. De eerder in het proefschrift bestudeerde draadloze afstandsschattingen stonden relatieve lokalisatie toe, maar leidden in bepaalde situaties wel tot een slechte observeerbaarheid van relatieve lokaties. Tengevolge hebben drones in die stituaties – geïdentificeerd met behulp van "Lie" afgeleiden – een minder precieze schatting van de relatieve lokaties van andere drones. Een nonlineaire modelvoorspellingsbesturingsmethode (nonlinear model predictive control, NMPC) wordt onderzocht om zowel de determinant van de observeerbaarheidsmatrix te maximaliseren als de positiefouten tijdens een formatievlucht te minimaliseren. Dit terwijl de besturing zich ook aan alle beperkingen houdt, zoals motorlimieten en grenzen in de toestandsruimte. Simulatieresultaten valideren de lokalisatie- en besturingseffectiviteit van de voorgestelde NMPC methode voor relatieve lokalisatie op basis van draadloze afstandsmetingen. De methode resulteert in een snellere convergentietijd en een nauwkeurigere

lokalisatie dan de eerder in het proefschrift voorgestelde willekeurige bewegingsrichtingen tijdens de initialisatie.

In het algemeen draagt dit proefschrift bij aan volledig zelfstandige zwermen van vliegende mini-drones, door middel van snelle afstandsbepalingen, nauwkeurige relatieve lokalisatie, observeerbaarheidsanalyses, en gecoördineerde navigatie. Het stelt daarnaast verscheidene verbeteringen voor, waaronder zelflerende methodes voor toestandschatting en visuele relatieve lokalisatie, en optimale besturing voor maximale observeerbaarheid van de relatieve lokaties van andere drones.

1

INTRODUCTION

Over the last decade, aerial swarms as a special case of multi-robot systems have drawn increasing attention from the robotic community. These multiple micro aerial vehicles (MAVs) are suitable for remote sensing and operations, and are qualified for numerous cumbersome works by parallel execution and cooperation. The increasing interest is also due to their potential applications such as entertainment of light shows with up to 5,164 MAVs [1], construction of cubic structures [2], collaborative payload transportation [3], and inspection of power lines [4]. The focus in this thesis is on multiple aerial robots with small size (e.g., 33 grams in [5]), which are harmless and agile, allowing operations in narrow indoor environments and close to humans. Creating autonomous swarms of tiny aerial robots will open up new applications ranging from crop monitoring in greenhouses to search and rescue [5] or gas source localization [6].

Most swarm intelligence algorithms have been successfully validated in simulated environments [7]. Few real-world experiments have been demonstrated on multiple tiny flying robots, however, not only due to the tough management of a large number of robots, but also because of the *limited multi-level autonomy* of aerial swarms [8]. The low-level autonomy consists of attitude and position estimation for a single drone that allows for an autonomous flight [9]. This is normally achieved by fusing inertial sensor data with other position information provided by external infrastructures such as global positioning systems [10], motion capture systems (MCS) [11], and wireless beacons [12]. Since external signals are not always available, each drone should be capable of full onboard localization, wherein optimal filter parameters are important to more accurate state estimation. Existing methods for autonomous flight include optical flow [13], Visual Inertial Odometry (VIO) [14] and Simultaneously Localization and Mapping (SLAM) [15]. Moreover, obstacle avoidance is another prominent component for individual autonomy, tackled typically by utilizing ranging-detection [5], LIDAR [16], and RGB-D sensors [17]. Notice that the last two sensors are too heavy for implementation on tiny aerial robots.

High-level autonomy contains the inter-robot relative localization and coordinated control without relying on any external infrastructure. To perceive other team members, different onboard sensors have been deployed, including infra red [18], sound [19], Blue-

1

Figure 1.1: Example of aerial swarms in nature, i.e., a flock of birds. Photo is from https://commons.wikimedia.org/wiki/Main_Page.

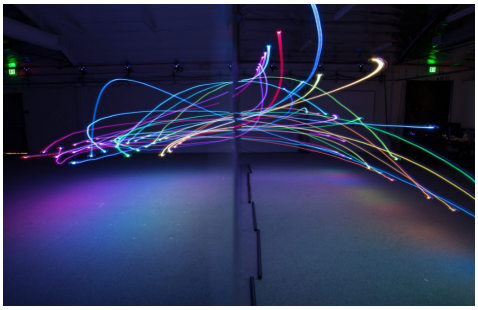
tooth [20], Ultra Wide-Band (UWB) [21], and vision [22]. In addition, aerial swarms need to combine inter-robot perception with the relative motion model to calculate the relative positions. Still, there are currently no stable and accurate relative localization solutions that can be run "out of the box" for multi-robot applications. UWB tends to be the most promising sensor for relative positioning as it consists of both a ranging and a communication capability. The corresponding requirements of implementing UWB for multi-MAVs localization involve a high-speed ranging communication technology, on top of which an efficient relative localization should be designed. Moreover, the communication network is strong for exchanging essential information not only for the relative localization but also allowing coordinated operation. Based on relative positions, aerial swarms require distributed control that consists of several essential capabilities such as inter-swarm collision avoidance, trajectory tracking, and visual navigation. The limited hardware on pocket drones makes the system design even more challenging to integrate all above capabilities.

This thesis will focus on the whole system design of autonomous swarms of tiny flying robots, providing efficient and accurate relative localization, distributed and coordinated control, and visual navigation ability. Before diving into the solutions, the remainder of this chapter will give an overview of the state-of-the-art strategies, followed by the research questions and thesis outline.

1.1. AERIAL SWARMS

Inspired by nature, scientists in the area of aerial swarms strive to design individual interactions to achieve collective and self-organizing behavior for swarms of robots [7]. Normally, flying robot swarms take inspiration from wild animal groups such as bird flocks as shown in Figure 1.1, to perform a collective swarm behavior only with local perception information and distributed control laws. However, to obtain mutual perception among robots is a vital problem to be addressed. The corresponding solutions also draw inspiration from animal swarms, the elements of which rely on smell, sound, or vision to perceive each other and alter the group dynamics [23].

A wealth of strategies for mutual perception and collective control among multiple



(a) An indoor swarm of 32 Crazyflie quadrotors [27]

(b) An outdoor flocking of 30 drones [10]

Figure 1.2: Example of aerial swarms mimicked by multiple quadrotors. Both swarm systems leverage external positioning systems.

aerial robots have been summarized in [24]. For multiple robots, several essential features inspired by natural aerial swarms are expected: mutual perception, coordinated motion, environmental perception, and collaboration. Environmental perception has consistently been an active area studied by robotic scientists and engineers, to enhance the individual robot autonomy for working in cluttered environments. For example, an individual flying robot is capable of self-localization with visual-inertial odometry [25] and avoids obstacles by means of tiny laser sensors [5]. In contrast, this thesis will focus on mutual perception and coordinated motion within a swarm. Nevertheless, the top priority is to build a stable, robust, effective, and autonomous swarm system with multiple flying robots. The following will discuss the state-of-the-art achievements in this area.

1.2. PREVIOUS RESEARCH

Worldwide, scientists and engineers have placed significant effort on mimicking the behavior of bird swarms using multiple drones. Most research related to aerial swarms has been summarized recently in two survey papers, [24] and [26]. However, due to the fast pace of swarm robotics, new works have been published that are not included in these review papers. This section provides an updated and brief survey, with most emphasis on mutual perception, as this is a central topic in the thesis. The most state-of-the-art aerial systems are divided into three categories according to their localization approaches: (i) external infrastructures, (ii) onboard non-visual perception, and (iii) onboard visual perception.

1.2.1. EXTERNAL INFRASTRUCTURE

External infrastructures primarily consist of motion capture systems (MCS) [11], positioning satellites [10], and ultra wide-band (UWB) beacons [28]. Although the swarm depends on the availability of these external systems, the works are still relevant, as they feature impressive demonstrations and control strategies.

For example, a recent study in [29] proposed a centralized nonlinear model predictive controller (MPC), which guaranteed the safe trajectory and fast speed of collective mo-

tion of five Crazyflie quadrotors in cluttered environments, based on OptiTrack MCS for providing the robot states and environment information. In addition, a large number of aerial robots formed beautiful patterns in Figure 1.2, where quadrotors are positioned by the Vicon MCS in Figure 1.2a and relative localized by Global Navigation Satellite System (GNSS) in Figure 1.2b, respectively.

Alternatively, fixed wireless UWB beacons are another external system that can localize multiple drones simultaneously. With these beacons, multiple quadrotors can be controlled simultaneously with two common-used localization technologies: Time Difference of Arrival (TDoA) [12, 30], and Two-Way Time-of-Flight Ranging (TWR) [28].

External positioning systems have been instrumental in illustrating the potential of drone swarms, of how they can pass through a window together [31] or how they can perform beautiful choreographies [11]. However, the availability of the above external infrastructures is too limited to allow for autonomous flight in unknown and potentially GPS-denied environments. In nature, a swarm of birds does not rely on any external infrastructure for relative positioning. Instead, they perceive swarm members and environments by relying on sound [32], and vision [33]. Therefore, several works also explore how to achieve onboard relative localization.

1.2.2. ONBOARD NON-VISUAL PERCEPTION

Cooperative tasks require the relative position of peer robots to enable inter-swarm collision avoidance, coordination, and more. Therefore, the development of accurate relative localization technologies is a challenge to be tackled.

Table 1.1: Advantages and disadvantages of different onboard localization strategies for multiple aerial robots based on non-visual sensors. The frequency means the ranging observation update interval between twin robots. The degree and meter represent the direction and position estimation error. The best performance in terms of the update frequency and range combination is typeset in bold - and is part of this thesis.

Reference	Sensors	Dependence				Outcomes	
		IMU	Velocity	Height	North	Position Direction	Frequency & Accuracy
[18], 2012	infrared					P	100Hz, ~0.4m
[19], 2016	sound					D	~4°
[20], 2018	bluetooth	•	•	•	•	P	5Hz, ~0.8m
[34], 2017	UWB	•	•	•	•	P	20Hz, ~0.8m
[35], 2018	UWB	•	•	•	•	P	~0.15m, ~30°
[36], 2019	UWB	•	•	•	•	P	40Hz, ~0.2m
[21], 2020	UWB	•	•	•		P	25Hz, ~0.22m
[37], 2020	UWB	•	•	•		P	**300Hz**, **~0.2m**
[38], 2020	UWB	•	•	•	•	P	10Hz, ~1.4m
[39], 2021	UWB	•			•	P	10Hz, ~0.6m
[40], 2021	2xUWB	•			•	P	16Hz, ~0.4m

Onboard sensors benefit an autonomous swarm of drones by removing the dependence on external infrastructures. These sensors provide relative information among mul-

tiple flying robots, underlying a wide range of distributed and cooperative tasks. In the last decade, different onboard sensors enabled mutual relative localization of multiple aerial robots. They have different dependence on the prior known information, while the output relative estimation performance also varies, as summarized in Table 1.1.

Arrays of infrared (IR) sensors [18] or sound-based microphones [19] were studied first for multi-MAV detection. However, these works featured a low range accuracy and limited operating range. Recently, wireless sensors have been adopted fully onboard for relative localization of aerial swarms thanks to its light weight and ID-deterministic ranging measurements even with long distances. Multiple MAVs can estimate the relative positions by fusing the inter-robot distances with other sensory information (e.g., velocity, yaw rate, height) exchanged with antennas. The first work in this area focused on bluetooth [20], which was published on the arxiv in 2016, influencing subsequent work on UWB relative localization [21, 34–40]. Note that our proposed technology in Chapter 3 [37] contributes to the first stable, high-speed, and accurate relative localization on a swarm of tiny drones as shown with bold text in Table 1.1. Importantly, we removed the dependence on orientation observation by the magnetometer, since this sensor often fails due to environment interference.

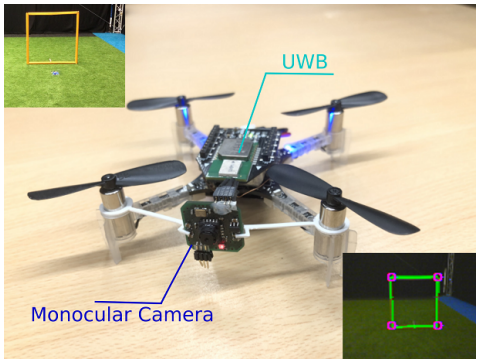
In summary, wireless-signal-based aerial swarms are stable and omni-directional, but with poor scalability for large numbers of quadrotors given the constant communication bandwidth. Recently, though, a distributed and UWB-based swarm ranging protocol was presented to increase the scalability capability [41], which has not been used for inter-robot localization. Of course, the perception of the environment still demands other sensors, e.g., for obstacle avoidance and localization.

1.2.3. ONBOARD VISUAL PERCEPTION

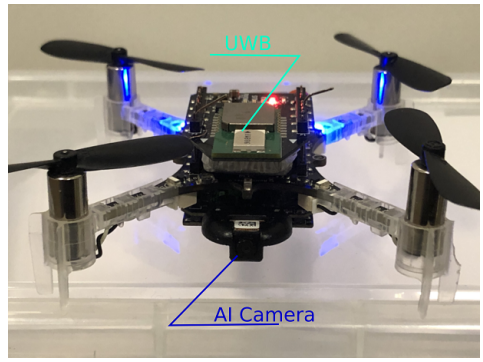
Other works leverage computer vision technology to achieve aerial swarms for robot detection and environment perception. Cameras are low-cost, small-size, light-weight, and due to the passive nature of the sensor, a camera scales well to detecting many other robots in a swarm. Most studies in the vision domain address the problem of self-localization of the individual robot or target localization. For multi-robot localization, the existing visual algorithms need to be modified for relative positioning, and expanded with a tracking procedure. Vision-based localization approaches for aerial swarms can be divided into three main categories: marker-based localization, markerless absolute localization, and markerless relative localization.

Marker-based visual methods can simplify both multi-MAVs detection and localization procedures. For example, a simple marker with known pattern shape and size allows computationally efficient visual relative localization of quadrotors [42]. Others include ultraviolet markers to overcome unfavorable lighting scenarios and to increase the distance at which drones can localize each other [43]. However, marker occlusions heavily degrade the relative localization errors, which is even worse when using tiny pocket drones. Moreover, markers can be heavy to carry for small drones.

One visual solution for aerial swarms is to compute the absolute positions of the individual robots without relying on any specific textures. For example, multiple drones execute formation flights based on the global positions calculated by visual-inertial odometry (VIO) [14]. These VIO-based multi-drone systems rely on the initial robot positions and



(a) In 2019, navigation of a leader drone with a tiny customized monocular camera



(b) In 2020, mutual detection among multiple drones with an AIdeck - a camera module for deep learning

Figure 1.3: Example of Crazyflie2.0 quadrotors used in this thesis. Apart from the different visual sensors, both platforms equip the UWB deck and flow deck.

require a centralized controller, and the absolute positions are subject to drift over time. In contrast, SLAM-based aerial swarms build an environment map such that the drift problem is eliminated. In [44], an aerial swarm with visual SLAM is capable of mapping an unknown environment, while the inter-drone measurements are from radio signal strength (RSS). This system is further developed with collaborative localization and mapping for multiple MAVs [45]. More recently, a fully decentralized visual SLAM is proposed for multiple robots [46], which however has not been tested on real robots. In contrast to pure simulation, the DOOR-SLAM for multi-UAV localization is designed and run onboard two quadrotors in a distributed manner [15].

The last vision-based strategy is to have the robots detect and localize each other directly. These strategies consist of traditional visual algorithms [47, 48] and learning-based methods. Deep learning related methods include YOLOv3-tiny [22], tiny-YOLO [49], EVPropNet [50], YOLOv2-tiny and depth maps [51], etc. However, the range estimation for the above works requires prior knowledge of drone size, which varies regarding different new drones. This thesis proposes to predict the center pixel positions and the corresponding depth of other MAVs from monocular images [52] in a self-supervised manner with support from range-based localization.

Overall, apart from inter-robot detection, cameras also provide information about the environment. The challenge remains that vision processing is typically computationally expensive, limiting the implementation and performance on pocket drones.

1.3. RESEARCH OBJECTIVES AND QUESTIONS

This thesis focuses on the system design of an open-source and open-hardware fully autonomous tiny aerial swarm. The main research goal is formulated as follows.

Research Goal

To develop a fully autonomous swarm of flying robots that can operate without any external infrastructure, which can be used for various multi-robot tasks.

This goal requires swarm autonomy concerning four essential capabilities: (1) individual state estimation, (2) relative localization, (3) distributed control, and (4) visual navigation. Therefore, the key challenges involve the following aspects:

- Explore optimized individual state estimation and control to improve the swarm motion behaviors.
- Design effective and efficient methods for all above procedures that can run onboard a pocket drone simultaneously.

We split the research goal into four research questions summarized in the remaining part of this section. First, as is well known, state estimation is the fundamental part of robot design, as it can output robot attitude or position, which provides crucial information for robot control and navigation. Typically, the filter parameters for state estimation are tuned manually, based on the measurements and groundtruth values. However, this tuning process becomes tedious for large numbers of drones, which vary in terms of body, sensors and actuators. Hence, to make the individual robot more autonomous at its state estimation, the first research question is:

Research Question 1

Given a physical model and a filter, how can an individual robot learn its filter parameters automatically in an unsupervised manner?

Inheriting the idea of range-based relative localization with three Bebop drones [21], we further develop a similar but more efficient system framework for relative localization on commercial open-source tiny quadrotors Crazyflie2.0. The purpose is to allow peers to use the autonomously-localized, multiple Crazyflies for their own multi-robot experiments and applications. Therefore, the second research question is:

Research Question 2

With limited computation resources on pocket drones, how can we design fully autonomous and efficient relative localization technology that allows for multi-robot tasks?

Because wireless communication-based localization has bandwidth limitations, we employ deep visual learning for relative localization among a swarm of aerial robots. Also, the configuration with deep learning and vision supports future potential obstacle avoidance and navigation required by swarm robotics. However, mutual detection considers drone appearance, of which the learning procedure requires groundtruth such as masks as discussed in most references. Hence, the third research question is:

1

Research Question 3

How can we design a deep learning setup for 3D multi-robot relative localization without manual labeling?

The last research question is related to the second one, in which the initialization procedures need optimization for a better relative localization performance. Random velocities do not explicitly consider weak observability, thus they do not guarantee a high estimation performance. We consider using optimal control strategies to calculate the velocity inputs for range-based multi-robot systems, leading to the fourth research question:

Research Question 4

How can we design an optimal control scheme that maximizes multi-robot observability, to achieve a faster localization convergence?

Each research question is studied in a separate chapter, and their relationship forms the thesis outline discussed in the next section.

1.4. DISSERTATION OUTLINE AND APPROACHES

This thesis involves six Chapters to finalize the design and improvement of a fully autonomous swarm of tiny aerial robots. It starts with the current Chapter 1, the Introduction, presenting the research review and research questions. The final chapter, Chapter 6, summarizes the answers to these questions, provides conclusions and discusses future work.

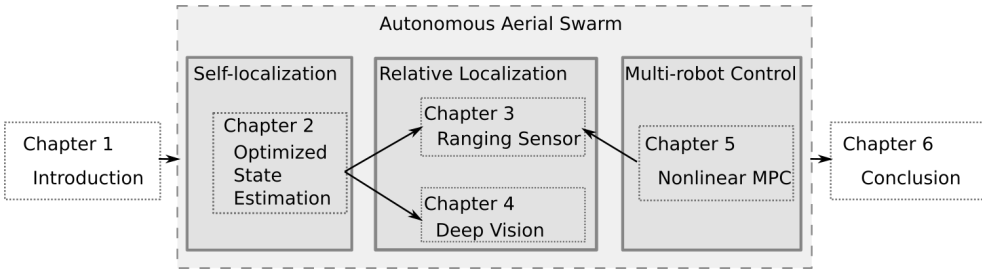


Figure 1.4: The outline of the thesis.

The detailed investigations into the individual research questions form the four intermediate chapters. In Chapter 2, the estimation autonomy for the individual robot is studied, in which a novel cost function is proposed for unsupervised filter parameter tunning without relying on groundtruth. With finite learning iterations, the optimized parameters enable an effective attitude and position estimation.

After solving the autonomy for self-estimation, Chapter 3 studies the autonomous aerial swarm by proposing a ranging sensor-based relative localization. This chapter finalizes a fully autonomous swarm of tiny flying robots, with full-stack solutions including high-speed ranging communication, velocity-based relative localization, distributed

control for coordination, tracking, and navigation. This proposed system design is open-source and forms a fundamental tool for many other multi-robot tasks.

Based on the developed range-based relative estimation, Chapter 4 provides the deep visual relative localization for pocket drones. Visual detection has better scalability, and it benefits other visual tasks such as obstacle detection. The methodology consists of self-supervised training aided by automatic annotations, a small network that outputs the pixel positions and depth, and implementation on a tiny AI chip.

Chapter 5 incorporates the nonlinear optimal control into the range-based aerial swarm to maximize the weak observability. The target function is composed of two cost terms. One is to maximize the determinant value of the observability matrix. The second is to minimize the formation reference tracking errors. Besides, the control satisfies the constraints of velocity input limitations, state bounds, and safe distance keeping. This nonlinear optimal control exploits the software Acados, an open-source solver for calculating the control inputs. The proposed optimal solution supports the incorporation of new multi-robot tasks.

REFERENCES

[1] Guinness World Records, *Most unmanned aerial vehicles (UAVs) airborne simultaneously,* (2021).

[2] Q. Lindsey, D. Mellinger, and V. Kumar, *Construction of cubic structures with quadrotor teams,* Proc. Robotics: Science & Systems VII (2011).

[3] M. Gassner, T. Cieslewski, and D. Scaramuzza, *Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors,* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 5196–5202.

[4] T. Uzakov, T. P. Nascimento, and M. Saska, *Uav vision-based nonlinear formation control applied to inspection of electrical power lines,* in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2020) pp. 1301–1308.

[5] K. McGuire, C. De Wagter, K. Tuyls, H. 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** (2019).

[6] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. H. E. de Croon, *Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments,* accepted by IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021).

[7] M. Dorigo, G. Theraulaz, and V. Trianni, *Swarm robotics: Past, present, and future,* Proceedings of the IEEE **109**, 1152 (2021).

[8] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, *Aerial swarms: Recent applications and challenges,* Current Robotics Reports , 1 (2021).

[9] 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.

[10] 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** (2018).

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

[12] M. Hamer and R. D'Andrea, *Self-calibrating ultra-wideband network supporting multi-robot localization,* IEEE Access **6**, 22292 (2018).

[13] G. C. H. E. de Croon, C. De Wagter, and T. Seidl, *Enhancing optical-flow-based control by learning visual appearance cues for flying robots,* Nature Machine Intelligence **3**, 33 (2021).

[14] 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).

[15] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, *Door-slam: Distributed, online, and outlier resilient slam for robotic teams,* IEEE Robotics and Automation Letters **5**, 1656 (2020).

[16] R. Sabatini, A. Gardi, and M. A. Richardson, *Lidar obstacle warning and avoidance system for unmanned aircraft,* International Journal of Mechanical, Aerospace, Industrial and Mechatronics Engineering **8**, 718 (2014).

[17] M. C. Santos, L. V. Santana, A. S. Brandao, and M. Sarcinelli-Filho, *Uav obstacle avoidance using rgb-d system,* in *2015 International Conference On Unmanned Aircraft Systems (ICUAS)* (IEEE, 2015) pp. 312–319.

[18] 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).

[19] M. Basiri, F. Schill, P. Lima, and D. Floreano, *On-board relative bearing estimation for teams of drones using sound,* IEEE Robotics and Automation letters **1**, 820 (2016).

[20] 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).

[21] 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).

[22] F. Schilling, F. Schiano, and D. Floreano, *Vision-based drone flocking in outdoor environments,* IEEE Robotics and Automation Letters **6**, 2954 (2021).

[23] R. Menzel, J. Fuchs, A. Kirbach, K. Lehmann, and U. Greggers, *Navigation and communication in honey bees,* in *Honeybee Neurobiology and Behavior* (Springer, 2012) pp. 103–116.

[24] 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).

[25] J. Delmerico and D. Scaramuzza, *A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots,* in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 2502–2509.

[26] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, *A survey on aerial swarm robotics,* IEEE Transactions on Robotics **34**, 837 (2018).

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

[28] K. Guo, Z. Qiu, C. Miao, A. H. Zaini, C.-L. Chen, W. Meng, and L. Xie, *Ultra-wideband-based localization for quadcopter navigation,* Unmanned Systems **4**, 23 (2016).

[29] E. Soria, F. Schiano, and D. Floreano, *Predictive control of aerial swarms in cluttered environments,* Nature Machine Intelligence **3**, 545 (2021).

[30] W. Zhao, J. Panerati, and A. P. Schoellig, *Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots,* IEEE Robotics and Automation Letters **6**, 3639 (2021).

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

[32] A. Farnsworth, *Flight calls and their value for future ornithological studies and conservation research,* The Auk **122**, 733 (2005).

[33] A. Strandburg-Peshkin, C. R. Twomey, N. W. Bode, A. B. Kao, Y. Katz, C. C. Ioannou, S. B. Rosenthal, C. J. Torney, H. S. Wu, S. A. Levin, *et al.*, *Visual sensory networks and effective information transfer in animal groups,* Current Biology **23**, R709 (2013).

[34] 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).

[35] B. Broecker, K. Tuyls, and J. Butterworth, *Distance-based multi-robot coordination on pocket drones,* in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 6389–6394.

[36] K. Guo, X. Li, and L. Xie, *Ultra-wideband and odometry-based cooperative relative localization with application to multi-uav formation control,* IEEE transactions on cybernetics **50**, 2590 (2019).

[37] 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).

[38] S. Güler, M. Abdelkader, and J. S. Shamma, *Peer-to-peer relative localization of aerial robots with ultrawideband sensors,* IEEE Transactions on Control System Technology (2020).

[39] 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).

[40] 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).

[41] F. Shan, J. Zeng, Z. Li, J. Luo, and W. Wu, *Ultra-wideband swarm ranging,* IEEE INFOCOM (2021).

[42] 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* (IEEE, 2013) pp. 993–998.

[43] V. Walter, M. Saska, and A. Franchi, *Fast mutual relative localization of uavs using ultraviolet led markers,* in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2018) pp. 1217–1226.

[44] M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J.-D. Decotignie, K.-M. Doth, F. Fraundorfer, L. Kneip, D. Gurdan, *et al., Sfly: Swarm of micro flying robots,* in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012) pp. 2649–2650.

[45] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, *Collaborative monocular slam with multiple micro aerial vehicles,* in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2013) pp. 3962–3970.

[46] T. Cieslewski, S. Choudhary, and D. Scaramuzza, *Data-efficient decentralized visual slam,* in *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2018) pp. 2466–2473.

[47] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, *Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems,* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2016) pp. 1556–1561.

[48] M. Vrba, D. Heřt, and M. Saska, *Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system,* IEEE Robotics and Automation Letters **4**, 3402 (2019).

[49] M. Vrba and M. Saska, *Marker-less micro aerial vehicle detection and localization using convolutional neural networks,* IEEE Robotics and Automation Letters **5**, 2459 (2020).

[50] N. J. Sanket, C. D. Singh, C. M. Parameshwara, C. Fermüller, G. C. H. E. de Croon, and Y. Aloimonos, *Evpropnet: Detecting drones by finding propellers for mid-air landing and following,* arXiv preprint arXiv:2106.15045 (2021).

[51] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, *Drone detection using depth maps,* in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2018) pp. 1034–1037.

[52] S. Li, C. De Wagter, and G. C. H. E. de Croon, *Self-supervised monocular multi-robot relative localization with efficient deep neural networks,* arXiv preprint arXiv:2105.12797 (2021).

2

UNSUPERVISED TUNING OF FILTER PARAMETERS APPLIED TO AERIAL ROBOTS

Autonomous robots heavily rely on well-tuned state estimation filters for successful control. This chapter presents a novel automatic tuning strategy for learning filter parameters by minimizing the innovation, i.e., the discrepancy between expected and received signals from all sensors. The optimization process only requires the inputs and outputs of the filter without ground-truth. Experiments were conducted with the Crazyflie quadrotor, and all parameters of the extended Kalman filter (EKF) are well tuned after one 10-second manual flight. The proposed method has multiple advantages, of which we demonstrate two experimentally. First, the learned parameters are suitable for each individual drone, even if their particular sensors deviate from the standard, e.g., by being noisier. Second, this manner of self-tuning allows one to effortlessly expand filters when new sensors or better drone models become available. The learned parameters result in a better state estimation performance than the standard Crazyflie parameters.

Parts of this chapter have been published in IEEE Robotics and Automation Letters 4 (4), 4102-4107 (2019) [1].

2.1. INTRODUCTION

Perception heavily relies on the difference between predictions and sensory observations [2]. For robots, perception is typically implemented in the form of a filter that estimates the states based on an optimal combination of predictions and observations. For example, aerial robots combine a model of their dynamics with gyroscope and accelerometer observations for attitude estimation [3] and optical flow for motion estimation [4]. For the performance of state estimation filters such as the Kalman filter, it is essential that the parameters representing the covariance of both the prediction and observation model are set correctly. These unknown parameters are mostly determined by the human designer, with the help of expensive external measurement setups that give “ground-truth" measurements corresponding to the observations. Determining a model’s parameters in this manner is time-consuming, while the parameters are in principle only valid for the single robot with which the measurements were made. This makes robotic perception less autonomous than animal and human perception. Automatic tuning of filter parameters is an important challenge for achieving intelligent perception of robots.

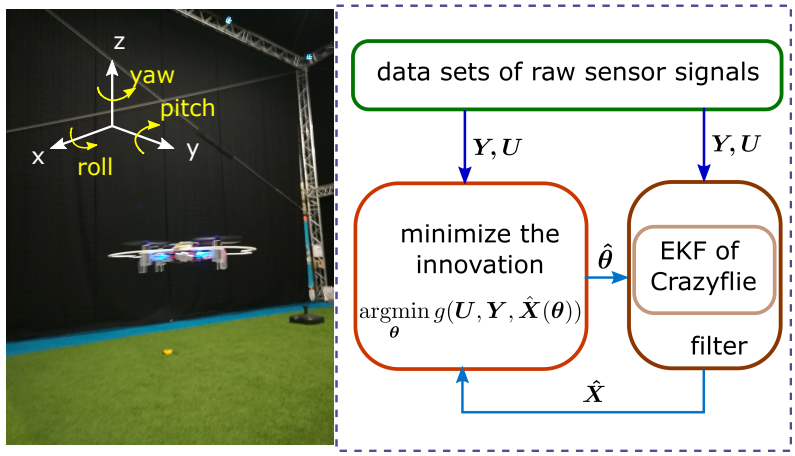


Figure 2.1: Overview of the unsupervised filter parameter tuning method. **Left:** experimental setup where a small Crazyflie quadrotor is flown manually to get an onboard data set of observations $\boldsymbol{Y}$ and control inputs $\boldsymbol{U}$, while the OptiTrack motion tracking system provides ground-truth for validation. **Right:** the unsupervised tuning process where the filter operates with parameters $\hat{\boldsymbol{\theta}}$, while the estimated states $\hat{\boldsymbol{X}}$ are utilized by the optimization block until the innovation based goal function is minimal.

A less well-known use of the difference between prediction and observation, termed “innovation" in the filtering literature, is to learn the filter’s parameters. Adaptive filters have been proposed to identify the noise covariance parameters based on algebraic methods, such as correlation [5], Bayesian [6], covariance matching [7, 8], and maximum likelihood [9, 10]. The advantage of these methods is that theoretical guarantees can be given, when the assumptions of the filter are correct. Initially, mathematical derivations were made for “simple" systems, e.g., linear systems with a single output [11, 12]. Later work extended these methods to more complex, nonlinear systems [13], even giving guarantees

on upper bounds of the covariance [14]. However, the above methods require a particular filter formula that is amenable to mathematical derivation, in order to calculate the covariance or likelihood function. This leads to a loss of generality to arbitrary filters or systems.

To reduce the dependency on the mathematical structure of the filter, more recently tuning methods have been proposed that can learn filter parameters based on input and output data of the filter, treating the filter as a black box. Many of these methods require ground-truth measurements of states which are not available to the robot itself. Examples include a deep learning method proposed for covariance estimation [15], and another filter tuning method represented by an optimization problem [16]. Other methods assume knowledge of some of the noise characteristics. For example, in [17], an optimization based EKF estimated process noise without prior knowledge, while the measurement noise was assumed to be known. Finally, there are methods that can function without prior knowledge or ground-truth measurements. A self-tuning mixture model was proposed in [18], which not only tuned the distribution parameters but also estimated the states without any prior knowledge. In [19], several criteria for parameter learning of Kalman filter were proposed, and the selected one maximized the measurement likelihood. A tuning method for an unscented Kalman filter (UKF) was proposed based on optimization method without ground-truth to maximize the likelihood of measurements [20]. Until now, these studies have only considered the measurement likelihood rather than the whole sensor likelihood meaning that the filter inputs are not considered for determining the process noise. And since most of these studies focus only on simulation, their accuracy for real robots still remains largely unknown.

In this work, an unsupervised tuning method (illustrated in Fig. 1) is proposed for determining filter parameters. It only requires the information available to the robot, without a need for ground-truth measurements. The method is independent of a filter's structure such that it allows for different filters and different sensor arrangements. The use of control inputs can make process noise identification more accurate since acceleration sometimes is modelled as a control input to the filter. The contributions of this work consist of: 1) automatic filter parameter tuning without relying on ground-truth; 2) novel intuitive optimization with generality to any filter; 3) scalability to more sensor inputs and measurements, and the inclusion of control inputs; and 4) experimental results showing that the approach can improve upon the state estimation of a commercial flying robot.

The rest of this chapter is organized as follows. Section 2.2 introduces the model preliminaries and the tuning problem. In Section 2.3, the optimization-based unsupervised tuning method is presented. Section 2.4 illustrates the effectiveness of the proposed filter tuning method using several tests. Section 2.5 presents our conclusions.

2.2. PRELIMINARY

To introduce the tuning problem of filter parameters, a generalized nonlinear model of robots will be given in this section. This model shows the prediction and measurement process of state estimation, followed by a specific EKF filter implemented on a quadrotor as a study case, with its unknown noise covariance parameters.

2.2.1. NONLINEAR STOCHASTIC MODELING

Consider a robot with a nonlinear discrete model described by following equations:

$$\begin{aligned} \boldsymbol{x}_k &= f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{q}, \\ \boldsymbol{y}_k &= h(\boldsymbol{x}_k) + \boldsymbol{r} \end{aligned} \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{q} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^p$ and $\boldsymbol{r} \in \mathbb{R}^p$ denote states, inputs, process noise, measurements and measurement noise, respectively. $f(\cdot)$ and $h(\cdot)$ are transition function from time step $k-1$ to k and observation function. Assume Gaussian distributions with zero means for the process and measurement noise and corresponding covariance matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{p\times p}$, such that $\boldsymbol{y}_k \sim \mathcal{N}(h(\boldsymbol{x}_k), \boldsymbol{R})$ and $\boldsymbol{x}_k \sim \mathcal{N}(f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}), \boldsymbol{Q})$.

Based on the above assumptions, various filters are designed to estimate robot states represented by $\hat{\boldsymbol{x}}_k$. The tuning method of this chapter will only rely on inputs $\boldsymbol{u}_k$, observations $\boldsymbol{y}_k$ and estimate states $\hat{\boldsymbol{x}}_k$ from an arbitrary filter. As EKF is arguably the most widely used estimator for the nonlinear systems, it is chosen for carrying out the filter parameter tuning in this chapter. The specific definition of the commonly used EKF can be found in [16].

2.2.2. PROBLEM FORMULATION

The model and EKF filter for state estimation of a Crazyflie commercial quadrotor, are in line with the general model Eq. (2.1) and the normal EKF in [16]. As the specific equations are given in [21], this chapter will not repeat them except for some necessary variables.

In the EKF of Crazyflie, the state vector is $\boldsymbol{x} = [\boldsymbol{\xi}, \boldsymbol{\rho}, \boldsymbol{\delta}]$, where $\boldsymbol{\xi} = [x, y, z] \in \mathbb{R}^3$ denotes the three dimensional position of the quadrotor in earth coordinate, $\boldsymbol{\rho} = [vx, vy, vz] \in \mathbb{R}^3$ is the 3-axis velocity in body coordinates, and $\boldsymbol{\delta} \in \mathbb{R}^3$ represents the attitude error vector which is used to update the three attitude angles of pitch ϕ, roll θ and yaw ψ. For simplicity, following analysis will consider $\boldsymbol{\delta}$ as attitude.

Unlike theoretical physical models, most practical aerial robots utilize sensor signals as *control* inputs into the filter. For example, the accelerometer readings are inserted into the filter as controlled accelerations, while the gyro readings are represented as controlled rotation rates. Therefore, the process noise stems mainly from these sensors rather than from the uncertainty in the controlled motions. Suppose all elements of noise are stochastically uncorrelated. Then the unknown noise covariance parameters $\boldsymbol{Q}$ and $\boldsymbol{R}$ can be formulated as diagonal matrices represented by $\boldsymbol{Q} = \text{diag}[q_1\ q_2\ ...\ q_n]$ and $\boldsymbol{R} = \text{diag}[r_1\ r_2\ ...\ r_p]$.

The unknown process noise covariance in the Crazyflie EKF is composed of 9 elements, in which $q_1 = q_2 = \sigma_{axy} t^2/2$, $q_3 = \sigma_{az} t^2/2$, $q_4 = q_5 = \sigma_{axy} t$, $q_6 = \sigma_{az} t$, $q_7 = q_8 = \sigma_{gxy} t$ and $q_9 = \sigma_{gz} t$, where σ_{axy} and σ_{az} denote the standard deviation of horizontal and vertical acceleration noise (m/s^2) in body coordinates. σ_{gxy} and σ_{gz} are standard deviation of gyroscope noise (rad/s) around x, y, and z axis shown in Fig. 2.1, respectively .

The measurement noise covariance matrix is defined by $\boldsymbol{R} = \text{diag}[\sigma_{fxy}\ \sigma_{lz}]$, in which σ_{fxy} and σ_{lz} represents the standard deviation of the 2-axis optical flow noise in pixels and in which σ_{lz} represents the range noise (m) from a tiny laser pointing to ground. Therefore, the final filter parameters of quadrotor to be determined are given as follows:

$$\boldsymbol{\theta} = [\sigma_{axy}\ \sigma_{az}\ \sigma_{gxy}\ \sigma_{gz}\ \sigma_{fxy}\ \sigma_{lz}\ k_{va}], \tag{2.2}$$

where k_{va} is an auxiliary parameter to calculate acceleration with velocity, of which the role will be discussed later. Obviously, the parameters in $\boldsymbol{\theta}$ play an important role in estimation performance of EKF. In the next section, an optimization method will be designed to calculate the unknown parameters $\boldsymbol{\theta}$ in Eq. (2.2).

2.3. METHOD

This section will approach the filter parameter tuning as an optimization problem. First, a novel scalar goal function $g(\cdot)$ is proposed to evaluate the filter performance. Then the tuning process would consist in iterations of minimizing the scalar goal function, realized by a stochastic gradient descent method.

2.3.1. GOAL FUNCTION

The literature on adaptive Kalman filtering typically sets the goal function with innovation to maximize the likelihood of measurements as follows.

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \| h(\hat{\boldsymbol{x}}_i) - \boldsymbol{y}_i \|_2. \tag{2.3}$$

This function represents the average of measurement error norm.

This chapter extends this traditional measurement based goal function in (2.3) by also maximizing the likelihood of input signals, such as in this case $\boldsymbol{u} = [ax\ ay\ az\ gx\ gy\ gz]$ including 3-axis acceleration and 3-axis gyroscope, which are not directly represented by states. To this end, we use the physical relationship between the filter states and inputs, i.e., in this case the approximated linear relation between velocity and drag acceleration denoted by $[ax\ ay] = -k_{va}[vx\ vy]$ in wind still conditions. Thus, one can obtain $\boldsymbol{u} = \mathscr{F}(\boldsymbol{x}) = [-k_{va}vx\ -k_{va}vy\ \dot{vx}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]$. Incorporating this "input innovation" into the goal function Eq. (2.3), the overall goal function is given by

$$g(\boldsymbol{U}, \boldsymbol{Y}, \hat{\boldsymbol{X}}(\boldsymbol{\theta})) = \frac{1}{N} \sum_{i=1}^{N} (\| h(\hat{\boldsymbol{x}}_i) - \boldsymbol{y}_i \|_1 + \| \mathscr{F}(\hat{\boldsymbol{x}}_i) - \boldsymbol{u}_i \|_1) \tag{2.4}$$

$$\hat{\boldsymbol{\theta}}_{\text{best}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, g(\boldsymbol{U}, \boldsymbol{Y}, \hat{\boldsymbol{X}}(\boldsymbol{\theta})) \tag{2.5}$$

Where data sets $\boldsymbol{Y}$, $\boldsymbol{U}$ and $\hat{\boldsymbol{X}}$ are length N sequences of observations $\boldsymbol{y}$, control inputs $\boldsymbol{u}$ and estimated states $\hat{\boldsymbol{x}}$, respectively. Here, we select the L1 loss function as it does not overpenalize large but unlikely errors, and is therefore more robust to non-Gaussian distributions with outliers [22], which are typical for many real-world sensors.

In real implementation, expectations $h(\hat{\boldsymbol{x}}_i)$ and $\mathscr{F}(\hat{\boldsymbol{x}}_i)$ tend to be smooth with $\hat{\boldsymbol{\theta}}_{\text{best}}$. Therefore, noisy signals are smoothed generally to keep the goal function more relevant to filter results. Also, since the laser measurement is considerably precise and it is coupled with the optical flow innovation, the laser cost function is weighted as $50|h_{lz}(\hat{\boldsymbol{x}}_i) - y_{lz}|$ in Eq. (2.4), in which we have left out the weights for brevity.

2.3.2. TUNING PROCESS

The entire unsupervised tuning process is shown in Fig. 2.1. It starts with an initial parameters $\boldsymbol{\theta}_0$ in which all elements are set to zero. The filter runs once with the initial parameter

and outputs the sequence of estimated states $\hat{\boldsymbol{X}}$. Then the optimization block utilizes $\hat{\boldsymbol{X}}$ to infer all sensor data, and minimizes the difference between inferred and real sensor signals for tuning $\hat{\boldsymbol{\theta}}$. This tuning step iterates until a minimal value of the goal function is obtained.

Given the goal function, this chapter deploys stochastic gradient descent as the optimization method. Since our proposed method regards the goal function as a black box problem, it would be possible to use any other metaheuristic optimization method.

The incorporated random process can help jump out of local minima to some extent. Because after one gradient descent, the method will select a fixed number of random parameter initalizations for comparison to see if there exists a parameter set that has a smaller goal value. A more systematic description of this tuning method is given in the following algorithm. $\boldsymbol{\theta}_{lb}, \boldsymbol{\theta}_{ub}, N_s, fm()$, and $random()$ denote the lower bound, upper bound, number of searches that improve the current estimation result, fmincon() function and random() function in matlab, respectively.

Algorithm 1 Unsupervised tuning of EKF

procedure $f_{\text{opti}}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_{\text{lb}}, \boldsymbol{\theta}_{\text{ub}}, N_{\text{s}})$ ▷ Tuning function
  $\hat{\boldsymbol{\theta}}_{\text{opti}} \leftarrow fm(g(\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{x}}(\boldsymbol{\theta})), \boldsymbol{\theta}_0, \boldsymbol{\theta}_{\text{lb}}, \boldsymbol{\theta}_{\text{ub}})$ ▷ gradient descent function e.g. fmincon in matlab
  $g_{\text{opti}} \leftarrow g(\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{x}}(\hat{\boldsymbol{\theta}}_{\text{opti}}))$
  $n_{\text{loop}} \leftarrow 0$
  while $n_{\text{loop}} < N_{\text{s}}$ **do**
    $\hat{\boldsymbol{\theta}}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{ub}} \times random(length(\boldsymbol{\theta}), 1)$
    if $g(\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{x}}(\hat{\boldsymbol{\theta}}_{\text{new}})) \leq g_{\text{opti}}$ **then**
      $\hat{\boldsymbol{\theta}}_{\text{opti}} \leftarrow fm(g(\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{x}}(\boldsymbol{\theta})), \boldsymbol{\theta}_{\text{new}}, \boldsymbol{\theta}_{\text{lb}}, \boldsymbol{\theta}_{\text{ub}})$
      $g_{\text{opti}} \leftarrow g(\boldsymbol{u}, \boldsymbol{y}, \hat{\boldsymbol{x}}(\hat{\boldsymbol{\theta}}_{\text{opti}}))$
      $n_{\text{loop}} \leftarrow 0$
    else
      $n_{\text{loop}} \leftarrow n_{\text{loop}} + 1$
  $\hat{\boldsymbol{\theta}}_{\text{best}} \leftarrow \hat{\boldsymbol{\theta}}_{\text{opti}}$ ▷ return the best parameter

2.3.3. HARDWARE SETUP

To validate the effectiveness of the proposed tuning method for EKF, experiments are conducted on the commercial quadrotor Crazyflie 2. It uses a MPU-9250 IMU consisting of a 3-axis gyroscope and a 3-axis accelerometer. These are considered as inputs of the onboard EKF. Also, VL53L0x and PMW3901MB sensors provide the range to ground and optical flow measurements, respectively. These sensors are fused by the onboard EKF with preset parameters from the factory.

The tuning method relies on one manual flight of this vehicle. Maneuvering the drone in all dimensions is key to collecting a suitable sensor data set. Data will be stored on a SD card, which is further used by an offline computer to calculate all required filter parameters. This process takes several minutes for laptop with i7-6600U CPU at 3.40 GHz. In principle, a slower version of this optimization process could take place onboard the

Crazyflie when not in flight.

To validate filter performance, a motion capture system OptiTrack is used for providing external measured ground-truth of 3-axis attitude and 3-axis position.

2.4. EXPERIMENTAL RESULTS

The performance of the proposed unsupervised tuning method is illustrated by several experimental scenarios shown in this section, in order to validate its tuning results, estimation performance, robustness to extra noise and efficiency for expanding filters. The data sets $\boldsymbol{Y}$ and $\boldsymbol{U}$ for tuning are collected from 20 manual flights of the quadrotor operated in all dimensions, whereas half data sets are used for tuning and the other half are used for testing. The duration of these data sets ranges from 5 to 15 seconds with 1000Hz sampled frequency.

2.4.1. RESULTS OF UNSUPERVISED TUNING

By implementing Algorithm 1 on sensor data collected from a real Crazyflie quadrotor, all related noise parameters are calculated and given as $\hat{\boldsymbol{\theta}}_{\text{best}}$. Table 2.1 gives insight into the optimized parameter values, by showing the median values over the 10 different data sets. The distribution of $\hat{\boldsymbol{\theta}}_{\text{best}}$ tuned from 10 training data sets is shown in Fig. 2.2 with 25th and 75th percentiles as the boundaries and the few outliers can be neglected. σ_{axy} tends to be far from the preset value in the Crazyflie simply because horizontal acceleration only works for the onboard EKF when the quadrotor is in freefall or carried.

At the same time, expected and real sensor data are shown in Fig. 2.3 to show the effectiveness of the goal function on a training data set. From the figure, we can see that all dimensional raw sensor data are well inferred by utilizing this tuning method.

2.4.2. FILTER RESULTS WITH TUNED PARAMETERS

In this subsection, we compare the performance of the tuned parameter set with that of the onboard EKF with preset parameters. Estimation performance on position, velocity and attitude using the filter with $\hat{\boldsymbol{\theta}}_{\text{best}}$ is shown in Fig. 2.4. Instead of taking a specific parameter set, we here take the median values as shown in Table I for the tuned filter.

Fig. 2.4 indicates that both the EKF with tuned parameters and onboard EKF with original parameters have accurate position estimation. The tuned EKF tends to be closer to the ground-truth at various times. We can also see the tuned filter has less attitude estimation error than that with original parameters within some time. Filters with both sets of parameters can estimate the velocity with similar precision as seen from Fig. 2.4. Traditional tuning only with measurement likelihood has larger error than our method, especially at x, y and yaw axes due to the lack of the input likelihood. The corresponding tuned parameter $\hat{\boldsymbol{\theta}}_{\text{noInput}}$ in Table 2.1 also shows its wrong estimation of the yaw covariance and the drag coefficient.

Fig. 2.5 summarizes the comparison between the automatically tuned filter parameters and the preset parameters, showing the error distribution of all estimated states on 10 test data sets. Root mean square error (RMSE) is used in this figure. The state estimates of the automatically tuned filter attain a lower RMSE for all states.

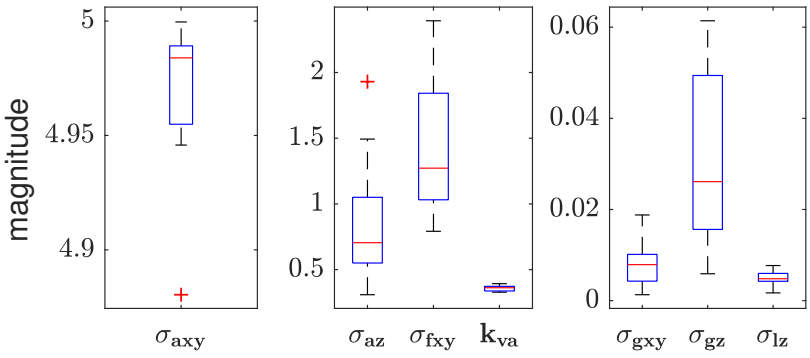


Figure 2.2: Distribution of filter parameters tuned based on 10 training data sets, in which corresponding units are $\sigma_{axy}(m/s^2)$, $\sigma_{az}(m/s^2)$, $\sigma_{gxy}(rad/s)$, $\sigma_{gz}(rad/s)$, $\sigma_{fxy}(pixel)$, and $\sigma_{lz}(m)$.

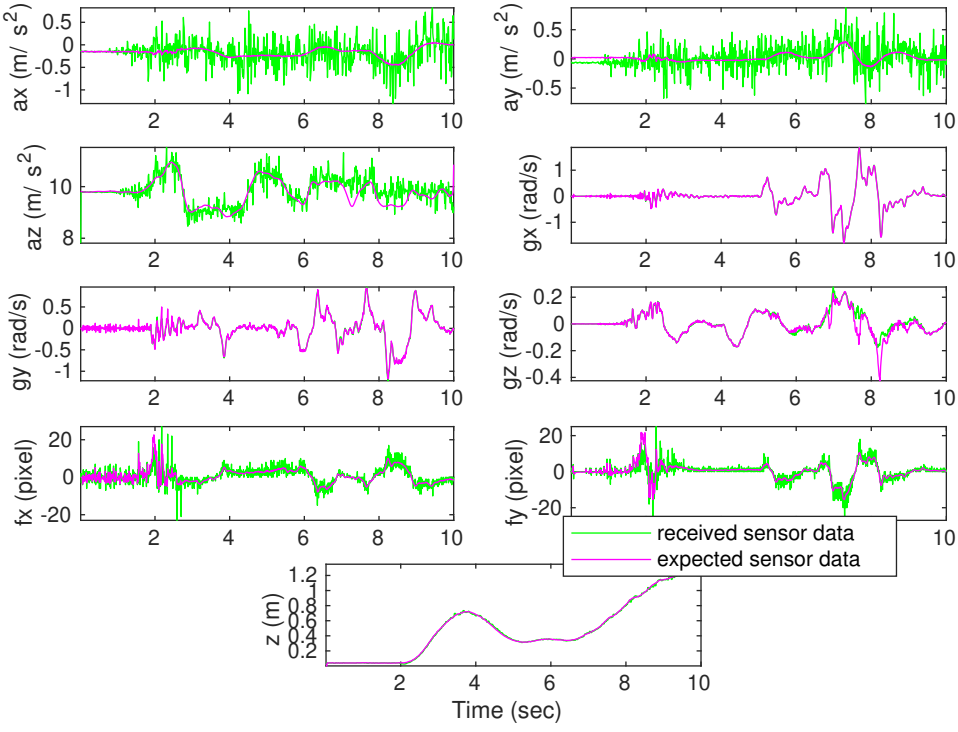


Figure 2.3: Expected (magenta) and received (green) sensor data for all sensors including acceleration of ax, ay and az, gyroscope of gx, gy and gz, flow of fx and fy, and altitude range z, based on a training data set.

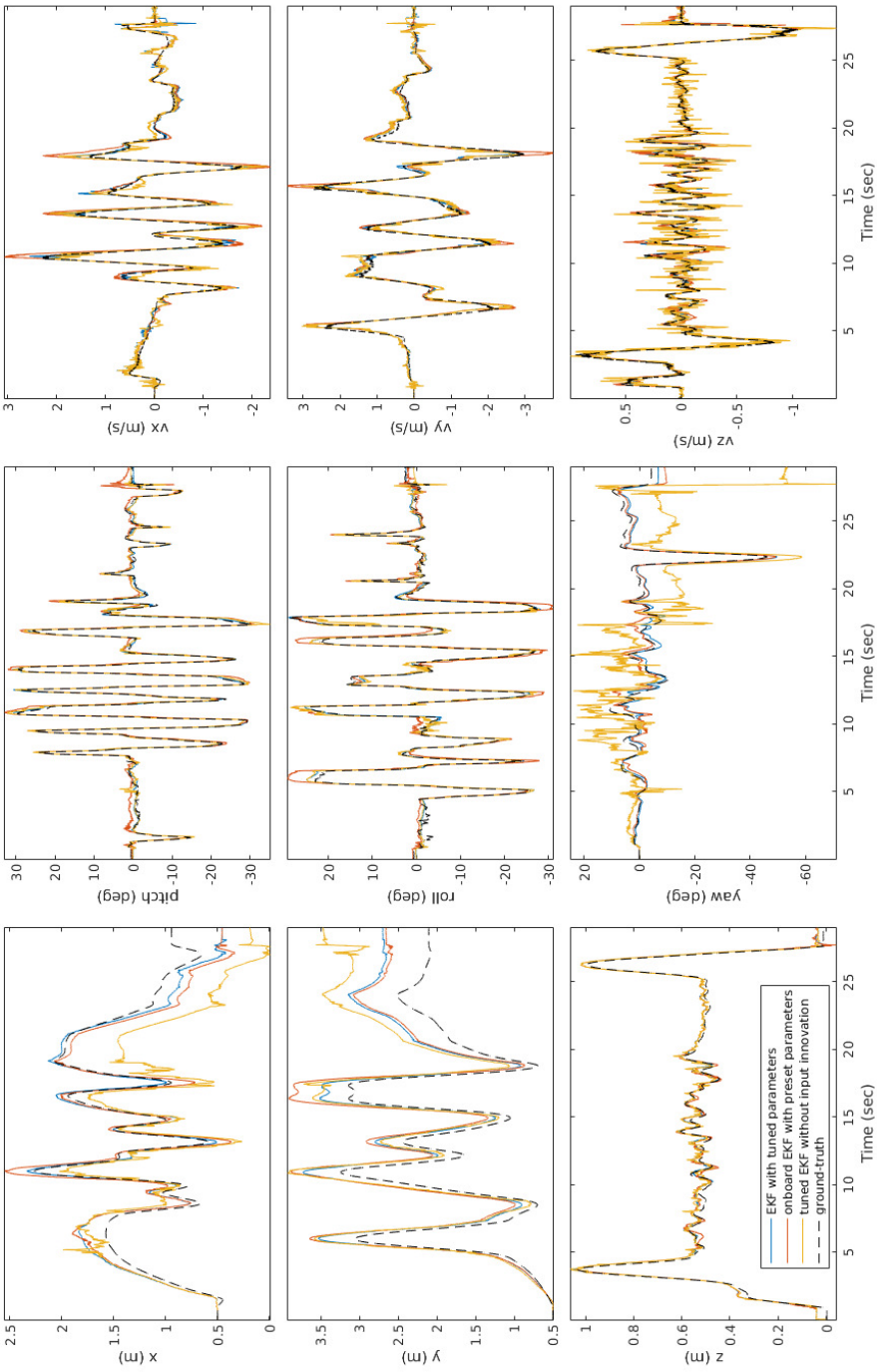


Figure 2.4: The 3-dimensional position, attitude and velocity estimated by EKF with tuned parameters (blue), EKF with Crazyflie parameters (red), tuned EKF without the input innovation term (yellow), as well as ground-truth from OptiTrack (black), respectively, based on a test data set.

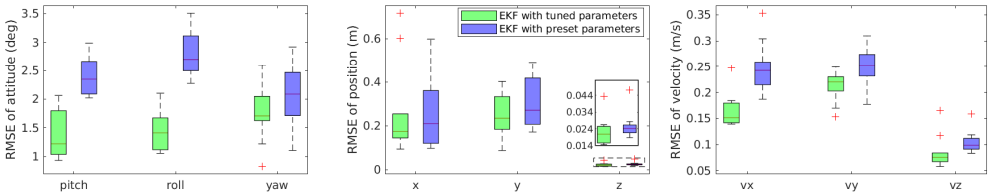


Figure 2.5: Comparison between the self-tuned filter and the standard Crazyflie filter based on 10 test data sets.

2.4.3. TUNING RESULTS WITH EXTRA NOISE AND EXPANDED MODEL

This section continues analyzing the effectiveness of the method with regards to extra noise and expanded filter structure on a training data set.

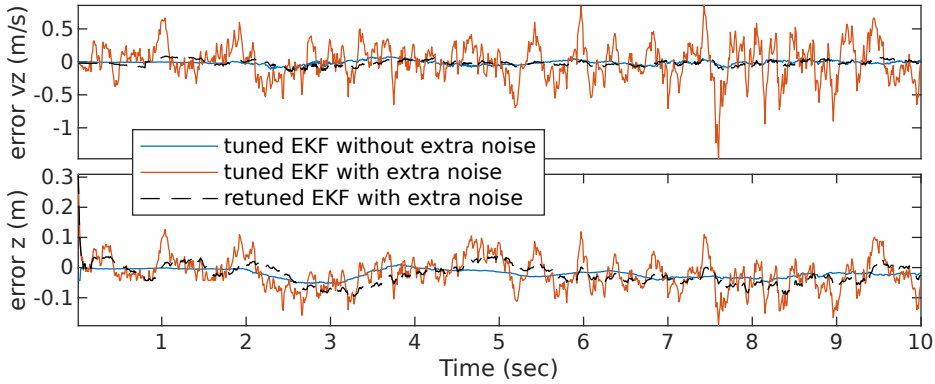


Figure 2.6: Altitude and corresponding velocity estimation error with respect to extra noise on laser range measurement on a training data set.

The first scenario is adding extra noise to some dimensions of the data sets. For example, a white Gaussian noise with 5dB is added to the normally accurate altitude measurement, which is created by AWGN function in MATLAB. Fig. 2.6 shows that the EKF with previous tuned parameter has a larger vertical velocity and position estimation error due to the extra noise. After retuning the filter with the extra noise, the new parameters enable the filter to largely reduce the estimation error. This can also be seen from Table 2.1; the standard deviation of the laser measurement increases with a factor 10 due to the extra noise. Moreover, σ_{az} slightly decreases so that the vertical velocity relies more on the accelerometer readings. Also, Fig. 2.7 illustrates that the retuned EKF has less estimation error in all states compared to that with unchanged parameters.

Another test is constructed for an expanded filter model, which uses thrust from command rather than from accelerometer to predict velocity. The latter one is sometimes noisy due to mechanical vibrations. The thrust command is smooth but mapping it to thrust requires knowledge of mass which varies significantly between drones. In fact, the

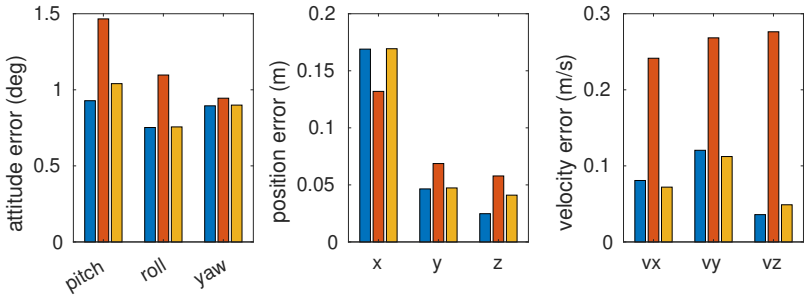


Figure 2.7: Estimation error on all states of the EKF with tuned parameter without noise (blue), previous tuned parameter with noise (red), and retuned parameter with noise (yellow), respectively, on a training data set.

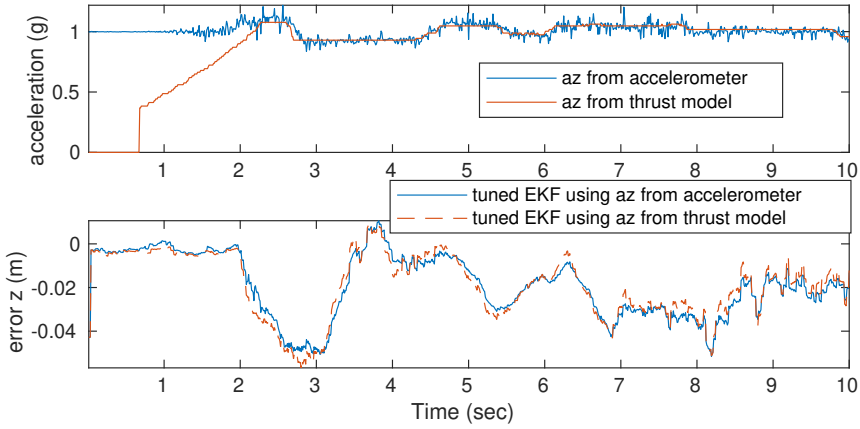


Figure 2.8: Tuning of a filter with the trust model on a training data set.

Crazyflie filter has a term for the thrust, but this is commented out in the code mentioning this mass variability problem. The proposed self-tuning method can simply learn the thrust function for the specific drone by adding a mapping coefficient k_{Ta} into $\hat{\boldsymbol{\theta}}$. After training, the unsupervised tuning method gives an exact $k_{Ta} = 1.4654$ that maps thrust command to acceleration as $az = k_{Ta} * T_{cmd}/65536$ shown in Fig. 2.8. We use this thrust model only when it is flying because the ground also provides a force to the robot when it is not flying between 0s-2s in Fig. 2.8. Also, this figure shows that even if the input is replaced by the thrust model, the vertical position estimation error is still comparable.

2.4.4. TUNING OF A UKF ON THE EUROC MAV DATASET

The method is further tested on the commonly used UKF formulated in [20] for attitude estimation, and the IMU data is from the public EuroC MAV dataset. The unknown 3-axis gyroscope covariance $\boldsymbol{Q}$ and 2-axis accelerometer covariance $\boldsymbol{R}$ are tuned from initial

Table 2.1: EFK parameters from Crazyflie, median of 10 tuned tests, tuning without input innovation, tuning without extra noise, tuning with extra noise, tuning with thrust model, respectively.

$\boldsymbol{\theta}$	σ_{axy}	σ_{az}	σ_{gxy}	σ_{gz}	σ_{fxy}	σ_{lz}	k_{va}	k_{Ta}
$\boldsymbol{\theta}_{\text{CF}}$	0.5000	1.0000	0.1000	0.1000	0.2500	0.0025		
$\hat{\boldsymbol{\theta}}_{\text{median}}$	4.9839	0.7048	0.0079	0.0261	1.2720	0.0048	0.3621	
$\hat{\boldsymbol{\theta}}_{\text{noInput}}$	3.4744	2.5060	0.0723	1.8408	0.8837	0.0013	2.4981	
$\hat{\boldsymbol{\theta}}_{\text{noNoise}}$	4.9832	0.7513	0.0056	0.0500	2.5656	0.0095	0.3193	
$\hat{\boldsymbol{\theta}}_{\text{noise}}$	4.1874	0.4380	0.0069	0.0461	2.6429	0.0926	0.3514	
$\hat{\boldsymbol{\theta}}_{\text{thrust}}$	4.5725	0.7404	0.0157	0.0897	3.7352	0.0300	0.3233	1.4654

[1,1,1,1,1] to [5.8e-8,2.8e-7,0.6443,2.1987,2.5532]. Results from the Fig. 2.9 show that the tuned UKF can estimate the attitude accurately, which validates the efficacy of the proposed method on different filters and data sets.

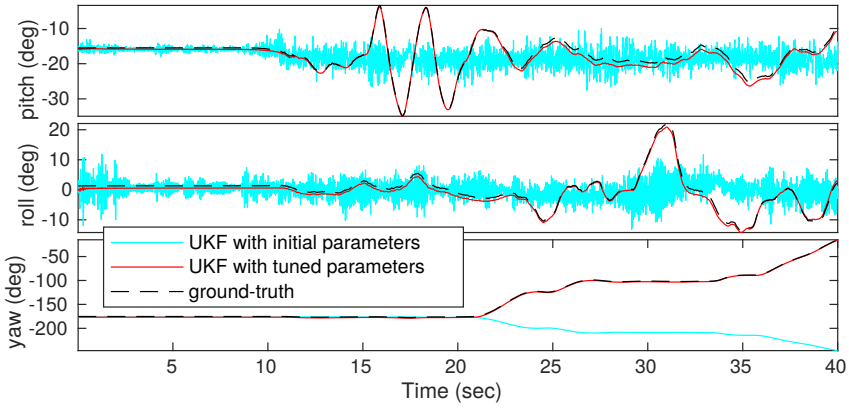


Figure 2.9: Attitude estimation of UKF with initial parameters, tuned parameters, and ground-truth on the public EuroC MAV dataset.

2.5. CONCLUSIONS

We have proposed an unsupervised parameter tuning method for arbitrary filters by minimizing the difference between predictions and observations. The effectiveness of the proposed method has been validated using real-world data of a tiny Crazyflie quadrotor. The results show that the filter with self-tuned parameters has more precise estimation than the filter with preset parameters. Moreover, we have shown that this tuning method can reject extra noise, by adding noise to the sensor data, and it can also easily extend to an expanded filter model, identifying new unknown parameters. Finally, the method is validated to work on other filters like a UKF.

However, there are some limitations of this method. For example, in the current chapter we have based the input innovation term on our knowledge of the physical system. Future work could focus on learning the relation between states and control inputs if this relationship is unknown. Of course, if there is no inherent relationship between the state and the control inputs, then the innovation term will not be beneficial. Furthermore, the performance may degrade if some sensors have a covariance that changes heavily over time. Future work could address issues such as largely varied covariance and automatic selection of the innovation weight for the laser.

REFERENCES

[1] S. Li, C. De Wagter, and G. C. H. E. de Croon, *Unsupervised tuning of filter parameters without ground-truth applied to aerial robots,* IEEE Robotics and Automation Letters **4**, 4102 (2019).

[2] H. E. Den Ouden, P. Kok, and F. P. De Lange, *How prediction errors shape perception, attention, and motivation,* Frontiers in psychology **3**, 548 (2012).

[3] M. Karásek, F. T. Muijres, C. De Wagter, B. D. Remes, and G. C. H. E. de Croon, *A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns,* Science **361**, 1089 (2018).

[4] A. Santamaria-Navarro, J. Sola, and J. Andrade-Cetto, *High-frequency mav state estimation using low-cost inertial and optical flow measurement units,* in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (IEEE, 2015) pp. 1864–1871.

[5] P. S. Maybeck, *Stochastic models, estimation, and control*, Vol. 2 (Academic press, 1982).

[6] H. Hu and G. Kantor, *Introspective evaluation of perception performance for parameter tuning without ground truth.* in *Robotics: Science and Systems* (2017).

[7] K. Myers and B. Tapley, *Adaptive sequential estimation with unknown noise statistics,* IEEE Transactions on Automatic Control **21**, 520 (1976).

[8] B. J. Odelson, A. Lutz, and J. B. Rawlings, *The autocovariance least-squares method for estimating covariances: application to model-based control of chemical reactors,* IEEE Transactions on Control Systems Technology **14**, 532 (2006).

[9] R. Mehra, *Approaches to adaptive filtering,* IEEE Transactions on Automatic Control **17**, 693 (1972).

[10] A. Mohamed and K. Schwarz, *Adaptive kalman filtering for ins/gps,* Journal of geodesy **73**, 193 (1999).

[11] R. Reynolds, *Robust estimation of covariance matrices,* IEEE Transactions on Automatic Control **35**, 1047 (1990).

[12] P. R. Bélanger, *Estimation of noise covariance matrices for a linear time-varying stochastic process,* Automatica **10**, 267 (1974).

[13] F. V. Lima, M. R. Rajamani, T. A. Soderstrom, and J. B. Rawlings, *Covariance and state estimation of weakly observable systems: Application to polymerization processes,* IEEE Transactions on Control Systems Technology **21**, 1249 (2013).

[14] J. Duník, O. Straka, and M. Šimandl, *On autocovariance least-squares method for noise covariance matrices estimation,* IEEE Transactions on Automatic Control **62**, 967 (2017).

[15] K. Liu, K. Ok, W. Vega-Brown, and N. Roy, *Deep inference for covariance estimation: Learning gaussian noise models for state estimation,* in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 1436–1443.

[16] T. D. Powell, *Automated tuning of an extended kalman filter using the downhill simplex algorithm,* Journal of Guidance, Control, and Dynamics **25**, 901 (2002).

[17] M. Karasalo and X. Hu, *An optimization approach to adaptive kalman filtering,* Automatica **47**, 1785 (2011).

[18] T. Pfeifer and P. Protzel, *Robust sensor fusion with self-tuning mixture models,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 3678–3685.

[19] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, *Discriminative training of kalman filters.* in *Robotics: Science and systems*, Vol. 2 (2005) p. 1.

[20] L. A. Scardua and J. J. Da Cruz, *Complete offline tuning of the unscented kalman filter,* Automatica **80**, 54 (2017).

[21] 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.

[22] C. Robert, *The Bayesian choice: from decision-theoretic foundations to computational implementation* (Springer Science & Business Media, 2007).

3

AN AUTONOMOUS SWARM OF MICRO FLYING ROBOTS WITH RANGE-BASED RELATIVE LOCALIZATION

Accurate relative localization is an important requirement for a swarm of robots. This chapter presents an autonomous multi-robot relative localization technique based on wireless ranging with ultra wide-band (UWB), which features the fastest inter-robot ranging communication to date (333 Hz). We investigate the following aspects of the system, which have been largely ignored in the literature for other wireless-ranging relative localization schemes. First, we study the effect of unobservable states, proving both theoretically and empirically that these do not form as substantial a problem as previously assumed. Second, we introduce an automatic initialization procedure and study the robots' filters' convergence times (on average 20 seconds). Third, we investigate the scalability of the proposed technique to larger numbers of robots. For a minimal update rate of 5 Hz, the maximal group size is 12 for a fully connected communication scheme and 67 for a leader-follower communication scheme. Real-world experiments are conducted on teams of up to five tiny 33-gram Crazyflie drones, demonstrating autonomous formation flight and coordinated flight through a window. All results indicate the effectiveness of the proposed relative positioning method for a broad range of multi-robot systems. Video and code can be found at `https://shushuai3.github.io/autonomous-swarm/`

Parts of this chapter have been submitted to IEEE Transactions on Robotics [1]

3.1. INTRODUCTION

Aerial multi-robot systems have been widely studied recently because of their advantages, such as efficiency of parallel task processing [2], the cooperative ability of performing team missions [3, 4], and the ability of smaller drones to operate safely in confined spaces and near humans [5, 6]. These systems require the relative position of peer robots to enable intra-swarm collision avoidance and coordination. Therefore, the development of accurate relative localization technologies is an important challenge to be solved.

One solution to the above is to use external positioning systems. For example, there are many indoor examples of teams of multiple quadrotors which are localized with a motion capture system [2, 7, 8]. In [9], 30 drones exhibit outdoor flocking behavior by relying on a Global Navigation Satellite System (GNSS) for positioning. Global positioning system (GPS) is employed for formation control of multiple quadrotors in [10]. Alternatively, fixed wireless UWB beacons are another external system that can provide positions for multiple robots [11]. All these systems have illustrated the potential of drone swarms, of how they can pass through a window together [7] or how they can perform beautiful choreographies [8]. However, the availability of the aforementioned systems is too limited to allow for autonomous flight in unknown and GPS-denied environments.

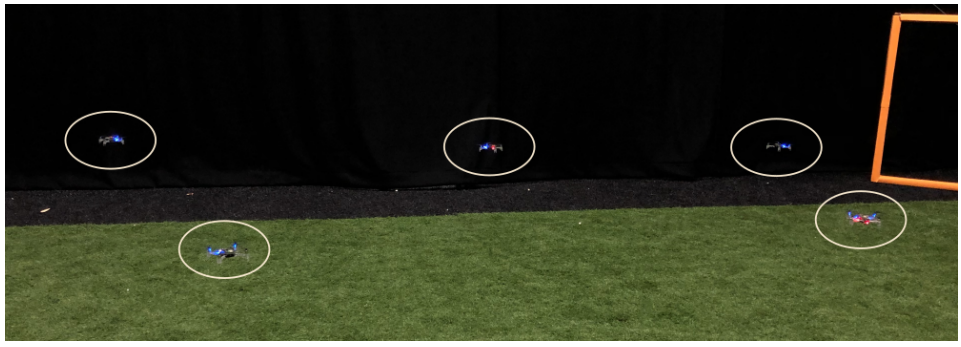


Figure 3.1: Autonomous flight of multiple Crazyflie2 quadrotors without GPS or motion capture system, fully based on relative localization using onboard sensing information of velocity, yaw rate, height and ranging measurements between any two robots.

Other techniques allow for onboard relative localization between multiple robots. Several implementations are based on vision. To simplify the task, [12] and [13] designed a simple pattern that could be detected by a monocular camera to calculate the relative position of other vehicles. In [4], a follower tracks an April Tag mounted on the leader, enabling the duo to perform a collaborative task. Some more recent techniques involve detecting active LED tags [14] and ultraviolet light [15]. Vision-based methods have intrinsic scalability, i.e., no limitation on the number of robots. However, these vision-based methods are sensitive to the visibility of the markers or robots, which depends on aspects such as the size of the marker or robots, and the field of view of the robots' cameras. Markerless detection requires heavy onboard computation such as [16] and [17], and a large localization error occurs when inter-robot distance getting far. This creates an important limitation for its application in tiny exploration robots. Another vision-based strategy is

not to have the robots detect and localize each other, but to perform a formation flight based on visual-inertial odometry [18]. However, this system requires a known initial position of each drone, and can drift over time, potentially leading to losing formation or even collisions.

Other studies have also seen the exploration of alternative technologies such as sound-based relative localization [19], or infra-red relative localization [20], although these require larger sensor arrays to be mounted on the robot, which is impractical for low-cost micro multiple robots.

As an alternative approach, relative localization based on wireless communication between drones has the advantages of being light-weight and omnidirectional. Here, the robots use antennas to exchange sensory information (e.g., velocity, yaw rate, height) and combine these with relative range measurements obtained from the antennas. This method was explored by [21] and [22, 23] for aerial robots. These studies required the knowledge of a common orientation, requiring the use of e.g. interference-prone magnetometers. This limitation was overcome in [24], demonstrating a system of 3 drones in leader-follower flight.

Although wireless-ranging-based relative localization is promising for use in robot swarms, it has a few potential fundamental issues. First, there are considerable parts of the relative state space that are *unobservable*, meaning that the EKFs may not converge to the true state. It is essential to study the effects of this, since the unobservable part of the state includes important cases such as that of parallel robot velocities - as happens in formation flight. Second, when starting a mission, robots have to initialize their filter. Before the filter is converged, they cannot reliably localize and avoid other robots. So it is important to develop an automatic initialization procedure, a matter ignored by the literature. Third, bandwidth for communication is always limited. Since the robots need to communicate in order to localize with respect to each other, there is a limit to the number of robots present within a given communication range. The scalability of wireless-based relative ranging is currently unclear.

In this chapter, we propose a novel wireless-ranging-based relative localization scheme, for which we study the three potential fundamental issues mentioned above. We show that the proposed localization system works well for multi-robot groups. It has many advantages over recent research, such as light and low-cost estimation compared to visual-inertial UWB fusion [25]; and substantially higher speed (333 Hz) ranging communication than that in previous studies, e.g., 40 Hz [26] and 10 Hz [27]. These fast ranging measurements enhance the estimation update speed, localization precision, and are beneficial for scaling up compared to previously proposed wireless-ranging schemes. Specifically, this work makes the following contributions:

- A novel, fully autonomous, onboard relative localization scheme implemented on multiple *nano* (33 grams) flying robots, reaching the fastest inter-robot ranging communication to date (333 Hz).
- A proof showing that unobservable relative states are not as problematic as commonly assumed in the literature. In particular, state drift and sensory noise lead to control actions that make the state observable again. This *self-regulated estimation convergence* is validated by simulation and real-world experiments.

- An automatic *initialization procedure* for dealing with large unknown initial state errors, and investigation of the filters' convergence time (on average 20 s).
- We investigate the *scalability* of the system. For fully connected communication, it scales up to 12 robots in the communication range and for leader-follower communication, it scales up to 67 robots, assuming a desired update rate ≥ 5 Hz.
- Case studies of formation flight and leader-follower flight. For the latter case study, the tiny leader robot performs onboard vision-based window fly-through, while guiding its "blind" followers.
- Public release of the code to the community. It can be run on off-the-shelf Crazyflie quadrotors by peers.

The remainder of this chapter is organized as follows. Section 3.2 introduces the preliminaries of sensory inputs, the multi-robot model, and the relative localization problem. Section 3.3 proposes the fast communication protocol and the relative localization method, followed by an observability analysis. Section 3.4 presents the filter initialization method. Section 3.5 discusses distributed formation control, and the self-regulated estimation convergence under unobservable conditions. Section 3.6 and Section 3.7 show the simulation and experiment results to verify the effectiveness of the proposed multi-robot localization and control. In Section 3.8 conclusions are drawn.

3.2. SYSTEM DEFINITION

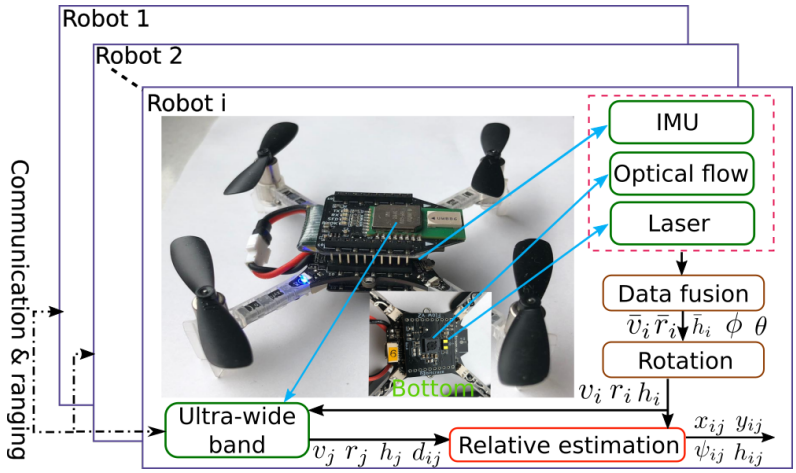


Figure 3.2: The scheme of the multi-robot system and all onboard sensors. Specifically, each robot has an inertial measurement unit (IMU), an optical flow sensor, and a downward-pointing laser sensor for obtaining acceleration, rotation rates, velocities, and height. This information is fused by an onboard filter to get the body-frame velocity, yaw rate, and height, which is further rotated to get the horizontal-frame velocity, yaw rate and height. Combined with the other robots' state information received via UWB communication, the relative positions and yaw are estimated.

3.2.1. SENSORY INPUTS

For a swarm of robots, essential information for each individual is the relative position of other robots. For clarity of analysis, the model of two arbitrary robots is discussed here, in which robot i needs to estimate the relative position of another robot $\{j \mid j \in \mathbb{N}, j \neq i\}$, where $\mathbb{N} = \{1, 2, ..., N\}$, and N is the number of robots.

Before introducing the system model, we define the onboard sensing data as shown in Fig. 3.2. For each aerial robot, the 3-axis velocity $\bar{\boldsymbol{v}} = [\bar{v}^x, \bar{v}^y, \bar{v}^z]^T$, pitch θ, and roll ϕ attitude in the body frame can be obtained by fusing IMU, height, and optical flow measurements [28]. The yaw rate $\bar{r}$ in the body frame is provided by a gyroscope. The range d_{ij}, meaning the distance between robots i and j, can be measured by UWB sensors. The variable h is the vertical height calculated from a downward laser measurement $\bar{h}_i$ and the attitude. The final output x_{ij}, y_{ij}, and h_{ij} denote the relative position between the i^{th} and j^{th} robots.

3.2.2. SYSTEM MODEL

Each drone localizes other drones in an ego-centered *horizontal frame* with a purely vertical z-axis, as shown in blue lines in Fig 3.3.

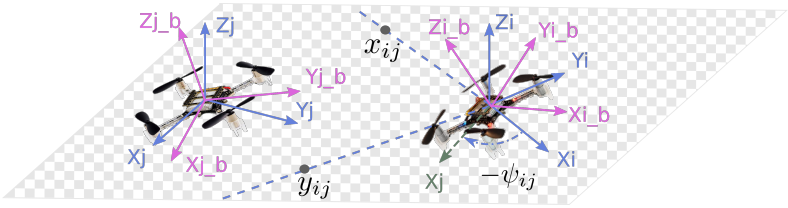


Figure 3.3: The diagram of the relative kinematic model, composed by two robots shown in a horizontal plane for simplicity (as they can be at different heights with the relative height h_{ij}). 3D purple axes represent the body frame of each robot, while the 3D blue axes denote the horizontal frame with a vertical z-axis. The relative 2D position $[x_{ij}, y_{ij}]$ and relative yaw ψ_{ij} of j^{th} robot is shown in i^{th} robot horizontal frame in this figure.

The 2-axis velocity $\boldsymbol{v} = [v^x, v^y]^T$ in the horizontal frame for each robot can be obtained from body-frame velocity $\bar{v}$ and the attitude based on the rotation matrix:

$$\boldsymbol{v} = \mathrm{R}_{xy}^{[0:2,0:3]} \bar{\boldsymbol{v}} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\phi)s(\theta) & c(\phi) & -c(\theta)s(\phi) \end{bmatrix} \bar{\boldsymbol{v}}, \tag{3.1}$$

where $\mathrm{R}_{xy}^{[0:2,0:3]}$ means the first two rows of the rotational matrix R_{xy} defined by

$$\mathrm{R}_{xy} = \mathrm{R}_x \mathrm{R}_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix}, \tag{3.2}$$

where $s(\cdot)$, $c(\cdot)$ and $t(\cdot)$ denote $sin(\cdot)$, $cos(\cdot)$ and $tan(\cdot)$, respectively. In addition, according to the relationship between angular velocity vector and angular velocity, the yaw rate r in horizontal frame can be calculated with the gyros:

$$r = \dot{\psi} = -s(\theta)/c(\phi)\bar{p} + c(\theta)/c(\phi)\bar{r} \tag{3.3}$$

which is derived from another rotation matrix

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\theta)t(\phi) & 1 & -c(\theta)t(\phi) \\ -s(\theta)/c(\phi) & 0 & c(\theta)/c(\phi) \end{bmatrix} \begin{bmatrix} \bar{p} \\ \bar{q} \\ \bar{r} \end{bmatrix}. \tag{3.4}$$

in which $\bar{p}$, $\bar{q}$, and $\bar{r}$ are three body rotation rates.

The estimation scheme depends on three types of onboard measurements. First, the body rates are measured with gyros. Furthermore, the height difference between drones is known, since they communicate the filtered vertical height h from the accurate ranging sensor (VL53L1X). The last sensory input comes from the ultra wide-band, which provides both a distance measurement and a communication ability such as transmitting the velocity, height and yaw rate to other robots.

3.2.3. PROBLEM FORMULATION

Given an arbitrary pair of robots i and j, as shown in Fig. 3.3, the inputs for estimation are represented by $\boldsymbol{U}_{ij} = [\boldsymbol{v}_i^T, r_i, \boldsymbol{v}_j^T, r_j]^T$. The measurements consist of h_i, h_j and d_{ij}. Define the relative state of j^{th} robot in the i^{th} robot's horizontal frame as $\boldsymbol{X}_{ij} = [x_{ij}, y_{ij}, \psi_{ij}]^T$, representing 2-axis relative position and the relative yaw as shown in Fig. 3.3. This relative state is the core problem of this chapter and needs to be estimated based on the inputs and measurements.

Here we introduce a formulation of the problem that is similar to [24] in the sense that no common orientation frame is assumed. However, we reformulate it as a kinematic model, leaving out accelerations. This seemingly small change results in a smaller state space, which was instrumental for reducing memory usage and computational effort for the extremely limited onboard STM32F4 processor.

The kinematic model of the swarm of aerial robots can be derived based on Newton's formulas, and the model takes the transformed velocity and yaw rate as the inputs directly. The continuous model $f(\boldsymbol{X}_{ij}, \boldsymbol{U}_{ij})$ is given as

$$\dot{\boldsymbol{X}}_{ij} = f(\boldsymbol{X}_{ij}, \boldsymbol{U}_{ij}) = \begin{bmatrix} R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i - Sr_i\boldsymbol{p}_{ij} \\ r_j - r_i \end{bmatrix} \tag{3.5}$$

where $\boldsymbol{v}_i = [v_i^x, v_i^y]^T$ and $\boldsymbol{v}_j = [v_j^x, v_j^y]^T$ represent the 2-axis horizontal velocity of two robots; $\boldsymbol{p}_{ij} = [x_{ij}, y_{ij}]^T$ is a part of the relative state $\boldsymbol{X}_{ij}$ meaning 2-axis relative position. $R(\cdot)$ is the rotation function from j^{th} horizontal frame to i^{th} horizontal frame, and S is a skew-symmetric matrix:

$$R(\cdot) = \begin{bmatrix} c(\cdot) & -s(\cdot) \\ s(\cdot) & c(\cdot) \end{bmatrix}, \; S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{3.6}$$

3.3. FAST COMMUNICATION AND RELATIVE LOCALIZATION

This section gives details of the fast communication protocols, relative state estimation method and the corresponding observability analysis. We extend the two-way-ranging (TWR) communication for bidirectional ranging and signal-loss detection. An EKF is used here for estimation because it is efficient compared to other filters such as the particle filter, which is vital for micro-robots with limited computation power.

3.3.1. FAST COMMUNICATION AND RANGING

The two-way-ranging method can provide accurate distance measurements when using UWB with a standard deviation of 0.025 m [29]. Instead of communicating with beacons [29], this chapter proposes a dynamic recurrent two-way-ranging method, which allows for robust and high-speed communication and ranging between any two robots.

The communication design can be divided into the high-level communication scheme and low-level protocols. The communication scheme can be designed independently from the low-level protocol. For fully-connected communication we propose the scheme shown in Fig. 3.4.

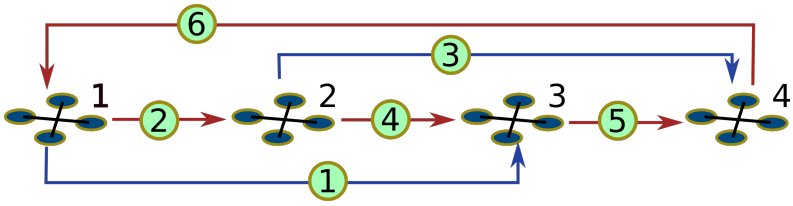


Figure 3.4: Communication scheme for multiple robots in an infinite loop. The scheme is illustrated for 4 robots, starting from the most left robot as the only sender, and following the sequence of the numbers in green circles. A red or blue arrow means a low-level communication procedure. However, two robots will swap the sender and receiver mode in red arrows, while in blue arrows the sender will communicate with the next receiver.

Specifically, in this communication loop, all robots are assigned an incremental ID from 1 to N. All robots' UWB modules are set to be in receiver mode except for the 1st robot which is in sender mode at the beginning. Then the 1st robot communicates with robots from $N-1$ to 2, and changes mode into the receiver while the 2nd robot changes into sender mode and starts its own communication. In this way, the ranging communication can be run on an arbitrary number of robots infinitely with no communication conflicts. Additional logic is designed to deal with drones exiting or entering the communication network.

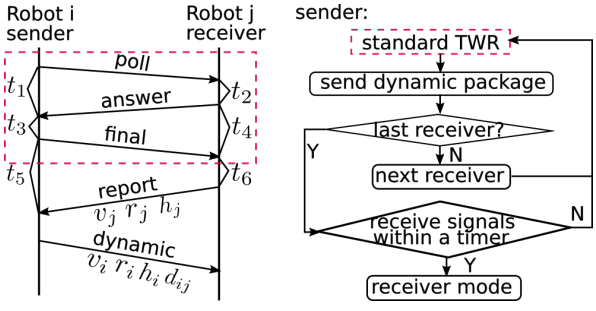


Figure 3.5: Left: extended TWR communication, in which an extra 'dynamic' chain is designed for bidirectional ranging and communication fault detection. Right: the details of the 'dynamic' protocol for fast robust sender-receiver mode transformation by signal detection.

The low-level communication protocol in each arrow of Fig. 3.4 is an extension of the common TWR method with improvements to allow for bidirectional ranging and dynamic mode changing. The brief review of the standard TWR is shown in the red box in Fig. 3.5. After three communication steps of 'poll', 'answer', and 'final', the j_{th} robot can calculate the distance d_{ij} based on the signal flight time $t_f = (t_1 t_4 - t_2 t_3)/(t_1 + t_2 + t_3 + t_4)$. Normally, the i_{th} robot would have to start a new TWR communication for also knowing the distance. Instead, here we introduce an additional step termed 'report'. With this step, the i_{th} robot can also calculate the distance d_{ij} with the flight time $t_f = (t_4 t_5 - t_3 t_6)/(t_3 + t_4 + t_5 + t_6)$. Only the latter flight time is calculated to save on computational effort, and the i^{th} robot sends the determined distance together with its state information to the j^{th} robot in the final 'dynamic' step.

The new extra communication step 'dynamic' extends the standard TWR, such that the sender can swap to receiver mode after communication with all other agents. Via this step, the sender sends the distance measurement back to the receiver and checks if it is changing into receiver mode. If the mode swapping succeeds, the high-level communication topology continues with a new sender. Otherwise, the robots retry the swapping procedure. Details are shown in the right diagram which enables robust recurrent communication between multiple robots.

Overall, this extended communication allows bidirectional ranging in multi-robot networks with fast ranging frequency. It will be compared with the state of the art in Section 3.7.

3.3.2. EKF FILTER FOR RELATIVE LOCALIZATION

We employ the following EKF for state estimation, using a discrete model $F(\hat{\boldsymbol{X}}_k, \boldsymbol{U}_k)$ of (3.5):

$$\begin{aligned} \hat{\boldsymbol{X}}_{k+1|k} &= F(\hat{\boldsymbol{X}}_k, \boldsymbol{U}_k) = \hat{\boldsymbol{X}}_k + \dot{\hat{\boldsymbol{X}}}_k \Delta t, \\ \boldsymbol{P}_{k+1|k} &= \boldsymbol{A}_k \boldsymbol{P}_{k|k} \boldsymbol{A}_k^T + \boldsymbol{B}_k \boldsymbol{Q}_k \boldsymbol{B}_k^T, \end{aligned} \tag{3.7}$$

where Δt is the interval time of updating the Kalman filter, the predicted state is represented by $\hat{\boldsymbol{X}}_{k+1|k}$, and $\hat{\boldsymbol{X}}_{k|k}$ is the estimated state at time step k. Furthermore, the first equation in (3.7) shows the prediction result using the nonlinear model of (3.5). The second equation denotes the update of error covariance $\boldsymbol{P}$ caused by the prediction step and input noise covariance $\boldsymbol{Q}$. To update $\boldsymbol{P}$, the state Jacobian matrix $\boldsymbol{A}$ and input Jacobian matrix $\boldsymbol{B}$ are calculated as follows.

$$\begin{aligned} \boldsymbol{A} &= \frac{\partial F}{\partial \boldsymbol{X}} = \begin{bmatrix} 1 & r_i \Delta t & (-s(\psi_{ij}) v_j^x - c(\psi_{ij}) v_j^y) \Delta t \\ -r_i \Delta t & 1 & (c(\psi_{ij}) v_j^x - s(\psi_{ij}) v_j^y) \Delta t \\ 0 & 0 & 1 \end{bmatrix} \\ \boldsymbol{B} &= \frac{\partial F}{\partial \boldsymbol{U}} = \begin{bmatrix} -1 & 0 & y_{ij} & c(\psi_{ij}) & -s(\psi_{ij}) & 0 \\ 0 & -1 & -x_{ij} & s(\psi_{ij}) & c(\psi_{ij}) & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.8}$$

After the prediction update, the Kalman filter fuses the predicted state with the observation of the distance between two robots, represented by

$$z = h(\boldsymbol{X}_{ij}) = \sqrt{\boldsymbol{p}_{ij}^T \boldsymbol{p}_{ij}} = \sqrt{x_{ij}^2 + y_{ij}^2 + (h_j - h_i)^2}. \tag{3.9}$$

Therefore, the Jacobian matrix of observation is

$$\boldsymbol{H} = \frac{\partial h}{\partial \boldsymbol{X}} = [x_{ij}/z, y_{ij}/z, 0]. \tag{3.10}$$

The rest of the Kalman filer process is shown as follows.

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}, \\ \hat{\boldsymbol{X}}_k &= \hat{\boldsymbol{X}}_{k|k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}_k\hat{\boldsymbol{X}}_{k|k-1}), \\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1} \end{aligned} \tag{3.11}$$

where $\boldsymbol{K}$ is the Kalman gain. Here, both $\boldsymbol{Q}$ and $\boldsymbol{R}$ are noise covariance parameters and can be formulated as diagonal matrices denoted by $\boldsymbol{Q} = \text{diag}([q_v^2, q_v^2, q_r^2, q_v^2, q_v^2, q_r^2])$ and $\boldsymbol{R} = \text{diag}([r_d^2])$. q_v, q_r and r_d denote the standard deviation of the velocity, yaw rate and distance measurements.

3.3.3. OBSERVABILITY ANALYSIS

Since we employ a kinematic model that is different from previous work, we perform a nonlinear observability analysis with Lie derivatives. From [30], a local weak observability analysis can be performed with model (3.5) and observation (3.9). The observability matrix $\boldsymbol{O}$ is composed by different orders of Lie derivatives.

$$\boldsymbol{O} = \begin{bmatrix} \nabla\mathscr{L}_f^0 h \\ \nabla\mathscr{L}_f^1 h \\ \nabla\mathscr{L}_f^2 h \end{bmatrix} = \begin{bmatrix} (\partial\mathscr{L}_f^0 h)/(\partial\boldsymbol{X}) \\ (\partial\mathscr{L}_f^1 h)/(\partial\boldsymbol{X}) \\ (\partial\mathscr{L}_f^2 h)/(\partial\boldsymbol{X}) \end{bmatrix} \tag{3.12}$$

where $\mathscr{L}_f h$ is the Lie derivative of the model function f, and $\nabla\mathscr{L}_f h$ is the differential operator of the Lie derivatives. For simplicity, the power form $\boldsymbol{p}_{ij}^T\boldsymbol{p}_{ij}/2$ is taken as $h(\boldsymbol{X}_{ij})$ in this subsection. Substituting in the system model and observation function, we obtain:

$$\begin{aligned} \mathscr{L}_f^0 h &= h(\boldsymbol{X}_{ij}) = \boldsymbol{p}_{ij}^T\boldsymbol{p}_{ij}/2 \\ \nabla\mathscr{L}_f^0 h &= \begin{bmatrix} \boldsymbol{p}_{ij}^T & 0 \end{bmatrix} \\ \mathscr{L}_f^1 h &= \nabla\mathscr{L}_f^0 h \cdot f = \boldsymbol{p}_{ij}^T(R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i - Sr_i\boldsymbol{p}_{ij}) \\ \nabla\mathscr{L}_f^1 h &= \begin{bmatrix} (R\boldsymbol{v}_j - \boldsymbol{v}_i)^T & \boldsymbol{p}_{ij}^T R(\psi_{ij})S\boldsymbol{v}_j \end{bmatrix} \\ \mathscr{L}_f^2 h &= \nabla\mathscr{L}_f^1 h \cdot f = \boldsymbol{v}_j^T\boldsymbol{v}_j - 2\boldsymbol{v}_i^T R\boldsymbol{v}_j + \boldsymbol{v}_i^T\boldsymbol{v}_i \\ &\quad + \boldsymbol{v}_i^T Sr_i\boldsymbol{p}_{ij} + \boldsymbol{p}_{ij}^T R(\psi_{ij})S\boldsymbol{v}_j r_i \\ \nabla\mathscr{L}_f^2 h &= \begin{bmatrix} \boldsymbol{v}_i^T Sr_i + r_j\boldsymbol{v}_j^T S^T R^T & -2\boldsymbol{v}_i^T RS\boldsymbol{v}_j - \boldsymbol{p}_{ij}^T R\boldsymbol{v}_j r_j \end{bmatrix}^T \end{aligned} \tag{3.13}$$

According to the local weak observability theory, the system is observable only if observability matrix $\boldsymbol{O}$ is full rank. In other words, the determinant of the matrix in (3.12) should be non-zero. The determinant is calculated as

$$\begin{aligned} |\boldsymbol{O}| = &-\boldsymbol{p}_{ij}^T RS\boldsymbol{v}_j(\boldsymbol{v}_i^T Sr_i + r_j\boldsymbol{v}_j^T S^T R^T)S\boldsymbol{p}_{ij} \\ &-(2\boldsymbol{v}_i^T RS\boldsymbol{v}_j + \boldsymbol{p}_{ij}^T R\boldsymbol{v}_j r_j)(-\boldsymbol{v}_i^T + \boldsymbol{v}_j^T R^T)S\boldsymbol{p}_{ij} \end{aligned} \tag{3.14}$$

Although it is difficult to get the full analytical solution of $|\boldsymbol{O}| \neq 0$, we can extract three intuitive and practical unobservable conditions. The first intuitive condition is that $|\boldsymbol{O}|$ tends to be zero when $\boldsymbol{p}_{ij}$ is close to zero. This means compact movements of this multi-robot system will cause lower estimation accuracy. The second intuitive unobservable condition is that $\boldsymbol{v}_j$ cannot be zero, simply because the i^{th} robot cannot find out the heading of j^{th} robot if j^{th} robot is not moving. However, $\boldsymbol{v}_i$ could be zero according to (3.14), because the relative state is represented in the body frame of i^{th} robot such that it knows its own heading, even if it is static. The third unobservable condition occurs when the relative velocity $-\boldsymbol{v}_i^T + \boldsymbol{v}_j^T R^T$ is zero. This will cause the second term of (3.14) to be zero, and the first term is also zero when yaw rates of both robots remain zero. This is a potentially seriously limiting condition, as this is exactly what happens in most formation flights. This matter will be studied in the following sections. Finally, please note that the nonlinear observability analysis we have performed is based on the continuous system, whereas the estimation is performed with a discretized EKF. Depending on the system, this may lead to slightly different outcomes [31].

3.4. AUTOMATIC INITIALIZATION PROCESS

3.4.1. STOCHASTIC INITIALIZATION

In practical scenarios the initial relative states $\boldsymbol{X}_{ij}^0$ between robots are usually unknown. Manual measurements of the initial positions are time-consuming and make the system less autonomous. Therefore, an automatic initialization method is designed to have the relative localization errors approximate zero before executing cooperative tasks.

Assumption 1. *For simplicity, we assume the control input of the yaw rate for each robot remains zero during the whole flight, i.e., $r_i = r_j = 0$. As the drones are in control of their yaw rates, this assumption can be made true by design.*

Based on this assumption, the observability equation of (3.12) can be reduced to

$$\boldsymbol{O}_r = \begin{bmatrix} \boldsymbol{p}_{ij}^T & 0 \\ (R\boldsymbol{v}_j - \boldsymbol{v}_i)^T & \boldsymbol{p}_{ij}^T R(\psi_{ij}) S\boldsymbol{v}_j \\ \boldsymbol{0} & -2\boldsymbol{v}_i^T RS\boldsymbol{v}_j \end{bmatrix} \tag{3.15}$$

The corresponding determinant is reduced to

$$|\boldsymbol{O}_r| = -2\boldsymbol{v}_i^T RS\boldsymbol{v}_j(-\boldsymbol{v}_i^T + \boldsymbol{v}_j^T R^T)S\boldsymbol{p}_{ij} \tag{3.16}$$

With this assumption, denote the control inputs for each robot as $\boldsymbol{u}_i = [v_i^x, v_i^y, r_i]^T$. The initialization inputs are set to:

$$\boldsymbol{u}_i(t) = \begin{cases} [v_{\text{xR}}, v_{\text{yR}}, 0]^T, & t \in [2kT, (2k+1)T) \\ -[v_{\text{xR}}, v_{\text{yR}}, 0]^T, & t \in [(2k+1)T, 2(k+1)T) \end{cases} \tag{3.17}$$

where v_{xR} and v_{yR} are two-axis velocities generated randomly within the range of $(0, v_{\max}]$ at time $t = 2kT$ in local clocks. Here, k, $v_{\max} = 1$ m/s and $2T$ denote $[0, 1, 2...]$, the maximum velocity and the periodic time interval, respectively. Specifically, each robot does

periodical maneuvers, starting with random velocity $\boldsymbol{u}_i$ in the first time interval T, following by a reversed velocity $-\boldsymbol{u}_i$ in the second time interval T. The initialization time is set to 30 seconds since this procedure takes 10-30 seconds to converge as shown in the simulation and experiments.

Remark 1. *This initialization process prevents the robots from flying away from the initial position in a short time such that a safe flight is guaranteed. In addition, bounded velocities and known time interval T limit the flight radius and furthermore guarantee collision avoidance among multiple robots even if the initial relative positions are unknown, provided that they do not start too close to one another.*

3.4.2. CONVERGENCE OF THE INITIALIZATION PROCESS

This subsection studies the convergence of the proposed initialization process. Specifically, we will first study what the state estimate converges to in observable conditions and then in unobservable conditions. This analysis is simplified by the fact that due to the predesigned non-zero velocity inputs (3.17), the unique unobservable condition is (cf. (3.16)):

$$R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i = 0, \tag{3.18}$$

which means both i^{th} and j^{th} robots are flying in same direction with same velocities.

Theorem 1. *If the system input satisfies $R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i \neq 0$, all relative states of the Kalman filter converge and are exponentially bounded.*

Proof. The observability determinant (3.16) is not zero when $R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i \neq 0$. Therefore, the system satisfies the nonlinear observability rank condition. According to [32], the corresponding estimator converges exponentially and the estimation error is bounded. The detailed convergence proof is omitted for the weak observable systems as many references have already proved it. □

Theorem 2. *For multiple robots with dynamic estimation model* (3.5)*, if the control inputs follow the initialization process* (3.17) *and there is an unobservable condition* (3.18)*, then the estimated relative state of the Kalman filter will converge to an unobservable subspace, i.e.*

$$\lim_{t\to\infty} \hat{\boldsymbol{X}}_{ij}(t) \to \{x, y, \psi | \sqrt{x^2+y^2} = z_{\text{GT}}, \psi = \psi_{\text{GT}}\}, \tag{3.19}$$

and all states $[x_{ij}, y_{ij}, \psi_{ij}]^T$ drift slowly once they reach the subspace. z_{GT} and ψ_{GT} denote the constant distance measurement and constant relative yaw hold by (3.18)*.*

Proof. From (3.15), (3.16) and (3.18), we can get the observable dimension $\text{Rank}(\boldsymbol{O}_{\text{r}}) = 2$ in the reduced system. Therefore, the relative estimation system has 2-dimensional weak observability. In order to find the two variables that could be observed and converged based on the Kalman filter, the estimation error is analyzed as follows

$$\tilde{\boldsymbol{X}}_{ij} = \boldsymbol{X}_{ij} - \hat{\boldsymbol{X}}_{ij}. \tag{3.20}$$

The derivative of the estimate state $\hat{\boldsymbol{X}}$ can be written as:

$$\dot{\hat{\boldsymbol{X}}}_{ij} = f(\hat{\boldsymbol{X}}_{ij}, \boldsymbol{U}_{ij}) + \boldsymbol{K}(z - h(\hat{\boldsymbol{X}}_{ij})). \tag{3.21}$$

According to [33], the Kalman gain $\boldsymbol{K}$ and the derivative of the error covariance matrix $\boldsymbol{P}$ can be represented by

$$\begin{aligned} \boldsymbol{K} &= \boldsymbol{P}\boldsymbol{H}^T\boldsymbol{R}^{-1} \\ \dot{\boldsymbol{P}} &= \boldsymbol{A}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}^T - \boldsymbol{P}\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{P} + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^T. \end{aligned} \tag{3.22}$$

Based on the definition of the Kalman function, the optimal gain $\boldsymbol{K}$ always satisfies the following equations:

$$\frac{\partial \mathrm{tr}(\boldsymbol{P})}{\partial \boldsymbol{K}} = 0,\ \boldsymbol{P} = \mathrm{cov}(\boldsymbol{X} - \hat{\boldsymbol{X}}) = \mathrm{cov}(\tilde{\boldsymbol{X}}), \tag{3.23}$$

which means that gain $\boldsymbol{K}$ in (3.21) can minimize the state error to a transformed subspace. Therefore, if a unique equilibrium space of $\tilde{\boldsymbol{X}}$ can be found, the relative estimation under the unobservable condition will converge to that space.

The equilibrium space can be found by setting $\dot{\tilde{\boldsymbol{X}}} = \dot{\boldsymbol{X}}_{ij} - \dot{\hat{\boldsymbol{X}}}_{ij}$ to zero. $\dot{\boldsymbol{X}}_{ij} = [0,0]^T$ can be derived by combining (3.5), Assumption 1, and (3.18). Hence, substitute (3.21) into $\dot{\tilde{\boldsymbol{X}}} = -\dot{\hat{\boldsymbol{X}}}_{ij} = 0$ which yields

$$\begin{bmatrix} R(\hat{\psi}_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i \\ 0 \end{bmatrix} + \boldsymbol{K}(z - h(\hat{\boldsymbol{X}})) = 0. \tag{3.24}$$

A two-dimensional time-invariant solution for Eq. 3.24 is:

$$\begin{cases} \hat{x}_{ij}^2 + \hat{y}_{ij}^2 = z_{\mathrm{GT}}^2, \\ \hat{\psi}_{ij} = \psi_{\mathrm{GT}}. \end{cases} \tag{3.25}$$

Here we prove that (3.25) is the unique time-invariant solution by studying all cases. Case 1: $R(\hat{\psi}_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i = 0$ and $\boldsymbol{K} = 0$; Case 2: $R(\hat{\psi}_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i = 0$ and $z - h(\hat{\boldsymbol{X}}) = 0$; Case 3: $R(\hat{\psi}_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i \neq 0$ and $\boldsymbol{K}(z - h(\hat{\boldsymbol{X}})) \neq 0$, but they sum to zero.

Case 1 holds only if $\boldsymbol{P}\boldsymbol{H}^T = 0$ according to (3.22), which furthermore leads to $\dot{\boldsymbol{P}} = \boldsymbol{A}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^T$. Hence, $\boldsymbol{P}$ is independent of distance measurement z from (3.8), while $\boldsymbol{H}$ is dependent on z from (3.10). Since $\boldsymbol{H}$ always varies over time due to measurement noise while $\boldsymbol{P}$ does not, $\boldsymbol{P}\boldsymbol{H}^T = 0$ will be a transient condition. Case 2 corresponds to the time-invariant solution in (3.25). In case 3, $\boldsymbol{K}$ is time variant as it contains the integration of state variables which are in matrix $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{H}$ according to (3.22). Thus, this solution is also transient. Therefore, (3.25) is the unique time-invariant equilibrium state space, and the estimated states will converge to the equilibrium space as shown in (3.19).

Furthermore, after the relative states reach the equilibrium space, the state update equation in (3.7) and (3.11) can be rewritten as

$$\hat{\boldsymbol{X}}_{k+1} - \hat{\boldsymbol{X}}_k = \begin{bmatrix} R(\hat{\psi}_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i \\ 0 \end{bmatrix} \Delta t + \boldsymbol{K}_k(z_k - \boldsymbol{H}_k\hat{\boldsymbol{X}}_{k-1}) \tag{3.26}$$

When the state drifts along the circle trajectory $\hat{x}_{ij}^2 + \hat{y}_{ij}^2 = z_{ij}^2$ in the equilibrium space, the drift magnitude $\|\hat{\boldsymbol{X}}_{k+1} - \hat{\boldsymbol{X}}_k\|$ has a positive correlation to σ_v, σ_z (noise of the the velocity and distance measurement) and Δt. However, the drift is slow due to the unbiased noise direction. □

The above theorems show that for non-parallel velocities, the estimate will converge to the true state, and that for parallel velocities, the estimate will converge to the equilibrium space defined by (3.25) with slow circular drift. The initialization procedure works mostly under observable conditions because of the velocity inputs that are picked randomly and started asynchronously. This makes the probability that two robots are flying with continuously identical velocities at the same times extremely unlikely. Moreover, the robots additionally experience different external disturbances and actuation noise, further improving observability. Hence, the initialization procedure ensures with high probability that the state $\hat{\boldsymbol{X}}$ converges to the true values after a finite-time flight.

Remark 2. *In the initialization, convergence speed is influenced by the inputs. For example, large velocities keep the observability matrix away from singular points, thus enhancing convergence performance.*

3.5. DISTRIBUTED CONTROL AND SELF-REGULATED ESTIMATION CONVERGENCE

This section will discuss the distributed controller design for relative position control based on the relative estimation. We will subsequently show what the influence is of using the estimates in the control loop on the observability. Finally, we describe the setup for the leader-follower experiments.

3.5.1. DISTRIBUTED FORMATION CONTROL

For formation flight, the reference setpoint is $\bar{\boldsymbol{p}}_{*i} = [\bar{x}_{*i}, \bar{y}_{*i}]^T$ for the i^{th} robot in the frame of the $*^{\text{th}}$ robot. For simplicity, $*$ is set to be 1 which means the 1^{st} robot's position is the reference origin. Thus, the desired relative states in each robot's frame can be obtained as

$$\bar{\boldsymbol{p}}_{i1} = -R(\hat{\psi}_{i1})\bar{\boldsymbol{p}}_{1i} \tag{3.27}$$

Therefore, the control error of the relative position is

$$\boldsymbol{e}_{i1} = \hat{\boldsymbol{p}}_{i1} - \bar{\boldsymbol{p}}_{i1} = \hat{\boldsymbol{p}}_{i1} + R(\hat{\psi}_{i1})\bar{\boldsymbol{p}}_{1i} \tag{3.28}$$

where $\hat{\boldsymbol{p}}_{i1}$ is the relative position estimation. Considering the relative system dynamics

$$\dot{\boldsymbol{p}}_{ij} = R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i - Sr_i\boldsymbol{p}_{ij}, \tag{3.29}$$

Inspired by feed-forward controllers of [34, 35], a dynamic inversion formation control law is proposed as follows

$$-k_{\text{DI}}\boldsymbol{e}_{i1} = R(\hat{\psi}_{i1})\boldsymbol{v}_1 - \boldsymbol{v}_i - Sr_i\hat{\boldsymbol{p}}_{i1}. \tag{3.30}$$

k_{DI} denotes the control gain. Hence, the velocity command for the i^{th} robot is

$$\boldsymbol{v}_i = k_{\text{DI}}\boldsymbol{e}_{i1} + R(\hat{\psi}_{i1})\boldsymbol{v}_1 - Sr_i\hat{\boldsymbol{p}}_{i1} \tag{3.31}$$

For the formation flight task, each robot should avoid other robots when changing the formation pattern. Therefore, a repulsive velocity is introduced to the system model (3.29)

for collision avoidance.

$$\dot{\boldsymbol{p}}_{ij} = R(\psi_{ij})\boldsymbol{v}_j - \boldsymbol{v}_i - Sr_i\boldsymbol{p}_{ij} - \alpha \sum_{\substack{m=1 \\ m\neq i}}^{N} \text{sign}(\boldsymbol{p}_{im}) \circ |\boldsymbol{p}_{im}|^{-1} \tag{3.32}$$

where α is a positive constant which represents the repulsive gain, and the last item in (3.32) leads to high repulsive velocity when distance is close to avoid collision inter robots. Therefore, the dynamic inversion formation control with collision avoidance is designed as

$$\boldsymbol{v}_i = k_{\text{DI}}\boldsymbol{e}_{i1} + R(\hat{\psi}_{i1})\boldsymbol{v}_1 - Sr_i\hat{\boldsymbol{p}}_{i1} - \alpha \sum_{\substack{m=1 \\ m\neq i}}^{N} \text{sign}(\boldsymbol{p}_{im}) \circ |\boldsymbol{p}_{im}|^{-1}. \tag{3.33}$$

Lemma 1. *Given a formation flight task with the reference states of* (3.27)*, if estimated state $\hat{\boldsymbol{X}}$ converges to the real state $\boldsymbol{X}$ with a small error $\|\hat{\boldsymbol{X}} - \boldsymbol{X}\| < \delta_{\boldsymbol{x}}$, the relative estimation system becomes unobservable.*

Proof. Suppose the control gains in (3.33) are selected appropriately, such that $\lim \boldsymbol{e}_{i1} \to 0$. In addition, from $\|\hat{\boldsymbol{X}} - \boldsymbol{X}\| < \delta_{\boldsymbol{x}}$ we can get that real relative yaw $\boldsymbol{\psi}_{i1} \approx \hat{\boldsymbol{\psi}}_{i1} = C_\psi$ is nearly constant due to assumption of $r_i = r_1 = 0$. Therefore, the real relative position $\boldsymbol{p}_{i1} \approx \hat{\boldsymbol{p}}_{i1} = -\boldsymbol{e}_{i1} - R(C_\psi)\bar{\boldsymbol{p}}_{1i}$ approximates a constant due to the constant formation planning of $\bar{\boldsymbol{p}}_{1i}$. Thus, the following holds:

$$\dot{\boldsymbol{p}}_{i1} = 0 = R(\psi_{i1})\boldsymbol{v}_1 - \boldsymbol{v}_i \tag{3.34}$$

This leads to a zero determinant in (3.16) such that the stable states of formation control cause an unobservable condition for the relative estimation system. □

Remark 3. *Notice that this unobservable condition is different from what is discussed in Section 3.4. Here the unobservable velocity inputs are derived based on correct estimation and formation control, instead of the predefined time-varying velocity inputs in the last section. Therefore, the following subsection will give a different proof of the self-regulated estimation stability under the unobservable formation planning.*

3.5.2. SELF-REGULATED ESTIMATION CONVERGENCE

In this subsection, we will give a proof that thanks to the active use of estimates in closed loop control and real-world factors such as sensor and actuation noise and external disturbances, unobservable states occurring in successful formation flight will always lead again to observable states. Hence, in formation flight, the system will be continuously transitioning from unobservable states to observable states and back again, keeping the actual drift bounded.

Assumption 2. *The estimated relative yaw $\hat{\psi}_{i1}$ can be assumed to approximate the real value ψ_{i1} for a certain time when the system switches to an unobservable condition. This holds because of low-noise yaw rate measurement from the gyroscope and appropriate noise covariance selection for yaw rate in the filter. In addition, this assumption holds also because $\psi = \psi_{\text{GT}}$ belongs to the convergence subspace from* (3.19) *under unobservable control inputs.*

Problem 1. *The input and measurement noise is omnidirectional and hence typically has a component tangent to the unobservable circle trajectory. This leads to the estimation drift of relative states, hence* $\hat{p} \neq \bar{p}$.

Theorem 3. *Given the converged state estimation* $\boldsymbol{p}_{i1}$ *and* ψ_{i1} *according to Theorem 2, and the invariant* $\hat{\psi}_{i1}$ *in Assumption 2 and the estimation drift in Problem 1. The estimation error will remain converged and bounded even if the multi-robot system is under unobservable maneuvers such as the formation flight.*

Proof. First, $\boldsymbol{p}$, $\hat{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}$ denote the real value, estimation and control reference of the relative position, respectively. After the initialization and the formation control, relative states satisfy $\boldsymbol{p} = \hat{\boldsymbol{p}} = \bar{\boldsymbol{p}}$. The reference $\bar{\boldsymbol{p}}$ is constant for a formation flight. There are two unobservable cases.

Case 1: Define the estimation drift in Problem 1 as $\Delta\boldsymbol{p}$. The incorrect relative estimation has the following relationship to the real and reference relative positions:

$$\hat{\boldsymbol{p}} = \Delta\boldsymbol{p} + \boldsymbol{p} \neq \bar{\boldsymbol{p}} \tag{3.35}$$

Substitute (3.28) into (3.31), and consider the zero yaw rate in Assumption 1, we can get

$$\boldsymbol{v}_i = k_{\mathrm{DI}}(\hat{\boldsymbol{p}}_{i1} - \bar{\boldsymbol{p}}_{i1}) + R(\hat{\psi}_{i1})\boldsymbol{v}_1. \tag{3.36}$$

In view of (3.34) and (3.35), the state will become observable again due to the ensuing control actions:

$$R(\psi_{i1})\boldsymbol{v}_1 - \boldsymbol{v}_i = k_{\mathrm{DI}}(\bar{\boldsymbol{p}}_{i1} - \hat{\boldsymbol{p}}_{i1}) \neq 0 \tag{3.37}$$

Hence, based on Theorem 1, the estimated relative position $\hat{\boldsymbol{p}}$ will converge again to the real value $\boldsymbol{p}$.

Case 2: The system is possibly unobservable when $\hat{\boldsymbol{p}} = \bar{\boldsymbol{p}}$ but $\hat{\boldsymbol{p}} \neq \boldsymbol{p}$, which means the relative estimation is incorrect and the system is unobservable. In this case, the relative position will converge to the subspace (the circle trajectory) according to Theorem 2. However, the measurement noise on $\boldsymbol{v}_1$ and $\boldsymbol{v}_i$ is omnidirectional, so it has components orthogonal to the equilibrium state, leading to case 1 and hence observability. Moreover, external disturbances and actuation noise will lead to non-zero $R(\psi_{i1})\boldsymbol{v}_1 - \boldsymbol{v}_i$, and hence observability. □

In summary, the combination of active formation control (case 1) and sensor and actuation noise or external disturbances (case 2) substantially limit the effects of unobservable states. We term this phenomenon 'self-regulated estimation convergence'. This novel finding, which is highly relevant to ranging-based relative localization, is corroborated in the next sections by simulation and real-world experiments.

3.5.3. VISUAL CONTROL

Besides formation control, we also study leader-follower flight. A tiny camera with onboard processor is mounted on the leader robot, such that the leader robot can make motion decisions based on the camera input. 'Blind' followers fly the same safe path as the leader robot based on the relative estimation and distributed control. Here we study the

task of flying multiple robots through a window. The leader needs visual algorithms for detecting the window and creates velocity inputs autonomously, and this function is written as

$$\boldsymbol{v}_1 = f_{\mathrm{VL}}(\mathrm{Img}, \mathrm{Thr_u}, \mathrm{Thr_l}) \tag{3.38}$$

where the function input Img represents 2D 3-channel RGB image. $\mathrm{Thr_u}$ and $\mathrm{Thr_l}$ denote the upper and lower color thresholds of the targets. The leader uses a histogram of the target color for control, centering the window when in view and performing an open loop turn when it is not.

For the follower robots in the vision task, the i^{th} robot creates the velocities based on the states of the $(i-1)^{\mathrm{th}}$ robot, which is written as

$$\boldsymbol{v}_i = f_{\mathrm{VF}}(\boldsymbol{v}_{i-1}, \boldsymbol{p}_{i,i-1}, d_{\mathrm{VF}}) \tag{3.39}$$

This leader-follower flight of f_{VF} is achieved by setting the reference position as $\bar{\boldsymbol{p}}_{i-1,i} = d_{\mathrm{VF}}\boldsymbol{v}_{i-1}/\|\boldsymbol{v}_{i-1}\|$, which is a constant distance along the reversed velocity direction in the leader's frame. After rotation, the reference position in follower i^{th} robot's frame is $\bar{\boldsymbol{p}}_{i,i-1} = -R(\hat{\psi}_{i,i-1})\bar{\boldsymbol{p}}_{i-1,i}$. Therefore, the corresponding velocity for the i^{th} robot can be calculated by position reference $\bar{\boldsymbol{p}}_{i,i-1}$ and control methods proposed in Section 3.5.1.

3.6. SIMULATION

In this section, the relative localization method for multiple robots is validated in simulation. The simulator models robots as points with headings and velocities. Reference velocities take some time to be satisfied, as an abstract model of inertia and actuator dynamics. Range measurements are generated by adding Gaussian noise to the ground-truth distances. The results in this section show the estimation accuracy, convergence performance such as time to convergence, and estimation efficiency in unobservable conditions. The simulation code has been implemented using Python and is available at `https://github.com/shushuai3/multi-robot-localization`

3.6.1. LOCALIZATION PERFORMANCE

In this simulation, the relative state between two robots is estimated and compared to the ground-truth relative position and yaw to verify the localization accuracy. This simulation is configured with a time interval dt=0.01 s and a maximum moving velocity of 1 m/s. The settings of the simulation are: input noise deviation of 0.25 m/s and 0.01 rad/s, and a distance measurement deviation 0.1 m. The initial estimated states are set to zero, while the ground-truth initial states are set randomly and uniformly in a range of [-3,3] m and [-1,1] rad. The parameters of the relative EKF are set to be $\boldsymbol{Q} = \mathrm{diag}([0.25^2, 0.25^2, 0.4^2, 0.25^2, 0.25^2, 0.4^2])$, $\boldsymbol{R} = 0.1^2$, and $\boldsymbol{P} = \mathrm{diag}([10, 10, 0.1])$, based on the simulated estimation performance.

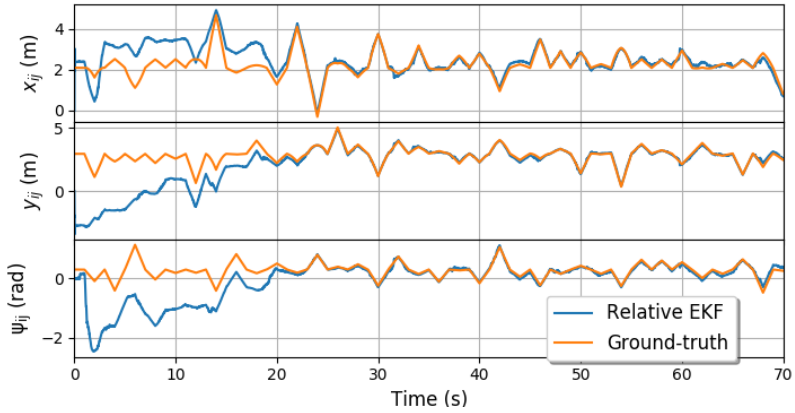


Figure 3.6: Simulation results of the relative state estimation between two robots on x_{ij}, y_{ij}, and ψ_{ij}. Both robots are randomly initialized at unknown position and yaw, and they are 2 meter far away each other. Then each robot flies a start-up procedure with 2-second periodic random settings of velocity and yaw rate (1-sec positive velocity and 1-sec negative velocity to guarantee that it flies within 1 meter and not collide with other robots). The orange line represents the ground-truth relative states, while the blue line means the relative states estimated by EKF.

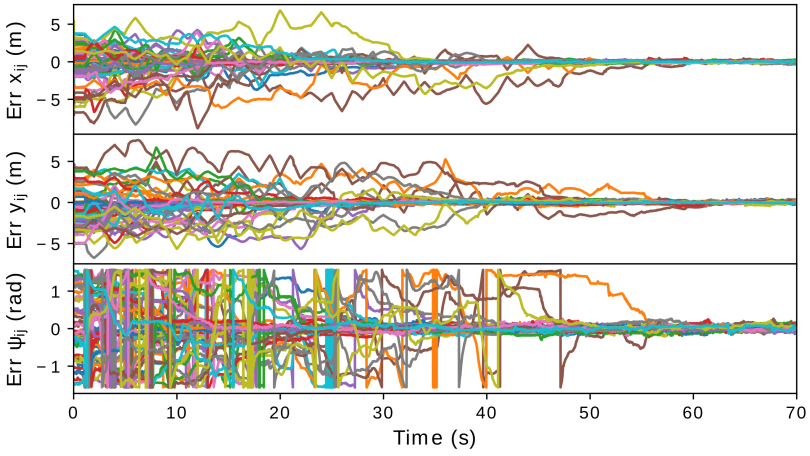


Figure 3.7: Simulation results of estimation error convergence. 3 dimensional relative states are shown from 50 tests with different configurations. Each line with different color represents a different estimation test in three states of x_{ij}, y_{ij}, and ψ_{ij}. All errors are calculated by comparing the estimated states with the ground-truth.

The robots perform random maneuvers at start-up as explained in Section 3.4. These allow the filter to converge. Relative localization results are shown in Fig. 3.6, where we can see that the relative position and yaw approximate the ground-truth after a random flight.

3.6.2. CONVERGENCE TIME

The metric of most interest is the expected time to convergence of the estimation. This will dictate how long an initialization maneuver should be before the filter has converged and the swarm can begin to perform coordinated tasks.To evaluate this, we extracted the performance over a set of 50 simulations. These tests are conducted with different random initial position and yaw for each robot, and the inputs of velocity and yaw rate are also randomized.

The results are shown in Fig. 3.7, which shows the relative estimation error. In all 50 different random tests, the errors of three relative states x_{ij}, y_{ij}, and ψ_{ij} tend to be zero after a certain amount of seconds of random flight. The average convergence time is 20 seconds, while the largest convergence time is 55 seconds, potentially due to inactive inputs, the initialization being further away from the true initial state, or unobservability-inducing flight behaviors. For most multi-robot systems this is a rather short time. Of course, for tiny flying robots such as the Crazyflie drones, which can fly only for ~ 3 minutes, it is still considerable (on average ~ 11% of the flight time).

3.6.3. UNOBSERVABILITY AND SELF-REGULATED CONVERGENCE

In Section 3.3.3, we analytically showed that some flight conditions are unobservable. This subsection will study the influence of unobservable flight behavior on the relative localization *after* estimation convergence in practice. Two situations that lead to unobservability will be discussed: 1) Formation flight that causes $-\boldsymbol{v}_i^T + \boldsymbol{v}_j^T R^T = 0$, while the yaw rates of r_i and r_j remain zero; 2) the j^{th} robot has zero velocity, i.e. $\boldsymbol{v}_j = 0$, so that the relative yaw ψ_{ij} should be unobservable.

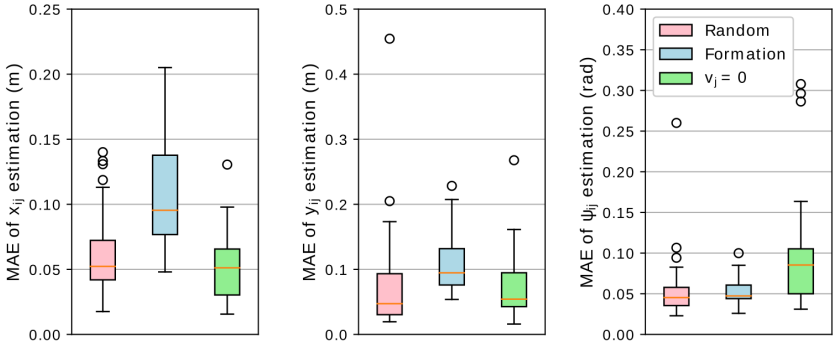


Figure 3.8: Error distribution of relative localization in different unobservable situations. For each situation, the mean absolute errors (MAE) are obtained from 50 different tests, during the 20 seconds after the estimation convergence. Boxes of red, blue, and green color represent random flight, formation flight, and zero velocity of the j^{th} robot, respectively.

In Fig. 3.8, relative localization performance with unobservability-inducing control inputs are shown. By comparing the red and blue boxes, the effects of formation flight are: 1) An increase of estimation errors on all relative states; 2) The relative estimation is still rather accurate with position error less than 0.2 m. This validates the self-regulated estimation convergence theory in Section 3.5.2. The result indicates that once the estimation

is not correct, robots will deviate from their role in the formation (in terms of velocity and position), which in turn makes the system observable again. Hence, this unobservability problem is a self-stabilizing phenomenon that operates within acceptable precision bounds as has been discussed in Section 3.5.

In the case that the robot to be localized has zero velocity, the relative yaw estimation has a larger error compared to normal random flight, which can be seen from the green box. However, as indicated in Section 3.3.3, the relative position of x_{ij} and y_{ij} is still observable. Therefore, the green boxes show an estimation error similar to the red boxes in axes of x_{ij} and y_{ij}.

3.6.4. CIRCLE DRIFT

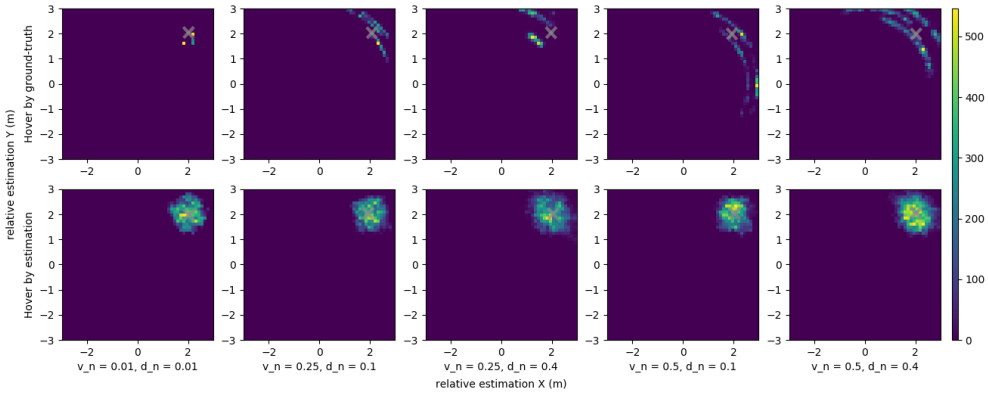


Figure 3.9: Relative localization estimates in two different cases, involving two robots. One robot is hovering with the help of ground truth position at (0,0)m, while the desired relative position of the second drone is at (2,2)m, shown as a grey cross. Each figure represents the second robot's estimate of its position with respect to the first robot for 3 tests with 140 seconds for each flight. Top row: Passive estimation case. Both robots hover with the help of ground truth coordinates. The estimate plays no role in control. Bottom row: Active case, in which the second robot tries to control a constant relative position with respect to the first robot based on the relative estimation.

In this section, the state drift on the circle subspace is discussed during unobservability. To show the extent of the estimation drift along the circle path, Fig. 3.9 shows two simulation tests, where v_n and d_n denote the velocity input noise and distance measurement noise, respectively.

Case 1 on the top row: The drones both hover perfectly with the help of ground truth measurements. In this case, the EKF is a passive observer, and estimation errors do not lead to control corrections that make the state observable again. The top row shows the estimated relative position by drone 2 under this unobservable condition, with varying amounts of noise. We can conclude that in such a passive estimation condition, the relative estimation tends to drift in the unobservable subspace (the circle trace). The drift is slow, but increases when systems have larger noise in velocity and distance measurements. This is the type of drift that one may intuitively expect, overlooking the self-regulating

effect of using the estimates in the closed control loop.

Case 2 on the bottom row: Drone 1 at the center perfectly hovers with ground truth, drone 2 uses its estimated relative position to hover at $\bar{\boldsymbol{p}} = [2, 2]$ m. We can see that the estimation drift will be in a circle area for robots with formation control and constant relative position. Input and measurement noise have less influence on the estimation drift, and the relative estimation keeps stable. Comparing case 2 to case 1, we can get that with the same measurement characteristics, the relative estimation error stay bounded if the estimates are used for active closed loop control in the bottom row, whereas in the passive case in the top row the estimates can drift all along the circle. These results validate the self-regulated estimation convergence proof in Section 3.5.2.

3.6.5. CONVERGENCE RATE

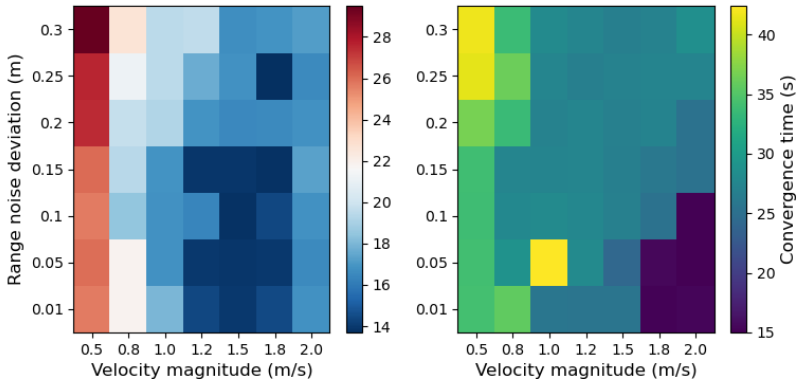


Figure 3.10: The localization convergence rate with respect to different velocity magnitude and the ranging sensor noise deviation. There are two examples, where each example shares the same initial conditions.

This subsection investigates the convergence rate as a function of the velocity magnitude and the range sensor noise. As shown in Fig. 3.10, larger velocity magnitude and smaller range noise deviation lead to less convergence time of the relative localization. Multiple robots with smaller velocity require more time to reach the accurate relative localization. Therefore, less range noise and larger velocity are important for the initialization procedure.

3.7. REAL-WORLD EXPERIMENTAL RESULTS

Using simulations, we have shown that the filter is capable of converging to correct estimates by means of randomized flight maneuvers, after which it can be effectively used for cooperative flight. This section presents an experimental setup in order to further illustrate the relative estimation efficiency in a real-world multi-robot system. The test scenarios consist of formation flights and leader-follower flights.

3.7.1. HARDWARE SETUP

The swarm of the aerial robot system consists of 5 commercial Crazyflie2 quadrotors. Each quadrotor is equipped with a 3-axis accelerometer, 3-axis gyroscope, flow deck (VL53L1x height sensor and PMW3901 optical flow sensor), and loco deck (DWM1000 ultra wide-band sensor). The flow sensor can provide velocity at 100 Hz, and the distance measurement frequency can reach over 333 Hz. The processor is an STM32F4 running at 168 MHz, on which both relative estimation and control are running.

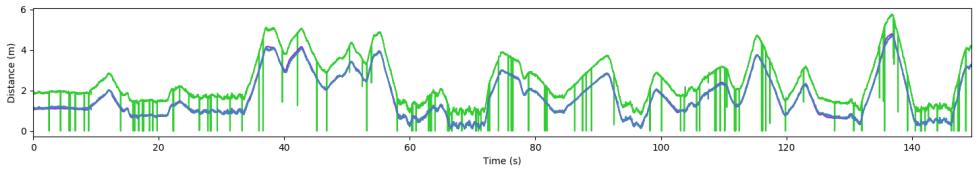


Figure 3.11: UWB measurements and data processing. The green line shows the original distance measurements with large outliers. The blue line shows the outlier-rejected and bias compensated distance data. And the purple line is the ground-truth distance from OptiTrack.

An OptiTrack motion capture system is used for tracking the ground-truth position and yaw of each robot. The OptiTrack data is only used for post-processing to validate the relative estimation performance, and has not been used for any other purpose.

3.7.2. DATA PROCESSING AND COMMUNICATION PERFORMANCE

The raw ranging measurements from UWB can have outliers and unknown biases. Thus, a median filter is applied to reject the outliers, and a bias function is predetermined by data fitting only once based on the ground-truth distance from OptiTrack.

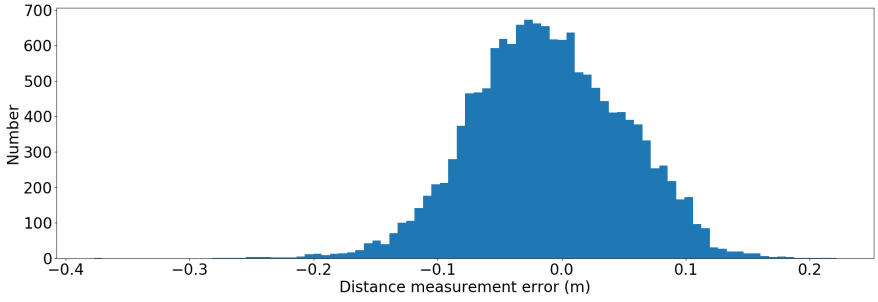


Figure 3.12: The distribution of distance measurement errors between the processed distance measurements and ground-truth distance from OptiTrack. These measurements come from a swarm of 3 Crazyflie quadrotors with a 160-seconds flight.

The linear bias fitting function is related to the distance, represented by $b(d_{ij}) = 0.072d_{ij} + 0.62$ where b denotes the ranging bias compared to the ground-truth distance. By median filtering and subtracting the bias, the processed distance is accurate and approximates the ground-truth distance as shown in Fig. 3.11. In Fig. 3.12, the distance measurement error

between two robots is less than 0.1 m and the bias is compensated by the proposed fitting function. This ranging technique has more accurate and less-biased measurements than that in [24].

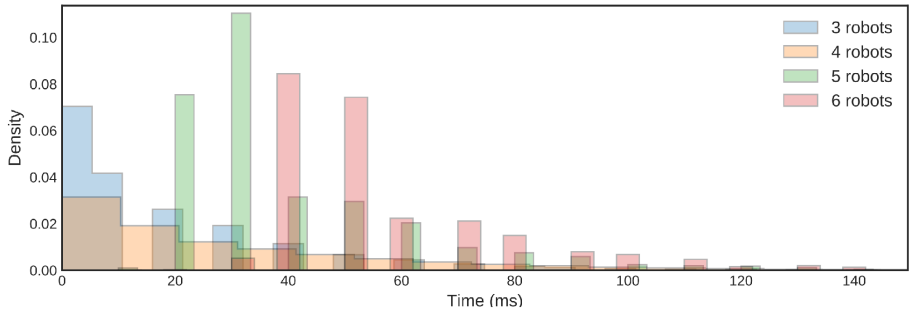


Figure 3.13: The distribution of ranging and communication frequency between each two robots with the increase of the number of robots.

3.7.3. COMMUNICATION SCALABILITY

Each ranging measurement in our proposed scheme takes $1/(22C_6^5) = 0.003$ s. Thus, the proposed communication and ranging protocols have a frequency of $F = 333$ Hz. This is substantially higher than the state of the art, with 48 Hz in [24], 40 Hz in [26], and 10 Hz in [27].

The faster ranging measurement improves the scalability of the wireless-ranging based relative localization. Based on the ranging frequency, we can make a coarse estimate of the ranging update rate for different numbers of robots. Assuming the proposed fully connected ring communication scheme, the number of range measurements for a swarm of N drones are $M = \frac{N!}{2!(N-2)!}$. The time t_c it takes to send all these messages perfectly in sequence is $\frac{M}{F}$.

Fig. 3.13 shows the time t_c of the proposed communication in Section 3.3.1 for different numbers of robots. As seen from Fig. 3.13, the time of each round increases with the increase of the number of robots, approximately as expected by theory. The theoretically expected times for $N = \{3,4,5,6\}$ robots is $t_c = \{9,18,30,45\}$ ms, while the measured 50th percentile averages are $t_c = \{10.6,23.8,31.6,49.7\}$ ms. As expected theoretically, the ranging update frequency in a fully connected communication scheme as Fig. 3.4 is still 22Hz for 6 drones.

One can also retrieve the maximal number of robots for a fully connected scheme, given a minimal desired ranging frequency $f*$. The formula for this is:

$$N^* = \left\lfloor \frac{1}{2} + \sqrt{\frac{2F}{f^*} + \frac{1}{4}} \right\rfloor. \tag{3.40}$$

Filling this in for a desired minimal frequency of $f^* = 5$ Hz, gives 12 robots. This is a substantial improvement over the state of the art. If we fill in the same formula for the $F = 48$ Hz from [24], we arrive at only 4 robots. Still, it may sound a little disappointing if one

thinks of swarms of thousands of robots as honeybees in the air. However, one should keep in mind two things. First, this finding determines the number of drones that can be *within communication range*. This range is typically 10 m, but it can be tuned down to, e.g., 1 m. Having 12 drones within a 1 m area is a very high density, so much that the precision of the wireless-ranging based localization will start to be the main problem. Second, this finding is valid for a fully connected communication scheme. Other schemes can make use of our fast low-level protocol to scale up further. For instance, in leader-follower flight, the leader only ranges to all followers, which makes the number of connections $M = N-1$. This leads to $N^* = \lfloor 1 + \frac{F}{f^*} \rfloor$, which gives 67 robots within communication range for $f^* = 5$ Hz.

3.7.4. RELATIVE ESTIMATION IN REAL EXPERIMENTS

First, the real-world relative estimation performance is shown Fig. 3.14, which indicates the short convergence time and accurate estimation on real robots. A system with a greater number of robots has a longer convergence time due to the lower frequency of communication and EKF updates.

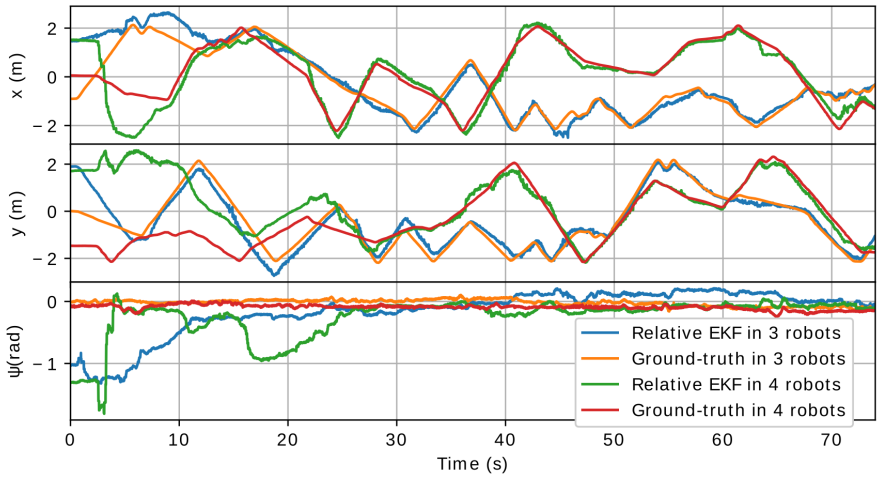


Figure 3.14: Real-world relative localization in 3-robot and 4-robot systems respectively. Here, x, y and ψ denote the absolute XY position and yaw of the 2$^{\text{nd}}$ robot, calculated by the relative EKF from the 1$^{\text{st}}$ robot, and compared with ground-truth from OptiTrack.

For explicit analysis, the 3-dimensional estimation error is given in the following figure. From Fig. 3.15, we can see that the unknown initial states can be estimated in 15 seconds for 3 robots and 25 seconds for 4 robots respectively. Fewer robots need less time for estimation convergence because they have a higher estimation update rate as shown in Fig. 3.13. After convergence, the absolute estimation errors remain converged.

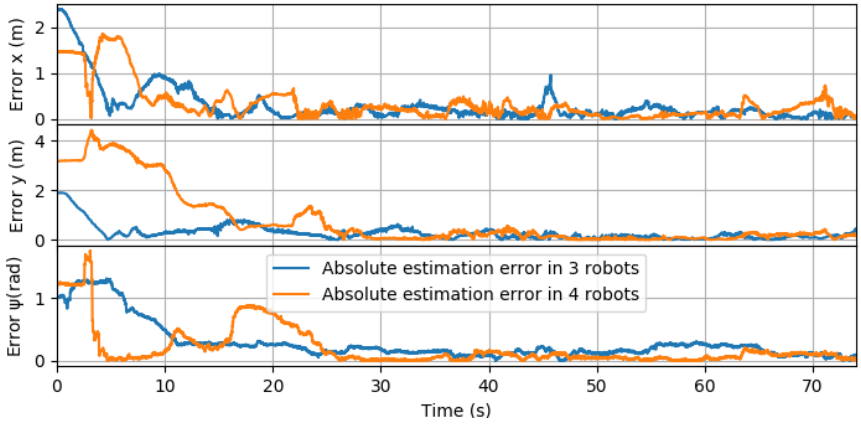


Figure 3.15: Absolute error of real-world relative localization in 3-robot and 4-robot systems respectively. Here, x, y and ψ denote the absolute error of XY position and yaw of the 2nd robot, calculated by the relative EKF from the 1st robot, and compared with ground-truth from OptiTrack.

3.7.5. FORMATION FLIGHT

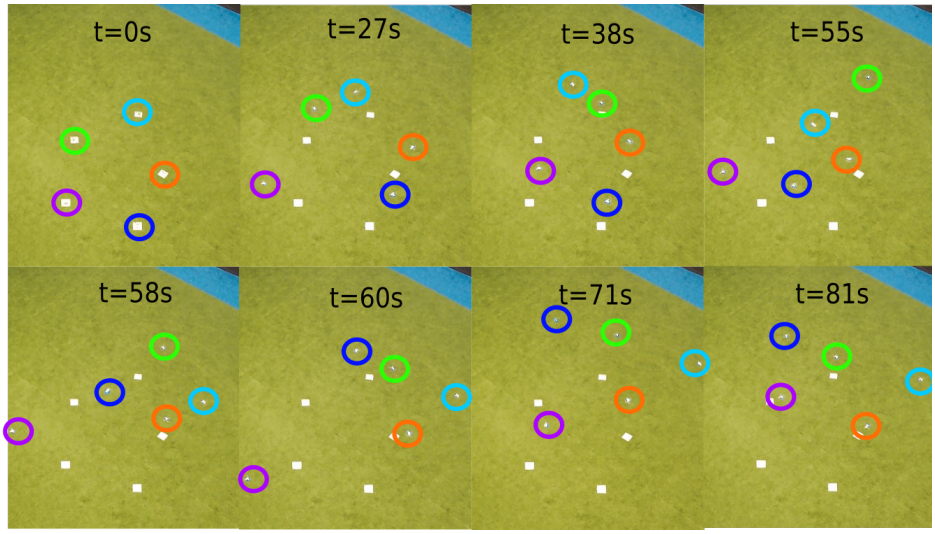


Figure 3.16: Top view of the formation flight of 5 robots. Nine figures show different flight status such as take-off, initialization procedure, distributed control for formation flight, and hovering. Five circles with different colors show the positions of five tiny drones, respectively. Full flight details can be found in the video link `https://www.youtube.com/playlist?list=PL_KSX9GOn2P9sgaX3DHnPsnBCJ76fLNJ5`

Fig. 3.16 shows how the robot team achieves a formation flight based on the proposed relative localization and distributed control. At $t = 0$ s, all tiny flying robots take off from

5 random unknown positions with unknown random yaw angles. After the 30-seconds initialization procedure, all robots have an accurate relative position and yaw estimation of other robots, and start flying to the desired formation positions with respect to the 1st robot with the orange circle. As seen from the figures at $t = 38$ s, $t = 55$ s, $t = 58$ s, robots with green, blue, and dark blue fly to the desired relative positions which are far away from the initial positions.

Starting from $t = 60$ s all robots did a formation flight with constant relative positions to the 1st robot which performs a random flight. From the last three figures, we can see that the robot with the purple circle initially has a wrong relative position estimate, likely due to unobservability. Eventually, also this robot's estimate converges to the true state and the robot flies to the desired formation position. Finally, in the right-bottom figure, five robots form an Olympic-flag-like shape. This shape is maintained by all robots even in the hovering state, which is unobservable for the multi-robot system. This experiment shows that when the proposed relative localization method is used in the control loop, it has consistent convergence in practical experiments even under different unobservable states such as formation flight or hovering. Hence, it corroborates our proof on the self-regulated convergence, just like the simulation experiments.

Figure 3.17: Outdoor formation flights of three Crazyflies. The leader is controlled manually for safety, while two followers fly fully autonomously purely based on onboard relative localization. The environments consist of winds, grass and sunlight. The full experiment can be found in the video.

Fig. 3.17 demonstrates the outdoor formation flight of three Crazyflies. Two followers coordinate to keep a formation flight with respect to the leader drone. Both relative localization and distributed control are calculated onboard the follower drones. Therefore, the proposed relative localization scheme allow multi-robot system executing tasks autonomously even in challenging outdoor environments.

3.7.6. AUTONOMOUS VISUAL TASK

This part further explores the proposed localization ability for an autonomous task by multiple heterogeneous flying robots. The leader robot is equipped with a tiny onboard camera and processor (TCM8230MD camera and STM32F4 processor). This enables the 1st robot to detect the four points of the window based on the proposed color filtering and histogram method. The detection result is shown in Fig. 3.18. The leader robot first controls its two-axis velocity to move through the center of the window and then makes a large turn to get in front of the window again, so that it performs window fly-throughs in-

definitely. At the same time, another robot coordinates with the leader robot in order to fly through the window. As seen in Fig. 3.18, the follower robot has stable following behavior with respect to the leader robot, and flies through the window without having a camera only based on the relative localization. In a real-world application, these follower robots could be equipped with different gas sensors (one robot with a CO-sensor, the other with a CH4 sensor, etc.).

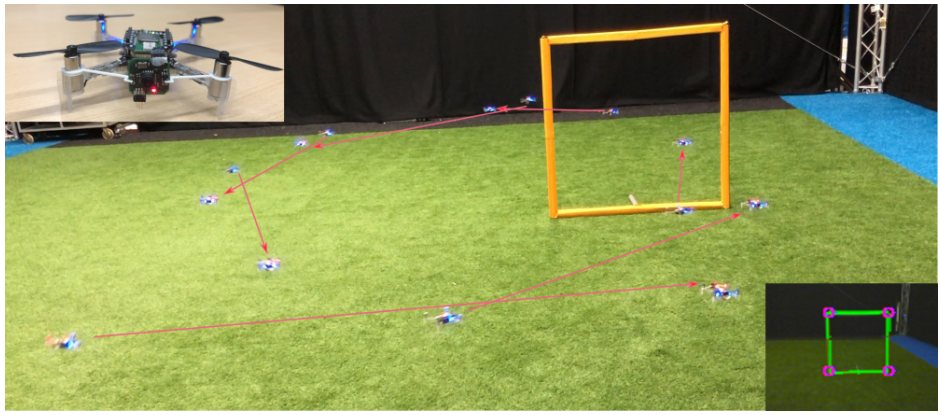


Figure 3.18: Experimental results of coordinated leader-follower flight through a window. Two drones connected with a line are captured at a specific time, where the arrow points to the leader. Only the leader drone is equipped with a monocular camera shown in the left-up corner, and the follower robot maneuvers based on the estimated relative position. The window detection results and its four points are shown in the right-down corner.

3.8. CONCLUSIONS

This chapter proposes a novel, fast relative localization method for fully autonomous swarms of resource-constrained robots. The novel proposed communication protocol achieves bidirectional ranging at 333 Hz. Consequently, the proposed scheme can scale up to larger numbers of drones present within the communication range than previous methods. With a minimal desired update rate of 5 Hz, a fully connected communication scheme scales up to 12 robots and a leader-follower communication to 67 robots. We consider this scaling to be sufficient for most practical applications in which even swarms of robots do not have to be very close to each other, such as search and rescue, monitoring industrial plants or gas source localization. However, future work could further improve scalability by designing more flexible communication schemes allowing for more dynamic networks.

Furthermore, in the article, we have analyzed the commonly assumed fundamental problem of unobservable subspaces in ranging-based relative localization. We have shown both theoretically and empirically that unobservability is not such a pressing problem when the state estimates are actively used in the control loop. Self-regulating estimation convergence arises thanks to sensor and motor noise and control actions in response to state estimation errors. We have also investigated an initialization procedure that reaches estimation convergence in on average 20 s. Although this is quite fast for

most multi-robot systems, this is still a considerable time for the Crazyflies used in our experiments, which fly for 3–5 minutes. Concerning unobservability, future work should explore a control scheme that deliberately maximizes observability along with the mission's control objectives.

Finally, we have performed both formation flights and leader-follower flights with 33 g Crazyflie drones using only their onboard resources. These experiments show the enormous application potential of the proposed technique. We have made the code open source, so that it can be easily used by the community. The proposed scheme is relevant not only for tiny drones but can be readily applied to many other, less resource-restricted robots, as they only need a yaw rate sensor, velocity estimate, and height sensor if they are flying.

REFERENCES

[1] 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).

[2] Q. Lindsey, D. Mellinger, and V. Kumar, *Construction of cubic structures with quadrotor teams,* Proc. Robotics: Science & Systems VII (2011).

[3] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, *Cooperative quadrocopter ball throwing and catching,* in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012) pp. 4972–4978.

[4] M. Gassner, T. Cieslewski, and D. Scaramuzza, *Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors,* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 5196–5202.

[5] M. A. Estrada, S. Mintchev, D. L. Christensen, M. R. Cutkosky, and D. Floreano, *Forceful manipulation with micro air vehicles,* Science Robotics **3** (2018).

[6] K. McGuire, C. De Wagter, K. Tuyls, H. 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** (2019).

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

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

[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** (2018).

[10] X. Dong, B. Yu, Z. Shi, and Y. Zhong, *Time-varying formation control for unmanned aerial vehicles: Theories and applications,* IEEE Transactions on Control Systems Technology **23**, 340 (2014).

[11] M. Hamer and R. D'Andrea, *Self-calibrating ultra-wideband network supporting multi-robot localization,* IEEE Access **6**, 22292 (2018).

[12] 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* (IEEE, 2013) pp. 993–998.

[13] 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).

[14] V. Walter, M. Saska, and A. Franchi, *Fast mutual relative localization of uavs using ultraviolet led markers,* in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2018) pp. 1217–1226.

[15] V. Walter, N. Staub, A. Franchi, and M. Saska, *Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs,* IEEE Robotics and Automation Letters **4**, 2637 (2019).

[16] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, *Drone detection using depth maps,* in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2018) pp. 1034–1037.

[17] F. Schilling, F. Schiano, and D. Floreano, *Vision-based drone flocking in outdoor environments,* IEEE Robotics and Automation Letters **6**, 2954 (2021).

[18] 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).

[19] M. Basiri, F. Schill, D. Floreano, and P. U. Lima, *Audio-based localization for swarms of micro air vehicles,* in *2014 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2014) pp. 4729–4734.

[20] 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).

[21] 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).

[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).

[23] T.-M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, *Distance-based cooperative relative localization for leader-following control of mavs,* IEEE Robotics and Automation Letters **4**, 3641 (2019).

[24] S. van der Helm, M. Coppola, K. N. McGuire, and G. C. de Croon, *On-board range-based relative localization for micro air vehicles in indoor leader–follower flight,* Autonomous Robots , 1 (2019).

[25] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, *Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 8776–8782.

[26] K. Guo, X. Li, and L. Xie, *Ultra-wideband and odometry-based cooperative relative localization with application to multi-uav formation control,* IEEE transactions on cybernetics **50**, 2590 (2019).

[27] S. Güler, M. Abdelkader, and J. S. Shamma, *Peer-to-peer relative localization of aerial robots with ultrawideband sensors,* IEEE Transactions on Control System Technology (2020).

[28] M. Greiff, *Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation,* MSc. Thesis (2017).

[29] 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.

[30] R. Hermann and A. Krener, *Nonlinear controllability and observability,* IEEE Transactions on automatic control **22**, 728 (1977).

[31] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, *A first-estimates jacobian ekf for improving slam consistency,* in *Experimental Robotics* (Springer, 2009) pp. 373–382.

[32] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, *Stochastic stability of the discrete-time extended kalman filter,* IEEE Transactions on Automatic control **44**, 714 (1999).

[33] K. Reif, F. Sonnemann, and R. Unbehauen, *An ekf-based nonlinear observer with a prescribed degree of stability,* Automatica **34**, 1119 (1998).

[34] T. Lee, M. Leok, and N. H. McClamroch, *Geometric tracking control of a quadrotor uav on se (3),* in *49th IEEE conference on decision and control (CDC)* (IEEE, 2010) pp. 5420–5425.

[35] T. J. Koo and S. Sastry, *Differential flatness based full authority helicopter control design,* in *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, Vol. 2 (IEEE, 1999) pp. 1982–1987.

4

SELF-SUPERVISED MONOCULAR MULTI-ROBOT RELATIVE LOCALIZATION WITH EFFICIENT DEEP NEURAL NETWORKS

Relative localization is an important ability for multiple robots to perform cooperative tasks in GPS-denied environment. This chapter presents a novel autonomous positioning framework for monocular relative localization of multiple tiny flying robots. This approach does not require any groundtruth data from external systems or manual labelling. Instead, the proposed framework is able to label real-world images with 3D relative positions between robots based on another onboard relative estimation technology, using ultra-wide band (UWB). After training in this self-supervised manner, the proposed deep neural network (DNN) can predict relative positions of peer robots by purely using a monocular camera. This deep learning-based visual relative localization is scalable, distributed and autonomous. We also built an open-source and light-weight simulation pipeline by using Blender for 3D rendering, which allows synthetic image generation of other robots, and generalized training of the neural network. The proposed localization framework is tested on two real-world Crazyflie2 quadrotors by running the DNN on the onboard AIdeck (a tiny AI chip and monocular camera). All results demonstrate the effectiveness of the self-supervised multi-robot localization method.

Parts of this chapter have been submitted to IEEE International Conference on Robotics and Automation [1]

4.1. INTRODUCTION

Relative localization is necessary for a robot to interact with peer robots, underlying a wide range of distributed and cooperative tasks, e.g., formation flight [2], cooperative construction [3], flocking behaviour [4], etc. However, most of the multi-robot systems rely on external devices such as the global positioning system (GPS) or motion capture systems, for providing the relative positions between robots. These systems cannot work in unknown, GPS-denied environments.

Onboard relative localization methods have been recently proposed for achieving fully autonomous operation of multi-robot systems. Relative estimation based on sound [5] or infra red [6] is impractical for nano robots as larger sensor arrays need to be mounted. Relative localization with communication chips is very suitable for tiny robots thanks to their light weight. For example, multiple tiny quadrotors can avoid each other based on the received signal strength (RSS) [7]. More precise ranging from UWB can be implemented on the same tiny robots for more accurate relative estimation [8]. However, these methods suffer from band-width limitations, leading to poor scalability for a larger number of robots.

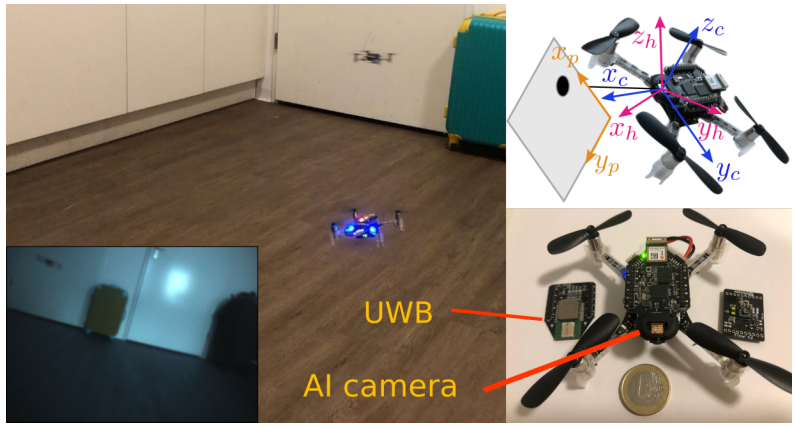


Figure 4.1: Multiple tiny Crazyflie quadrotors localize peer robot 3D positions with deep neural networks based on self-supervised labels from an ultra-wide band relative localization method. Top right: different coordinate frames. Bottom right: a tiny quadrotor, the AI camera and the UWB module. Bottom left: onboard captured image.

Vision-based methods are scalable and distributed for multi-robot localization. These methods can be divided into two main categories: marker-based traditional methods and marker-less learning based detection. The methods with markers consist of relative localization for multiple micro aerial vehicles (MAVs) with onboard markers [9], collaborative localization for a swarm of MAVs relying on salient external features for sparse reconstruction [10], estimation of relative pose between two ground robots by observing a pair of 3D points [11], and two drones that track the same target cooperatively [12]. The performance of these methods is easily degraded due to the size of markers (which should be very small for tiny robots) and the marker pose in the image. Although there is a swarm of quadrotors based on general textures by using visual inertial odometry (VIO) [13], the relative

pose will drift with time and they must take off from known locations. An improvement for VIO-based relative localization is combining it with the UWB measurement [14]. However, the VIO part requires considerable computation power, which is impracticable for tiny flying robots.

Learning-based visual localization is marker-independent and directly robot-oriented. For example, 3D positions of a drone can be estimated by using depth images and deep neural networks [15]. DeepURL proposes a deep estimation method for relative localization of underwater vehicles based on keypoint prediction and PnP which, however, requires each robot's 3D model information [16]. Another state-of-the-art work uses YOLOv3-tiny to detect the drones by training the network with mask images from a static camera and background subtraction method [17], in which the deep neural network is too heavy to run on tiny drones. Besides, its training dataset has a simple background such that the detection will be subject to a larger reality gap caused by common, more cluttered environments, and potentially by motion blur and lighting conditions.

We also review the related references in computer vision and robotic grasping. Classical pose networks require manual annotations such as SSD [18], PoseCNN [19], and PoseNet [20]. Model-based methods can extract more pose information without or with less annotations. For example, EPOS predicts 3D fragment coordinates only with coordinate annotation, and then uses PnP-RANSAC to get 6D object pose [21]. Instead of using coordinate annotation, a deep neural network is designed by detecting 2D projections of robot joints, combined with PnP and model information to estimate the camera-to-robot pose from a single image [22]. However, annotations are time-consuming, and 3D model information is not significant for tiny robots.

Extended visual information can facilitate the 6D pose estimation, such as RGB-D images for object pose prediction, based on a model [23], or in a self-supervised way by capturing images from different views [24]. A pair of images is used for self-supervised depth estimation [25]. These extended visual sensors are usually heavy, power hungry, and high-cost for tiny robots compared with a monocular camera.

This chapter proposes a self-supervised framework for autonomous, scalable and low-cost relative localization of multiple tiny robots. The proposed network draws from the object detection research YOLOv3, but is adapted to the multi-robot localization domain. We do not predict bounding boxes. Rather, we predict the 2D pixel position of the robot center and depth from camera to robot. Here we adopt our work of UWB-based relative localization in previous chapter as an auxiliary localization method to label the images automatically [8], to make the training process self-supervised. The **contributions** of this chapter are summarized as: 1) an efficient deep neural network for integrated monocular multi-robot detection and depth prediction; 2) a novel self-supervised system framework for data labelling and network training; 3) a light-weight 3D rendering simulation for multi-robot image generation with arbitrary pose states; 4) the first implementation of deep neural networks into light tiny (33 gram) flying robots for visual relative localization; 5) Public release of all code and dataset [1].

The remainder of the chapter is organized as follows. Section 4.2 introduces the system definition and the relative localization problem. Section 4.3 gives the detailed design of the proposed framework and the deep neural network. Section 4.4 shows the 3D multi-

[1] Code: `https://github.com/shushuai3/deepMulti-robot`

robot visual rendering pipeline, and performance of the deep relative localization on synthetic images. Section 4.5 validates the localization efficacy on real-world experiments, including dataset collection, network refining, and deep inference onboard an AI camera.

4.2. PRELIMINARIES

A monocular camera mounted on one robot can observe n peer robots in a single RGB image. Before exploring 3D relative pose between the camera and peer robots, this section gives the preliminaries of the multi-robot visual system, the auxiliary onboard relative localization, and the problem definition.

4.2.1. MULTI-ROBOT SYSTEM

For clarity, the spatial model of two robots is considered. As shown in Fig. 4.1, we define three coordinates: 1) image coordinate with yellow axes, where $\boldsymbol{\xi}_p = [x_p, y_p, 1]^T$ denotes pixel positions of peer-robot center in the image with top-left origin; 2) camera coordinate with blue axes, where $\boldsymbol{\xi}_c = [x_c, y_c, z_c]^T$ represents 3D positions of peer robot with respect to the camera; 3) horizontal coordinate with red axes, which is an inertial frame fixed to the robot with a vertical z axis, while x and y axes point forward and left horizontally.

Camera coordinates can be transformed to image coordinates by the intrinsic matrix M_{itr} of the camera. The intrinsic parameters of the AIdeck camera is calibrated with a chessboard, and R_c means the rotation of the camera coordinate, which are shown as follows:

$$x_c\boldsymbol{\xi}_p = M_{\text{itr}}R_c\boldsymbol{\xi}_c = \begin{bmatrix} 183.73 & 0 & 166.90 \\ 0 & 184.12 & 77.51 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \boldsymbol{\xi}_c \tag{4.1}$$

The auxiliary 3D relative estimation is represented in the horizontal coordinate, denoted as $\boldsymbol{\xi}_h = [x_h, y_h, z_h]^T$, which facilitates real-world 3D multi-robot control. This coordinate can be transformed into camera coordinate by rotation in xy sequence

$$\boldsymbol{\xi}_c = R(\phi,\theta)\boldsymbol{\xi}_h = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\phi)s(\theta) & c(\phi) & -c(\theta)s(\phi) \\ -c(\phi)s(\theta) & s(\phi) & c(\theta)c(\phi) \end{bmatrix} \boldsymbol{\xi}_h, \tag{4.2}$$

where ϕ and θ denote roll and pitch attitude along axis x_c and y_c respectively.

4.2.2. ONBOARD AUXILIARY LOCALIZATION

This subsection gives a brief review of the UWB-based relative estimation [8], which is used for generating labels to teach the deep neural network to learn peer-robot positions from monocular images.

As can be seen in Fig. 4.2, the yellow box illustrates the auxiliary localization. The i^{th} robot takes as inputs the peer velocity v_j, peer yaw rate r_j, peer height h_j, self velocity v_i, self yaw rate r_i, self height h_i, and range d_{ij}. Afterwards, a Kalman filter is implemented to estimate the relative position $[x_{ij}, y_{ij}, h_{ij}]^T$, which is equal to $\boldsymbol{\xi_h}$. More details can be found in [8].

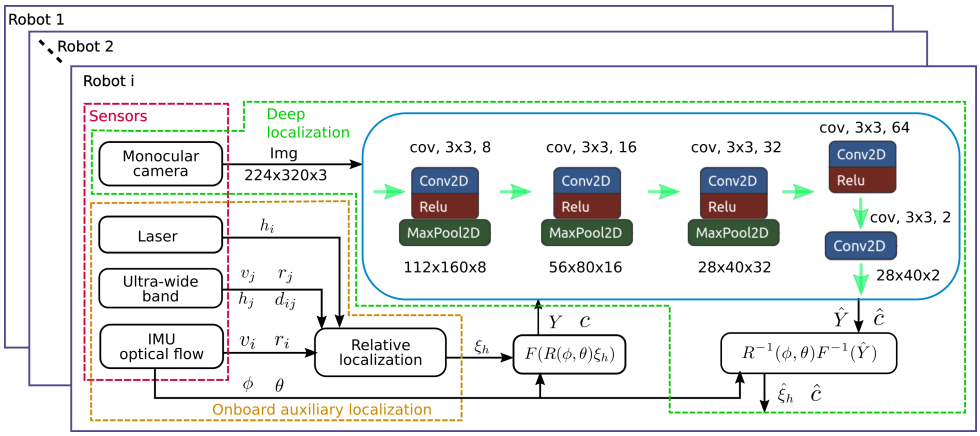


Figure 4.2: System framework for deep relative localization of tiny flying robots. Red block shows all onboard sensors. Yellow block is the onboard auxiliary localization, which outputs the relative position label. This data is transformed to pixel position and depth for training the neural network as shown in the green block. After training, the deep localization can use purely monocular image to estimate the 3D multi-robot relative pose. In addition, the network architecture consists of convolution layers, Relu activation, and max pooling layers as shown in the blue block. It also shows the kernel size, input and output channels, and the image size for each step.

4.2.3. SELF-SUPERVISED LOCALIZATION PROBLEM

Suppose the multi-robot system is equipped with the aforementioned localization techniques. Each robot captures the monocular image and obtains the corresponding label $\boldsymbol{\xi}_h$ automatically. Given the image Im with a peer robot in it, the self-supervised deep localization problem is to find a deep neural network $f_n(Im)$ that can predict peer-robot relative positions $\boldsymbol{Y}$ which satisfies $\boldsymbol{Y} = [x_p, y_p, x_c]^T$. According to (4.1) and (4.2), the desired relative positions $\boldsymbol{Y}$ can be derived from automatic label $\boldsymbol{\xi}_h$ as:

$$\begin{aligned} \boldsymbol{Y} &= F(R(\phi,\theta)\boldsymbol{\xi}_h) = [\boldsymbol{\xi}_p^{[0:2]}; x_c] \\ &= [(M_{\text{itr}} R_c R(\phi,\theta)\boldsymbol{\xi}_h / x_c)^{[0:2]}; x_c] \end{aligned} \tag{4.3}$$

where $\boldsymbol{\xi}_p^{[0:2]}$ means the first two rows of the vector $\boldsymbol{\xi}_p$. In addition, multiple robots can appear on one monocular image. Thus, the deep inference function $f_n(Im)$ must be able to predict 3D positions of multiple robots.

4.3. METHODOLOGY

This section describes the detailed approach to self-supervised monocular relative localization. The system framework, network architecture and loss functions are explained, respectively.

4.3.1. NETWORK OUTPUT AND SELF-SUPERVISED DATASET

The data flow of the whole system is shown in Fig. 4.2. As can be seen, it starts from the auxiliary localization block, which generates the labeled dataset automatically for self-supervised training. The dataset consists of monocular images and the corresponding inter-robot positions $\boldsymbol{Y} = [x_p, y_p, x_c]^T$. To detect multiple robots, the network output $f_n(Im)$ is designed as a 28x40 grid map with the predicted depth channel $\hat{d}(i,j)$ and the confidence channel $\hat{c}(i,j)$. A higher dimensional grid map leads to more accurate pixel localization, which however increases the ambiguity between detections at neighbor pixels. The grid labels are created by the following rules:

$$\begin{cases} c(i,j) = 1, d(i,j) = x_c, & \text{if(i,j)} = (\mathrm{x_p}/8, \mathrm{y_p}/8) \\ c(i,j) = 0, d(i,j) = 0, & \text{otherwise} \end{cases} \tag{4.4}$$

which means a grid that contains a robot has confidence of 1 and the depth of x_c in camera coordinate. Since $x_p \in [0,320)$ and $y_p \in [0,224)$, $(x_p, y_p)/8$ fits with the network output size.

In order to obtain a more generalized network, we pretrain the network on a synthetic dataset. In this dataset, the grid labels can be created automatically by the masks of robots during 3D rendering. The synthetic dataset contains more different backgrounds and arbitrary random attitudes and positions of the drones.

4.3.2. NETWORK ARCHITECTURE

Our deep network is inspired by YOLOv3 network which can detect multiple small objects with bounding boxes [26]. We modify the YOLOv3 network to solve the localization problem as described in Section 4.2.3. The proposed network predicts the pixel position and depth of peer robots as can be seen in the blue block in Fig. 4.2, instead of bounding boxes as the original YOLOv3 network. The feature maps and layers of the original network are largely reduced, as we only predict one class of robots. Also, this simplified network fits with the implementation on a resource-limited tiny AI chip, the GAP8 microprocessor, which was first demonstrated for autonomous corridor following in [27].

Specifically, the proposed network is an encoder with eight main layers including convolution and max pooling. The activation adopts the Relu function, and there is batch normalization during the training process. No anchors are required as it focuses on the center of the object. The output layers are modified to predict the confidence grid map for localizing the robot center, and the depth grid map.

4.3.3. LOSS FUNCTIONS

The loss functions are also different from those in object detection networks. The total loss of the proposed network is composed by two individual items:

$$l = l_\mathrm{d} + l_\mathrm{c} \tag{4.5}$$

Depth loss l_d denotes the mean of square errors between estimated depth $\hat{x}_c$ and real value x_c in grid (i,j), which is represented by

$$l_\mathrm{d} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}} \sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}} \mathrm{c(i,j)} \cdot (\hat{\mathrm{d}}_\mathrm{c}(\mathrm{i,j}) - \mathrm{d(i,j)})^2) \tag{4.6}$$

where $c(i,j)$ is the real confidence of whether there exists an robot center in grid (i,j) defined in (4.4). $N_{yc} = 28$ and $N_{xc} = 40$ are the sizes of output grid maps in the proposed network.

Confidence loss item l_{c} is designed by softmax cross entropy function [26], and the formula is

$$l_{\mathrm{c}} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}}\sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}}(\mathrm{c(i,j)}-\hat{\mathrm{c}}\mathrm{(i,j)})^2[-\mathrm{c(i,j)}\cdot\log(\hat{\mathrm{c}}\mathrm{(i,j)})-(1-\mathrm{c(i,j)})\cdot\log(1-\hat{\mathrm{c}}\mathrm{(i,j)})]) \tag{4.7}$$

where $\hat{c}(i,j)$ and $c(i,j)$ are the predicted and real confidence in grid (i,j) defined in (4.4).

An optional loss item is to distinguish multiple classes of robots by using the one-hot vectors. l_{prob} demonstrates the class probability error, written as

$$l_{\mathrm{p}} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}}\sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}}\mathrm{c(i,j)}\cdot[-\mathrm{p(i,j)}\cdot\log(\hat{\mathrm{p}}\mathrm{(i,j)})-(1-\mathrm{p(i,j)})\cdot\log(1-\hat{\mathrm{p}}\mathrm{(i,j)})]) \tag{4.8}$$

where $\hat{p}(i,j)$ and $p(i,j)$ are predicted and real probability of different classes in grid (i,j). This item has been tested to be effective for robot classification, but not included in this chapter for a better quantization of the network to run in the microprocessor.

4.3.4. TRAINING AND POST PROCESSING

The proposed deep relative localization network is trained from scratch in the environments of Anaconda and Tensorflow2. There are 25 epochs for total training including 2 warm epochs to reduce the primacy effect and avoid the early over-fitting. Given 800 training images and a batch size of five images, each epoch has 160 steps. The learning rate changes adaptively with Adam method. The training can be either run on GPU or CPU as the network size is very small.

After training the network on synthetic images, the network is refined on the real-world dataset (192 training images) captured on two Crazyflies. The refined model file is quantized into int8 format by the GAP8 tool chains. Finally, it is compiled to a C function that can run on the GAP8 microprocessor.

Based on the prediction $\hat{c}(i,j)$ and $\hat{d}(i,j)$ from the network $f_n(Im)$, the final relative position $\boldsymbol{Y}$ can be calculated by

$$[\hat{x}_p, \hat{y}_p, \hat{x}_c]^{\mathrm{T}} = \{[8\cdot i, 8\cdot j, \hat{d}(i,j)]^{\mathrm{T}}) | \hat{c}(i,j) > T_c\}. \tag{4.9}$$

where T_c denotes a proper confidence threshold, which was empirically selected as 0.33 for synthetic dataset and 0.23 for real-world tests.

4.4. SIMULATION

This section shows the developed simulation pipeline for synthetic image generation, training and testing results of the network on the synthetic dataset. This simulated environment is not necessary for an onboard deep neural network, but it can generate a flexible and rich multi-robot visual dataset easily for preliminary validation of the DNN. Refining the network on the simulated dataset saves training time and promises more generality of the neural network.

4.4.1. SIMULATED PIPELINE FOR 3D MULTI-ROBOT RENDERING

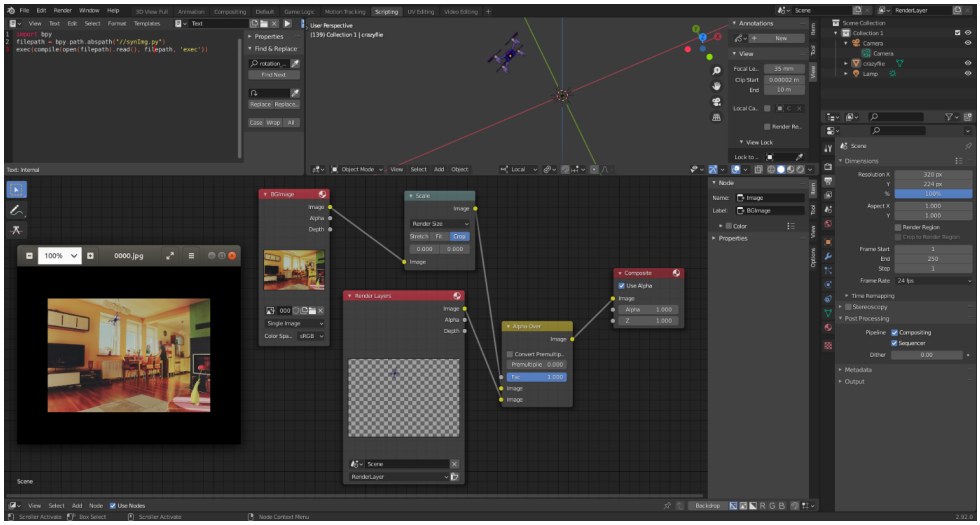


Figure 4.3: The developed 3D rendering simulation environment. It can render multiple 3D robots in any background images. The attitudes and positions of robots and camera can be set in python code.

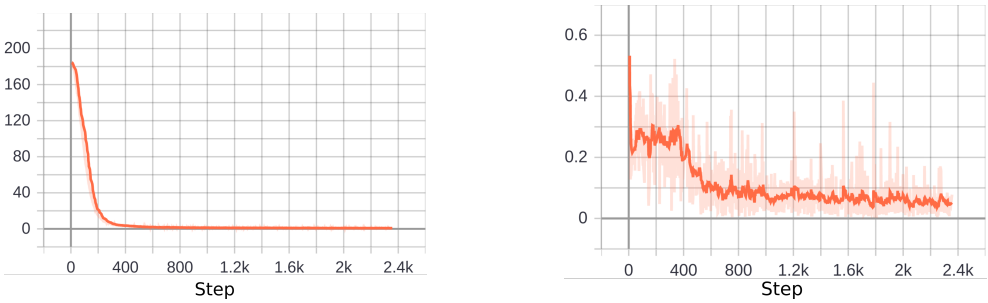
We built a Blender-based 3D rendering environment as shown in Fig. 4.3 for generating synthetic images in a multi-robot domain. The whole pipeline is light and can render 3D images with a random number of robots in the images with random attitude and positions. The attitude of the camera can be also set arbitrarily. The background images are from the COCO dataset, specifically the 2017-Val-images set including 5000 images. Annotation of robot positions and depth to the camera can be obtained from the prior known groundtruth in Blender. The rendered images can be seen in Fig. 4.5. This tool is light-weight, open-source and easily modified for generating multi-robot images for other applications.

4.4.2. TRAINING ON SYNTHETIC DATASET

From Fig. 4.4, we can see all loss items decrease as the training steps increase. They are stable at about 1000 steps. Specifically, confidence error tends to converge to zero which indicates that the deep neural networks encode the robot pixel position in the output grid maps. The stable depth error is about 0.05m, which is not approximating zero probably because different robot attitudes lead to slight depth changes. However, it is accurate enough for robot position estimation.

4.4.3. TESTING RESULTS ON SYNTHETIC IMAGES

The prediction errors are shown in this subsection. All testing results come from network prediction on the test dataset which contains 200 new images with different backgrounds and different poses of peer robots.



(a) Confidence loss

(b) Depth loss (meter)

Figure 4.4: Loss changes with training steps on synthetic dataset. The network is trained from scratch on a training dataset with 800 images. Two individual loss items are shown in this figure.

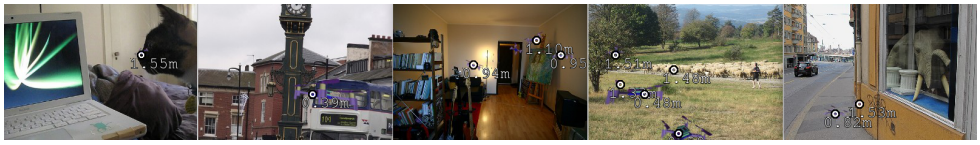


Figure 4.5: Testing results of multi-robot localization on synthetic images. The white circle with an outer black circle represents the predicted robot position in the image. Different robot attitude and position with respect to the camera are demonstrated in these figures, as well as multiple robots in indoor and outdoor environments.

The testing results on synthetic images are shown in Fig 4.5. From these figures, we can see the proposed network can detect different sizes of robots in outdoor and indoor environments. In addition, thanks to the underlying (modified) YOLOv3 framework, the network is capable to detect multiple robots in one image at same time, even though it is only trained on dataset with a single robot in each image. There are a false-positive detection on the fourth image and a true-negative detection on the third image, potentially due to similar background and less features. These outliers can be rejected by filters in real-time sequences of images. Note that these five test images are selected randomly. Hence, the deep network has similar effectiveness on the other testing images.

For explicit demonstration, the statistical 3D estimation error is depicted by Fig. 4.6. From the left figure, we can see the robot localization prediction in image has a zero average pixel error. Most position error is within 20 pixels and with few outliers caused by a few false detections. The right figure demonstrates the error distribution of depth predictions, where mean and medium values are approximating zero.

Therefore, compared to other state-of-the-art research in relative localization with deep learning, our proposed network is much smaller and thus efficient for both training (20 minutes on i7 CPU) and testing. Besides, the drone depth is predicted simultaneously with the drone detection, while other references require the prior-known drone size to estimate its depth.

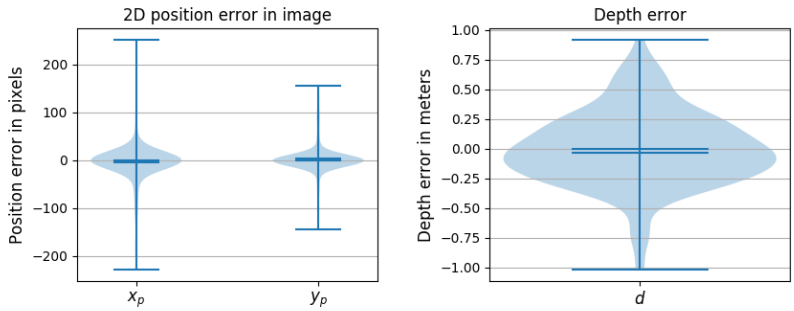


Figure 4.6: 2D robot position error (left figure) in image and depth error (right figure) between prediction and groundtruth are shown. This distribution is based on 200 testing images with single robot in each image.

4.5. EXPERIMENTS

This section presents practical experiments: flight for real-world dataset collection, refining of the neural network on the real dataset, porting the Tensorflow-based network to AIdeck microprocessor, and onboard deep visual localization.

4.5.1. HARDWARE

The multi-robot platform is composed by two Crazyflie2 quadrotors, which are tiny flying robots with only 33 grams and pocket size (12.5 cm in diameter). Three decks are required for our work. The first one is AI deck [27], which is composed by a GAP8 RISC-V processor, a Himax HM01B0 RGB camera, 512 Mbit HyperFlash for storing dataset, and an ESP32 wifi module for remote streaming. Another deck is the optical flow deck for estimating robot velocity and altitude. The last one is the loco deck with a UWB module for inter-robot ranging measurements and auxiliary localization.

The last two decks are used for generating relative position labels automatically, while only the first deck is used for deep visual multi-robot localization.

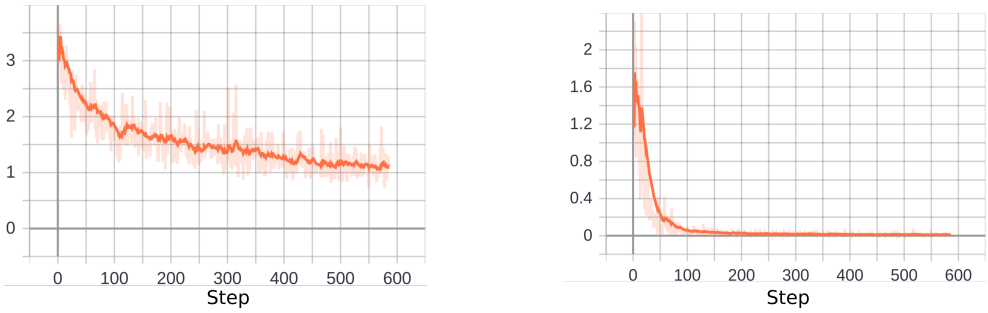
4.5.2. REAL-WORLD DATASET ACQUISITION

Due to the reality gap between synthetic and real images, a real-world dataset with 240 images is collected for refining the network. During data collection, one drone flies with randomly velocities by a remote controller, while another drone flies with random velocities but in the view of the camera on the first drone. The example dataset can be found in following sections.

To get dataset among two flying tiny drones, a specific procedure is designed: a quadrotor sends its 2D attitude (roll and pitch) and 3D auxiliary relative position to the GAP8 chip, which combines these variables with the monocular image data and store them in HyperFlash memory. Afterwards, the GAP8 sends the data from flash memory to a computer via a Olimex ARM-JTAG-20-10 debugger.

4.5.3. REFINING AND TESTING ON REAL-WORLD DATASET

The training process on the real-world dataset is the same as that in Section 4.4. The loss changes during refining process are shown in the following figure.



(a) Confidence loss

(b) Depth loss (meter)

Figure 4.7: Loss changes of refining the neural network on real-world dataset. The initial weights are those trained by the synthetic images in last section. All training configurations remain the same during refining.

From Fig. 4.7, we can see all loss items drop down again within 600 steps. The confidence loss decreases a bit slowly than that of depth loss, because the appearance changes largely while drone sizes are easily to learn. The real-world dataset is divided into 192 training images and 48 testing images with self-supervised labels of relative position, all obtained from two randomly flying Crazyflies, without relying on any external systems such as motion capture system or GPS.

Figure 4.8: Testing results of the refined network on randomly selected testing images. The white circle in images shows the predicted pixel position of peer robot center. The value on the image means the depth of the robot from the camera. All testing images are captured onboard with different flying attitude and velocity of both quadrotors.

Fig. 4.8 shows the localization performance of the refined neural network. Both training and testing images have three light conditions (evening, afternoon and morning, re-

spectively). These testing results indicate that the refined network can localize the real-world flying robots with high accuracy due to the previous training on a more generalized synthetic dataset. It also works even with large motion blur, partial occlusion, and different positions and attitudes of the robots.

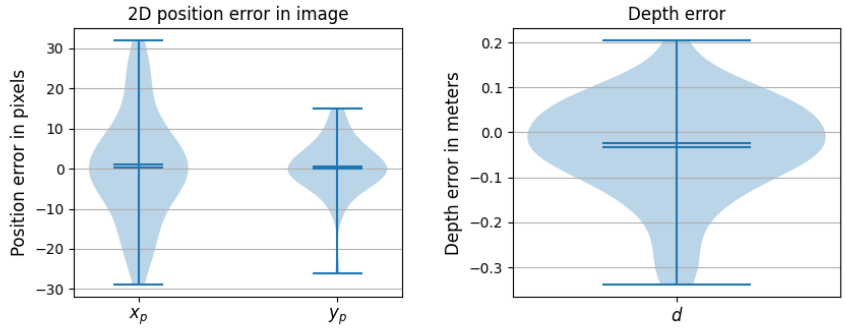


Figure 4.9: Left: the 2D pixel position error of robot center between deep inference and UWB localization. Right: the depth error distribution in camera coordinate. These results are based on testing dataset with 48 new real-world images with size of 224x320.

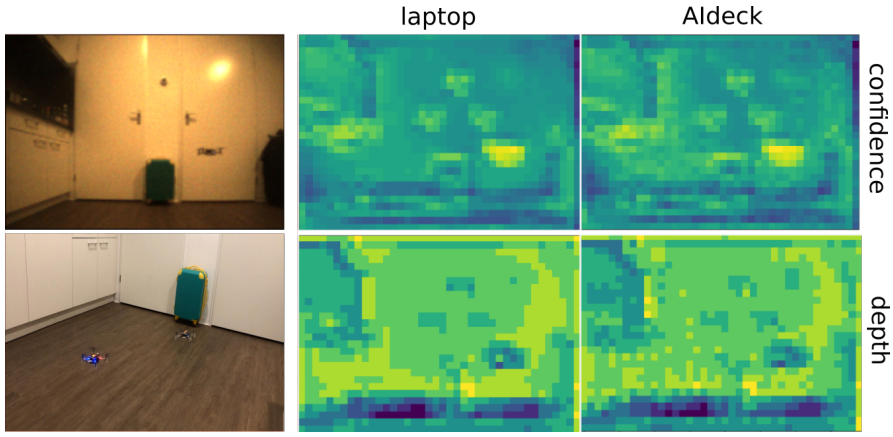


Figure 4.10: Comparison of the deep visual relative localization between running on the laptop and the onboard AI chip. Top left: an example captured image during flight; bottom left: the experimental flight of two quadrotors; top middle: confidence channel predicted on laptop; bottom middle: depth channel predicted on laptop; top right: confidence channel predicted on AIdeck; bottom right: depth channel predicted on AIdeck.

Fig. 4.9 demonstrates the statistical prediction errors in image coordinate, by comparing the deep visual localization and auxiliary UWB localization. The refined network has even smaller localization and depth prediction errors than on the synthetic dataset.

4.5.4. ONBOARD DEEP RELATIVE LOCALIZATION WITH AIDECK

This subsection shows the implementation of the deep localization network into a low-cost and ultra low-power AI chip. After quantization from float to int8, the network can run on the AIdeck microprocessor to predict peer-robot position onboard a tiny flying robot.

Fig. 4.10 compares the deep inference results on both laptop (the middle column) and the edge AI chip (the right column) for the same flight image (top left). The network outputs on PC multiply the quantization scale calculated by the GAP8 tools. Two right images on first row shows the confidence prediction, where both laptop and AIdeck have similar inference of drone localization in the image. Though they have slight differences in depth prediction as seen in the right two images on second row, the depth values with respect to the robot center area are similar.

4.6. CONCLUSIONS

This chapter proposes a novel multi-robot relative localization framework which are self-supervised, autonomous, low-cost, and scalable. Both simulation and experimental results indicate that the proposed deep network can predict peer-robot relative positions with monocular images, without using any external positioning system. In addition, this chapter solves a challenging problem, i.e., implementation of deep neural network into a low-cost tiny AI chip that enables visual multi-robot localization of tiny flying robots.

Future work could include real-world dataset collection in more scenes, and control of a large number of flying robots with the proposed localization method.

REFERENCES

[1] S. Li, C. De Wagter, and G. C. H. E. de Croon, *Self-supervised monocular multi-robot relative localization with efficient deep neural networks,* arXiv preprint arXiv:2105.12797 (2021).

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

[3] Q. Lindsey, D. Mellinger, and V. Kumar, *Construction of cubic structures with quadrotor teams,* Proc. Robotics: Science & Systems VII (2011).

[4] 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** (2018).

[5] M. Basiri, F. Schill, D. Floreano, and P. U. Lima, *Audio-based localization for swarms of micro air vehicles,* in *2014 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2014) pp. 4729–4734.

[6] 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).

[7] K. McGuire, C. De Wagter, K. Tuyls, H. 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** (2019).

[8] 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).

[9] 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).

[10] S. Vemprala and S. Saripalli, *Monocular vision based collaborative localization for micro aerial vehicle swarms,* in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2018) pp. 315–323.

[11] R. T. Rodrigues, P. Miraldo, D. V. Dimarogonas, and A. P. Aguiar, *A framework for depth estimation and relative localization of ground robots using computer vision,* in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019) pp. 3719–3724.

[12] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Bülthoff, M. J. Black, and A. Ahmad, *Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles,* IEEE Robotics and Automation Letters **3**, 3193 (2018).

[13] 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).

[14] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, *Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 8776–8782.

[15] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, *Drone detection using depth maps,* in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2018) pp. 1034–1037.

[16] B. Joshi, M. Modasshir, T. Manderson, H. Damron, M. Xanthidis, A. Q. Li, I. Rekleitis, and G. Dudek, *Deepurl: Deep pose estimation framework for underwater relative localization,* in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2020) pp. 1777–1784.

[17] F. Schilling, F. Schiano, and D. Floreano, *Vision-based drone flocking in outdoor environments,* IEEE Robotics and Automation Letters **6**, 2954 (2021).

[18] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, *Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 1521–1529.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, *Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,* arXiv preprint arXiv:1711.00199 (2017).

[20] A. Kendall, M. Grimes, and R. Cipolla, *Posenet: A convolutional network for real-time 6-dof camera relocalization,* in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 2938–2946.

[21] T. Hodan, D. Barath, and J. Matas, *Epos: estimating 6d pose of objects with symmetries,* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 11703–11712.

[22] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, *Camera-to-robot pose estimation from a single image,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 9426–9432.

[23] K. Wada, E. Sucar, S. James, D. Lenton, and A. J. Davison, *Morefusion: multi-object reasoning for 6d pose estimation from volumetric fusion,* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 14540–14549.

[24] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, *Self-supervised 6d object pose estimation for robot manipulation,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 3665–3671.

[25] J. Hur and S. Roth, *Self-supervised monocular scene flow estimation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 7396–7405.

[26] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement,* arXiv preprint arXiv:1804.02767 (2018).

[27] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, *A 64-mw dnn-based visual navigation engine for autonomous nano-drones,* IEEE Internet of Things Journal **6**, 8357 (2019).

5

NONLINEAR MODEL PREDICTIVE CONTROL FOR IMPROVING RANGE-BASED RELATIVE LOCALIZATION BY MAXIMIZING OBSERVABILITY

Wireless ranging measurements have been proposed for enabling multiple Micro Air Vehicles (MAVs) to localize with respect to each other. However, the high-dimensional relative states are weakly observable due to the scalar distance measurement. Hence, the MAVs have degraded relative localization and control performance under unobservable conditions as can be deduced by the Lie derivatives. This chapter presents a nonlinear model predictive control (NMPC) by maximizing the determinant of the observability matrix in order to generate optimal control inputs, which also satisfy constraints including multi-robot tasks, input limitation, and state bounds. Simulation results validate the localization and control efficacy of the proposed MPC method for range-based multi-MAV systems with weak observability, which has faster convergence time and more accurate localization compared to previously proposed random motions.

Parts of this chapter have been accepted by the International Micro Air Vehicle Conference and Competition (2021).

5.1. INTRODUCTION

The use of multiple aerial robots has been studied deeply in recent years for more complicated tasks and challenging environments [1]. For example, a predictive control is proposed for flights of a swarm of five quadrotors despite cluttered obstacles [2]. In outdoor confined spaces, multiple drones are controlled with the evolutionary optimization method for flocking flights [3]. Multiple flying robots coordinate with simultaneous localization based on ranging measurements with beacons [4]. These recent studies show the state-of-art aerial swarm methods. However, most of them rely on extra positioning systems such as indoor optiTrack [2], outdoor GPS [3] or beacons [4].

To remove the dependence of the external infrastructure such as positioning systems, onboard sensors are deployed for developing an autonomous swarm of drones. For example, 3D relative direction can be estimated by sound-based microphone arrays and allows for leader-follower flights of micro aerial vehicles [5]. An array of infrared sensors can also enable relative positioning and inter-robot spatial-coordination [6]. However, these sensor arrays are too heavy and power-consuming for tiny flying robots. In [7], fully distributed and autonomous multiple tiny flying robots explore unknown environments with finite state machine. However, the relative localization is not very accurate due to the direct usage of signal strength, which may not fulfil the precise cooperative tasks.

Vision is the most widely used solution for multi-robot relative localization. Outdoor flocking of multiple drones localize each other with deep neural network and cameras for a safe navigation [8], which requires heavy AI hardware to run the deep network, also for [9] and [10]. Marker-based localization requires simple computation such as recognizing black circles [11] or April tags [12]. But these visual methods are easily influenced by the field of view or lighting conditions that lead to detection failure and localization disaster.

Wireless ranging sensors provide omnidirectional and low-cost ranging measurements, and recently have been used frequently for relative localization. It was initially proposed in [13], where use was still made of Bluetooth in order to fit on tiny MAVs. In [14], an ultra-wide band (UWB) based cooperative relative localization was proposed to estimate the neighbor drones' position based on the distance and self-displacement measurements under common orientation. Furthermore, [15] removes the orientation assumption and achieves the relative localization purely using the distance measurement and acceleration model. However, these experiments assumed high-order dynamic model and has low ranging frequency, which is not efficient for a large number of tiny robots.

In [16], a simplified velocity model and robust ranging protocol are designed for multiple tiny flying robots with self-regulated localization convergence. However, the initialization procedure with random velocity inputs is not efficient. Thus, this chapter considers using nonlinear MPC to design the multi-robot controller by maximizing the task performance and degree of observability, while satisfying the constraints such as input velocity bounds and state bounds.

There are some related papers discussing the control of bearing-based or range-based multi-robot systems [17]. Most papers use persistent excitation methods by setting specific active control patterns to maintain observability, which is not flexible nor optimal for other tasks or constraints.

The main contribution of this chapter is leveraging weak observability theory to optimize the multi-robot control inputs, which has not yet been presented, to the best knowl-

edge of authors. Specifically, the proposed NMPC framework maximizes the nonlinear observability condition derived by Lie derivatives, which is coupled with the velocity inputs and relative states. This leads to faster localization convergence and higher estimation accuracy even after convergence, compared to the random control inputs.

The rest of the chapter is organized as follows. Section 5.2 states the problem including the range-based multi-MAV model, weak observability condition and the problem definition. Section 5.3 proposes the nonlinear MPC method with the cost function and corresponding constraints. Section 5.4 gives the simulation results of the proposed control with Acados, an integrated nonlinear MPC tool. The conclusion is discussed in Section 5.5.

5.2. PRELIMINARIES

This section briefly introduces the multi-MAV kinematic model and relative Kalman filter. Based on the relative model and distance observations, the observability matrix is determined with Lie derivatives. Finally, the control problem is defined by considering both the model and observability.

5.2.1. RELATIVE MULTI-MAV MODEL

The model of twin MAVs is described in this subsection, as the relative localization is distributed and triggered by the ranging event among arbitrary two MAVs. The simulated relative model has been tested in real experiments in our previous work, thus it has a small gap compared to the real-world multi-robot system. For details, consult in [16].

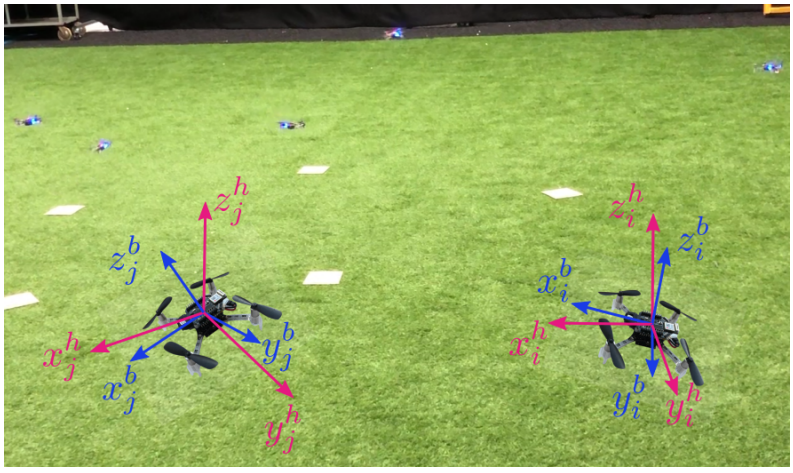


Figure 5.1: The diagram of a twin-MAV kinematic model, and two coordinated frames. Body frames and horizontal frames are shown with blue axes and red axes, respectively. Both frames are fixed to the robot, while the horizontal frames always have a vertical Z axis. The background images shows previous experiments of multi-MAV relative localization but without optimal control.

For simplicity, we assume the yaw rate of both robots to be zero. This assumption has no influence on the 3D movements of each robot. The control input vector $\boldsymbol{u} = [v_i^x, v_i^y, v_j^x, v_j^y]^T$

represents the XY-axis velocities of the i^{th} and j^{th} robots in their horizontal frames as shown in Fig. 5.1. The velocities in horizontal frame can be obtained by rotating the measured velocities in body frame, so that the Z axis in the horizontal frame aligns with gravity. The relative state is denoted by $\boldsymbol{x} = [x_{ij}, y_{ij}, \psi_{ij}]^T$, which represents the j^{th} robot's position and relative yaw in the horizontal frame of the i^{th} robot.

The nonlinear relative kinematic model can be derived from Newton formulas by considering the states $\boldsymbol{x}$ and velocity inputs $\boldsymbol{u}$, which can be written as follows

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} cos(\psi_{ij})v_j^x - sin(\psi_{ij})v_j^y - v_i^x \\ sin(\psi_{ij})v_j^x + cos(\psi_{ij})v_j^y - v_i^y \\ 0 \end{bmatrix}. \tag{5.1}$$

Here, 2D single-integrator model is introduced as the velocity and height can be estimated and measured by the sensors.

A distance measurement d comes from the DWM1000 ranging sensors, and has the following relation to model states:

$$d = h(\boldsymbol{x}) = \sqrt{x_{ij}^2 + y_{ij}^2 + (h_j - h_i)^2}, \tag{5.2}$$

where h_i and h_j are the altitudes measured directly from the height sensors. The function $h(\cdot)$ represents the scalar nonlinear observation.

5.2.2. RELATIVE ESTIMATION

This subsection briefly reviews the Extended Kalman filter (EKF) for the relative localization. The discrete prediction is formulated as:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1|k} &= F(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \hat{\boldsymbol{x}}_k + \dot{\boldsymbol{x}}_k \Delta t, \\ \boldsymbol{P}_{k+1|k} &= \boldsymbol{A}_k \boldsymbol{P}_{k|k} \boldsymbol{A}_k^T + \boldsymbol{B}_k \boldsymbol{Q}_k \boldsymbol{B}_k^T \end{aligned} \tag{5.3}$$

where Δt is the update interval, $\boldsymbol{P}$ is the error covariance, $\boldsymbol{A} = \partial F / \partial \boldsymbol{x}$ and $\boldsymbol{B} = \partial F / \partial \boldsymbol{u}$ are the Jacobians of states and inputs, and $\boldsymbol{Q}$ is the process noise covariance.

The final state estimation is estimated by using the distance measurement as shown below:

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}, \\ \hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (d_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1}), \\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1} \end{aligned} \tag{5.4}$$

where $\boldsymbol{K}$ is the Kalman gain, $\boldsymbol{H} = \partial h(\boldsymbol{x}) / \partial \boldsymbol{x}$ is the observation Jacobian, $\boldsymbol{R}$ is the observation noise covariance, and $\boldsymbol{I}$ is the identity matrix.

Remark 4. *The kinematic model and EKF-based relative localization have been validated in real-world experiments [16].*

5.2.3. OBSERVABILITY CONSTRAINT

Observability of nonlinear systems can be analyzed by Lie derivatives [18]. The corresponding observability matrix is defined as

$$\boldsymbol{O} = [\nabla \mathscr{L}_f^0 h, \nabla \mathscr{L}_f^1 h, \nabla \mathscr{L}_f^2 h]^T \tag{5.5}$$

where $\mathscr{L}_f h$ means the Lie derivative of function f. The iterations satisfy three conditions: 1) $\mathscr{L}_f^0 h = h(\boldsymbol{x})$; 2) $\mathscr{L}_f^{i+1} h = \nabla \mathscr{L}_f^i h \cdot f$; 3) $\nabla \mathscr{L}_f^i h = \partial \mathscr{L}_f^i h / \partial \boldsymbol{x}$.

Therefore, the relative states are observable only when the observability matrix $\boldsymbol{O}$ is full rank. That means that the determinant should be non-zero, which is expressed as:

$$\begin{aligned} |\boldsymbol{O}| = f_O(\boldsymbol{x}, \boldsymbol{u}) = -2[&-v_i^x v_j^x s(\psi) + v_i^y v_j^x c(\psi) \\ &- v_i^x v_j^y c(\psi) - v_i^y v_j^y s(\psi)] \cdot [-v_j^x y_{ij} c(\psi) + v_j^y y_{ij} s(\psi) \\ &+ v_i^x y_{ij} + v_j^x x_{ij} s(\psi) + v_j^y x_{ij} c(\psi) - v_i^y x_{ij}] \end{aligned} \tag{5.6}$$

where $s(\cdot)$, $c(\cdot)$, and ψ are simplifications of $sin(\cdot)$, $cos(\cdot)$, and ψ_{ij}, respectively.

5.2.4. PROBLEM STATEMENT

The optimal control problem P_O with respect to observability $|O|$ for this multi-MAV system is defined as:

$$\max_{u^*, k \in \{1,2 \ldots \mathrm{N}\}} P_O(|O_k|) = \sum_{k=1}^{\mathrm{N}} |f_O(\boldsymbol{x}_k, \boldsymbol{u}_k)| \tag{5.7}$$

where $\boldsymbol{u}^*$ is the optimal control vector at current time, which is normally taken from an control sequence. The appropriate control input sequence guarantees the strong observability of the multi-robot system in the future, which can improve the relative localization in both convergence speed and estimation accuracy. The problem is how to calculate the optimal control input sequence.

5.3. METHODOLOGY

This section proposes a nonlinear model predictive control for solving the optimal problem as described in (5.7). Then the cost function is further extended for multi-robot tasks such as formation control and motion tracking. In the end, the solver settings for the nonlinear problem (NLP) are presented.

5.3.1. NONLINEAR MPC

The intuitive solution for NLP is MPC, which can achieve the target by minimizing the cost function. Hence, the nonlinear MPC for the proposed problem is designed as follows

$$\boldsymbol{u}_{0|t} := \min_{\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}} J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) \tag{5.8a}$$

$$\text{s.t. } \boldsymbol{x}_{k+1|t} = f(\boldsymbol{x}_{k|t}, \boldsymbol{u}_{k|t}) \delta_t + \boldsymbol{x}_{k|t}, \tag{5.8b}$$

$$\boldsymbol{x}_{0|t} = \mathrm{sat}_{\boldsymbol{x}_\mathrm{l}}^{\boldsymbol{x}_\mathrm{u}}(\hat{\boldsymbol{x}}_\mathrm{t}), \tag{5.8c}$$

$$\|\boldsymbol{p}_{\cdot|t}\|_2 - d_{\text{safe}} \geq 0, \tag{5.8d}$$

$$v_l \leq v_{i,\cdot|t}^x, v_{i,\cdot|t}^y, v_{j,\cdot|t}^x, v_{j,\cdot|t}^y \leq v_u, \tag{5.8e}$$

$$p_l \leq x_{ij,\cdot|t}, y_{ij,\cdot|t} \leq p_u \tag{5.8f}$$

where $\boldsymbol{x}_{\cdot|t}$ and $\boldsymbol{u}_{\cdot|t}$ stands for the sequence of states and control inputs in the prediction horizon. The first control value $\boldsymbol{u}_{0|t}$ is taken as the input for the robots. The relative position is denoted by $\boldsymbol{p} = [x_{ij}, y_{ij}]^T$. Saturation function sat() clips data with lower bound $\boldsymbol{x}_l$ and upper bound $\boldsymbol{x}_u$.

The overall objective function $J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ is composed by several cost functions, which will be designed later. The remaining equations represent the constraints, which guarantee that the controller satisfies the system dynamic as (5.8b), the initial state condition related to the current estimated state as (5.8c), the safe distance for collision avoidance as (5.8d), the upper and lower bounds of input velocities as (5.8e), and the relative position bounds as (5.8f).

Remark 5. *The constraint of initial state is related to the estimated state which is not correct before localization convergence. Hence, the limitation of initial value is necessary to avoid singularity when solving the NLP. A saturation function is employed to limit the the initial value as shown in* (5.8c)*. This is reasonable as many nonlinear robust MPC methods for systems with uncertain states have their stability proof by assuming bounds on the uncertain state [19].*

5.3.2. COST FUNCTIONS

A nonlinear least square (NLS) method is deployed for minimizing the objective function of (5.8), which is written as:

$$J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) = J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) + J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) \tag{5.9}$$

where $J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ and $J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ represent the reformulated observability cost and multi-robot formation coordination cost, respectively.

To maximize the observability with the NLS method, the observability objective (5.7) is reformulated as the following cost function.

$$J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) = \sum_{k=0}^{\mathrm{N}-1} \omega_O \| \frac{a_O}{f_O(\boldsymbol{x}_k, \boldsymbol{u}_k) + \epsilon_O} \| \tag{5.10}$$

where ω_O, a_O and ϵ_O denote the constant weight, amplitude of cost value, and a small value preventing the singularity.

The coordination cost of $J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ is designed for multi-robot tasks such as motion tracking. Given the reference position sequence $\widetilde{\boldsymbol{p}}_{\cdot|t} = [\widetilde{x}_{ij,\cdot|t}, \widetilde{y}_{ij,\cdot|t}]^T$, the motion tracking cost function is designed as:

$$J_C(\boldsymbol{x}_{\cdot|t}) = \sum_{k=0}^{\mathrm{N}-1} \omega_C \| \boldsymbol{p}_{k|t} - \widetilde{\boldsymbol{p}}_{k|t} \| \tag{5.11}$$

where $\boldsymbol{p}_{\cdot|t}$ is the predicted relative position sequence in the proposed MPC. Specially, if the reference sequence is constant such that $\widetilde{\boldsymbol{p}}_{\cdot|t} \equiv [a_x, a_y]$, the multi-robot motion tracking reduces to formation control.

Remark 6. *Since the coordination task is inaccurate before the localization convergence, the weight ω_C can be set dynamically for the control stability according to the localization accuracy, e.g., the trace of the estimation error covariance* $\mathrm{tr}(\boldsymbol{P})$*. However, a constant ω_C is enough for the following formation task.*

Sometimes, a penalty cost can be introduced to smooth the control inputs as follows:

$$J_U(\boldsymbol{u}_{\cdot|t}) = \sum_{k=1}^{\mathrm{N}-1} \omega_U \| \boldsymbol{u}_{k|t} - \boldsymbol{u}_{k-1|t} \| \tag{5.12}$$

Other multi-robot motion control can also be incorporated into the cost function of $J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$. This chapter does not discuss the details of those cost functions such as flocking, swarming, and cooperative coordination.

5.3.3. ACADOS SOLVER

The nonlinear MPC solver we use in this chapter is Acados, which is an open-source and high-performance library for fast optimal control [20]. This software supports Python and is finely tuned for multiple CPU. As for the model definition and differentiation, CasADi is employed to deal with the constraints and model calculations [21].

The brief process of the solver setting is summarized as below. First, the continuous optimal problem is discretized by the multiple shooting method. Furthermore, real-time iteration (RTI) is selected to solve the sequential quadratic programming (SQP). The corresponding Hessian approximation is based on Gauss-Newton. The quadratic problems (QP) in SQP are solved with the partial condensing HPIPM, which is based on linear algebra library BLASFEO. Overall, this solver has a competitive computation speed compared to other stat-of-the-art NMPC solvers.

5.4. SIMULATION RESULTS

This section shows the improvement of the proposed nonliear MPC on the relative localization performance compared to the stochastic initialization procedure studied in [16]. The statistics of the localization errors and convergence speed are analyzed to validate the efficiency of the proposed controller. In addition, adaptive formation flight of multiple MAVs is studied as example application.

5.4.1. SIMULATION SET-UP

The following simulation experiments are conducted on a Dell Latitude 7480 laptop with a i7-6600U CPU with 4 cores at 2.60GHz and 8GB of RAM. For the simulation experiments, the corresponding EKF parameters are chosen as $\Delta t = 0.01s$, $t_{\text{sim}} = 40s$, $\boldsymbol{Q} = \text{diag}([0.25, 0.25, 0.01])$, and $\boldsymbol{R} = \text{diag}([0.1])$. The initial estimated relative states are set to zero. In contrast, the initial ground-truth positions of each robot are randomly generated, such that the EKF estimation has no prior knowledge of the initial state information. The error covariance is initialized as $\boldsymbol{P} = \text{diag}([10, 10, 0.1])$.

As for the parameters of the proposed nonlinear MPC, the horizon is set to N = 50 and prediction time to $T_f = 1s$, which means each control prediction takes $\delta_t = 0.02s$. Larger prediction horizon has long-term constraint guarantees but with more computation burden. In the observability cost function, the parameters are $a_O = 0.021$, $\epsilon_O = 0.001$, and $\omega_O = 1$.

For the constraint settings, the saturation parameters for the initial state vector are chosen as $\boldsymbol{x}_l = [-4, -4, -15]$ and $\boldsymbol{x}_u = [4, 4, 15]$. The safe distance is set to $d_{\text{safe}} = 0.1m$. The velocity input is bounded between $v_l = -2m/s$ and $v_u = 2m/s$. The minimum and maximum relative positions are set to $p_l = -4m$ and $p_u = 4m$, which prevents them flying far from each other.

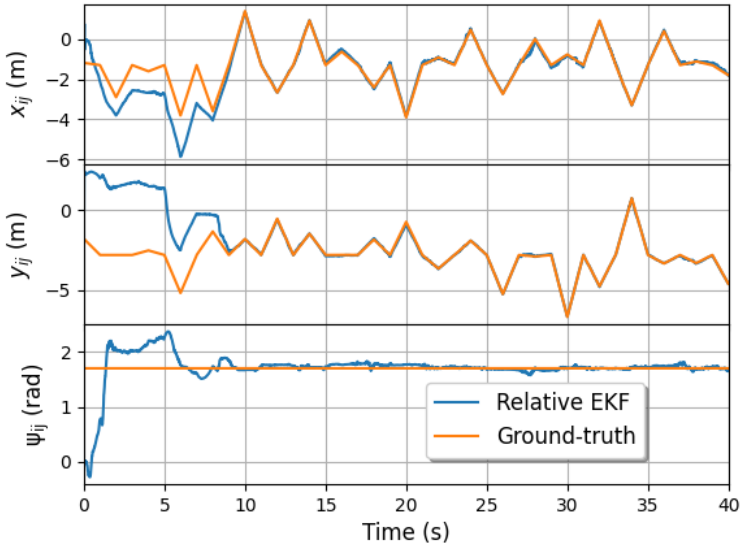


Figure 5.2: Relative state from EKF estimation and ground-truth between two MAVs under the random velocity inputs. The data consists of 2-axis relative positions and 1-axis relative orientation.

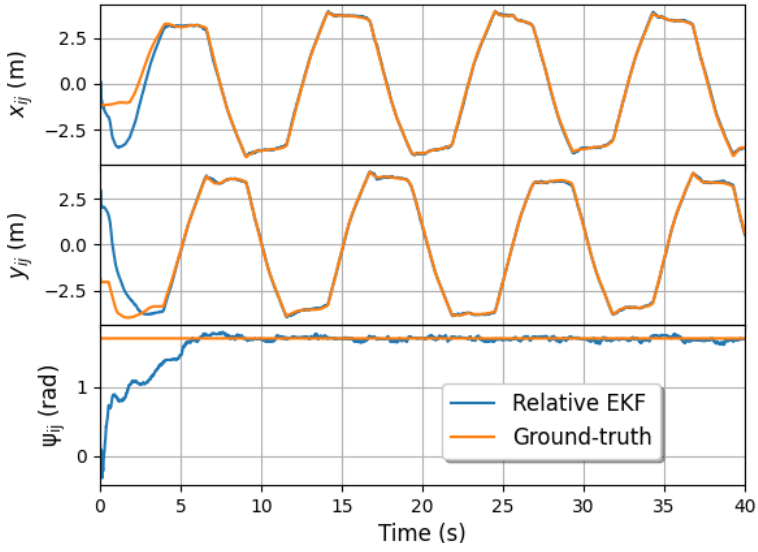


Figure 5.3: Relative state from EKF estimation and ground-truth between two MAVs under the nonlinear MPC controller. The data consists of 2-axis relative positions and 1-axis relative orientation.

5.4.2. IMPROVEMENT ON RELATIVE LOCALIZATION

This subsection compares stochastic initialization with nonlinear MPC, in order to verify that the proposed controller with consideration of pure observability cost has better lo-

calization performance than the former one. In this subsection, the multi-robot task cost J_C is set to zero.

Fig. 5.2 and Fig. 5.3 shows the relative localization performance with the same initial relative states and same parameters for the EKF. Be notified that the three initial states are completely unknown for both the EKF and the controllers. Additionally, the maximum velocities for both controllers are set to be 2 m/s. From these two figures, we can see that the relative positioning with optimal controller has a faster convergence time (about 5s) compared to that of the random controller (about 9s). Especially, observability optimized NMPC has straight convergence in the axis of relative yaw, while the random control leads to overshooting as shown in the third subplot of Fig. 5.2. Therefore, the proposed controller with observability consideration excites all relative states which become more observable even with the unknown initial state errors.

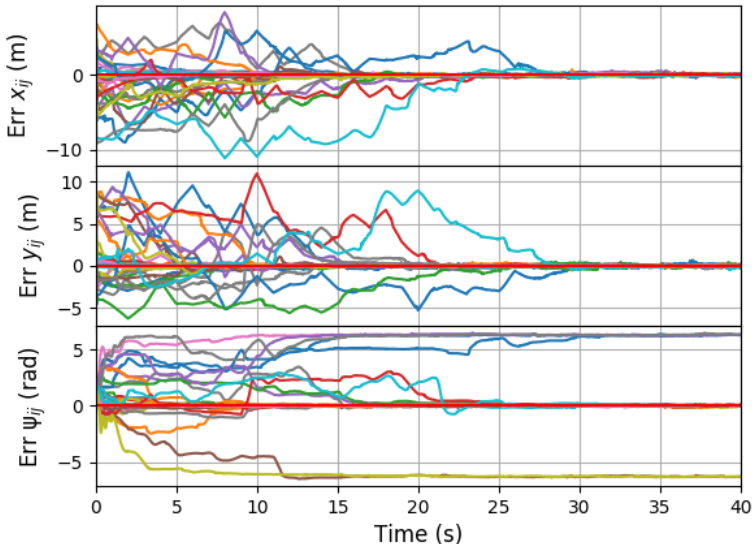


Figure 5.4: 30 simulation experiments of the stochastic controller from [16] with random initial MAV positions. This figure shows the estimation errors of 2-axis relative positions and 1-axis relative orientation.

In addition, after the localization is converged in Fig. 5.3, the optimal controller automatically generates a periodic motion pattern which is similar to the manual-designed persistent excitation motions. In addition, even with incorrect relative states, they still can avoid each other as shown in Fig. 5.3, because the observability cost penalizes the collision situation during which the observability determinant approximates zero.

To validate the general efficacy of the proposed NMPC, we gather more statistics on the performance. As shown in Fig. 5.4 and Fig. 5.5, 30 random simulation experiments are conducted for each controller. During each simulation epoch, the initial positions for both robots are generated randomly. Moreover, the velocity and distance measurement noise are also created randomly. Both figures imply that the proposed NMPC controller

has in general a faster localization convergence speed.

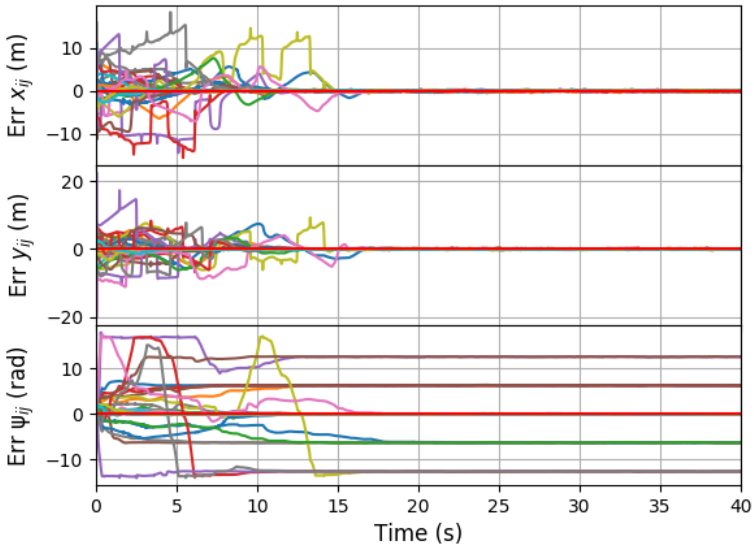


Figure 5.5: 30 simulation experiments of the optimal controller with random initial MAV positions. This figure shows the estimation errors of 2-axis relative positions and 1-axis relative orientation.

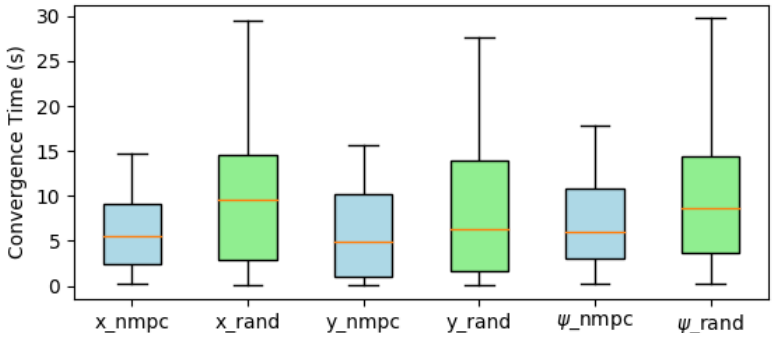


Figure 5.6: The statistics of convergence time of three-dimensional relative localization under 30 random tests. Blue: the proposed nonlinear MPC; Green: the stochastic control.

Fig. 5.6 shows the comparison of the detailed convergence time of two controllers with 30 random tests. From it we can see that the average convergence time of the NMPC on all axes is smaller than that of the random controller. Besides, NMPC with observability constraint has a lower maximum convergence time compared to random control inputs.

Another interesting result is the localization accuracy after estimation convergence. Fig. 5.7 shows the distributions of position estimation errors in the last 5 seconds of two

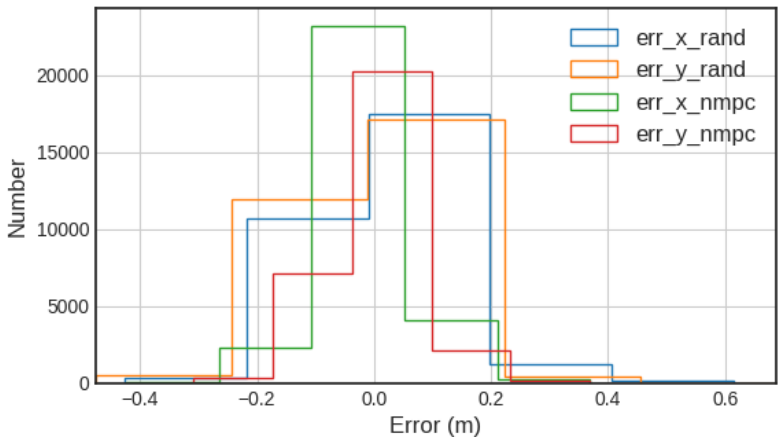


Figure 5.7: Localization error of two controllers after estimation convergence. Each distribution has total 15000 data on these 30 random tests, which is taken from the last 5 seconds when all estimators have converged.

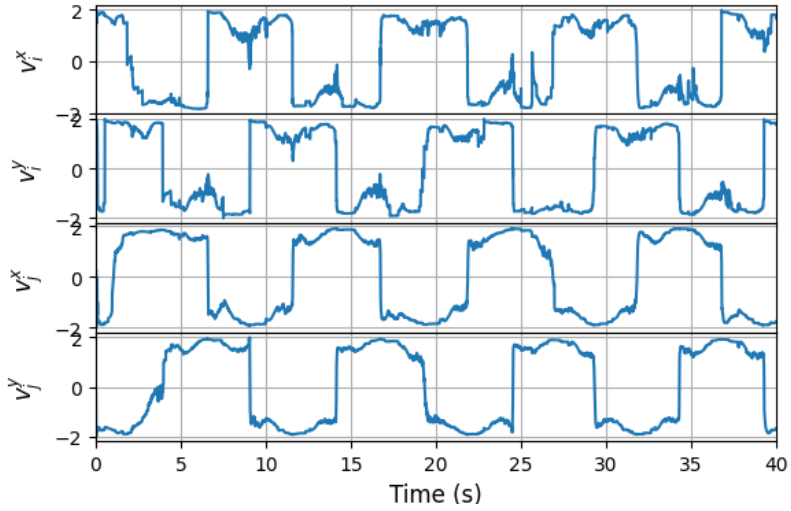


Figure 5.8: The control inputs generated by the proposed NMPC with observability optimization. These sequences show the velocity input values corresponding to the simulation in Fig. 5.3.

controllers in the 30 random tests. Obviously, the proposed NMPC has lower averaged position estimation errors compared to the stochastic controller. Therefore, the behaviours after convergence are still meaningful to the localization performance. To study it, the control input $\boldsymbol{u}$ for two MAVs is shown in Fig. 5.8. From which we can see that all 4-channel velocities are approximating the maximum value of 2m/s. The oscillations and

changes of velocity direction occur due to the state bounds and velocity limitation. The four velocities are assigned different phases equally of the periodic motion pattern. This asynchronous behaviour has not been considered before, but NMPC can generate it automatically.

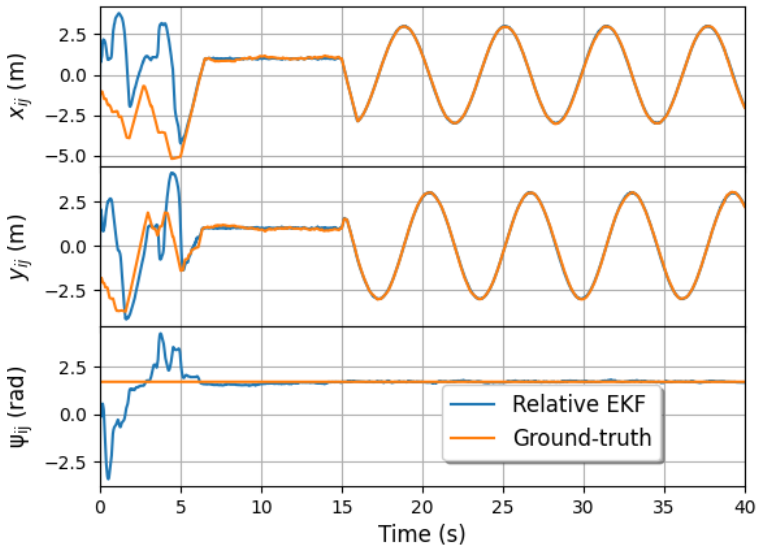


Figure 5.9: Relative localization and ground-truth between two MAVs under the proposed optimal control method and formation tracking multi-robot tasks. The target relative position is constant before $t = 15s$, and variant after $t = 15s$.

5.4.3. FORMATION CONTROL WITH NMPC

This subsection uses the NMPC controller for multi-robot tasks. Examples of formation flight and dynamic motion tracking are given below. At the beginning, a constant relative position is set in the task cost J_C, where $\widetilde{\boldsymbol{p}}_{\cdot|t} = [1,1]m$. The other settings of the solver remain unchanged. After 15s, a variant relative motion reference is introduced, which is defined as $\widetilde{\boldsymbol{p}}_{\cdot|t} = [2cos(t), 2sin(t)]m$. This leads to a circle motion of the second MAV around the first MAV.

The corresponding control results are shown in the following figures. In Fig. 5.9 we can see that the proposed NMPC has fast and stable tracking performance given the formation and dynamic tracking tasks at $t = 5$ and $t = 15s$ respectively. In addition, the observability cost keeps being optimized simultaneously by the NMPC.

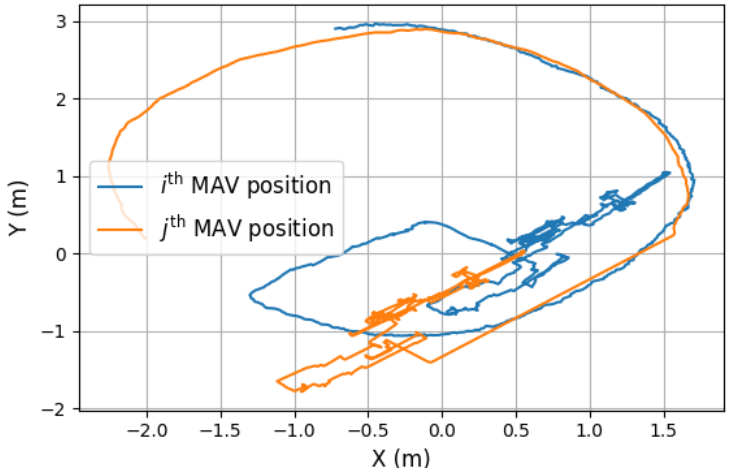


Figure 5.10: The world-frame trajectories of both MAVs under the proposed optimal control method and formation tracking multi-robot tasks. The time range of the data is between 10s and 20s.

To view the motion of each MAV in world-frame, the trajectories of both MAVs are plotted in Fig. 5.10. For the formation flight during 10-15s, both MAVs move slowly with constant relative positions. During 15-20s, both MAVs move to achieve the circle tracking and keep optimizing the observability according to the asynchronous behaviours.

5.5. CONCLUSIONS

This chapter proposes a novel nonlinear MPC controller with an observability cost to improve range-based multi-MAV relative localization. Simulation results demonstrate its faster localization convergence and lower estimation errors with respect to previously studied stochastic motion. Future work involves the implementation of this controller in real-world micro air vehicles for better localization and control.

REFERENCES

[1] 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).

[2] E. Soria, F. Schiano, and D. Floreano, *Predictive control of aerial swarms in cluttered environments,* Nature Machine Intelligence **3**, 545 (2021).

[3] 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** (2018).

[4] M. Hamer and R. D'Andrea, *Self-calibrating ultra-wideband network supporting multi-robot localization,* IEEE Access **6**, 22292 (2018).

[5] M. Basiri, F. Schill, P. Lima, and D. Floreano, *On-board relative bearing estimation for teams of drones using sound,* IEEE Robotics and Automation letters **1**, 820 (2016).

[6] 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).

[7] K. McGuire, C. De Wagter, K. Tuyls, H. 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** (2019).

[8] F. Schilling, F. Schiano, and D. Floreano, *Vision-based drone flocking in outdoor environments,* IEEE Robotics and Automation Letters **6**, 2954 (2021).

[9] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, *Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 8776–8782.

[10] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, *Drone detection using depth maps,* in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2018) pp. 1034–1037.

[11] 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* (IEEE, 2013) pp. 993–998.

[12] M. Gassner, T. Cieslewski, and D. Scaramuzza, *Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors,* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 5196–5202.

[13] 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).

[14] 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).

[15] 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).

[16] 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).

[17] Z. Han, K. Guo, L. Xie, and Z. Lin, *Integrated relative localization and leader–follower formation control,* IEEE Transactions on Automatic Control **64**, 20 (2018).

[18] R. Hermann and A. Krener, *Nonlinear controllability and observability,* IEEE Transactions on automatic control **22**, 728 (1977).

[19] J. Köhler, F. Allgöwer, and M. A. Müller, *A simple framework for nonlinear robust output-feedback mpc,* in *2019 18th European Control Conference (ECC)* (IEEE, 2019) pp. 793–798.

[20] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, *acados—a modular open-source framework for fast embedded optimal control,* Mathematical Programming Computation , 1 (2021).

[21] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, *Casadi: a software framework for nonlinear optimization and optimal control,* Mathematical Programming Computation **11**, 1 (2019).

6

CONCLUSION

6.1. ANSWERS TO RESEARCH QUESTIONS

The previous four chapters demonstrate the theoretical and technological solutions to the four research questions listed in the introduction. The summarized answers are presented below for each question.

Research Question 1

Given a physical model and a filter, how can an individual robot learn its filter parameters automatically in an unsupervised manner?

Inspired by human perception, the state estimation of robots can be optimized by minimizing the discrepancy between the predicted and measured sensor data. Chapter 2 translates this concept into a nonlinear target function, solved by the gradient-based descent method to find the minimum. This tuning process does not require any groundtruth states. Instead, the raw sensor data from a short manual flight with variation in flight maneuvers are already sufficient for the tuning. After optimization iterations, the tuned filter parameters ensure precise attitude and position estimation with the EKF on Crazyflie. This unsupervised tuning method is also effective for UKF parameter identification for attitude estimation based on the Euroc MAV dataset. Experiments show that an individual robot can learn its filter parameters for estimating attitude and position in an unsupervised manner.

Research Question 2

With limited computation resources on pocket drones, how can we design fully autonomous and efficient relative localization technology that allows for multi-robot tasks?

For multiple tiny flying robots, Chapter 3 presents an efficient system design for relative localization. The developed technology begins with low-level but high-speed ranging

communication among multiple Crazyflie quadrotors, which guarantees the update frequency of the estimation without relying on external systems. To demonstrate the system's ability for multi-robot tasks, we show the pattern formation flight of five Crazyflie quadrotors based on the relative localization. With the proposed method, velocity-based relative localization and distributed control are computationally efficient and can run onboard a microprocessor. In addition, we design the monocular visual navigation for a leader drone getting through a window with other "blind" robots following it through in a cooperative task. Overall, the swarm system has higher localization accuracy and efficiency compared to other state-of-the-art systems.

Research Question 3

How can we design a deep learning setup for 3D multi-robot relative localization without manual labeling?

Most related studies borrow the object detection network from computer vision to recognize the bounding boxes of other quadrotors. Instead of detecting the bounding box, our method in Chapter 4 designs a deep neural network with the following characteristics:

- Predict the center pixel position and its corresponding depth of other robots from a monocular image, and
- Leverage the developed UWB range-based relative localization for providing the position annotation instead of manually labeling.

These novel characteristics allow the neural networks to learn different and new drones, provided that they also possess the UWB ranging capability. In other words, the whole system provides an approximate self-supervised manner for multi-robot detection and localization, without relying on manual labelling and prior known drone size. In addition, this network is efficient and can run on a swarm of pocket drones, based on a tiny AI chip.

Research Question 4

How can we design an optimal control scheme that maximizes multi-robot observability, to achieve a faster localization convergence?

Given Research Question 2, we accomplished a fully autonomous aerial swarm but without optimizing the observability. As a consequence, the initialization procedure takes up a considerable amount of the flight time, and in principle the system can become unobservable again during execution of the task. To tackle this issue, Chapter 5 casts the observability optimization into an optimal control problem. By maximizing the determinant value of the observability matrix, the optimal controller gives future velocity commands for future N steps of all robots. The first velocity commands guarantee the future observability of the multi-MAV system. This optimal control leads to particular multi-robot behaviors (including both relative positions and velocities). Experiments show that the localization convergence is quicker and has lower localization errors than random control inputs.

6.2. CONCLUSIONS

This thesis achieves the research goal of designing a fully autonomous swarm of tiny flying robots by considering the four levels of swarm autonomy: individual estimation, relative localization, distributed control, and swarm navigation. The final conclusions are drawn based on the approaches and results of this thesis.

Research Goal

To develop a fully autonomous swarm of flying robots that can operate without any external infrastructure, which can be used for various multi-robot tasks.

On the autonomy of individual estimation

- Each individual robot must be able to estimate its attitude and position with fully onboard sensors.
- The state estimation parameters can be tuned autonomously with an unsupervised policy, without relying on groundtruth.

On the autonomy of relative localization

- As an important component of aerial swarms, the relative positions between MAVs can be estimated based on either UWB ranging sensors or visual sensors.
- Ranging communication with robustness to package loss and high frequency, has a positive effect on relative localization accuracy.
- The distributed ranging communication needs further developments for increasing the swarm scalability.
- A deep neural network can be trained in a self-supervised manner to localize other robots autonomously, in the absence of prior-known masks and robot size.

On the autonomy of distributed control

- Distributed control only considers the local relative positions, which guarantees the swarm scalability.
- Direct velocity control is effective and efficient for the low-cost microprocessor with limited computation power.

On the autonomy of swarm navigation

- Instead of implementing SLAM, a simple visual navigation on the leader makes the swarm of drones fly in a coordinated fashion, indicating the proposed aerial swarm is capable of different multi-robot tasks.

Overall, all four-level autonomy aspects are designed in the most efficient manner and are integrated fully onboard a tiny microprocessor. This was very challenging due to the limited memory and computation power of the processor that tiny drones can carry. With thorough system design, we have been able to show the first autonomous swarm of tiny flying robots with accurate relative localization and high-speed ranging capability, accomplishing the main research objective of this thesis.

6.3. APPLICATION OF THE DEVELOPED METHODS

The developed autonomous aerial swarm system has great application value as it provides necessary states for general multi-robot purposes. In addition, since the code has been made available open-source and the used hardware is commercial off-the-shelf, others begin to utilize the range-based aerial swarm functionality for their own projects. As an example application, our group has exploited this swarm system for a cooperative gas-seeking project in a GPS-denied environment [1].

Figure 6.1: Example application of the proposed tiny aerial swarms for a gas-seeking task [1]. It proposes a bug algorithm for an autonomous swarm of quadrotors, to localize a gas source in cluttered environments.

As shown in Figure 6.1, multiple tiny aerial robots are designed to localize the gas source cooperatively. This is a complex task as the aerial swarm should be able to use only onboard sensing and computation to achieve fully autonomous waypoint tracking, obstacle avoidance, relative localization, communication and gas sensing. More specifically, the technical solution is to have each robot follow the waypoints generated by the particle swarm optimization (PSO). But PSO achieves the collision avoidance and collaborative gas seeking based on accurate relative positions and communicating observed gas concentrations, which builds on the work of Chapter 3 in this thesis. Without these fundamental tools, the PSO can never be accomplished for this collaborative task. Therefore, our proposed aerial swarms are proved to be stable, reliable, and easy-to-use. This application also indicates the achievement of the research goal, i.e., a good tool for testing novel swarm theories in GPS-denied environments.

6.4. FUTURE WORK

This thesis solves the most fundamental challenge of designing a stable and autonomous swarm of tiny flying robots. However, there are still some limitations of the proposed technology for various real-world applications. For example, the bandwidth limitations for inter-robot ranging, the short flight time due to the limited battery capacity, the lack of recovery solutions for accidental crashes, etc.

In addition, there are several aspects that could make the aerial swarm even more intelligent and stable. For example, the swarm of flying robots possesses the autonomous decision-making ability that can perform tasks more efficiently than a pre-defined task

setting. Reinforcement learning or evolutionary learning fit well with this concept. For low-level specifications, an individual robot needs to perceive the environmental information for obstacle avoidance and visual navigation to the task points. Overall, a mature product of aerial swarm system should possess the following features:

- Scalable and accurate mutual relative localization,
- Environmental perception for obstacle avoidance,
- Navigation in order to reach locations important to the task, and typically in order to get back to a base / recharging station, and
- Automatic decision making or task assignments to adapt the behavior flexibly depending on the task and environment.

The most promising theoretical and technical solution for an intelligent tiny aerial swarm may rely on a comprehensive deep neural network that can achieve all the above features simultaneously. Specifically, current AI has limited computation on tiny pocket drones. Integration of relative localization, obstacle avoidance, and visual navigation into the tiny drone with a customized deep visual solution, may move the swarm robotics from the lab to the real world.

REFERENCES

[1] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. H. E. de Croon, *Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments,* accepted by IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021).

A

APPENDICES: COORDINATE DEFINITION AND 3D RELATIVE LOCALIZATION

A.1. COORDINATE DEFINITION

In Chapter 3, there are two different rotational matrices. One is used for the movement transformation from the body frame to the earth-fixed frame. Another matrix describes the relation of angular velocity between the body frame and the earth-fixed frame.

To derive the rotation matrices, a right-hand coordinate is defined. Compared to the Crazyflie coordinate as defined in Figure 2.1, this coordinate only has an inverse pitch value such that it is right-hand. The rest remains the same.

In (3.2), the rotation matrix is derived by rotating the earth-fixed coordinate in sequences of Z-X-Y. First, the rotation around Z-axis is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c(\psi) & s(\psi) \\ -s(\psi) & c(\psi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c(\psi) & -s(\psi) \\ s(\psi) & c(\psi) \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}. \tag{A.1}$$

Similarly, all three rotation matrices are given as follows.

$$\mathrm{R}_z = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathrm{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix}, \mathrm{R}_y = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix}. \tag{A.2}$$

Therefore, the rotation from earth-fixed frame to body frame is denoted by

$$\begin{bmatrix} x''' & y''' & z''' \end{bmatrix}^T = \mathrm{R}_y^{-1}\mathrm{R}_x^{-1}\mathrm{R}_z^{-1} \begin{bmatrix} x & y & z \end{bmatrix}^T. \tag{A.3}$$

Obviously, $[x\ y\ z]^T = \mathrm{R}_z\mathrm{R}_x\mathrm{R}_y[x'''\ y'''\ z''']^T$. Thus, the full rotation matrix from body frame

A

to earth-fixed frame is

$$\mathrm{R} = \mathrm{R}_z\mathrm{R}_x\mathrm{R}_y = \begin{bmatrix} c(\theta)c(\psi) - s(\phi)s(\theta)s(\psi) & -c(\phi)s(\psi) & s(\theta)c(\psi) + s(\phi)c(\theta)s(\psi) \\ c(\theta)s(\psi) + s(\phi)s(\theta)c(\psi) & c(\phi)c(\psi) & s(\theta)s(\psi) - s(\phi)c(\theta)c(\psi) \\ -c(\phi)s(\theta) & s(\phi) & c(\phi)c(\theta) \end{bmatrix}. \quad \text{(A.4)}$$

The following is the derivation of the rotation matrix that can transform the body-frame angular velocity to earth-frame angular velocity. Angular velocity $\boldsymbol{\omega}_b$ in body frame is equal to the gyroscope measurement. Euler angle rate $\dot{\boldsymbol{\eta}} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ is relative to the axes fixed to the earth. The derivation of the transformation between $\boldsymbol{\omega}_b$ and $\dot{\boldsymbol{\eta}}$ can be found in `http://www.euclideanspace.com/physics/kinematics/angularvelocity/`. In this section, we give the rotation derivation under the given rotation sequences of Z-X-Y. Note that X, Y and Z represent the three axes, where X' means one rotation around X, and X'' means double rotation around X. Apparently, the body-frame angular velocity satisfies

$$\boldsymbol{\omega}_b = \dot{\psi} Z + \dot{\phi} X' + \dot{\theta} Y''. \quad \text{(A.5)}$$

Furthermore, we have the following relation between all rotated axes.

$$[X, Y, Z]^T = \mathrm{R}_z [X', Y', Z']^T \quad \text{(A.6)}$$

$$[X', Y', Z']^T = \mathrm{R}_x [X'', Y'', Z'']^T \quad \text{(A.7)}$$

$$[X'', Y'', Z'']^T = \mathrm{R}_y [X''', Y''', Z''']^T. \quad \text{(A.8)}$$

Therefore, we can get the following equations:

$$Z = -c(\phi)s(\theta)X''' + s(\phi)Y''' + c(\phi)c(\theta)Z''' \quad \text{(A.9)}$$

$$X' = X'' = c(\theta)X''' + s(\theta)Z''' \quad \text{(A.10)}$$

$$Y'' = Y'''. \quad \text{(A.11)}$$

In view of (A.5), we can get the angular velocity rotation matrix by clustering the elements according to $\boldsymbol{\omega}_b = [\cdots X''', \cdots Y''', \cdots Z''']^T$, which is shown below.

$$\boldsymbol{\omega}_b = \begin{bmatrix} c(\theta) & 0 & -c(\phi)s(\theta) \\ 0 & 1 & s(\phi) \\ s(\theta) & 0 & c(\phi)c(\theta) \end{bmatrix} \dot{\boldsymbol{\eta}} \Leftrightarrow \dot{\boldsymbol{\eta}} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\theta)t(\phi) & 1 & -c(\theta)t(\phi) \\ -s(\theta)/c(\phi) & 0 & c(\theta)/c(\phi) \end{bmatrix} \boldsymbol{\omega}_b. \quad \text{(A.12)}$$

A.2. THREE-DIMENSIONAL RELATIVE LOCALIZATION

The 2D relative localization in Chapter 3 is extended for 3D scenarios, which means the absolute height information is unknown for all robots. Instead, the vertical velocity will be added to the communication package of the UWB. This 3D relative localization has been validated experimentally on multiple Crazyflies and flapping-wing robots. Furthermore, the formation flight of multiple flapping-wing robots has been achieved based on the 3D relative localization.

The 3D state is $\boldsymbol{X}_{ij} = [x_{ij}, y_{ij}, z_{ij}, \psi_{ij}]^T$, and the input is $\boldsymbol{U}_{ij} = [v_i^x, v_i^y, v_i^z, r_i, v_j^x, v_j^y, v_j^z, r_j]^T$. The relative motion model is augmented by the vertical movement as $z_{ij} = v_j^z - v_i^z$. There-

fore, the state and input Jacobian matrices are changed into

$$\begin{aligned}
\boldsymbol{A} &= \frac{\partial F}{\partial \boldsymbol{X}} = \begin{bmatrix} 1 & r_i\Delta t & 0 & (-s(\psi_{ij})v_j^x - c(\psi_{ij})v_j^y)\Delta t \\ -r_i\Delta t & 1 & 0 & (c(\psi_{ij})v_j^x - s(\psi_{ij})v_j^y)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\boldsymbol{B} &= \frac{\partial F}{\partial \boldsymbol{U}} = \begin{bmatrix} -1 & 0 & 0 & y_{ij} & c(\psi_{ij}) & -s(\psi_{ij}) & 0 & 0 \\ 0 & -1 & 0 & -x_{ij} & s(\psi_{ij}) & c(\psi_{ij}) & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{A.13}$$

The observation function is

$$d = h(\boldsymbol{X}_{ij}) = \sqrt{\boldsymbol{p}_{ij}^T \boldsymbol{p}_{ij}} = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2}. \tag{A.14}$$

Thus, the corresponding Jacobian matrix is

$$\boldsymbol{H} = \frac{\partial h}{\partial \boldsymbol{X}} = [x_{ij}/d, y_{ij}/d, z_{ij}/d, 0]. \tag{A.15}$$

The EKF formula for 3D relative localization remains the same as that in Chapter 3. The implementation code of the 3D EKF and control on two Crazyflies can be found in `https://github.com/shushuai3/cf_onboard_swarm/tree/swarm3d`.

ACKNOWLEDGEMENTS

This dissertation is the result of my four-year research in Delft, where I have spent a wonderful time of my life. I would never forget the beautiful scenes and kind people in this small cozy town. At the end of my PhD career, I would like to thank all the people for their kind help to me in all aspects in recent years.

First, I would like to thank my promoters, Guido de Croon and Max Mulder. As also my daily supervisor, Guido not only guides me on solving academic problems but also encourages me to think independently. He taught me how to extract practical problems from experiments, and how to address them smartly. I have been always impressed by his endless passion for learning and open-mind thinking. Besides, his humor makes the research discussion and lunchtime very interesting and relaxing. I appreciate his detailed feedbacks on each paper revision and his consistent stimulation of my research. Max Mulder, thank you for giving me the chance to pursue my PhD at the C&S department. You are a very kind and courageous department leader. Your comments on my go/nogo meeting kept me working harder and deeper on the scientific problem. I admire your enthusiastic and efficient working styles. Besides, thanks for your thorough comments on this thesis, and the revisions on the propositions.

I am very proud to work in the MAVLab for four years, where we had many interesting activities like the Heidag, Drone Clash, and various demonstrations. Besides, this lab makes me feel a strong team spirit, that people dare to attempt any challenges with drones. For that, I would like to thank the MAVLab group members: Diana Olejnik, Tom van Dijk, Yingfu Xu, Federico Paredes Vallés, Sven Pfeiffer, Sunyou Hwang, Ziqing Ma, Jesse Hagenaars, Stein Stroobants, Alessandro Mancinelli, Sunyi Wang, Hang Yu, Yilun Wu, Mario Coppola, Shuo Li, Kimberly McGuire, Ewoud Smeur, Matěj Karásek, Sjoerd Tijmons, Kirk Scheper, Nilay Sheth, Julien Dupeyroux, Salua Hamaza, Erik van der Horst, Christophe De Wagter, Kevin van Hecke, Bart Remes, Freek van Tienen. My special thanks to Christophe for preparing different experimental hardware, to Eric for his help on my first Ubuntu installation and fixing the drones, to Mario for helping me with the paper writing and video editing, and to Kimberly for her consistent help on Crazyflie's problem.

Thanks to the people involved in my supervision and teaching activities. Daniel Willemsen and Matteo Barbera, thanks for cooperating as the teaching assistants for the MAV course. Qichen Deng, thanks for co-coaching the bachelor Design Synthesis Exercise. Besides, I appreciate that the master students I supervised have achieved significant research results through their hard work. Therefore, I would like to thank Stein Stroobants, Bart Duisterhof, Raoul Mink, and Chris Groen. We have been fighting together to solve the challenging problems.

A

In addition, I want to thank all the PhDs at the C&S department, for our leisure time such as coffee break, lunch time, BBQs, PhD drinks, PhD events, etc. We had lots of fun, and we also encouraged each other by proposing potential solutions for other's problem even if it is not in your area :). Hereby, my thanks go to: Dirk van Baelen, Jaime Junell, Malik Doole, Noor Nabi, Sihao Sun, Daniel Friesen, Jelmer Reitsma, Bo Sun, Jerom Maas, Isabel Metz, Sarah Barendswaard, Paolo Scaramuzzino, Dyah Jatiningrum, Tommaso Mannucci, Sophie Armanini, Annemarie Landman, Junzi Sun, Kasper van der El, Ye Zhang, Yingzhi Huang, Ying Yu, Marta Ribeiro, Emmanuel Sunil, Wei Fu, Ye Zhou, Julia Rudnyk, Xuerui Wang, Cheng Liu, Rowenna Wijlens, Gijs de Rooij, Yifei Li, Yiyuan Zou, Tiago Monteiro Nunes. And thank you, Bertine Markus, for your help on the administration stuff.

I wish to thank my friends outside the C&S department. We all witnessed the big snows in Delft and had many precious memories of the delicious food. They are: Hai Zhu, Jianliang Lin, Xiaoyan He, Zhen Cui, Tao Sun, Yidong Gan, Zhe Li, Kailun Yang, Yading Xu, Chenguang Wang, Hao Zhang.

Finally, I wish to express my deepest gratitude to my family. My parents, who raised me up and are always there for me. My two older brothers, who gave me a happy childhood and always care about my life and study. I sincerely thank my wife, Zhou, we have been together for 8 years. Thank you so much for your support and encouragement.

CURRICULUM VITÆ

Shushuai LI

09-05-1991 Born in Handan, China.

EDUCATION

2007–2010 Handan No.1 High School

2010–2014 B.Sc. in Automation
Hunan University

2014–2017 M.Sc. in Control Science and Engineering
Hunan University

Since 2017 PhD candidate in Swarm Robotics
Delft University of Technology

AWARDS

2016 National Scholarship

2017 Chinese Scholarship Council (CSC) Scholarship

2017 Distinguished Master Student

COACHING

2018 Design Synthesis Exercise

2019 Minor in Robotics

2019, 2020 Autonomous Flight of Micro Air Vehicles

LIST OF PUBLICATIONS

6. **Shushuai Li**, Christophe De Wagter and Guido C.H.E. de Croon, *Nonlinear model predictive control for improving range-based relative localization by maximizing observability*, Accepted by the International Micro Air Vehicle Conference and Competition (IMAV), 2021.

5. **Shushuai Li**, Christophe De Wagter, Guido C.H.E. de Croon, *Self-supervised monocular multi-robot relative localization with efficient deep neural networks*, Submitted to IEEE International Conference on Robotics and Automation, 2021.

4. **Shushuai Li**, Mario Coppola, Christophe De Wagter, Guido C.H.E. de Croon, *An autonomous swarm of micro flying robots with range-based relative localization*, Under revision, submitted to IEEE Transaction on Robotics, 2020.

3. **Shushuai Li**, Christophe De Wagter, Guido C.H.E. de Croon, *Unsupervised tuning of filter parameters without ground-truth applied to aerial robots*, IEEE Robotics and Automation Letters **4**, 4, 2019.

2. Bardienus P. Duisterhof, **Shushuai Li**, Javier Burgués, Vijay Janapa Reddi, Guido C.H.E. de Croon, *Sniffy Bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments*, Accepted by IEEE/RSJ International Conference on Intelligent Robots and Systems, 2021.

1. Veronica Munaro, Sven Pfeiffer, **Shushuai Li**, Alessandro Rizzo, Guido C.H.E. de Croon, *Three-dimensional relative localization and swarming of flapping-wing robots*, In preparation for Swarm Intelligence, 2021.