Titre: Title:	Accurate Localization with Ultra-Wideband Ranging for Multi-Robot Systems
Auteur: Author:	Yanjun Cao
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Cao, Y. (2020). Accurate Localization with Ultra-Wideband Ranging for Multi-Robot Systems [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5392/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Accurate Localization with Ultra-Wideband Ranging for Multi-Robot Systems

YANJUN CAO
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Août 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Accurate Localization with Ultra-Wideband Ranging for Multi-Robot Systems

présentée par **Yanjun CAO**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Giuliano ANTONIOL, président
Giovanni BELTRAME, membre et directeur de recherche
Guchuan ZHU, membre
Zeljko ZILIC, membre externe

DEDICATION

To all friends at the lab,
I will miss you...

ACKNOWLEDGEMENTS

The journey that leads to this point wouldn't have been possible without the support of many people and the grand-awarding organizations.

I am very grateful to Prof. Giovanni Beltrame, who opened the door to many valuable opportunities. I still remember the excitement when I received the admission from MISTLab. The following four years' experiences proved that coming here was one of the best choices of my life. It is my great honor to work with Giovanni, and it is a truly enjoyable journey. I learned a lot from his knowledge, thoroughness, and his philosophy of life. I would also like to express my particular thanks for his great help and support for my internships in Germany. The joyful experiences that happened in Tübingen with Giovanni and his family are unmemorable. In addition, Tübingen is such a lucky place that I met my love there.

I want to thank the members of MISTLab: Vivek, Chao, Guannan, Meng, Ivan, Jacopo, Hassan, and other alumni and current members. Thank Vivek for lots of help in various aspects as we started our Ph.D. journey at the same time. Thank Chao for getting me started in Montreal. Thank Guannan for his help in life as an old friend. Thank Meng for his guidance of my first publication in the doctoral stage. Thanks to Ivan for his generous help in my English and his humors. Thanks to Jacopo and Hassan for numerous discussions on various aspects. Thanks to Benjamin and Fang for their help in the proof of this dissertation. Thanks to all members of MISTLab for making our laboratory full of happiness and joy. I am also grateful to Prof. Zell and Prof. Knoll for hosting me in their labs in Germany. I want to thank my friends in Tübingen and Munich, including Chen Hao, Yapeng, Yuan, Hamd, and Rui. These special experiences would not have happened without the funding from NSERC, ReSMiQ, and FRQNT. These scholarships are my honors, and I am very grateful.

In addition, I wish to thank my examination committee: Prof. Giulio Antoniol, Prof. Guchuan Zhu, Prof. Zeljko Zilic, and Prof. Giovanni Beltrame. I know this dissertation and my graduation process have taken them a lot of time and efforts. Thank them for their contributions to the last stage of my doctoral life.

Finally, I am extremely thankful to my father and my sister. Their unconditional support has been with me throughout my Ph.D. studies. I would especially thank Min, who allows my isolated life for six months without feeling lonely in this special COVID-19 situation. I look forward to our wonderful future.

RÉSUMÉ

Avec l'avancement de la technologie matérielle et logicielle, l'application de l'automatisation et de la robotique se développe rapidement. Les systèmes multi-robots sont particulièrement prometteurs en raison de leur grande efficacité et robustesse. De tels systèmes peuvent être utilisés pour aider les humains à effectuer efficacement des tâches dangereuses ou pénibles, telles que l'intervention en cas de catastrophe, l'exploration souterraine, etc. Pour déployer un système multi-robot dans un environnement sans GPS, la coordination des robots dans le système est un défi crucial. Chaque robot doit avoir une estimation précise de sa propre position pour permettre aux robots du système de collaborer pour la réalisation de leur tâche. Comme cette direction de recherche est relativement nouvelle, les approches existantes ne sont pas encore abouties. Elles consistent principalement en des systèmes centralisés qui reposent sur des signaux GPS. La dépendance sur un signal GPS limite l'application aux espaces extérieurs ouverts. De plus, les systèmes centralisés sont confrontés au risque d'un point de défaillance unique, qui limite la robustesse du système. Par ailleurs, un système centralisé n'est pas toujours approprié à une taille grandissante, comme lors d'ajout de nouveaux groupes de robots ou lors de la fusion de différents groupes. Par conséquent, une solution distribuée, décentralisée, et adaptée à de larges groupes de tailles variables pouvant produire une estimation et un suivi du positionnement des robots dans un environnement sans GPS est souhaitée. Dans ce travail, nous adoptons une stratégie descendante pour relever ces défis.

La première partie de la recherche est la stratégie au niveau macroscopique. Un système de localisation distribué dynamique est proposé pour créer et maintenir un système de coordonnées unifié. Chaque robot du système est équipé d'un capteur de télémétrie qui mesure la distance avec ses voisins. Pour le système de localisation bidimensionnel, tous les robots sélectionnent trois robots en soumissionnant pour construire un système de coordonnées. Ensuite, chaque robot obtient sa position grâce à l'initialisation. Le système de coordonnées est ajusté dynamiquement en fonction de la position des robots. Nous utilisons un filtre de Kalman étendu (EKF) pour estimer de manière itérative la position de chaque robot. Nous proposons également une fonction de reconfiguration du système de coordonnées pour empêcher l'accumulation d'erreurs du suivi à long terme.

Les capteurs choisis utilisent la technologie Ultra Large Bande (ULB) en raison de la précision des mesures produites. Cependant, en raison du temps relativement long requis pour la mesure de télémétrie, le contrôle d'accès du support ULB partagé dans notre scénario fait face à certains défis: une télémétrie haute fréquence est nécessaire entre les robots pour

améliorer la précision du système; un ajustement rapide de la planification est nécessaire pour s'adapter aux changements de topologie du réseau; les critères de distribution doivent être suivis pour éviter un point de défaillance unique. Nous proposons un nouvel algorithme d'accès multiple par répartition dans le temps (TDMA), qui permet de planifier rapidement l'utilisation des médias ULB par un grand nombre de périphériques réseau sans conflits dans le voisinage du réseau local et d'éviter les conflits avec les terminaux cachés. De plus, il maximise l'utilisation des canaux du réseau ULB. Le système n'a pas de nœuds spéciaux (tous les nœuds sont les mêmes) et il est adaptatif pour prendre en charge la localisation de systèmes distribués.

Lorsque le système multi-robot dispose d'un système de coordonnées unifié et de mesures de distance fiables, chaque robot estime sa position dans le système. Dans la localisation collaborative, puisque chaque robot partage sa position avec ses voisins, une estimation précise de l'odométrie peut grandement améliorer la précision de localisation du système. Nous étudions d'abord les performances de localisation et de suivi dans les paramètres minimaux, comprenant une seule source de télémétrie ULB et un 9 unités de mesure inertielle DoF (IMU) à faible coût. Nous proposons un estimateur de vitesse qui utilise la mesure de distance UWB et dérive la vitesse du robot à partir des variations du signal. En combinant l'estimation de la vitesse avec l'estimation de l'orientation du capteur IMU, la position et l'orientation du robot peuvent être observée à l'aide de l'EKF. Cette solution permet également le suivi des appareils à faible coût avec uniquement IMU et signaux ULB, tels que les appareils IoT.

Pour les robots équipés de caméras monoculaires supplémentaires, une méthode exploitant l'odométrie visuelle-inertielle (VIO) est proposée pour un suivi précis. Les progrès récents de l'odométrie visuelle-inertielle peuvent fournir une estimation précise des changements de position du robot pendant une courte période. Cependant, sur de longues trajectoires, l'estimation peut dévier de manière importante, en particulier dans des environnements visuellement difficiles. En utilisant les capteurs ULB existants dans un système multi-robot, nous proposons une optimisation conjointe des mesures de vision, d'inertie et de télémétrie. Cette méthode utilise une seule ancre placée au hasard dans l'environnement et adopte une approche de fenêtre coulissante à double couche, qui utilise efficacement les mesures de distance. Cette méthode peut corriger efficacement l'erreur accumulée à chaque réception d'une mesure de l'ancre. Nous utilisons également cette configuration pour la localisation et la cartographie collaboratives: différents robots utilisent la télémétrie mutuelle (si disponible) et l'ancre commune pour estimer la transformation entre eux, permettant ainsi la fusion de cartes.

En abordant la localisation d'un système multi-robot à la fois au niveau macroscopique et

au niveau des robots individuels, cette recherche permet potentiellement à un grand groupe de robots d'avoir une autonomie à long terme dans un environnement où aucun signal GPS n'est disponible.

ABSTRACT

With the advancement of hardware and software technology, the everyday applications of automation and robotics are developing rapidly. Multi-robot systems are particularly promising because of their high efficiency and robustness. Such systems can be used to assist humans in performing dangerous or strenuous tasks, such as disaster response, subterranean exploration, etc. To deploy a multi-robot system in an environment without a global positioning system (GPS), coordinating the robots in the system is a crucial challenge. Each robot needs to have the correct tracking of its own and its teammates positions to enable the robots to cooperate. Because this research direction is relatively new, there are not many mature methods: existing approaches are mainly centralized systems that rely on GPS signals. The dependence on GPS restricts the application to the outdoors or indoor spaces with expensive infrastructure. Centralized systems also face the risk of a single point of failure, which is not acceptable for critical systems. In addition, centralized systems can be hard to scale both statically and dynamically (e.g. adding new groups of robots or merging different groups). Therefore, a distributed and scalable solution with accurate positioning and tracking in a GPS-denied environment is desired. In this work, we follow a top-down strategy to address these challenges.

The first part of this research is a strategy at the macroscopic level. We propose a dynamic distributed localization system to create and maintain a unified coordinate system assuming each robot is equipped with a sensor that can measures the distance of other robots in the system. For 2D localization, the system collectively select three robots, through a bidding mechanism, which construct a common coordinate system and we use an extended Kalman filter (EKF) to iteratively estimate the position of each robot. We also proposes a coordinate system reconfiguration function to prevent error accumulation from long-term tracking.

Ultra-WideBand (UWB) is a technology that can provide accurate ranging capabilities, and we selected it as our ranging sensor in the proposed system. However, due to the relatively long time required for obtaining UWB measurements, the access control of the shared UWB medium in our scenario faces some special requirements: high-frequency ranging between robots to improve system accuracy, rapid adjustment to changes in network topology, and a distributed architecture to avoid single points of failure. We propose a novel Time Division Multiple Access (TDMA) algorithm, which can quickly schedule the use of UWB media on a large number of network devices without conflicts in the local network neighborhood and avoid conflicts with hidden terminals, all while maximizing the channel usage of the UWB

network. Our system has no special nodes (all nodes are equal), and it is scalable to support distributed system localization.

Once a multi-robot system has a unified coordinate system and reliable ranging measurements, each robot can track its position. In collaborative localization, since each robot shares its position with its neighbors, accurate odometry estimation can greatly improve the overall localization accuracy of the system. We first study the localization and tracking performance using minimal infrastructure, minimally using a single UWB anchor and a low-cost 9 DoF Inertial Measurement Unit (IMU). We propose a speed estimator which continuously monitors the distance with the UWB anchor and derives the speed of the robot from its change patterns. By combining the speed estimation with the orientation estimation from the IMU sensor, the pose of the robot can be observed using an EKF. This solution also enables the tracking of low-cost devices with only IMU and UWB, such as IoT devices.

For robots that are equipped with an additional monocular camera, we propose a method fusing visual, inertial, and ranging measurements for accurate tracking. Recent advances in visual inertial odometry (VIO) can provide accurate odometry estimation for short periods. However, over long trajectories, drift can be significant, especially in environments that are visually challenging. Using the existing UWB sensors in the multi-robot system, we propose a joint optimization for visual, inertial, and ranging measurements. This method uses a single anchor randomly placed in the environment and adopts a double layer sliding window scheme, which uses distance measurements efficiently. This method can effectively correct the accumulated error whenever the anchor is visible. We also use this setup for collaborative simultaneous localization and mapping (SLAM): different robots use mutual ranging (when available) and the common anchor to estimate the reciprocal map transformation, thereby simplifying map fusion.

By addressing the localization for a multi-robot system both at the team level and for individual robots, this research potentially allows large groups of robots to have long-term autonomy in a GPS-denied environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ATE	Absolute Trajectory Error
CDMA	Code Division Multiple Access
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA	Carrier Sense Multiple Access
DOF	Degree of Freedom
EKF	Extended Kalman Filter
EM	Electromagnetic
FDMA	Frequency Division Multiple Access
GPS	Global Positioning System
IMU	Inertial Measurement Unites
IoT	Internet of Things
LiDAR	Light Detection And Ranging
LoRa	Long Range
LOS	Line of Sight
MAC	Medium Access Control
MAP	Maximum A Posterior
MEMS	Micro Electro Mechanical System
MWSN	Mobile Wireless Sensor Network
NLOS	Non-Line of Sight
RFID	Radio-Frequency identification
RMSE	Root Mean Square Error
RSSI	Received Signal Strength Indicator
SLAM	Simultaneous Localization and Mapping
TDMA	Time Division Multiple Access
TDOA	Time Difference of Arrival
TOA	Time of Arrival
ToF	Time of Flight
TWR	Two Way Ranging
UWB	Ultra-WideBand
VANET	Vehicular Ad Hoc Network
VIO	Visual Inertial Odometry
VIR-SLAM	Visual Inertial Ranging SLAM
VS	Virtual Stigmergy

VST	Virtual Stigmergy Table
WSN	Wireless Sensor Network

CHAPTER 1 INTRODUCTION

This dissertation includes the research work pursued to fulfill the requirements of the degree of Philosophiae Doctorate in computer engineering at Polytechnique Montréal, Université de Montréal. It recollects the research work conducted within the MIST laboratory of Polytechnique Montréal (Montreal, Canada), the Cognitive Systems Laboratory of the University of Tübingen (Tübingen, Germany), and the Robotics, Artificial Intelligence and Embedded System laboratory of The Technical University of Munich (Munich, Germany) from August 2016 to August 2020. This document follows the structure of a dissertation by articles — covering 4 published and submitted papers presented as separate chapters.

1.1 Context and Motivation

Localization via the Global Positioning System (GPS) has become an indispensable part of people's daily life. The easy access to global location information allows ordinary people to explore unknown places. As moving agents, robots also need to know their location to perform their tasks. For robot systems used to help humans perform tasks in areas that are dangerous and with limited access (such as disaster response or subterranean exploration), GPS cannot always be taken for granted. With the advancement of hardware and software technology, the application of automation and robotics to daily activities is developing rapidly, with multi-robot systems being particularly promising. The inherent advantages of "swarms" of robots in terms of robustness, flexibility, and scalability have attracted special attention. However, one of the great challenges of robotics is to coordinate a large number of robots in a distributed manner in a GPS-denied environment [1]. On top of this, a scalable system with few or no dependencies on fixed infrastructure is highly desirable.

Imagine deploying multiple robots in an unknown area to automate some tasks, such as searching for targets or establishing network infrastructure. To be efficient, robots need to work collaboratively and avoid duplication of work. If there is a central node as a supervisor that knows the state of all robots, it can optimally control operations. However, it is not trivial to realize a central node in a large-scale multi-robot system since the capacity of a single robot is always limited and a central node cannot obtain information from all nodes without delay. In addition, a centralized system may suffer from a single point of failure and cause the system crash. Therefore, a decentralized and distributed system is preferred, where any robot makes decisions based on its local information. Coordination in distributed systems faces the challenge of how to enable each robot to accurately locate itself in a common

coordinate system.

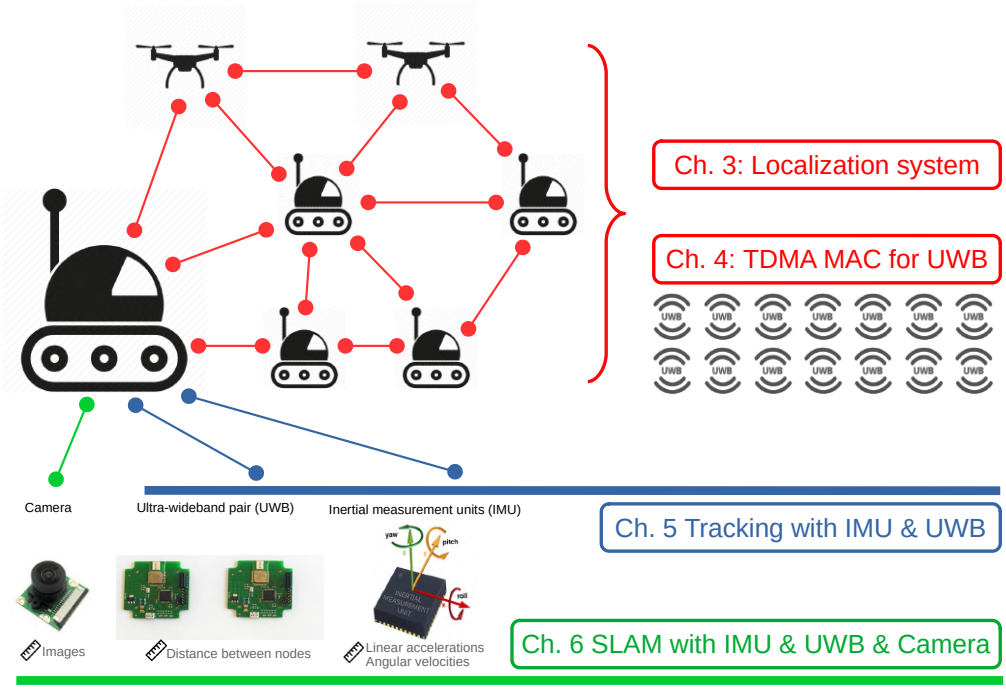


Figure 1.1 This dissertation covers four modules: a) a dynamic coordination system using ranging between robots to reach a common coordinate frame (Ch. 3); b) a distributed time division multiple access (TDMA) algorithm for a mobile UWB network that allows the full usage of the UWB channel (Ch. 4); c) a tracking solution with ranging to a single anchor (Ch. 5); and d) a SLAM system using visual, inertial, and ranging measurements (Ch. 6).

Localization for multiple agents has been extensively studied by the wireless sensor networks (WSN) community [2,3]. However, their approach focuses primarily on static sensor networks and approximate positions. In robotics, relative localization between multiple robots has been studied with a small numbers of robots [4]. Some algorithms (e.g., [5]) designed for swarm robotics can handle a large number of robots. However, as the collective behavior is their focus, the odometry of single robots and the estimated drift have received less attention [5]. Furthermore, swarm robotics research platforms are usually a large number of minimalist robots, which cannot be used for real-world tasks.

In this research project, we build a multi-robot positioning system for the real-world applications in GPS-denied environments. We adopt a top-down localization strategy with ranging sensors: first we establish a common coordinate system in a distributed manner, then we localize each robot in this frame of reference, and finally we use sensor fusion to improve tracking accuracy. The overall structure of this work is shown in Fig. 1.1.

Localization with ranging measurements between robots is attractive both for relative localization [6] and global localization [5]. EM-based localization is generally low-cost and suitable for multi-robot systems, robot swarms, and IoT applications. Wifi, RFID, Bluetooth, and UWB (Ultrawide Band) have their particular techniques for indoor positioning [7]. Wifi can have good accuracy but requires a customized antenna array [8]. RFID is primarily intended to automatically identify and track tags attached to objects, therefore a dense deployment of RFID readers is needed to have good accuracy [9]. The latest version of Bluetooth (Bluetooth 5.1) comes with a range and direction-finding option, which is announced to have centimeter-level accuracy [10]. Cominelli et al. [11] tested the new standard and achieved the sub-meter accuracy with a customized testbed. UWB [12, 13] uses ultrashort pulses to achieve very accurate (cm-level [14]) ranging as well as data transmission, which makes it an ideal choice in our scenario.

Most commercial localization applications using UWB are anchor-based [15, 16], meaning they require at least four calibrated anchors for 3D localization or three for 2D. Anchor-based solutions work fine in an empty space within a structured environment. However, it is not suitable for exploration of an unknown environment due to the pre-installed anchor requirement. In addition, most anchor-based systems use the Time Difference of Arrival (TDOA) method, which requires sub-nanosecond time synchronization in the system [17]. This high precision synchronization is not easy to maintain in a dynamic distributed system.

Localization with UWB can be done in different ways, such as Time of Arrival (TOA), TDOA, and *Two-Way Ranging* (TWR) [18](detailed in section 2.2.1). TWR allows arbitrary pairs of nodes to perform distance measurements. The main advantages of TWR are that it can be used without fixed anchors and it does not require synchronization between nodes. On the other hand, TWR takes a significant amount of time, which can reduce the number of ranging measurements leading to a less accurate localization system. To maximize the number of measurements, one must maximithe the UWB channel usage, requiring an appropriate Medium Access Control (MAC) mechanism among the robots in the system. *Time division multiple access* (TDMA) is widely used in communication networks for shared medium control [19]. Applying TDMA allows multiple devices to share the UWB channel by assigning time intervals (called *time slots*) in which nodes are allowed to communicate.

In this dissertation, we first address cooperative localization and accurate tracking of individual robots fusing UWB and inertial measurements. A *inertial measurement units* (IMUs) and UWB sensors are relatively cheap, this combination has the potential to be applied to the localization of low-cost Internet of Things (IoT) devices.

As many devices are also equipped with a camera, we design a simultaneous localization and mapping (SLAM) algorithm that combines the three modalities (inertial, UWB, and visual). Visual-inertial odometry (VIO) is accepted as the minimal sensor configuration for single robot state estimation and navigation [20]. Recent technical advances [21–23] make VIO more robust and stable. However, its drift caused by error accumulation is still hard to control without pose graph optimization [24] based on loop closures. Although the loop closure is a natural part of a SLAM system, the requirements to close a loop rely on the trajectory followed by the robot and other environmental factors. For example, generating high-quality closures requires revisiting the same location with a similar viewpoint and perception outliers caused by illumination, self-similar environments, etc. can disrupt the optimization process. Furthermore, detecting loop closures requires to keep a history of key frames [25], which adds to the memory and computation requirements of the robots. The problem is exacerbated when considering a multi-robot system that needs to detect inter-robot loop closures and perform distributed optimization to determine the relative coordinate transformation between robots in multi-robot SLAM [26]. In this dissertation, we propose a novel system to reduce the accumulated error and estimate the relative transformations using a UWB sensor.

1.2 Problem Statement

The research work in this dissertation aims at a scalable, accurate localization service for a distributed multi-robot system. In particular, we tackle the following challenges:

- lack of a localization strategy considering both the macroscopic (i.e. the robot team) and microscopic (i.e. single robot) levels;
- shared medium control when applying UWB as a ranging sensor to a distributed, dynamic multi-robot system;
- tracking and localization for robots that only have IMU and UWB;
- control of accumulated error from VIO and inter-robot transformation estimation from UWB measurements.

In the opinion of the author, addressing and resolving these challenges have the potential to substantially advance the application of multi-robot systems in real-world scenarios.

1.3 Research Objectives

The challenges listed in the previous section are addressed through the articles presented in Chapters from 3 to 6, by pursuing the following objectives:

1. propose a localization system to place robots in a common frame of reference. This system must self-organize and reconfigure at the macroscopic level;
2. design a distributed TDMA algorithm for mobile UWB networks that can quickly schedule the access of the shared UWB medium with high utilization;
3. design an algorithm for localization and tracking of robots using IMU and UWB;
4. design a SLAM solution that mitigates the accumulated error of VIO;
5. propose an algorithm that estimates the inter-robot transformations using UWB for multi-robot SLAM.

1.4 Contributions and Impact

To the best of our knowledge, this system is the first system designed for accurate localization service of a multi-robot system using UWB in a GPS-denied environment. We detail each contribution and its impact as follows:

- a dynamic coordination system using ranging measurements between robots;

  Our system allows robots to reach a consensus on a common coordinate frame and each agent to locate itself in the frame reference. This system dynamically changes the configuration to control the accumulated errors. This contribution has a potential impact on the landing of swarm robotics for real-world applications. With this system, robot swarms can perform long-term autonomy as a self-organized system. This work was published as a journal paper in *IEEE Access*.

  [27] Y. Cao, M. Li, I. Švogor, S. Wei, and G. Beltrame, "Dynamic range-only localization for multi-robot systems," *IEEE Access*, vol. 6, pp. 46527–46537, 2018.

- a novel distributed TDMA algorithm designed for mobile UWB ranging networks;

  Our proposed TDMA algorithm can quickly schedule the access of the shared UWB medium with high utilization. This algorithm results in a high range measurement update rate. Beyond the impact on distributed UWB network, any wireless system that requires high channel usage and has frequently changed topology can use this algorithm. This work is submitted to *IEEE Internet of Things Journal*.

  – Y. Cao, C. Chen, S.-O. David, and G. Beltrame, "Distributed TDMA for mobile UWB network localization service," *IEEE Internet of Things Journal*, 2020, (Submitted).

We also propose a UWB based framework to do localization in large scale areas using minimal infrastructure. This work shows the community a potential application scenario. This work is published as a conference paper in *2020 18th IEEE International New Circuits and Systems Conference*. We do not include this paper in the body of this dissertation since its contribution is included by the listed contribution.

[28] Y. Cao, D. St-Onge, and G. Beltrame, "Collaborative localization and tracking with minimal infrastructure," in *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 114–117, IEEE, 2020

- a novel solution for tracking robots using IMU and UWB;

  We propose a speed estimator enables the localization and tracking of robots using IMU and UWB. More than the impact on robots, any devices equipped with IMU and UWB can apply this algorithm, especially considering the low-cost IoT devices. This work is published in *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

  [29] Y. Cao, C. Yang, R. Li, A. Knoll, and G. Beltrame, "Accurate position tracking with a single UWB anchor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2344–2350, IEEE, May 2020.

- a SLAM system combines vision, IMU, and ranging.

  We design a novel two layers sliding window SLAM system using a camera, IMU, and UWB. The system can effectively reduce the accumulated odometry error for VIO. This work is submitted to *Autonomous Robots* journal.

  – Y. Cao and G. Beltrame, "VIR-SLAM: Visual, Inertial, and Ranging SLAM for single and multi-robot systems," *Autonomous Robots*, 2020, (Submitted).

1.5 Thesis Organization

The dissertation follows the structure of a thesis by article which dictates that the body of the contributed articles, published and submitted, is presented in separate chapters.

- Chapter 2 introduces the reader to the topics addressed throughout this dissertation and situates the work presented among the current literature.
- Chapters 3 to 6 mark the body of the dissertation and presents the published/submitted research articles:

  - Chapter 3 introduces a localization strategy for multi-robot system long-term autonomy, presented as a journal paper published on IEEE Access in 2018.
  - Chapter 4 details a UWB network control when we apply the system in Chapter 3 in hardware setup. The paper is submitted to IEEE Internet of Things Journal.
  - Chapter 5 is a published article in 2020 IEEE International Conference on Robotics and Automation. This paper details a novel solution for tracking devices that have only IMU and UWB.
  - Chapter 6 details a novel SLAM for robots equipped with a monocular camera, IMU, and UWB. This paper is submitted to Autonomous Robots journal.
- Chapter 7 discusses the contributions of the previous chapters and highlights the relations among them.
- Chapter 8 wraps the dissertation with concluding remarks and future work outlook.

CHAPTER 2 LITERATURE REVIEW

This chapter is dedicated to a review of fundamental concepts, classic results, and the state-of-the-art from three different directions: localization strategies for multi-robot systems, applications of UWB radios, robot state estimation through sensor fusion with UWB, IMU, and cameras. Each of these topics has its own related works detailed in the later chapters. Here we focus on the inter-topic relations and application background.

2.1 Localization Strategies for Multi-Robot Systems

As a fundamental requirement for most mobile agents, localization has been extensively explored in many different ways [2, 30–33]. Different systems are used in different scenarios , such as GPS [34] for outdoor localization or motion capture system [35] in laboratories. These systems are not suitable for exploration in unknown GPS denied environments. Localization strategies for distributed multi-robot systems ask for particular requirements [36, 37]. In this section, we provide a taxonomy on widely used localization systems and derive our requirements by comparisons. Then we give state-of-the-art localization strategies applied to distributed multi-robot systems.

2.1.1 Taxonomy

Different criteria have been used to categorize localization systems. Some surveys [2,3,38,39] have provided detailed summaries in WSN and robotics. We focus on a few classifications related to the motivations of this dissertation.

- **Indoor Versus Outdoor Localization**

  The main factor that differentiates the indoor localization from the outdoor counterpart is that indoor environments are usually known and controlled. Supportive infrastructure can be easily installed, such as Wi-Fi access points, RFID readers, camera arrays. With the popularization of smart phones and IoT devices, indoor localization has seen fast development within the last decade [3].

  By comparison, outdoor environments are typically more difficult to control. Access to GPS enables localization outdoors. However, because of the poor signal penetration capability of GPS signal [34], mobile robots that deployed in certain urban settings, underground, or off the planet cannot use it. Weather conditions can also limit the usage

of GPS. Other terrestrial predecessors of GPS, such as LoRa [40], cellular network, has a similar spirit. In addition, since these infrastructures are usually designed for long-range communication, the localization accuracy with the default setup is limited.

- **Static Versus Mobile Agents**
  Localization has been widely studied by the WSN community since 2000. However, the localization [41] research in the early stage was mainly designed for static nodes (or rarely moved nodes) in the network, especially for the range-free localization [42]. The range-free algorithms estimated the position based on hop distance or hop count information between anchor nodes and the sensor nodes. The development of wireless sensor networks, mobile sensors, and smart devices, promoted the research on the localization for mobile wireless sensor networks (MWSN) [39]. Localization for mobile agents faces other challenges. As the motion of agents keep changing the topology of the system, the network structure needs to be considered in terms of connectivity, density, etc. The amount of data to transmit, the connectivity between the nodes, access of shared medium can affect the performance of the system. Localization for multi-robot systems also fits into the MWSN localization, but usually requires higher accuracy.

- **Infrastructure-Based Versus Infrastructure-Free**
  Infrastructure-based solutions dominate most of the localization services in our daily life. GPS, with 31 satellites orbiting the planet [43], allows the receiver to locate itself. Infrared camera arrays are widely used in motion capture system [35] to provide accurate position estimation for reflective markers in the laboratory environments. Wi-Fi localization based on measurement from access points (e.g., RSSI, fingerprint, ToF) is a hot topic. UWB systems with preinstalled anchors have great potential for applications both in industry or public service. Generally, most of the infrastructure-based localization systems are designed to localize multiple agents with global position respect to the anchors in a controlled environment.

  Infrastructure-free localization is attractive because it requires minimal efforts to deploy and has the potential as a self-organized system. Anchor-free [44] or beacon-free [45] algorithms were proposed for WSN using relative localization. For a single robot, SLAM is an infrastructure-free solution that robot estimates its pose and maps at the same time. If there is no prior information about the map, SLAM works as relative localization in a coordinate system aligned to the camera pose at the initial stage. Using loop closure with a prior map or introducing absolute positions, the robot can find its global position on the map. By sharing maps to other robots, multi-robot SLAM systems can also create a global map and locate them on the map.

- **Centralized Versus Distributed**
  In centralized system, the locations of all agents are calculated by a central server. The common application of centralized localization is monitoring. A server connected to all anchors receives packets from the mobile agents and then calculates their positions. Centralized systems are generally not scalable, and exposed to the risk of single-point failures. Distributed systems allow each agent to calculate its position through its own measurements. GPS can be seen as a distributed system from the perspective of users. An agent with the GPS signal receiver infers its position based on the signals received from a minimal number of four satellites. Both centralized and distributed systems can be found in UWB anchor-based systems, depending on which side to estimate the position, the anchors or the tags. Because there is no dependence on a central server, a distributed system is scalable, with high robustness, and can be applied to large networks.

From the discussion above, we can see the advantages and the limitations of different categories. For the exploration in an unknown environment, preinstalled infrastructure is not practical. Multi-robot systems also require a high degree of robustness to achieve long-term autonomy. In this dissertation, we seek an infrastructure-free distributed localization system for a multi-robot system that is suitable for both indoor and outdoor environments.

2.1.2 Infrastructure-Free Distributed Localization System

Multi-robot and swarm robot systems have potential to solve various real-world challenges, such as subterranean exploration, establishing network coverage, search and rescue missions [36]. A key advantage of swarm robotics in contrast to classical multiple robots system is its distributed architecture. Swarm robot systems have inherent advantages in terms of robustness, flexibility, and scalability [37]. We will use the term "swarm " and distributed multi-robot system interchangeably throughout the dissertation.

When deploying a swarm of robots in an unknown area, the first thing is that the swarm needs to agree on a common coordinate system. With this frame of reference, measurements or information from neighbor robots can be used directly. A simple and straightforward approach is to use a fixed reference frame [5, 46–48]. This usually requires a set of beacons, which can either be static robots or external devices which are not part of the swarm. Indelman et al. [49] used a probabilistic approach to reach a consensus on the reference frame when dealing with spurious measurements (i.e., false negatives). In a recent publication, Shirazi and Jin [50] developed a localization method that required several static robots which acted as beacons to determine a common coordinate system. Using three robots, their algorithm

provided coordinates for other robots within the team using relatively trilateration. This work fits our scenario to start the localization, however, they mainly considered the application for a swarm of minimalist robots. Qin et al. [17] designed a BLAS system that used UWB for the localization of a multi-agent system. They divided the agents into parent and child groups: the parent agents acted as moving anchors and created a coordinate frame for the system. They proposed a distributed clock synchronization in parent agents to maintain a high clock precision. Because the parent nodes were fixed, the child nodes had to be kept within the range of parent nodes. In our system, we propose a dynamic way that allows the robots to bid for the reference nodes in order to create the coordinate frame. Furthermore, a reconfiguration strategy to maintain the coordinate center is designed to constrain the accumulated error. The localization of single robots is also addressed.

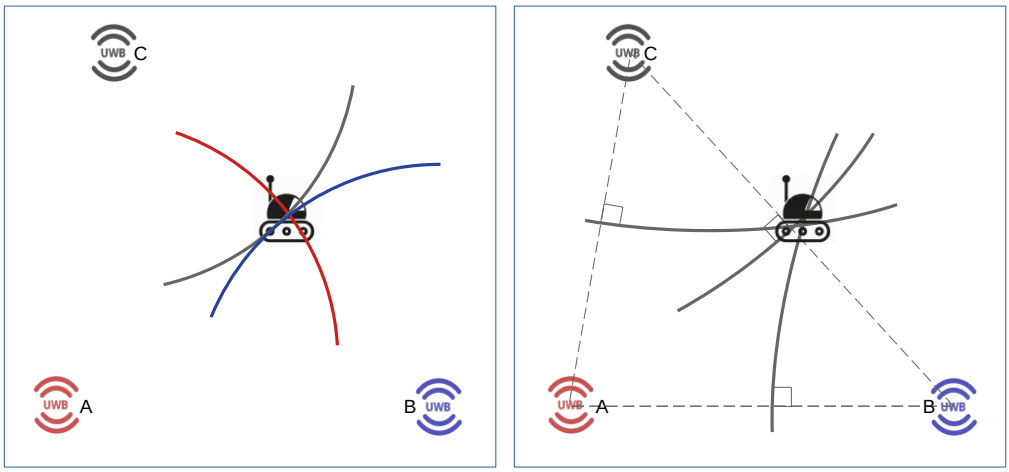
2.2 Ultra-WideBand Radio

Modern UWB radio systems were developed based on the impulse radio technology [51], which has been in development since the late 1960s [13]. Unlike the ME wave used in our daily life (e.g., Radio, Wi-Fi, Bluetooth), UWB is based on the transmission of ultra short impulses in the time domain (typically at nanosecond scale), but a wide bandwidth in the frequency domain [12]. This provides two main benefits: a) low energy consumption and robustness to interference (no carrier wave); b) excellent time resolution allowing accurately timestamped to mitigate multipath effects (ultra shot impulses). UWB has a good performance both in communication [52] and localization [53]. In this dissertation, we mainly focus on the localization application of UWB technology.

2.2.1 Localization With UWB

As introduced above, the ultra shot impulses enable the receiver to have accurate measurements for the timestamp when receiving packets. By leveraging the Time of Flight (ToF) principles, UWB can be used to perform localization and tracking. We now introduce the main strategies used for localization and explain their advantages and disadvantages.

- **Time of Arrival (TOA)**
  TOA is the straightforward application of ToF. By recording the signal propagation time, receivers can calculate the distance to the transmitters. Then the system can apply trilateration to find the intersection of circles for multiple transmitters (or receivers), namely anchors, as shown in Fig. 2.1-a. The main limitation of this strategy is the requirement of a very accurate synchronization among all transmitters and receivers. A nanosecond error in the synchronization can cause a 30 cm error in distance



(a) TOA system (b) TDOA system

Figure 2.1 UWB localization with Time of Arrival and Time-Difference of Arrival strategies. Both strategies use anchor setup and need synchronization in some extend.

estimation.

- **Time-Difference of Arrival (TDOA)**
  TDOA is based on measuring the time difference of arrival of two signals, which corresponds to the distance difference from the two anchors. As shown in Fig. 2.1-b, the robot is located in the intersections of the hyperboloids from anchors. Compared with TOA, TDOA is more practical because it only requires synchronization among all the anchors. The application of TDOA can be either centralized or distributed. In a centralized setup, anchors receive the signal from the tags. A server connected to all anchor calculates the position for the tag. In a distributed setup, anchors send signals in a predefined periodic manner. The tag listens to the signals and calculates its position (similar to how GPS works). Both solutions require precise synchronization among all anchors.

- **Two Way Ranging (TWR)**
  Different from the TOA and TDOA, TWR [18] just measures the distance between two nodes. TWR leverages the round trip delay of the signal to avoid the requirement of synchronization. As shown in Fig. 2.2, UWB device A first sends a message to B. Then B replies a message including the timestamp when it received the message (t^B_{RX}) and the timestamp when it replied the packet (t^B_{TX}). When A receives the reply, it can calculate the distance as Equ. 2.1, where c is the speed of light.

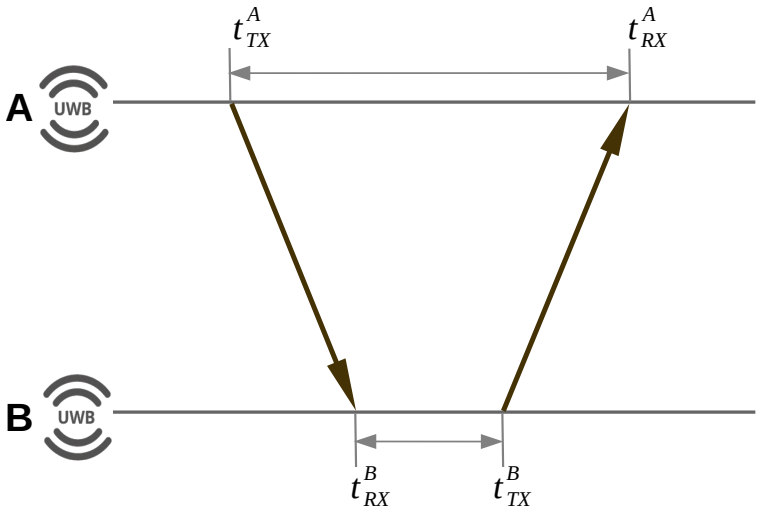


Figure 2.2 Principle of distance measurement with two way ranging. Assisted by the timestamp replied from UWB device B, device A can estimate the distance without any synchronization requirement.

$$d_{AB} = c \cdot \frac{(t^A_{RX} - t^A_{TX}) - (t^B_{TX} - t^B_{RX})}{2} \tag{2.1}$$

where t^i_{TX}, t^i_{RX} refer to the timestamps when node i transmits or receives a packet in its clock time.

Since no synchronization is required, TWR is very flexible and has mostly used in a distributed system. The disadvantage of TWR is that the time used for the two-way communication is higher than TOA or TDMA, which can limit the scalability of the system.

UWB technology has attracted substantial attention due to its high localization accuracy. Anchor-based UWB localization has been widely commercialized [15,16] using TDOA, a technique which requires a sub-nanosecond clock synchronization between anchors [17]. Compared with TOA and TDOA [54,55], TWR [18] is a more flexible solution, enabling arbitrary pairs of nodes to perform distance measurements at any time. Therefore, we choose TWR ranging solutions in our work.

2.2.2 UWB Network Control

TWR has started receiving more attention on multi-robot localization [56–59]. These results show the advantages of UWB for multi-agent systems. However, due to the small number of

UWB nodes in the experiments, network usage has not yet been explored. Ridolfi et al. [60] analyzed the scalability of UWB-based localization and showed that coordination protocols had the huge impact on scalability. Qin et al. [17] designed a BLAS system that used UWB for the localization of a multi-agent system. They divided the agents into parent and child groups. They proposed a distributed clock synchronization in parent agents to maintain a high clock precision. Then they used TOA to locate the child groups. The biggest advantage of this system is that the number of child agents is theoretically unlimited because they only passively receive the pings from the parents. Although the child nodes have no conflicts, the communication between parent agents still needs to be coordinated. Qin et al. use round robin as distributed TDMA to achieve collision-free broadcasting from parent agents. However, this solution assumes all the parent agents are fully connected, which limits the scalability of the system. Macoir et al. [61] also used an anchor-based TDOA strategy, but it is designed for relatively large-scale networks. The authors divided the usage of the network into multiple small cells to cover large areas. Unlike the passive child nodes from [17], Macoir et al. [61] scheduled active slots for mobile tags to broadcast messages and use a server connected with the anchors to calculate the positions. The server is also responsible for slot assignment and thereby forming a centralized system.

Zhu and Kia [62] proposed a negotiation-based dynamic TDMA algorithm across the UWB network, G.M. ter Horst [59] developed an anarchic TDMA algorithm based on the DESYNC algorithm [63]. Both methods only consider one-hop collisions, so that collision can occur for hidden nodes at neighborhood boundaries.

The most widely used MAC control on UWB is the IEEE802.15.4-2011 protocol [64], integrated with the Decawave 1000 [18] chip, which is the most popular commercial UWB chip on the market. The protocol uses Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) or slotted ALOHA [65, 66] to avoid collisions. However, these two strategies are only applicable to lightly loaded networks that have a small probability of collisions. In our case, we want to maximize channel usage for accurate localization. To the best of our knowledge, this work is the first to apply dynamic TDMA for UWB localization in multi-hop ad hoc networks with mobile devices.

Compared to the TDMA techniques available for communication networks, our application has several additional requirements: 1. the maximization of channel usage to increase the localization frequency; 2. rapid time slot scheduling to account for dynamic topologies; 3. decentralization to avoid the need for fixed infrastructure.

The TDMA slot assignment in a wireless network can be seen as an extension of the vertex

colouring problem in graph theory, with the additional constraint of needing to avoid collisions in 2-hop neighborhoods [67]. The problem was proven to be NP-complete [68], and several heuristic solutions were proposed [67–69] to get the near-optimal results with full knowledge of the network topology.

When considering a distributed system where nodes only receive messages from neighbors, some works (FPRP [70], DRAND [71], DICSA [72], PCP-TDMA [73]) propose negotiation-based algorithms to find the smallest frame with conflict-free scheduling. After scheduling, all nodes in a network need to agree on the same frame size to execute the slot schedule with the same frame of reference. We believe that this strategy is not the best strategy for a highly dynamic network, as it would require fast global consensus of the frame size across the network. Furthermore, when a node makes a proposal during negotiation, it has to wait for the feedback of all neighbors, potentially leading to long convergence times if the neighborhood size is large or the network topology is complex. For similar reasons, practical mobile networks use fixed-length frames such as USAP [74] for military telephone networks and VeMAC [75] for vehicular ad hoc networks (VANETs).

Since the size of the frame should be larger than the total number of nodes in the network, these protocols may have several unused time slots. Researchers found a balance between frame size stability and channel usage by doubling or halving the frame size, such as in USAP-MA [76] and Dynamic-TDMA [77]. Cao and Lee proposed VAT-MAC [78], a VANET with a changing frame size, relying on a roadside unit (RSU) infrastructure. In our method, we use a fixed frame size that equals the total number of nodes allowed in the system. Despite this static allocation, we have high channel usage since we always allocate all the available time slots to the neighborhood's devices.

2.3 Sensor Fusion with UWB/IMU/Camera

In this part, we assume the UWB network control allows each robot in the system can have collision-free ranging to its neighbors. All robots in the system reach a consensus on a coordinate frame. Then, we study how each robot can estimate its own odometry by fusing UWB and IMU measurements without or with camera image inputs.

Starting with the minimal configuration, we explore the literature of tracking with only IMU and UWB. Then we introduce the concept of SLAM to review the literature related to the odometry estimation using IMU, UWB, and a monocular camera. At last, we introduce the cooperative localization of the multi-robot systems.

2.3.1 Tracking With IMU and UWB

By introducing the use of UWB and a static anchor in the environment, the problem is single anchor localization or source localization.

Many researchers have studied single anchor localization, especially for underwater robotics [79, 80]. Underwater robots usually use acoustic sensors, top-of-the-line IMUs, and expensive doppler sensors. Guo et al. [81] proposed a cooperative relative localization algorithm. They used an optimization-based single-beacon localization algorithm to get an initial position for collaborative localization. However, they only observed a sine-like moving pattern and they required a velocity sensor. Similar with a recent work proposed by Nguyen et al. [82], they also used odometry measurements from optical flow sensors. In our study, we only use UWB and a low-cost IMU, dropping the need for a velocity sensor.

To better understand the single anchor localization problem, which is typically non-linear because of distance and angle, an observability study is necessary. Based on the groundwork of Hermann and Krener [83], researchers have studied the observability of range-only localization system, from one fixed anchor [80] to the relative range and bearing measurements between two mobile robots [84]. Bastista et al. [85, 86] used an augmented state to linearize the problem, enabling classical observability analysis methods. A recent study [87] explored the leader-follower experiment for drones with UWB ranging between robots, also with velocity measurement either from the motion capture system or optical flow. However, all these studies assumed the velocities were available as a direct measurement, which we do not have.

Although we do not have velocity sensors, the system still needs velocity to be observable. Getting a reliable velocity from a low-cost IMU or UWB is challenging: the integration of acceleration drifts dramatically using low-cost MEMS IMU sensors [88, 89]. For position estimation, IMUs are often combined with other sensor measurements, like GPS, multiple anchors [90], and cameras [91].

One straightforward way to estimate velocity is the distance change from a UWB anchor when the robot is moving along a radical line from the anchor. This situation is rarely lasting in reality, but the range changing pattern can be used as a speed estimator. We propose a method based on simple geometry relations under the assumption that the robot moves at constant velocity. The estimated speed coupled with data from the IMU gyroscope can provide a velocity estimate to keep the system observable. Finally, we use an EKF to fuse range, orientation, and velocity estimation to get the robot pose.

2.3.2 Simultaneous Localization and Mapping (SLAM)

SLAM has been studied intensively for more than thirty years [33, 92–94]. As the name suggests, SLAM is a process by which a mobile robot can build a map of the environment while localizing itself on the map. The map here represents a collection of features with the location information. For example, it can be a set of 3D visual features from visual sensors,or line and corner features from LiDAR sensors. We indicate this optimization state as $\mathcal{X} = \{\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n, \mathbf{f}_0, \mathbf{f}_i, \ldots, \mathbf{f}_k\}$, where $\mathbf{X}_i$ represent the pose of a robot at the time step i, n is the current time step, $\mathbf{f}_j$ represent the position of the feature j, and k is the maximum number of features tracked, as shown in Fig. 2.3. The goal is to get the optimal result of $\mathcal{X}$ with a set of measurements $Z = \{z_0, z_1, \ldots, z_q\}$.

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}|Z) \tag{2.2}$$

$$= \underset{\mathcal{X}}{\operatorname{argmax}}\, p(Z|\mathcal{X})p(\mathcal{X}) \tag{2.3}$$

For implementation, the architecture of a SLAM system can be divided into two parts called the front end and the back end, shown in Fig. 2.4. The front end depends on the sensor, which has a function converting the raw measurements into variables that the back end can easily process. The nature of the back end is a Maximum a posteriori (MAP) optimization as explained above. Taking vision-based SLAM as an example, the front end extracts the pixel localization of some distinguishable points in the image, using feature descriptors (BRIEF [95], ORB [96], etc.). The front end defines the measurement model $z_k = h_k(\mathcal{X}) + \epsilon$, where ϵ is the measurement noise and k indicates the kth feature. For the camera, $h()$ is the projection of a 3D point in a pixel in the image plane. With incoming observations, the back end is employed to solve the MAP (Equ. 2.2) interactively to estimate the 3D localization of the tracked points as well as the camera poses.

Assuming that the measurements in Z are independent, the MAP problem can be factorized into a factor graph format [97] as :

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}) \prod_{t=1}^{q} p(z_t|\mathcal{X}) \tag{2.4}$$

Since maximizing the posteriori is the same as minimizing the negative log posteriori, the

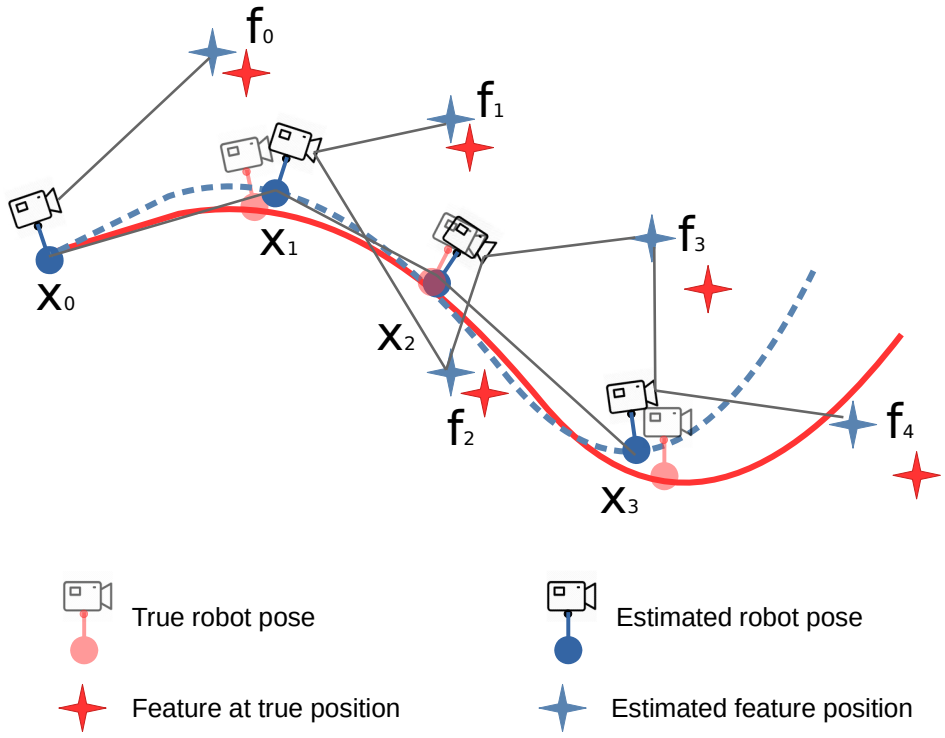


Figure 2.3 SLAM is the problem of constructing a map of an unknown environment while simultaneously localizing itself on the map.

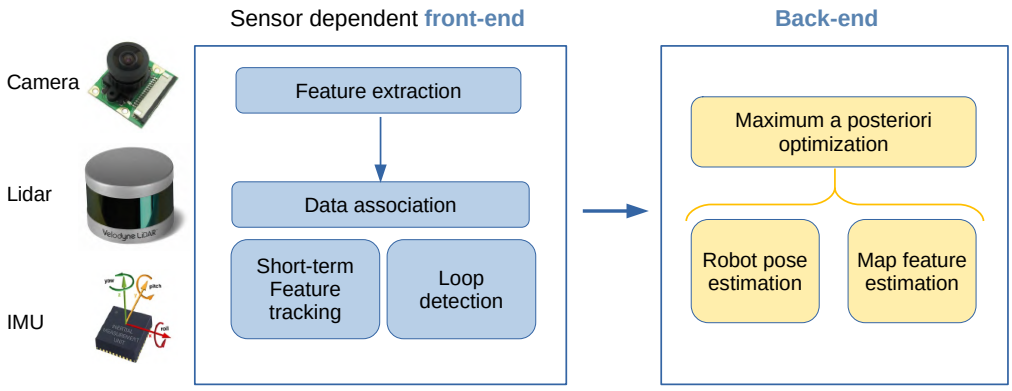


Figure 2.4 The architecture of a SLAM system. The front end is sensor dependent with a function to transform the raw measurements into the variables that the back end can easily process. The nature of the back end is a Maximum a posteriori optimization.

MAP estimation in Equ. 2.4 can be written to :

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} -\log\left(p(\mathcal{X})\prod_{t=1}^{q} p(z_t|\mathcal{X})\right) \tag{2.5}$$

$$= \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{t=1}^{m} p(z_t|\mathcal{X}) - p(\mathcal{X}) \tag{2.6}$$

Assume the measurement noise ϵ follows a zero-mean Gaussian distribution with an information matrix Ω (inverse of the covariance matrix), the measurement likelihood element in Equ. 2.5 follows :

$$p(z_t|\mathcal{X}) \propto \exp(||h_t(\mathcal{X}_t) - z_t||^2_{\Omega_t}) \tag{2.7}$$

$$p(\mathcal{X}) \propto \exp(||h_0(\mathcal{X}) - z_0||^2_{\Omega_0}) \tag{2.8}$$

Then Equ. 2.5 can be written into a more explicit nonlinear least squares problem as

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{t=0}^{m} ||h_t(\mathcal{X}_t) - z_t||^2_{\Omega_t} \tag{2.9}$$

The least square problem can be viewed as finding the set of robot poses and the map points which have the minimal residual with measurement functions. Optimization solvers such as Gauss-Newton are commonly used via successive linearization. Current SLAM libraries (e.g., GTSAM [98], g2o [99], Ceres [100]) are able to solve problems with large number variables in a few seconds.

Even though, as the robot moving, the set of state variables becomes larger and larger, especially in long-term autonomy. It is not practical to estimate the entire trajectory from the start. To keep the state size manageable, key frame selection [25, 101] and sliding window techniques [102, 103] are used to limit the number of variables in the optimization problem. When the old frames and features are removed from the sliding window, proper marginalization [23, 104] converts the marginalized factors into a prior factor for the optimization.

In our work in Chapter 6, we propose a novel double layer sliding window system to reduce the accumulated error using UWB and integrate it with the state-of-the-art VIO SLAM

framework.

2.3.3 SLAM with IMU, UWB, and Camera

Monocular visual-inertial odometry [20] is a popular choice as it provides good state estimation performance with a minimal sensor configuration. Although state-of-the-art VIO algorithms (e.g. SVO [22], VINS-Mono [23], DSO [105]) can reach very high accuracy in relative translation and orientation, the accumulated drift can still be an issue: any small orientation error can lead to large end-point error. Our system leverages UWB ranging measurements to correct the accumulated error. We developed our system based on VINS-Mono [23], which is a robust and versatile state estimator which uses a sliding window tightly coupled nonlinear optimization for visual and IMU measurements.

UWB technology, as a localization solution on its own, has attracted a lot of attention in recent years both in research and industry for its decimeter localization accuracy. However, most results are based on a well-calibrated multi-anchor setup [54, 106–109], which is not applicable for navigation in unexplored, unstructured environments. Single anchor setup is desired as the easy deployment [29]. Wang et al. proposed a system using cameras, IMU and UWB to bypass the complexity of loop closure [110] . However, they still used multiple preconfigured UWB anchors. Their UWB module provided coarse drift-free global position and VIO identifies the local trajectory. On the contrary, in this paper we use only one anchor, placed in an unspecified location. A system by [111] had a similar spirit to ours. They started with one UWB anchor and keep dropping anchors from a moving robot. Unfortunately, their experimental results were available only in simulations for one sequence of the EuRoC dataset [112], with five simulated anchors. A closely related work published recently [113] combined monocular camera with UWB ranging to a single anchor. The system was developed based on ORB-SLAM [25] and used UWB ranges to estimate the scale after recording the first batch of data. In our case, we also integrate the ubiquitous IMU to enable true scale all the time. We design a double layer sliding window algorithm, which effectively fuses accurate VIO and range constraints along the trajectory.

2.3.4 Multi-robot SLAM

Multi-robot SLAM has gained recent attention for the increased viability and accessibility of multi-robot systems. A survey [114] has shown a comprehensive survey of multi-robot SLAM and points out one key issue: relative pose estimation. Most current multi-robot SLAM systems solve this issue by analyzing inter-robot loop closures, either in centralized [115] or distributed [116, 117] fashion. The distributed approach is more robust, but it is harder to implement in practice: robots need to exchange map data to get the feature database for

future loop closures, and distributed optimization usually requires additional communication and computation. Ranging measurements can assist in relative pose estimation. Trawny et al. provided theoretical proofs and simulations that showed how six range measurements can be used to get the transformation matrix between two robots [6]. With the observed gravity direction. [57] adopt it to 4DOF relative pose estimation with a UWB setup, and used it for merging maps for VR applications. Both methods required mutual ranging measurements over long trajectories. In our solution, robots can estimate the transformation matrix as soon as they can get two measurements from their neighbors, which meets the requirement of real-time transformation estimation during robot rendezvous. These two methods [6, 57] are solutions when no common anchors are present in their records and can be combined with inter-robot loop closures to improve the transformation results. A recent work [58] presented a decentralized Visual-Inertial-UWB fusion for relative state estimation. They combined VIO, UWB and vision detectors of YOLOv3 [118] to track the relative state of neighbors.

CHAPTER 3 ARTICLE 1: DYNAMIC RANGE-ONLY LOCALIZATION FOR MULTI-ROBOT SYSTEMS

Preface: The starting point of this research is from the emergence of multi-robot system or swarm robotics. Multi-robot system have potential to help human in many applications, such as disaster response. Coordination and localization is important for the system to work collaboratively. In the article presented in this chapter, we propose a strategy for a group of robot deployed in a unknown GPS-denied environment. With their communication and ranging measurement, they can maintain a global coordinate frame and tracking their position in a distributed manner.

Full Citation: Y. Cao, M. Li, I. Švogor, S. Wei, and G. Beltrame, "Dynamic Range-Only Localization for Multi-Robot Systems," *IEEE Access*, vol. 6, pp. 46527–46537, 2018.

Personal Contributions: Y. Cao conceived of the presented ideas, designed and implemented the simulations. Y. Cao wrote the manuscript with support from M. Li, G. Beltrame, and I. Švogor.

Abstract

The localization problem for multi-robot teams has been extensively studied with the goal of obtaining precise positioning information, such as required by a variety of robotic applications. This paper proposes a dynamic localization approach which exploits multiple robots equipped with range-only ultra–wideband sensors to create and maintain a common self-adaptive coordinate system. For 2D localization, we use three robots with relative range measurements to build a global coordinate system. We recursively apply an extended Kalman filter, which results in accurate position estimates over time. We also propose a reconfiguration approach that prevents error accumulation from ultra–wideband sensors. The applicability of our approach is tested through a campaign of simulations, which show promising results.

3.1 Introduction

Multi-robot and swarm robotics systems have a high potential to solve various real–world challenges, such as underwater exploration, establishing network coverage, search and rescue missions, etc. [36]. A key advantage of swarm robotics in contrast to traditional approaches, is its decentralized architecture, which is most suited to highly dynamic and harsh environments, where it is hard to achieve sufficient system reliability and robustness using centralized

systems. In addition, with a decentralized approach, solutions tend to be simpler and more flexible [37]. In this paper, we focus on a decentralized system architecture for the purpose of robot localization.

As one of the fundamental challenges in mobile robotics, robot localization has been extensively studied in the past [119–121], using various approaches, sensors, and for different application scenarios [30, 122–124]. In general, based on their architecture, current localization methods can be classified in two main groups: *global/centralized* (e.g., using global positioning system, overhead cameras, motion capture systems, etc.) or *relative/decentralized* (e.g., using laser range finders, ultra–sonic sensors, onboard cameras, etc.).

Though a great number of publications address localization, there are still open challenges, specifically in the context of fully decentralized systems, as are swarms of robots.

In this paper, we propose a decentralized localization system for swarm robotics with ultra–wideband (UWB) ranging sensors in a two-dimensional plane. The selection of this particular sensor is due to its accuracy, low cost, and the fact that it is not limited to line–of–sight. However, localization using range-only information requires addressing of several issues.

First, the orientation of the robots needs to be estimated, and second, UWB range measurements become less accurate with distance, which requires to mitigate error accumulation. Our method addresses these issues in three stages: a) *initialization*, b) *localization*, and c) *reconfiguration* as illustrated by Fig. 3.1. In our approach, the swarm collectively creates and maintains a common coordinate system that is used for the localization of individual robots, while being continuously updated to minimize the position error coming from UWB measurements.

The primary goal of the *initialization* stage is to select three reference robots that are used to create a coordinate system. Initially, the swarm members elect the first robot, also called the leader. The choice of the leader is based on neighbor density, and the leader itself becomes one of the reference robots. After this step, the leader selects two additional robots which are then used to create a coordinate system. Finally, this coordinate system is shared with the other members in the swarm, which use the information to compute their position[1]. As the robots move to perform their tasks, their coordinates change and the distances can increase, thus building up estimation errors. In practice, pure trilateration is not suitable for continuous localization. This issue of the dynamics of the system is handled in the localization stage.

[1]The assumption used here is that all the robots within the swarm are able to communicate within the operating area.

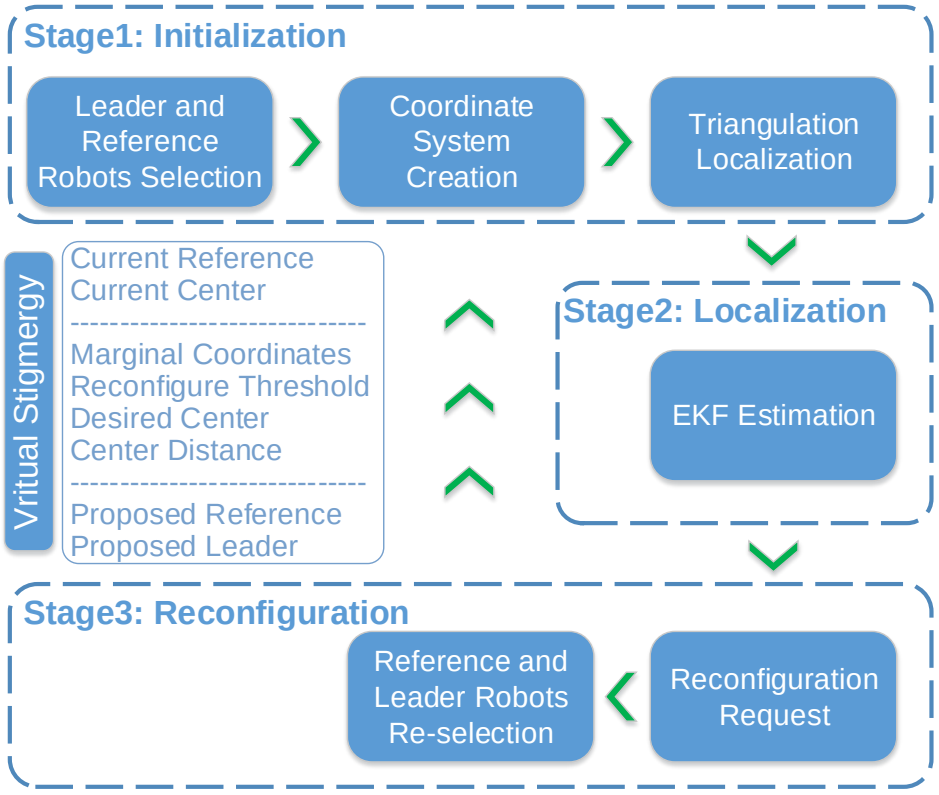


Figure 3.1 The system structure of dynamic range-only localization of multiple robots

In the *localization* stage, an extended Kalman filter (EKF) is applied for localization estimation. The EKF only requires the initial position of any given robot, and performs continuous localization using the common coordinate system. As the robots are moving, the UWB measurement error also builds up for the position of the reference robot, so this can potentially introduce additional error.

In the final stage, *reconfiguration*, we select a new set of reference robots based on the centroid of the swarm when the measurement uncertainty passes a given threshold, significantly reducing the estimation error. In the following sections, we describe all stages in detail, and show the accuracy measurements resulting from a simulation campaign.

To summarize, the contributions of this paper are:

(a) a dynamic self-adaptive approach for multi-robot localization based on range-only information,

(b) a method for determining a swarm central point and coordinate–system reconfiguration

for the purpose of mitigating UWB sensor measurement error,

(c) a reconfiguration mechanism for the dynamic adaptation of the formation dynamics and to avoid error accumulation, and finally,

(d) an application of EKF to obtain accurate position estimation of moving robots using UWB sensors.

The rest of the paper is organized as follows: Section 3.2 makes a brief review of localization sensors, strategies, and estimation models; Section 3.3 illustrates the proposed localization system; in Section 3.4, we introduce the EKF and we detail the dynamic localization estimation; we evaluate the proposed approach is evaluated in simulation in Section 3.5; finally, Section 3.6 concludes the paper.

3.2 Related Work

Localization is a well known and explored research subject: to present a structured and clear picture of the related work, this section is divided in three main topics in localization: sensors, strategies, and estimation models.

3.2.1 Localization Sensors

The application of traditional sensors, such as RGB or IR cameras, laser range finders, ultrasonic sensors, global positioning system (GPS), etc., in swarm robotics can be limiting. This is due to the fact that these sensors typically require heavy processing, or can only be used by a centralized system, or only work in certain types of environments.

Localization methods which use high-resolution RGB cameras usually rely on a vision subsystem which recognizes a set of QR markers (e.g. AprilTags [125]), which have a significant impact on processing resources [126]. To avoid this, researchers often use motion capture systems (e.g. Optitrack, Vicon [35, 127]) with a set of robots carrying physical IR markers that are unique for each robot. Although these methods are very precise, they are limited to a laboratory environment. In addition, the camera–marker solutions usually require centralized processing, which distributes the positions of the robots through a shared communication channel.

The use of GPS and derivatives (DGPS, RTK) is a convenient options, but its application is limited to a restricted set of environments. To ensure proper signal reception, it is often necessary to have a clear view of the sky, which mandates outdoor usage, with a *meter* scale precision. This is an ongoing issue for underwater, indoor, and robots for planetary

exploration. In such environments, researchers often apply acoustic-based sensors or laser range scanners to use relative measures of the robot's environment for its localization. While there have been numerous successful applications of these, the decision between the two usually involves a tradeoff between precision and price. Since swarm robotics assumes tens, hundreds, or even more robots, this can be an issue. In addition, the usage of these sensors is limited to line–of–sight and requires significant processing capabilities.

Ultra-Wide Band (UWB) sensors are a promising solution for localization, considering the cost per unit, accuracy, and the available bandwidth [54]. They are advantageous for their inherent data transmission features, while not being restricted to line–of–sight. However, UWB sensors did not receive much attention in robotics due to their drawbacks in resolution and accuracy [128]. Recent works by Gonzalez, et al. [106], Hollinger et al [129], Prorok and Martinoli et al. [54, 130, 131] have shown that UWB sensors can be successfully used for localization, with accuracy around 5 cm. In particular, UWB sensors are suitable for swarm robotics as they can be used in a fully decentralized manner [132].

3.2.2 Localization Strategies

To successfully perform localization, the swarm of robots needs to agree on a reference coordinate system, which is used to transform all relative measurements into position data for each member of the swarm. A simple and straightforward approach is to use a fixed reference frame [46, 47]. This requires a set of beacons, which can either be static robots or external devices which are not part of the swarm. However, with this approach the general assumption is that a reference coordinate frame is known beforehand. This presents a major challenge for a swarm which is to be deployed in an unknown environment, with no access to global positioning, e.g. an underwater sensor array. Such scenario does not assume *a-priori* information about the environment nor a consensus on the reference frame. While this issue has been addressed in the past, current solutions require static swarm members that other robots use for localization (e.g., [5, 48]). Indelman et al. [49] use a probabilistic approach to achieve a consensus for the reference frame when dealing with spurious measurements (i.e., false negatives). In a recent publication, Shirazi and Jin [50] developed a localization method that requires a swarm of robots to be surrounded by several static robots which act as beacons to determine a common coordinate system. Using three robots, their algorithm provides coordinates for other robots within the team using relative trilateration. While this approach is promising, it requires that the robots be stationary in the localization stage. A good overview of traditional distributed estimation techniques, such as Non-Bayesian, Bayesian and Factor graphs is given by Wymeersch et al. However, in contrast to our approach, authors combine communication networks with a positioning system [133]. In our paper, the communication

network is used only for information sharing. The range sensors and non–linear Kalman filters are then applied to perform localization and infer position information.

3.2.3 Estimation Models for Localization

In multi-robot system, collaborative localization is a more robust approach to the localization problem [134]. While there are a variety of strategies, the prevailing methods use probabilistic models, which caught on in the early days of simultaneous localization and mapping (SLAM). Thrun et al. provided a general framework which used probabilistic methods, such as Monte–Carlo and Markov chains to localize robots, while keeping track of uncertainty [31, 135]. In such approaches, each robot is assigned a probabilistic cloud of particles representing its position, which is reduced when a robot detects a known feature in the environment, or another robot. Prorok et al. proposed a similar approach for multi-robot localization using a range and bearing sensor [136]. Their goal was to minimize the overall complexity of the particle filter based approach by defining reciprocal sampling which allowed to reduce the number of necessary particles. Luft et al. recently proposed a fully decentralized localization algorithm based on the EKF [137]. The algorithm tracks inter–robot correlations and it does not require measurement storage. Their method focuses on localization, assuming limited communication to neighboring robots. In this paper, we adopt a similar approach using EKF filters while a relative coordinate system is built based on range-only UWB sensors.

3.3 Range-Only Self-Organized Localization

As a team of robots equipped with UWB sensors is deployed within an unknown area, their localization becomes a challenge. These robots need to use only range information to create a common coordinate system in which every robot can estimate its position and that of its team members. To be applicable in the real world, the entire procedure should be fully automated, self–organized, and maintained in time so that the robots can seamlessly perform their tasks.

To complete this challenge, we propose a localization procedure divided into three consecutive stages: a) localization system initialization, b) localization and state estimation using an EKF, and c) the localization system dynamic reconfiguration.

3.3.1 Stage 1: Localization System Initialization

As initial state, we assume that all members of a multi-robot team are scattered in an unknown region. To successfully perform localization, the robots first need to agree on a common reference frame, i.e. robots need to develop a common coordinate system. Since we

are dealing with range-only information, the approach to developing a common coordinate system in this paper requires three robots, namely a *leader* and two *reference robots*. The imaginary line connecting the reference robots then represents the x–axis, while the imaginary line crossing through the leader robot as an orthogonal projection to x–axis, represents the y–axis. After this, a common coordinate system can be shared between all robots, which can infer their positions using trilateration.

In general, the robots in the multi-robot team are classified into three categories:

- reference robots – after initialization, each reference robot keeps estimating its position. In addition, robots that have been localized can be added into a shared table as reference robots;
- non-reference robots – use the position of reference robots to estimate an initial position based on trilateration. The non-reference robots localized by the trilateration become reference robots;
- marginal robots – represent the robots at the boundary of the current reference frame (further described in the following).

The information between robots is shared by means of a bio–inspired distributed consensus system called Virtual Stigmergy. The *Virtual stigmergy* can be seen as a tuple space shared between all the robots in the swarm via a communication medium [138]. We refer to this tuple space as the virtual stigmergy table (VST). This means that, once a robot stores some data into the VST, this information will gradually propagate until it is shared by all swarm members within communication range, although not instantly. For this work, we described the robots' behavior with a domain–specific programming language for swarm robotics called Buzz, in which the VST and its propagation are built-in [139].

Using the VST, all robots reach a consensus on which robot is elected leader, along with two additional reference robots. To clarify further, consider the initialization phase, which is divided in three tasks: leader robot election, reference robot selection, and finally coordinate system creation and propagation.

Leader robot election

The election for the leader robot is by bidding on neighbor density. Therefore, the robot which has the most neighbors within a given range will be selected as a leader. Such choice is based on two reasons: a) the range of the UWB is limited, meaning that picking any random robot would not guarantee that it will have reference robots within its range, and also b)

selecting robots which are clustered closer together reduces the initial measurement error of the UWB (as it is increased with distance). We would emphasize that in the initialization stage, we develop an initial (possibly sub-optimal) coordinate system that is improved in the next stages.

Initially, each robot obtains the number of its neighbors and offers it as a bid in the VST. This information is propagated between the robots, and once the robot with most neighbors becomes known, it is selected as a leader, by its unique identifier (ID). Fig. 3.2 shows this step, and illustrates several differently colored candidates. In this example, the red candidate has the highest number of neighbors and it is selected as a leader.

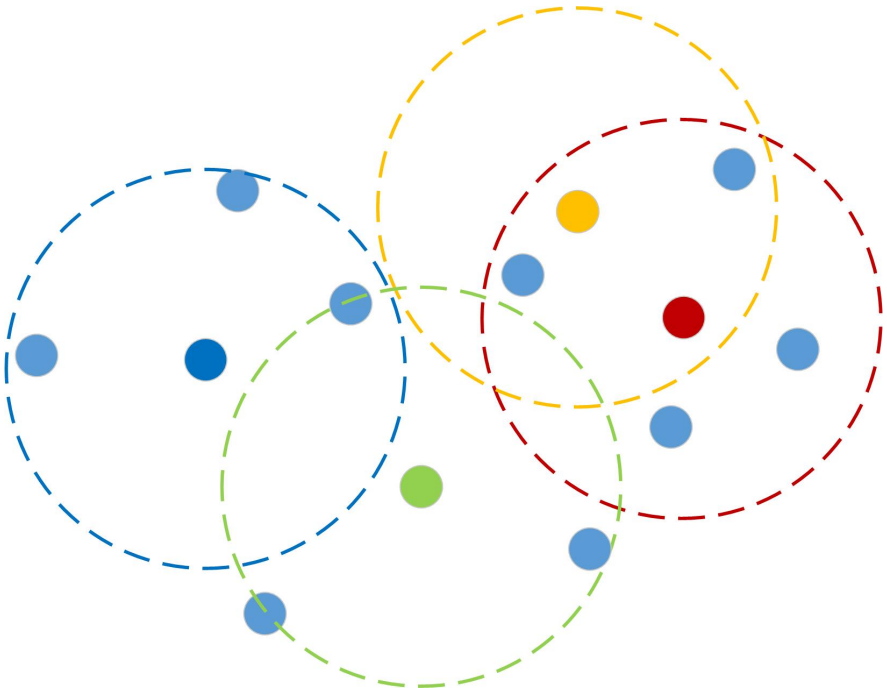


Figure 3.2 Leader election based on distributed consensus

Reference robots selection

Once the swarm determines the leader, it needs two more reference robots to create a common 2D coordinate system. The reference robots are selected based on the following criteria: a) a robot is within the UWB measurement range of the leader, b) robot is the closest to the leader, so that it has lower measurement error from the UWB sensor. Fig. 3.3 illustrates two selected robots (a, b, marked in yellow), which are used in the following step to create the coordinate system.

To clarify the proposed approach, Algorithm 1 (lines 1–27) provides in-depth details. The

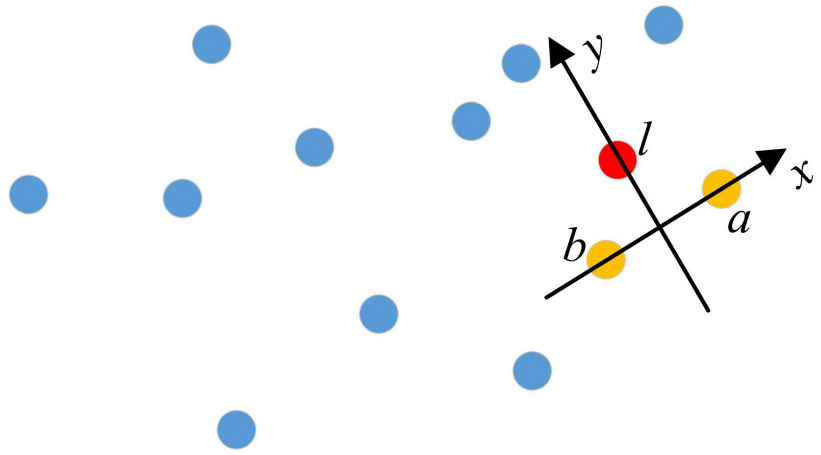


Figure 3.3 Reference robots (red and yellow dots) create coordinate system

algorithm starts by initializing all the necessary local variables (lines 1–3);

where *leader_id* stores the identifier of the elected leader, *reference_robot* stores the selection of reference robots, *leader_election* is a flag which indicates whether the procedure of electing the leader is finished, and finally, *bid* contains the number of neighbors of a current robot within a certain range (using a UWB sensor).

Once they reach the main while loop (line 4), robots read the current highest *bid*, i.e. number of neighbors that is shared via Virtual Stigmergy, using the `VST`.*get_leader_bid*() operation. If the current bid is lower than the one that a current robot has, it will swap its value, along with its identifier in the VST. This process is continuously repeated until a *barrier* is triggered.

In this context, a barrier refers to a mechanism which halts the further execution of the program until all robots reach consensus. In this case, they wait for everyone to have a chance to compare its bid. The barrier is a direct implementation of consensus among robots in the swarm. Essentially, it uses the VST, and a swarm table (a construct which contains the information about all swarm members in a decentralized manner) [138]. Moreover, once the operation `VST`.*leader_barrier*() returns True, the leader election is complete. Afterwards (lines 16–27), if the current robot is the leader robot, it sorts its neighbors based on distance and makes a *robot_list*. Every couple of robots in the list is checked not to be forming a line with the leader by function *check_in_line*. Finally, the leader selects its two closest non-colinear neighbors as references and writes their identifiers into the VST. Also, (lines 28–30) the same robot constructs the common coordinate system with the *create_coordinate_system*(), which

Algorithm 1: Coordinate system initialization phase

input : Virtual stigmergy table (VST)
output: Virtual stigmergy table with robots l, a, b and a coordinate system.

```
1  leader_id, reference_robot(.a, .b) ← ∅;
2  leader_election ← true;
3  bid ← neighbors.count();
4  while leader_election do
5      if bid < VST.get_leader_bid() then
6          leader_id ← VST.get_leader_id();
7      else
8          VST.get_leader_bid() ← bid;
9          VST.get_leader_id() ← id;
10         leader_id ← id;
11     end
12     if VST.leader_barrier() then
13         leader_election ← false
14     end
15 end
16 if id == leader_id then
17     robots_list = neighbors.foreach().sort_by_dist();
18     for robot_i, robot_{i+1} from robots_list do
19         if check_in_line() then
20             robots_list.delete(robot_i)
21         end
22     end
23     reference_robots.a ← robots_list[0];
24     reference_robots.b ← robots_list[1];
25     VST.reference_robots.a ← reference_robots.a;
26     VST.reference_robots.b ← reference_robots.b;
27 end
28 if id == leader_id then
29     create_coordinate_system();
30 end
```

uses the barrier mechanism to wait for all robots to confirm that the system was received. The details on the coordinate system creation are given in the following.

Building a common coordinate system

After electing the leader along and two companion reference robots, we can create a coordinate system. Although our approach was partially inspired by the work of Shirazi and Jin [50], we propose a more intuitive way of creating a common coordinate system, which is performed by the leader robot. Fig. 3.3 also illustrates that the reference robots a and b lie on the x–axis, while the leader robot l lies on a positive y–axis, which is an orthogonal projection on the x axis. We then define the coordinate system using the following equations:

$$\begin{aligned} z_{la}^2 &= x_a^2 + y_l^2 = (d_{la} + v_{la})^2, \\ z_{lb}^2 &= x_b^2 + y_l^2 = (d_{lb} + v_{lb})^2, \\ z_{ab}^2 &= x_a^2 + x_b^2 = (d_{ab} + v_{ab})^2, \end{aligned} \tag{3.1}$$

where z_{la}, z_{lb}, and z_{ab} represent the range measurements from robots l to a, from l to b, and from a to b, respectively. d_{la}, d_{lb}, and d_{ab} represent the true distance between two robots, while v_{la}, v_{lb}, and v_{ab} are the corresponding measurement noise, which are subsequently corrected in the filter. From the above equations, the coordinates of the leader and the two referent robots are solved in the following form; $(0,\ y_l)$, $(x_a,\ 0)$, and $(x_b,\ 0)$, respectively, which are determined by the leader using the following equations:

$$\begin{aligned} x_a &= \frac{1}{2}\left(z_{ab} + \frac{z_{la}^2 - z_{lb}^2}{z_{ab}}\right), \\ x_b &= \frac{1}{2}\left(\frac{z_{la}^2 - z_{lb}^2}{z_{ab}} - z_{ab}\right), \\ y_l &= \sqrt{z_{la} - {x_a}^2} = \sqrt{z_{lb} - {x_b}^2}. \end{aligned} \tag{3.2}$$

The leading robot l shares this information, along with the corresponding robot identifier using the VST. After the propagation of the VST data, all robots share the common coordinate system which is used to calculate their positions and positions of their neighbors. Fig. 3.4 shows how a robot c performs trilateration, i.e. signal intersection from the center points of the leader robot l (dashed red circle), and reference robots a (green dashed circle) and b (orange dashed line).

Only during this initial phase while creating the common coordinate system, we require the robots to be stationary. Afterwards, all robots execute the same EKF localization algorithm

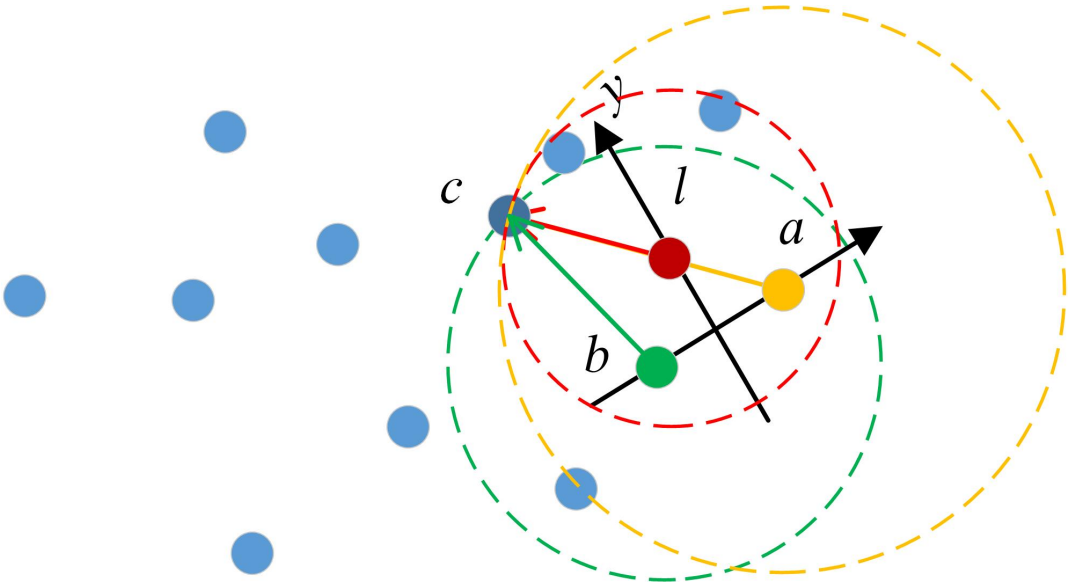


Figure 3.4 Localization of other robots within the coordinate system

and perform collaborative localization.

Comparing to other similar work by Kurazume et al., Corenejo et al, or Shirazi et al. [5,48,50], this is a significant improvement, as methods proposed in their work also require some robots to be static during the entire localization phase. Also, our approach does not require that the first three reference robots cover the distribution of the entire swarm. If the reference nodes are not within UWB range, neighbors who are already localized are used for trilateration, as implemented in our simulations. After the initialization, the robots use EKFs to estimate their positions with UWB measurements from their neighbors.

3.3.2 Stage 2: Localization and Estimation with the EKF

After the initialization is completed, the initial positions of all robots are fed to the EKF to continuously estimate their position. Further details on the EKF is provided in Section 3.4.

This concludes the process of creating a common coordinate system and performing the initial localization. As the robots continue to perform their tasks, the UWB error contributes to the overall localization error of each robot [140], so the following task is to mitigate the error by optimizing the selection of robots that are used to create a new and more suitable coordinate system.

3.3.3 Stage 3: Dynamic Coordinate System Reconfiguration

Having initialized a common coordinate system, in this stage the swarm is able to use their individual positions to mitigate erroneous measurements by the UWB, which are increasing with the distance. In particular, robots are now able to reduce this error by re—electing a new leader and new reference robots, that are roughly equally distant from all other members, i.e. those that are closest to the centroid of their distribution within the operating area, rather than just based on the neighbor density as in the initial stage.

We refer to this stage as the reconfiguration, and it takes place in two cases: immediately after the initialization of the initial coordinate system and b) whenever the distribution of the robots has drastically changed from the previous configuration.

The dynamic reconfiguration of the coordinate system involves three steps: marginal robot selection, reconfiguration triggering, and coordinate system reconfiguration.

Marginal robot selection

While in operation, robots continuously move and consequently their pose estimation changes. Therefore, the members of the multi-robot team need to keep track of their distribution, and in particular the change of this distribution since the previous reconfiguration.

For this purpose, we developed an approach by which robots continuously select marginal robots, i.e. robots closest to the edge of their distribution. This is performed in a similar way as the bidding for the leader. However, in this case, using the VST, robots compare their minimum and maximum x and y coordinates with the goal of determining the top-, bottom-, left-, and right-most robots. Note that we assume that initially all robots are randomly distributed within a 2D Euclidean plane. Therefore, each robot writes its position in VST only if its position is further away than what already written in the VST, hence becoming a *marginal* robot.

The position of marginal robots is then used to calculate the centroid dC of a tetragon bounded by their position. This centroid position is assumed to be the center point of the distribution of robots, and it is used as a candidate point around which the new leader should be found. Such tetragon, along with the centroid, is illustrated in Fig. 3.5. The point cC represents the origin of a current coordinate system.

Reconfiguration request trigger

The robots continuously calculate the distance between the current origin cC and the candidate origin dC, and if its value is beyond a given threshold, it triggers a reconfiguration

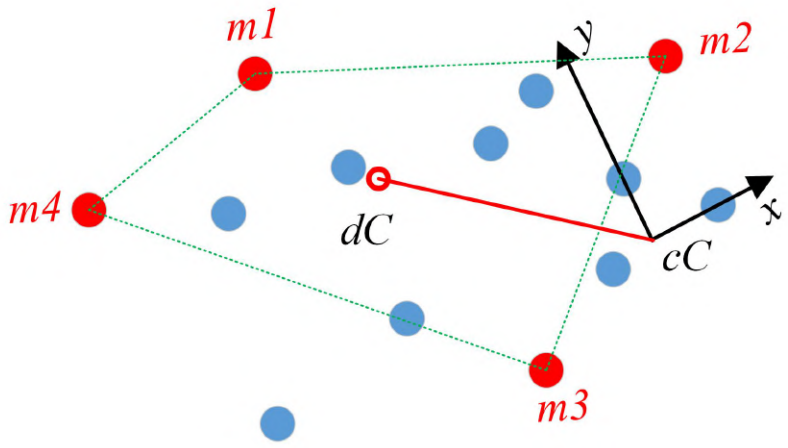


Figure 3.5 Selection of marginal robots (red dots)

request.

Coordinate system reconfiguration

Robots are notified that the reconfiguration is about to take place via the VST and go into the same process as in the initialization step described in Section 3.3.1.

Namely, robots start the selection process for the leader robot l, which is followed by the selection of two new reference robots a and b, as depicted in Fig. 3.6. Note, however, that the leader is selected as the robot closest to the calculated dC, as in this scenario the bidding criterion is the positional information of dC, instead of the neighbor density.

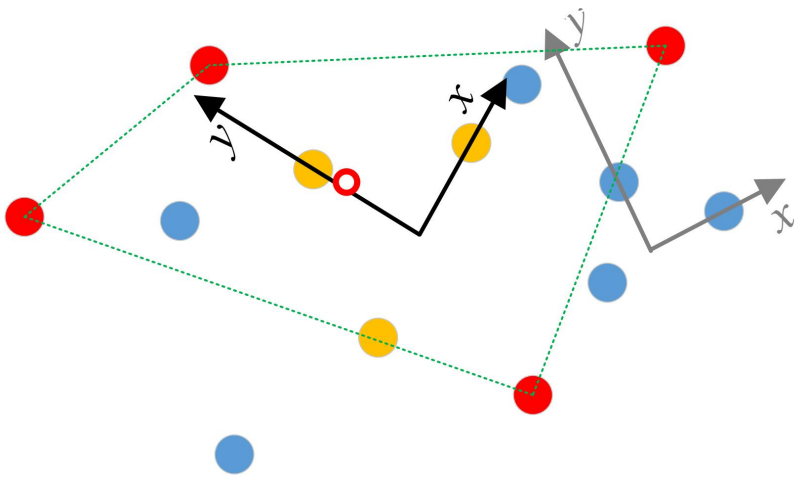


Figure 3.6 Reconfiguration of coordinate system

Furthermore, once the leader and reference robots are selected, the following procedure of coordinate system propagation and continuous localization are the same as previously described in Section 30. Only after the reconfiguration process is complete, robots start to use the new coordinate system, along with the new position within that reference frame.

Algorithm 2 shows the details of the proposed approach for continuous coordinate system reconfiguration. The input of the algorithm is the VST and the *position* of the robot.

Algorithm 2: Coordinate system reconfiguration

input : VST, *position*
output: VST with all robots sharing the new coordinate system

```
1  VST.marg_robots ← ∅ ;
2  reconfiguration_distance ← inf.;
3  new_ref_robots(.leader, .a, .b) ← ∅;
4  while VST.marg_robots_barrier() do
5  |  if compare(position, VST.marg_positions) then
6  |  |  update(VST.marg_robots, position);
7  |  end
8  end
9  VST.marg_centroid = center(VST.marg_robots);
10 if dist(VST.marg_centroid, VST.cur_centroid) > VST.marg_rob.threshold then
11 |  if min(dist(position, VST.marg_rob.centroid)) then
12 |  |  robots_list = neighbors.foreach().sort_by_distance();
13 |  |  new_ref_robots.leader ← robots_list[0];
14 |  |  new_ref_robots.a ← robots_list[1];
15 |  |  new_ref_robots.b ← robots_list[2];
16 |  end
17 |  VST.ref_robots ← new_ref_robots;
18 |  VST.leader_id ← new_ref_robots.leader;
19 |  if id == VST.ref_robots.leader : then
20 |  |  VST.coord_system=create_coord_system();
21 |  end
22 end
```

The algorithm starts by initializing local variables: VST.*marg_robots* which is used to store current marginal robots, and *new_ref_robots*(.*leader*, .*a*, .*b*) which stores the new set of reference robots (i.e. leader and two references). The while loop (lines 4–8) uses the barrier mechanism through which we ensure that all robots reach a consensus on marginal robots. Specifically, all robots compare their positions with the current position of marginal robots using the *compare* operation. This operation returns *True* if and only if the minimum and maximum coordinates are such that a marginal robot needs to be replaced, which is performed by the *update* operation. Once the barrier is reached, the following step (line 9) calculates the centroid of the tetragon bound by the marginal robots, and this value is stored

in VST.*marg_centroid*. After this, all robots are aware of the centroid position and are able to obtain reference robots. If the threshold is triggered (line 10), the closest robot (again agreed through the VST, line 11) obtains the list of all their neighbors sorted by distance (line 12). The first three robots are selected as reference robots (lines 13–15). Finally (lines 17–20), the new coordinate system is shared by the leader through the VST.

The proposed reconfiguration process is running continuously to minimize the erroneous measurements by the UWB minimal on average, with respect to the distribution of robots.

3.4 Dynamic Localization Estimation with EKF

In our system, every robot continuously estimates its location in the coordinate system after initialization or reconfiguration. In the beginning, a robot only needs the initial coordinates provided by trilateration, which is kept updated with an EKF estimator. Other high-order Kalman filters could also be considered as future work, when dealing with very high nonlinearities [141].

In this paper, we assume that a team of mobile robots is operating in a 2D Euclidean plane and the position tracking is based on range-only measurements, which is a marked difference from state-of-the-art scenarios using bearing information. It is reasonable to assume that the initial position of each robot can be accurately computed based on the range measurements in the initial frame of reference [142]. Assuming that robot i can move in 2D with speed v_i, in each time–step it can only move within a circular region centered in its position. Note that since the motion of a target robot can be reliably predicted for the next time step only, our objective is to locate and track the position of robots at consecutive time steps.

3.4.1 State Estimation

Extended Kalman filter

As physical processes and/or observation models are generally nonlinear, we cannot directly apply a linear Kalman filter. To overcome this issue, the standard approach is to use a linearized EKF model, obtained by continuously linearizing models before applying estimation techniques [143–145].

Let $\mathbf{x}_i(k) = [x_i(k)\ \dot{x}_i(k)\ y_i(k)\ \dot{y}_i(k)]^T$ be the i-th robot state at time t_k. $(x_i(k), y_i(k))$ is the i-th robot position in Cartesian coordinates and the dot notation indicates differentiation with respect to time. Considering a random walk, the motion model can be given in the following form:

$$\begin{aligned} \mathbf{x}_i(k) &= \mathbf{F}_k \mathbf{x}_i(k-1) + \mathbf{w}_k, \\ \mathbf{F}_k &= \mathbf{I}_2 \otimes \begin{bmatrix} 1 & T_k \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{3.3}$$

where $\mathbf{w}_k$ is the process noise, which we assume white Gaussian, $T_k = t_k - t_{k-1}$, $\mathbf{I}_2$ is a 2×2 identity matrix, and $\otimes$ denotes the Kronecker product. At time t_k, a robot produces range-only measurements to reference robots.

In most practical navigation applications, a reference trajectory does not exist beforehand. Therefore, the EKF uses the current estimated state at each time step k as a linearization point. If the filter operates properly, the linearization error around the estimated solution can be maintained at a reasonably small value [145, 146].

Generally, the EKF algorithm can be described as:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= F_k \hat{\mathbf{x}}_{k-1}, \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_{k-1} \mathbf{P}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{G}_{k-1} \mathbf{Q}_{k-1} \mathbf{G}_{k-1}^T, \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^T \right)^{-1}, \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h} \left(\hat{\mathbf{x}}_{k|k-1}, 0 \right) \right), \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_{k|k-1}, \end{aligned} \tag{3.4}$$

where $\hat{\cdot}$ stands for an estimate, $\mathbf{P}$ is the state covariance matrix, $\mathbf{Q}$ the process noise covariance matrix, $\mathbf{R}$ the measurement noise covariance matrix, $\mathbf{F}$ the state transition matrix, $\mathbf{K}$ the Kalman gain, $\mathbf{H}$ the observation matrix, $\mathbf{G}$ the Jacobian matrix with respect to process noise, and $\mathbf{M}$ the Jacobian matrix with respect to measurement noise. Note that the state-transition function is linear, and the observation function $\mathbf{h}(\cdot)$ is non–linear.

3.4.2 Measurement Model

At t_{k+1}, robot-i measures its range $d_j(k+1)$ to the reference robots, $j = 1, \cdots, N$, where N is the number of reference robots. Therefore the measurement equation is:

$$\mathbf{z}(k+1) = \begin{bmatrix} d_1(k+1) \\ \vdots \\ d_N(k+1) \end{bmatrix} + \begin{bmatrix} \mathrm{w}_{d_1}(k+1) \\ \vdots \\ \mathrm{w}_{d_N}(k+1) \end{bmatrix}, i = 1, \cdots, N. \tag{3.5}$$

$$d_j(k+1) = \sqrt{\Delta x_{i,j}^2(k+1) + \Delta y_{i,j}^2(k+1)}, \tag{3.6}$$

where $\Delta x_{i,j}(k+1) = x_i(k+1) - x_j(k+1)$ and $\Delta y_{i,j}(k+1) = y_i(k+1) - y_j(k+1)$ are the relative positions of i–th and the j–th reference robot, respectively, expressed in current coordinates. Note also that

$$\mathbf{w}(k+1) = [\mathrm{w}_{d_1}(k+1), \cdots, \mathrm{w}_{d_N}(k+1)]^T \tag{3.7}$$

is the noise in robot i's measurements, which we assume zero–mean white Gaussian noise with covariance

$$R_i(k+1) = E[\mathrm{w}_{d_i}(k+1)\mathrm{w}_{d_i}^T(k+1)] = diag(\delta_{d_i}^2) \tag{3.8}$$

and

$$\mathbf{R}(k+1) = E[\mathbf{w}(k+1)\mathbf{w}^T(k+1)] = diag(R_i(k+1)). \tag{3.9}$$

The measurement Equation (3.5) is a nonlinear function of the state variable $\mathbf{x}_i$. The measurement-error equation, obtained by linearizing Eq. 3.4 is

$$\begin{aligned} \tilde{\mathbf{z}}(k+1|k) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k) \\ &\approx H_{k+1}\tilde{\mathbf{x}}_i(k+1|k) + \mathbf{w}(k+1), \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} \hat{\mathbf{z}}(k+1|k) &= [\hat{z}_1^T(k+1|k), \hat{z}_2^T(k+1|k), \cdots, \hat{z}_N^T(k+1|k)]^T \\ \hat{z}_i^T(k+1|k) &= [\hat{d}_i(k+1|k)]^T \\ \hat{d}_j(k+1|k) &= \sqrt{\Delta\hat{x}_{i,j}^2(k+1|k) + \Delta\hat{y}_{i,j}^2(k+1|k)} \\ \Delta\hat{x}_{i,j}(k+1) &= \hat{x}_i(k+1|k) - x_j(k+1) \\ \Delta\hat{y}_{i,j}(k+1) &= \hat{y}_i(k+1|k) - y_j(k+1) \\ \tilde{\mathbf{x}}_i(k+1|k) &= \mathbf{x}_i(k+1) - \hat{\mathbf{x}}_i(k+1|k). \end{aligned}$$

Note that the Jacobian matrix of measurement is given by the following expression:

$$\begin{aligned} H_{d_j,k+1} &= \frac{\partial \mathbf{h}_{d_j,k+1}}{\partial \mathbf{x}_{k+1}}, \\ \mathbf{h}_{d_j,k+1} &= [d_1(k+1), \cdots, d_N(k+1)]. \end{aligned}$$

One can observe that the measurement Equation (3.6) is nonlinear and its first derivative exist, which justifies the use of the EKF.

3.5 Simulation Results

This section demonstrates and evaluates the proposed approach separately, with ARGoS [147] and Matlab simulations. Matlab was mainly used as a proof–of–concept, while ARGoS is a multi-physics simulator, which provides a concrete step towards the implementation of the algorithm on a team of robots. In addition, ARGoS natively supports Buzz which, as previously mentioned, implements the VST that is essential for this work.

3.5.1 Simulations of System Initialization and Reconfiguration

Consider a swarm of robots as shown in Fig. 3.7. Here, robots are involved in the bidding

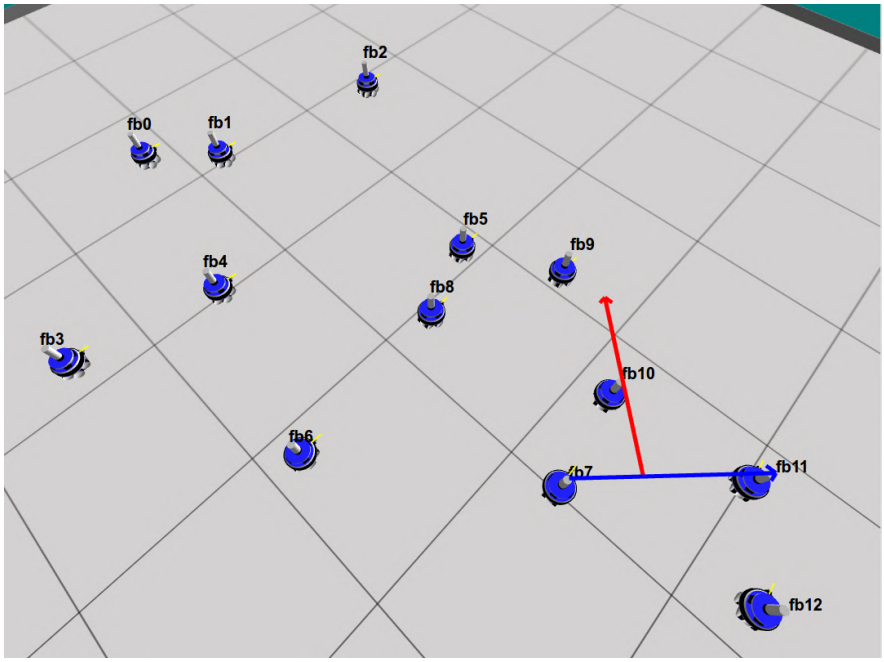


Figure 3.7 System initialization stage: three robots are selected as reference robots to build a coordinate system.

process to select a leader robot. Each robot has a range within which it can detect its neighbors, and for this simulation it is set to 2 m (one-floor rectangle represents one meter). All robots apply Algorithm 1 and achieve the consensus that the robot `fb10` is the initial leader. According to the proposed algorithm, its two closest neighbors, robots `fb7` and `fb11` are selected as reference robots which it uses to create the initial coordinate system (which is also illustrated in Fig. 3.7).

Suppose that after the initialization, the reconfiguration process is triggered and a new coordinate system needs to be constructed and shared with all robots. Since in this phase,

all robots are aware of their positions in a previously constructed coordinate system, four marginal robots are selected via distributed consensus. As shown in Fig. 3.8 these are `fb0, fb2, fb11, fb12`, which together form a tetragon.

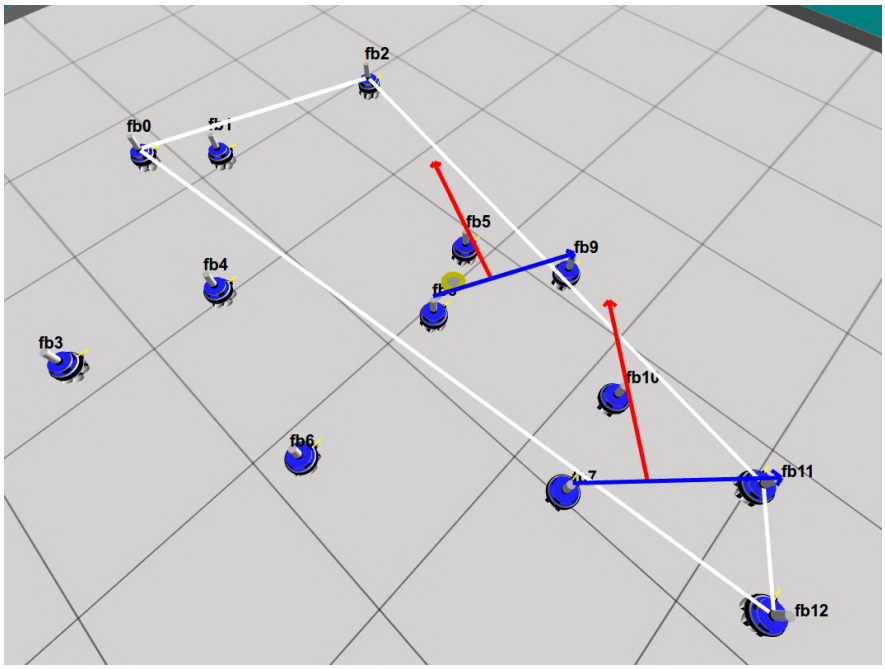


Figure 3.8 System reconfiguration stage: a new coordinate system is built with the help of four marginal robots.

The centroid of this tetragon is marked with a green circle, and the robot nearest to this point is selected as a new leader, i.e. robot `fb5`. Using the same strategy as in the initialization stage, robots `fb8` and `fb9` become reference robots, which together form a new coordinate system. Once this coordinate system is propagated to all the robots within the multi-robot team, the old coordinate system is dismissed.

3.5.2 Validation of Dynamic Localization

As presented in the previous sections, the EKF estimator can be applied for dynamic localization. In this subsection, this approach is validated with Matlab, having the following assumptions: a) all robots move randomly, b) the measurements are associated with the white Gaussian noise to simulate the noise of the real sensor, and finally c) the entire simulation (including measurements) are updated at 10 Hz.

In a first experiment, we place four robots randomly in an area of 40×40 m^2. The ground-truth trajectory of each robot, and its estimation are shown in Fig. 3.9, while the localization error and covariance matrix determinant of the EKF are given in Fig. 3.10. To statistically

evaluate our results, we run 30 simulations and acquire around 72000 estimations. We divide the localization error into 25 0.1-meter bins and we present the histogram of the error in Fig. 3.11 and note that for more than half of our samples, the error falls into the bins around 0.3 meters.

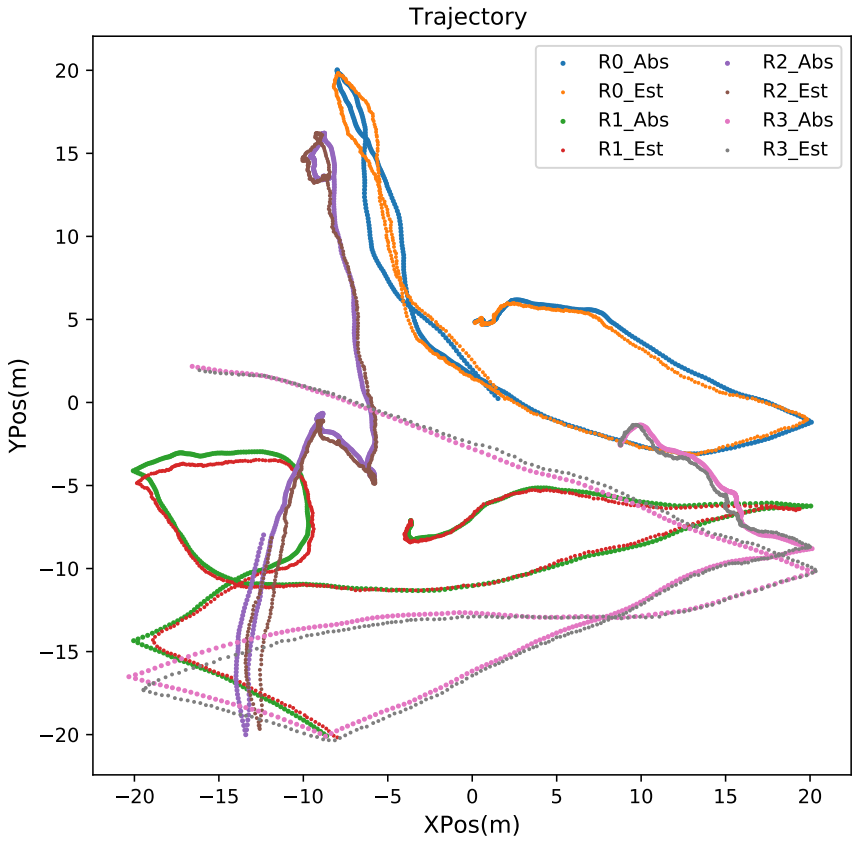


Figure 3.9 Localization of four robots with the EKF.

In our simulation, we assume that the range measurements arrive simultaneously at each time step. However, in a distributed system, it is very challenging to achieve such synchronization. This was analyzed by many researchers to provide performance limits [148] and solutions [149]. Other approaches, such as Unscented Kalman Filters, can also be applied to reduce the impact of poor synchronization. Since all of these approaches can be applied in our system to improve the estimation accuracy and mitigate the impact of asynchronism, we plan to address the issue in future work.

3.6 Conclusion

This work presents a dynamic localization approach based on range-only UWB sensors. By using a bio–inspired decentralized consensus technique, our approach leverages the commu-

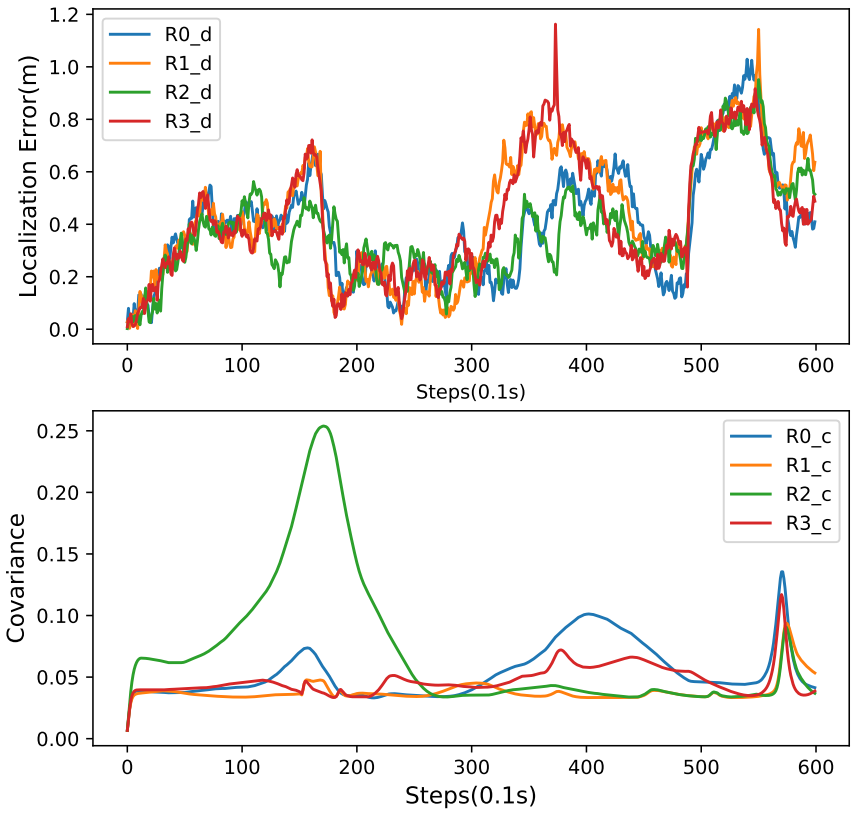


Figure 3.10 Localization error and covariance trend.

nication between robots to build a common coordinate system which allows for localization within an unknown region, with only three robots. Therefore, to the best of our knowledge, the presented work offers an advantage to existing approaches which require static robots which are used for localization. Furthermore, we developed an algorithm used for the initialization, and an additional algorithm which is used for the reconfiguration phase. During the initialization, for a brief time instance, robots which are selected by consensus are static, create a common coordinate system and propagate it to all other members of a multi-robot team. After this, a second algorithm, continuously, dynamically and automatically improves the localization by re–initializing the common coordinate system. The great advantage of this approach is that in the localization phase, it doesn't require robots to be static as it uses the EKF for pose estimation.

Through simulations in Matlab and ARGoS, we have validated and demonstrated that the EKF can be successfully applied in for the aforementioned purpose.

We plan further investigation into the error characteristics of UWB and robot formations to extend the presented method to support a variety of robot distributions. We believe we

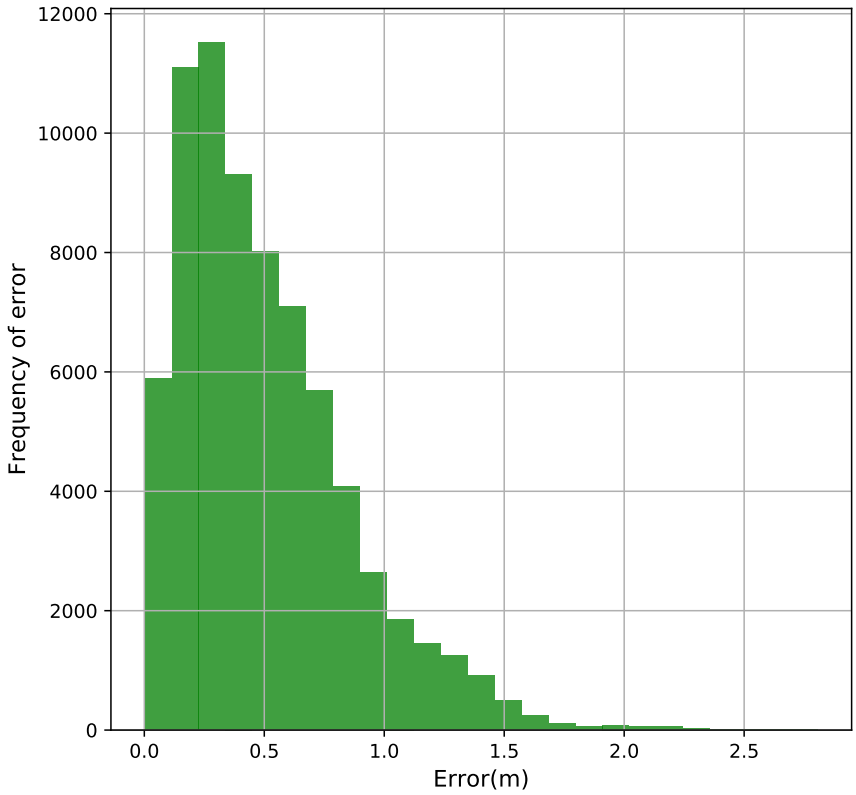


Figure 3.11 Histogram of the localization error over thirty experimental runs.

can achieve this by selecting a more appropriate algorithm to determine the marginal robots, as the current approach can be greatly improved from the standpoint of the necessary data exchange. Also, our goal is to implement the method on physical robots and consider different dynamic and complex environments.

3.7 Acknowledgements

We would like to thank Hamza Benzerrouk for the discussions on the EKF model, and Vivek Shankar Varadharajan and Jacopo Panerati, for their help with the ARGoS simulator and the Buzz programming language.

Errata:

1. The third equation of Equ 3.1, "$z_{ab}^2 = x_a^2 + x_b^2 = (d_{ab} + v_{ab})^2$" is replaced by "$z_{ab}^2 = (x_a - x_b)^2 = (d_{ab} + v_{ab})^2$";
2. The third equation of Equ 3.2, "$y_l = \sqrt{z_{la} - {x_a}^2} = \sqrt{z_{lb} - {x_b}^2}$" is replaced by "$y_l = \sqrt{z_{la}^2 - {x_a}^2} = \sqrt{z_{lb}^2 - {x_b}^2}$".
3. In page 37, paragraph 6, the "Considering a random walk" is replaced by "Considering a constant velocity model".
4. In fifth line of Equ 3.4, the "$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}, 0\right)\right)$" is replaced by "$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}\right)\right)$".

CHAPTER 4 ARTICLE 2: DISTRIBUTED TDMA FOR MOBILE UWB NETWORK LOCALIZATION SERVICE

Preface: To apply the system proposed in last chapter into real hardware, one serious challenge posed by available commercialized UWB system is the control of the network. MAC required in a distributed localization system require high channel usage, dynamic scheduling, and scalable performance. In this chapter, we propose a novel TDMA algorithm that well supports UWB shared medium control for the localization purpose. Combining the initialization stage in previous chapter, a real UWB based localization system is presented with hardware experiments.

Authors: Yanjun Cao, Chao Chen, David St-Onge, and Giovanni Beltrame

Submitted to: Internet of Things Journal, IEEE

Personal Contributions: Y. Cao conceived of the presented ideas, developed the theory, designed and carried out simulations and real hardware experiments. Y. Cao wrote the manuscript with support from G. Beltrame and Chao Chen.

Abstract

Many applications related to the Internet-of-Things, such as tracking people or objects, robotics, and monitoring require localization of large networks of devices in dynamic, GPS-denied environments. Ultra-WideBand (UWB) technology is a common choice because of its precise ranging capability. However, allowing access and effective use of the shared UWB medium with a constantly changing set of devices faces some particular challenges: high frequency of ranging measurements by the devices to improve system accuracy; network topology changes requiring rapid adaptation; and decentralized operation to avoid single points of failure.

In this paper, we propose a novel Time Division Multiple Access (TDMA) algorithm that can quickly schedule the use of the UWB medium by a large network of devices without collisions in local network neighborhoods and avoiding conflicts with hidden terminals, all the while maximizing network usage. Using exclusively the UWB radio network, we realize a decentralized system for synchronization, dynamic TDMA scheduling, and precise relative positioning on a multi-hop network. Our system does not have special nodes (all nodes are equal) and it is sufficiently scalable for real-world applications. Our method can be applied to implement device localization services in a large spaces without GPS and complex topologies, like malls, museums, mines, etc. We demonstrate our method in simulation and

on real hardware in an underground parking lot, showing the effectiveness of its TDMA schedule for relative localization.

4.1 Introduction

Indoor localization and tracking have the potential to unlock a plethora of new concepts and applications for public space enhancement [150].

The rapid development of hardware and software technology needs to address a higher demand for accurate localization services, such as person tracking and device localization. In particular, a scalable system with few or no dependencies on fixed infrastructure is highly desirable, allowing for faster deployment and reduced effort for users.

While many paths have been explored, it is still very challenging to deploy a scalable system with mobile and dynamic nodes [59]. Accurate tracking can be acquired with expensive sensors (e.g. multiple cameras, LiDARs, radars, etc.), but their cost limits their applicability for the Internet-of-Things (IoT). Building infrastructure support to provide localizationd services is the most common approach: satellite-based GPS or from cellular systems (5G, LoRa) are mostly for outdoor use and can have low localization accuracy depending on the environment. Camera arrays and radio beacon setups can provide indoor localization, but can be expensive and labor-intensive to set up, limiting their scalability.

EM-based localization is usually lower cost and more suitable for IoT devices. Wifi, Bluetooth, and RFID have respective techniques for indoor positioning, however with limited accuracy [7]. Ultra-Wide Band (UWB) is also used for localization, using ultrashort pulses to achieve very accurate and high-frequency ranging as well as transmitting data, which makes it an ideal choice for indoor localization.

Anchor-based UWB localization has been widely commercialized [15, 16]. Commercial systems mainly use Time Difference of Arrival (TDOA), a technique which requires nanosecond clock synchronization between anchors [17]. This limits the scalability of the system as a high level of synchronization accuracy is hard to maintain in distributed system. In addition, the positions of agents are usually calculated by a server in centralized fashioned, requiring an available low-latency communication infrastructure. To maximize scalability, we target a distributed system with minimal infrastructure support.

Two way ranging (TWR) [18] is a more flexible ranging solution, enabling arbitrary pairs nodes to perform distance measurements at any time. The key issue with TWR is the access to a shared medium, i.e. the UWB communication channel. Ranging using TWR takes

significantly longer than simply broadcasting a message, requiring a medium access control (MAC) mechanism to coordinate the measurements across all devices at a given location. Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA) are general strategy for MAC in many networks [19]. TDMA allows several devices to share the UWB channel by dividing the medium access into different time slots. For accurate localization, we want to maximize the number of ranging measurements, therefore requiring high utilization of the available time slots.

Our requirement of high channel usage for the system prevents the use of a contention-based TDMA [65] as the potential of collisions is very high. Another major category of distributed TDMA algorithms is based on request and response negotiations [70]. However, these techniques usually require several iterations to converge, which makes them difficult to apply in dynamic networks where the topology changes frequently. In this paper, we propose a novel distributed TDMA algorithm that allows conflict-free scheduling with almost full channel usage. In addition, our method supports multi-hop networks, where slots can be multiplexed and used by nodes that are two hops apart. Another major advantage of our method is that scheduling can be obtained and updated iteratively, making it suitable for dynamic networks of mobile devices. Without loss of generality we apply our TDMA algorithm to a UWB network.

The mobility of devices poses challenges not only for TDMA scheduling, but also for the localization of nodes in the network. Cooperative localization can help improve the localization performance for a large scale network [44]. In our system, each node cooperates with its neighbors to construct a local topological map, which is broadcast to recover the global map of the network, allowing for the localization of multi-hop device network.

Our contributions can be summarized as:

- a scalable decentralized system for relative localization in mobile ad-hoc networks;
- a novel TDMA algorithm that allows fast convergence and full channel usage;
- a neighborhood topological map construction algorithm for global cooperative global map recovery.

4.2 Related Work

UWB technology has attracted substantial attention due to its high ranging accuracy. Beside the centralized usage of anchor-based systems [54,55], point-to-point ranging started receiving more attention [56–59]. These results show the advantage of UWB for multi-agent systems,

but due to the small number of UWB nodes in the experiments, network usage has not yet been explored. Ridolfi et al. [60] analyzed the scalability of UWB-based localization and showed the huge impact of the coordination protocol on scalability. Qin et al. [17] designed a BLAS system that uses UWB for the localization of a multi-agent system. They divided the agents into parent and child groups: the parent agents act as moving anchors and create a coordinate frame for the system. They proposed a distributed clock synchronization in parent agents to maintain a high clock precision. This enables the child group to get the position by time-of-arrival (TOA) measurements. The biggest advantage of this system is the number of child agents is theoretically unlimited because they only passively receive the pings from the parents. Although the child nodes have no conflicts, the communication between parent agents still needs to be coordinated. Qin et al. use round-robin as distributed TDMA to achieve collision-free broadcasting from parent agents. However, this solution assumes all the parent agents are fully connected, which limits the scalability of the system. Macoir et al. [61] also uses an anchor-based TDOA strategy, but it is designed for relatively large scale networks. The authors divided the network into multiple small cells to cover large areas. Unlike the passive child nodes from [17], Macoir et al [61] schedule active slots for mobile tags to broadcast messages and use a server connected with the anchors to calculate the positions. The server is also responsible for slot assignment and thereby forming a centralized system.

Zhu and Kia [62] proposed a negotiation-based dynamic TDMA algorithm across the UWB network, G.M. ter Horst [59] developed an anarchic TDMA algorithm based on the DESYNC algorithm [63]. Both methods only consider one-hop collisions, so that collision can occur for hidden nodes at neighborhood boundaries.

The most widely used MAC control on UWB is the IEEE802.15.4-2011 protocol [64], integrated with the Decawave 1000 [18] chip, which is the most popular commercial UWB chip on the market. The protocol uses Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) or slotted ALOHA [65, 66] to avoid collisions. However, these two strategies are only applicable to lightly loaded networks that have a small probability of collisions. In our case, we want to maximize channel usage for accurate localization. To the best of our knowledge, this work is the first to apply dynamic TDMA for UWB localization in multi-hop ad-hoc networks with mobile devices.

Compared to the some TDMA techniques available for communication networks, our application has several additional requirements: 1. the maximization of channel usage to increase the localization frequency; 2. rapid time slot scheduling to account for dynamic topologies; 3. decentralization to avoid the need for fixed infrastructure.

The TDMA slot assignment in a wireless network can be seen as an extension of the vertex colouring problem in graph theory, with the additional constraint of needing to avoid collisions in 2-hop neighborhoods [67]. The problem was proven to be NP-complete [68], and several heuristic solutions were proposed [67–69] to get the near-optimal results with full knowledge of the network topology.

When considering a distributed system where nodes only receive messages from neighbors, some works (FPRP [70], DRAND [71], DICSA [72], PCP-TDMA [73]) propose negotiation-based algorithms to find the smallest frame with conflict-free scheduling. After scheduling, all nodes in a network need to agree on the same frame size to execute the slot schedule with the same frame reference. We believe that this strategy is not the best strategy for a highly dynamic network, as it would require fast global consensus of the frame size across the network. Furthermore, when a node makes a proposal during negotiation, it has to wait for the feedback of all neighbors, potentially leading to long convergence times if the neighborhood size is large or the network topology is complex. For similar reasons, practical mobile networks use fixed-length frames such as USAP [74] for military telephone networks and VeMAC [75] for vehicular ad-hoc networks (VANETs).

Since the size of the frame should be larger than the total number of nodes in the network, these protocols may have several unused time slots. Researchers found a balance between frame size stability and channel usage by doubling or halving the frame size, such as in USAP-MA [76] and Dynamic-TDMA [77]. Cao and Lee proposed VAT-MAC [78], a VANET with a changing frame size, relying on a roadside unit (RSU) infrastructure. In our method, we use a fixed frame size that equals the total number of nodes allowed in the system. Despite this static allocation, we have high channel usage since we always allocate all the available time slots to the neighborhood's devices.

The overall goal of our system is to localize nodes within the network. We use least square optimization to find the coordinate of the nodes by carefully selecting reference nodes to do trilateration [151] and iterative multilateration [27, 152] with located nodes. However, considering the distributed and dynamic characteristics of our network, applying two-way ranging between all nodes in the network in real-time can be challenging [41,153]. Therefore, we propose a two-stage strategy: we create a local relative localization map and then merge the local maps to get the whole network spatial information.

4.3 System Overview

With the goal of accurate localization in a dynamic UWB ad-hoc network, we developed a fully decentralized system that does not require any fixed infrastructure. By designing a novel MAC protocol, nodes can make full use of the channel to get ranging measurements across the network. Our system includes four main modules: synchronization (to align the frame boundaries), distributed TDMA (to get collision-free slot assignments), relative localization, and global map merging.

Synchronization and distributed TDMA are related to the MAC protocol of the UWB network, while relative localization and global map merging combine the devices' ranging measurements and are executed in the time slots assigned by the MAC protocol.

Fig. 4.1 shows the system architecture from the perspective of a node with ID i. All nodes are identified by a unique ID, are considered equal in the network, and run the same software. The overall localization runs in a cycle with n time slots, where n is the total number of nodes in the network. This choice is without loss of generality: this number is usually known for a given application, and not all devices need to be active or present.

Each robot is automatically assigned the slot with the same ID as the robot. At the beginning of each frame, each node checks its synchronization with the node with the lowest ID in range, ensuring that all frames start at the same time.

All the time slots in a frame can be divided into: own time slot, extra time slots, and passive time slots. The purpose of distributed TDMA is to find conflict-free extra slots to improve node's localization capability. Every time a device is in its own slot (predefined as the ID of the device), the node broadcasts TDMA scheduling packets. Nodes listen to the network during passive slots.

Finally the devices use relative localization to build a topological map of their neighborhood. Distributed TDMA helps nodes to gain extra slots, which enables them to do ranging and broadcast these measurements. Each node constructs the local map from its own ranging measurements as well as the measurements received from its neighborhood. Each node broadcasts its local map and merges the other nodes' maps when received, progressively converging to the global map.

4.4 Frame-Based Synchronization

Synchronization is usually needed for systems with pure TDMA scheduling [154]: nodes need to have the same clock to wait for the time of their own slot. Considering that the network

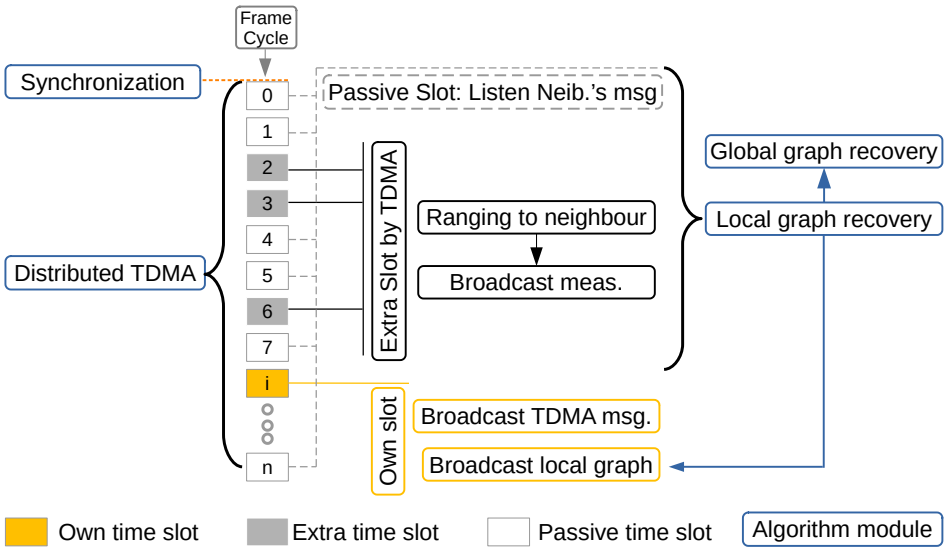


Figure 4.1 System architecture shown for a node with ID i. Each frame includes n time slot and runs in a cycle. Synchronization is checked at the beginning of each frame period to ensure the same start time for the whole neighborhood. All the slots in a frame can be divided into own time slot, extra time slots, and passive time slots. In its own slot (slot i for node i), a node broadcasts TDMA scheduling packets, and listen to neighbors in passive slots. As a result of scheduling, a node can be assigned extra slots that are used for ranging and broadcasting range measurements. Each node constructs a local node map from its own ranging measurements as well as the measurements received from its neighborhood. Once the map is stable, it is also broadcast. When a node receives a map, it can be merged into a global map if the nodes have common neighbors.

might be large and highly dynamic, it is not easy to effectively synchronize the entire system. In our case, we let the nodes synchronize the beginning of each frame instead of the true clock. We use two techniques to achieve distributed synchronization:

- Each node broadcasts the time offset to the starting of its frame;
- Each node listens to the offset of its neighbors and adjusts its offset using the neighbor with the smallest ID as reference.

The data usage used by transmitting an offset is less or at least not greater than for transmitting the clock. Using a reference (the lowest ID node in the neighborhood in our case) can also improve the synchronization speed with respect to a peer-to-peer gradient time synchronization setup [155].

4.5 Distributed TDMA

As long as all nodes in a neighborhood have the same frame start reference, the slot ID can be used as an index of the unique time period for controlling the access to the shared medium. Nodes also collect information from two-hop neighbors to avoid conflicts at the interface of the neighborhood. Many TDMA algorithms [156,157] use two-hop neighborhood information to assign unallocated slots: our main contribution is a strategy to quickly allocate all these free slots with minimal conflicts in a distributed manner. To describe our algorithm, we need some definitions:

- $U = \{u | u \in \mathbb{N}, 1 \leq u \leq n\}$: the set of node IDs, with n the largest ID in the system;
- N_i^k: the *k-hop neighbor set*, consisting of the IDs of nodes exactly at k hops from node i;
- aN_i^k: the *all neighbor set*, consisting of the IDs of nodes at most at k hops from node i:
$$aN_i^k = \bigcup_{j=1}^{k} N_i^j;$$
- sS_i: the *send slots set*, consisting of the IDs of slots assigned to node i;
- $fS_i = U \setminus sS_i$: the *free slots set*, consisting of the IDs of slots that are not assigned to node i;
- cS_i: the *candidate slots set*, consisting of the IDs of slots that are not assigned to any node in N_i^2:
$$cS_i = \bigcap_{j \in aN_i^2} fS_j;$$
- sN_i: the *sibling neighbor set*, consisting of the IDs of nodes sharing the same candidate slot sets in 2-hop range of node i:
$$sN_i : \{j | cS_j = cS_i, \forall j \in aN_i^2, i \neq j\};$$
- $shdS_i$: the *shared slots set*, consisting of the IDs of shared candidate slots of node i that are within 2 hops but 1. are not in sN_i 2. are in cS_j that is not a superset of cS_i:
$$\Omega : \{k | k \in (aN_i^2 \setminus sN_i), cS_k \not\supset cS_i\}$$

$$shdS_i = \bigcap_{j \in \Omega} cS_j;$$

4.5.1 Frame Structure

Each frame contains two cycles of communication slots, one for immediate neighbors, and one for two-hop neighbors. The number of slots in each cycle is equal to the maximum number of nodes allowed in the system. This represents the lower bound of the frame size if all nodes are to communicate in a fully connected network [68]. We believe that this constraint is not critical as our system can assign arbitrarily large numbers of slots to each device and fully use the available bandwidth, as well as guaranteeing that all devices get at least one communication slot.

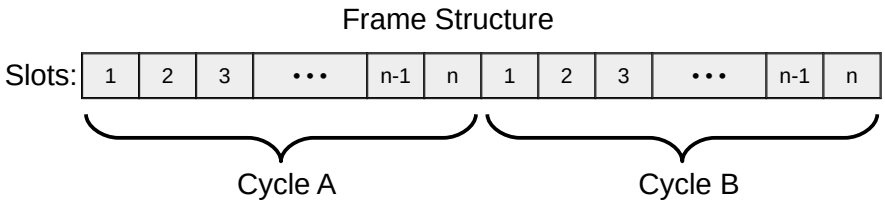


Figure 4.2 The frame structure of the system. Each frame includes two cycles of slots from 1 to n. n is the maximum number of nodes allowed in the system.

As shown in Fig. 4.2, each frame has a two-cycle structure. Each cycle has n slots, with n is the maximum number of nodes that the system can support.

By default in each cycle, node i takes slot i, i.e. the slot with the same ID (called own slot). This ensures a lower bound of two slots per frame when all nodes share the same neighborhood, meaning we have a fully connected network. When a node moves and leaves some other nodes' communication range, certain time slots become idle. When idle slots are detected, the nodes dynamically assign these slots to improve their update frequency, as described in Section 4.5.3.

The reason why we use two cycles (A and B) in each frame is to have a stable propagation of information of 2-hop nodes. Each node broadcasts its own ID, used and potential slots in its slot of cycle A, and the information of its neighbors in its slot of cycle B (see Fig. 4.3). Using this two-cycle broadcasting, each node acquires the latest two-hop neighborhood information at each frame. This information is fed into the distributed TDMA scheduler for conflict-free scheduling. As the slots schedule for two cycles are the same, we explain frame as n slot for simplicity.

4.5.2 Data Packet Format

We use different packet formats for the own slots in cycle A and B, as well as for extra slots. For own slots in cycle A the broadcast packet contains node ID, candidate slots (cS), and send slots (sS). Every node reached by the broadcast collects the information in its local memory. For own slots in cycle B, the packets contain every neighbor's cS and sS, effectively propagating 2-hop information across the network.

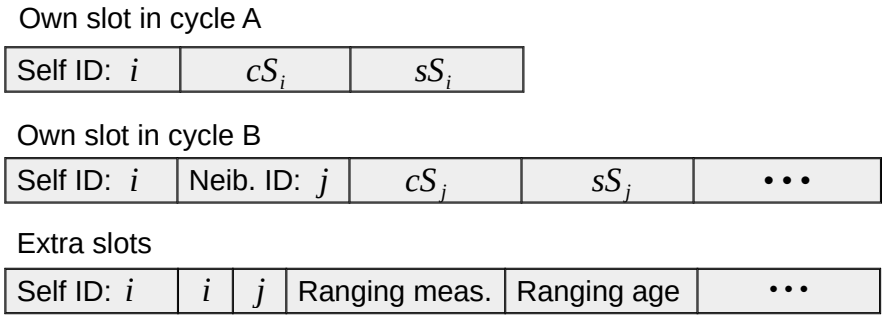


Figure 4.3 Communication data packet format used in three types of time slots in the system

The extra slots are used for ranging with neighbors. After the ranging task is completed, each node broadcasts a timestamped ranging result. This transmission avoids repeating the same measurement from two directions and provides global knowledge of inter-node distance across a neighborhood.

4.5.3 Scheduling

With the goal to improve channel usage through rapid scheduling, we propose an algorithm that solves the scheduling problem in small number of iterations. By listening to the data broadcast by neighbors in their own time slots, each node knows the free slots it can take over. The key idea behind how we solve the assignment is to first find the unique free slots set for the node itself, and then evenly distribute the free slots with neighbors. As shown in Fig. 4.4 where two nodes i and j have candidate slots cS_i and cS_j respectively. The idea is that i takes $\{e|e \in cS_i, e \notin cS_j\}$; j takes $\{e|e \in cS_j, e \notin cS_i\}$, and then i and j evenly share the remainder $\{e|e \in cS_i, e \in cS_j\}$.

However, when considering a real deployment with many nodes in a complex network topology, this is difficult to achieve. We propose Alg. 3, that can safely and quickly complete the allocation of all free slots. We show a complex but representative example of four nodes with four different candidate time slots in Fig. 4.6.

The input of the algorithm is the received packets from all neighbors. The packets include

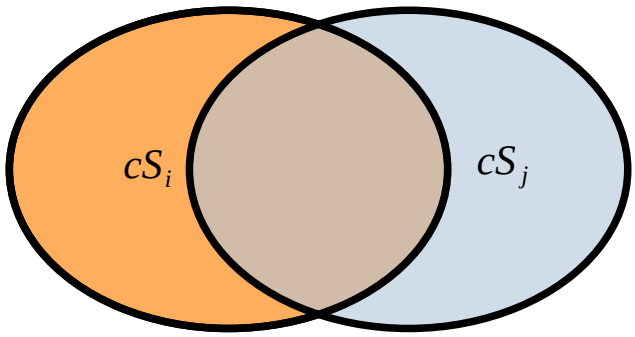


Figure 4.4 The key idea of how we solve the assignment is to first find the unique free slots set of the node itself, and then evenly distribute the shared free slots with neighbors. Two ellipses stand for the free candidate time slots cS_i and cS_j for nodes i and j respectively. The idea is i takes $\{e|e \in cS_i, e \notin cS_j\}$; j takes $\{e|e \in cS_j, e \notin cS_i\}$, then i and j evenly share $\{e|e \in cS_i, e \in cS_j\}$.

cS and sS for all neighbors in 2-hop range. As described in Section 4.5.1, this information is guaranteed to be received in one frame. This scheduling algorithm is executed once at the start of every frame based on the information received in last frame.

The algorithm has two key concepts among all definitions above: the *sibling neighbor set* (sN_i) and *shared free slots* ($shdS_i$). sN_i of a node i contains the neighbors within 2 hops which have the same candidate slots as as i (as previously defined). One example of sibling neighbors are nodes placed in close proximity that share the same neighbors and 2-hop neighbors. $shdS_i$ are slots to be assigned, either shared between sibling neighbors or to i.

The process is formalized by Algorithm 3, which is also represented graphically in Fig. 4.5. This algorithm is fully decentralized, and it is executed by every node independently (for this reason, Algorithm 3 omits the subscript notation for the sets). After listening for neighbors (line 1 in Alg. 3) for one frame, each node stores the received messages, including candidate slots, send slots of its 1-hop and 2-hop neighbors.

Fig. 4.5 shows the cS sets of four nodes (n_i, n_j, n_k, n_p) as ellipses, marking the possible intersections between the ellipses with letters from A to G. Each intersection set represents a set of time slots.

We call this initial state of the system *step 0*. Alg. 3 proceeds with the initialization of some variables, including the creation of an empty set of sibling neighbors (sN, line 2), a candidate slots set (cS, line 3) initialized with all slots from 1 to n (with n the total number of nodes in the network), and a set that includes all 1-hop and 2-hop neighbors (aN^2, line 4). Then the cS is updated by removing all the send slots (sS) of its neighbors and 2-hop neighbors.

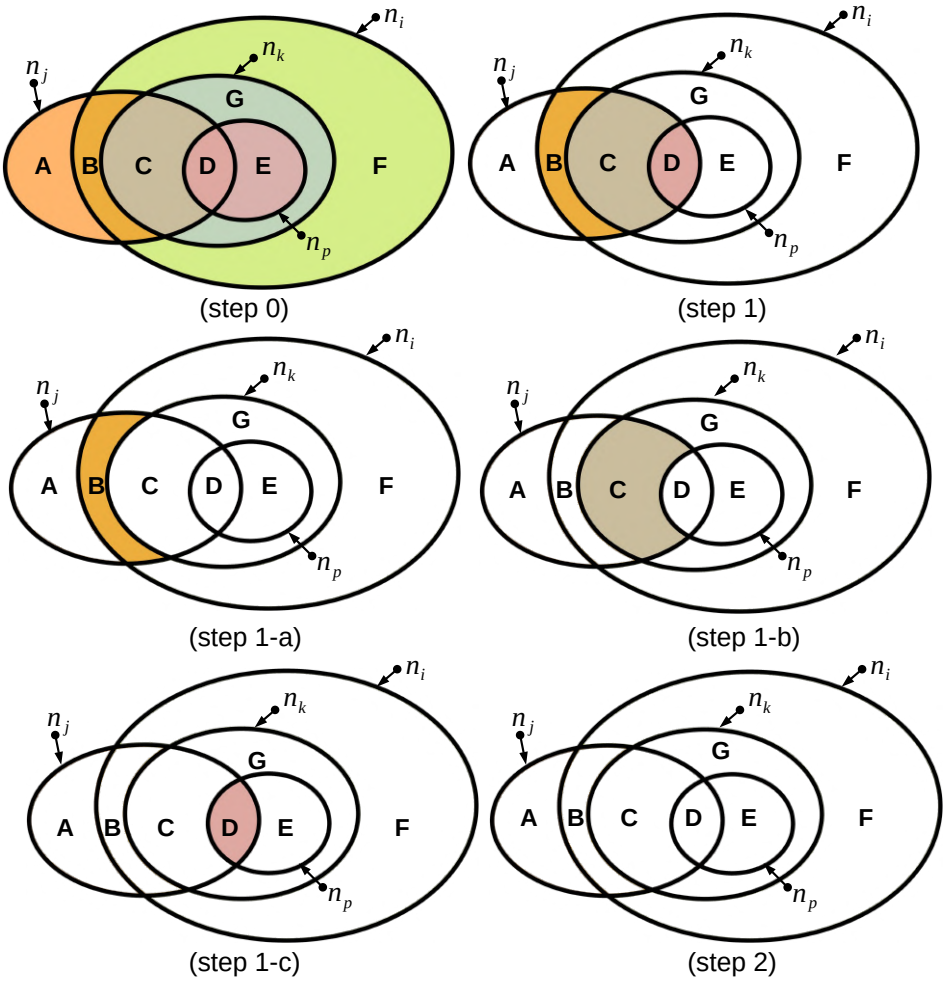


Figure 4.5 Slots scheduling process corresponding to explanation of Alg. 3. The candidate slots set of four nodes (n_i, n_j, n_k, n_p) are shown as ellipses with color, marking the possible intersections between the ellipses with letters from A to G. The disappeared color means the slots in the set are assigned after a step (step 0 $\rightarrow$ step1 $\rightarrow$ step2).

The remaining elements in cS are the *free slots*, which initialize shared slots ($shdS$, line 8).

The next loop identifies sibling neighbors and shared slots. Going through all neighbors, if a node finds a neighbor *ele* with the same cS, it adds *ele* to the sS (lines 10-11). If the neighbor is *not* a sibling and its cS is not a superset of the node's cS, the node removes the common candidate slots elements (i.e. the intersection) from $shdS$ (lines 13-15). This rule makes sure the nodes can find unique shared slots to allocate. Take node k in Fig. 4.5 as an example: as no other node has the same cS, n_k does not have sibling neighbors. The cS_k of n_k are {C, D, E, G}, which are used to initialize $shdS_k$. When n_k compares its cS_k with those of n_i and n_p, the intersection {C, D, E} is removed from $shdS_k$. The comparison with n_i does not have any effect since the cS_i of n_i ({B, C, D, E, F, G}) is a superset of n_k's ({C,

Algorithm 3: TDMA Schedule to for slots distribution

```
input : rcvPackets
output: cS, sS
1  listenNeighb(rcvPackets);
2  sS ⇐ ∅;
3  cS ⇐ {1, 2, 3, ..., n};
4  aN^2 ⇐ N^1 ∪ N^2;
5  forall ele ∈ aN^2 do
6      cS ⇐ cS \ ele.sS
7  shdS ⇐ self.cS;
8  forall ele ∈ aN^2 do
9      if self.cS == ele.cS then
10         sN.add(ele);
11     else if self.cS ⊄ ele.cS then
12         shdS ⇐ shdS \ (self.cS ∩ ele.cS);
13 if size(sN) > 0 then
14     sS ⇐ distribute(shdS)
15 else
16     sS ⇐ sS ∪ shdS
17 cS ⇐ cS \ shdS
18 broadcast(cS, sS)
```

D, E, G}).

Similarly for n_i, its $shdS_i$ is {F} after removing {B, C, D, E, G} as intersections. Doing so for each node leads to unique $shdS$: {A} for n_j, {F} for n_i, {G} for N_k, and {E} for N_p.

After performing the above operations, the elements in $shdS$ are safe to use for each node. Each node shares its $shdS$ with its sibling neighbors if it has any (lines 18-19), or else adds $shdS$ to its own sS (lines 20-21). Then each node updates cS by removing the remaining $shdS$. The new sS and cS are broadcast in the next frame. At this point, some slots are still not assigned, like slots in {B, C, D}. These slots are assigned in the next frame.

After each node broadcasts its information in its own time slot, all nodes receive the results from step 0 and reach the state as shown in step 1 in Fig. 4.5. In step 1, following the same procedures for step 0, nodes n_i and n_j become sibling neighbors as they all have the same cS {B, C, D}. After checking for intersections n_k and n_p (that are not siblings), have $shdS$={B}, as step 1-a of Fig. 4.5 shows in orange. The sibling neighbors distribute their shared slots evenly (line 19), which means both n_i and n_j get part of {B} without collisions. Shared slots can be distributed by fairness, ID, or other rules, as long as the rule allows for

unique assignments from local decisions. nodes n_k and n_p still do not have sibling neighbors and therefore they just do the intersection check: n_k ends with $shdS$={C} at step 1-b and n_p with $shdS$={D} at step 1-c. All free slots are assigned and no candidate slots are left in step 2.

A major advantage of our algorithm is that we can quickly and safely assign all free slots, no matter how many slots in each set from A to G. This allows our system to have a fast response to dynamic topology changes.

Let us consider a specific example of a multi-hop network in Fig. 4.6: the four nodes (n_i, n_j, n_k, n_p) are within each other's communication range and form a cluster connected by solid black lines. Other surrounding nodes with ID 1 to 6 are 2-hop neighbors for the cluster. The dotted line between two nodes means they are in 2-hop range or that they are a hidden node [19] for each other. Node 1 is connected with n_i, and broadcasts in time slot 1 (its own slot). Therefore, n_i must be silenced in time slot 1, otherwise the node between n_i and 1 would detect a collision. As the Fig. 4.6 illustrates, there exist candidate slots {2, 3, 4, 5, 6} for n_i, {1, 2, 3, 4} for n_j, {3, 4, 5, 6} for n_k, and {4, 5} for n_p. It takes 3 iterations to allocate all candidate slots as listed in Tab. 4.1.

It can happen that the $shdS$ becomes empty after removing intersections with the cS of neighbors. This situation can cause a deadlock: we let the node aggressively takes all the cS when detecting no changes in cS for more than 3 frames to break the deadlock.

4.5.4 Node Arrival and Departure

Algorithm 3 allocates all free slots, which means that in any 2-hop sub-graph, the shared channel medium is fully used. Therefore, when a new node enters the 2-hop neighborhood of a cluster of nodes, it generates unavoidable conflicts: at a minimum, its sending slot

Table 4.1 Slots distribution process for example in Fig. 4.6

		Step0	Step1	Step2	Step3
n_i	cS_i	{2,3,4,5,6}	{2,3,4,5,6}	{2,3,4}	∅
	sS_i	$\{s_i\}$	$\{s_i\}$	$\{s_i,2\}$	$\{s_i,2\}$
n_j	cS_j	{1,2,3,4}	{1,2,3,4}	{2,3,4}	∅
	sS_j	$\{s_j\}$	$\{s_j,1\}$	$\{s_i,1\}$	$\{s_i,1\}$
n_k	cS_k	{3,4,5,6}	{3,4,5,6}	{3,4}	∅
	sS_k	$\{s_k\}$	$\{s_k,6\}$	$\{s_i,6,3\}$	$\{s_i,6,3\}$
n_p	cS_p	{4,5}	{4,5}	{4}	∅
	sS_p	$\{s_p\}$	$\{s_p,5\}$	$\{s_i,5,4\}$	$\{s_i,5,4\}$

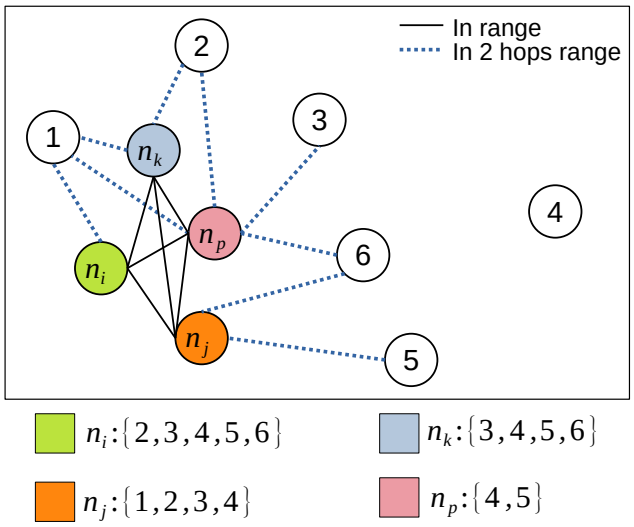


Figure 4.6 Example of the candidate slots of four nodes (n_i, n_j, n_k, n_p) in a multi-hop network. The number sets at the bottom are the candidate slots. A black solid line indicates that nodes are in direct communication range, and the blue dotted line indicates they are at 2-hop distance.

collides with the schedule of the cluster. We provide a two-step solution to quickly resolve the collision. Suppose node i enters the range of a cluster B, the first step is to have B release i's own slot. As we do not have precise clock synchronization, it is likely for nodes in B to receive a message from i. When a node in B receives a message from i, it adds i to its neighbor list and propagates its presence to the rest of B, freeing i's own slot.

A second step is to avoid collisions between the extra slots of i and B. We set that the node with the larger number of send slots must release the slots causing collisions. After i is allocated its own slot, i broadcasts its cS_i and sS_i and receives the same information from neighbors. When i finds it shares a slot s from sS_j with a 1- or 2-hop neighbor j (assume node $j \in B$, and $i \neq j$), namely $s \in (sS_i \cap sS_j)$, i checks the number of its sS_i: if it has more slots in sS_i than the conflicting neighbor, it removes s from sS_i. Conversely, if j has more sS_j than i, j releases s. If both nodes have the same number of send slots, the node with the lowest ID releases s:

$$sS_i = \begin{cases} sS_i & \text{if } size(sS_i) < size(sS_j) \\ sS_i & \text{if } size(sS_i) = size(sS_j) \text{ and } i > j \\ sS_i \setminus s & \text{if } size(sS_i) = size(sS_j) \text{ and } i < j \\ sS_i \setminus s & \text{if } size(sS_i) > size(sS_j) \end{cases} \tag{4.1}$$

With this solution, collisions can be solved in one frame. Note that the collision-free part of the previous scheduling result is maintained, which allows the system to adapt to a new schedule. This behavior is replicated if multiple nodes join the neighborhood at the same time.

If a node leaves the neighborhood, its slots should be reassigned. When a node i stops receiving messages from a neighbor j, i removes j from its neighbor list, as well as releasing all slots assigned to j. These time slots become candidate slots and participate in TDMA scheduling as described in Section 4.5.3.

4.5.5 Fairness and Extra Slots

Considering n number of slots in one frame, to guarantee a fair slot allocation, if a node i has a number of send slots $size(sS_i) > 2n/size(aN_i^2)$, meaning more than twice the average allotment of its neighborhood, n_i releases as many slots as needed to reach $size(sS_i) = n/size(aN_i^2)$. The released slots become candidate slots for its 2-hop neighborhood and follow the scheduling algorithm in Section 4.5.3.

The role of the extra slots depends on the application. For our localization purposes, each node performs two tasks in the extra slots:

1. ranging with one of its neighbors;
2. broadcast the ranging measurement.

Every node maintains an age list of its neighbors and selects the ranging target which has the oldest ranging measurement. It is worth noting that the measurements are also updated from the neighbors' measurement broadcast, avoiding repeated mutual ranging in short intervals.

4.6 Relative Localization

Some wireless sensor network localization systems use anchors or landmarks with known positions, and use centralized control to propagate the position results [158]. We propose a collaborative localization system based on ranging using UWB sensors [27].

The proposed TDMA algorithm maximizes the use of the UWB channel, allowing ranging measurements between all nodes at high frequency. Using the ranging information, each node can construct a local topological map of its neighbors' positions. By broadcasting, receiving, and fusing the local maps, the nodes can converge towards a global map.

In this work, we consider the 2D localization for the nodes in the system. Node i has a

neighbor set N_i^1, and it creates a graph of its surroundings as $G_i(V_i, E_i)$ where $V_i = \{v_t | t = i \text{ or } t \in N_i^1\}$ and $E_i = \{e_{kj} | k, j \in V_i, i \neq j\}$. For construction of a local map M_i, the goal is to find:

$$M_i = \{\mathbf{X}_j | \mathbf{X}_j = (x_j, y_j) \in \mathbb{R}^2,\ j \in V_i\}$$

Similarly, for the recovery of the global map, the goal is to find:

$$M_i^G = \{\mathbf{X}_j | \mathbf{X}_j = (x_j, y_j) \in \mathbb{R}^2,\ j \in W\}$$

where $W = \bigcup_{j \in N_i} V_j$.

4.6.1 Local Map Construction

To build the nodes' local maps, we use a similar method as the initialization stage of our previous work [27]. The difference in this work is that each node estimates the map locally. Given our hypothesis of a highly dynamic system, waiting for all ranging measurements to be propagated to build a global map as in [27] can introduce significant errors due to the use of outdated measurements. Therefore, each node creates a local map using the latest ranging measurements (which is therefore always up-to-date), and then merge the local maps at a later stage.

To build a local map, each node selects two neighbors as *reference nodes* to initialize the coordinate frame. The choice of reference nodes can greatly influence the localization accuracy. Yang [151] proposed the idea of the quality of trilateration to find appropriate reference nodes from nodes with known positions. Priyantha [44] proposed a relative localization algorithm by first selecting five reference nodes, and then applying an optimization process. In our case, the nodes do not have any neighbor with known positions. We propose a reference node selection that considering the expected trilateration performance, the number of neighbors, and the timestamps of the ranging measurements.

As showed in Algorithm 4, the system includes two parts: local map construction (lines 1-19) and global map recovery (lines 20-24).

Each node creates a map of its neighbors based on its own ranging measurements (named *rangeMeas*) as well as received measurements (named *receivedMeas*).

In Alg. 4, rangeMeas[tID, range, age] indicates an ego ranging measurement to the target neighbor tID, and receivedRange[sID, tID, range, age] indicates the received range measurement from node sID to node tID. All these range measurements are timestamped to allow the node to use the latest measurements.

Algorithm 4: Map construction for relative localization

input : neighbList, {rangeMeas[tID, range, age]}, {receivedMeas[sID, tID, range, age]}, receivedMaps
output: localMap, globalMap

$Origin \Leftarrow (0, 0)$;
$X\_Seed \Leftarrow Top(sortedCommonNeighb(Origin) \cap sortedRange(Origin) \cap sortedRangeAge())$;
$Y\_Seed \Leftarrow Top(sortedCommonNeighb(Origin) \cap sortedRange(Origin, X\_Seed) \cap sortedRangeAge())$;
$CS \Leftarrow coordinateFrame(Origin, X\_Seed, Y\_Seed)$;
$locatedNodes.add(Origin, X\_Seed, Y\_Seed)$;
$suspendSet \Leftarrow \emptyset$;
for $ele \in neighbList \setminus locatedNodes$ **do**
    if $rangeTo(Origin, X\_Seed, Y\_Seed)exist$ **then**
        $elePos \Leftarrow trilateration(Origin, X\_Seed, Y\_Seed)$;
        $locatdNodes.add(elePos)$;
    else
        $suspendSet.add(ele)$;
    end
end
for $ele \in suspendSet$ **do**
    $elePos \Leftarrow multilateration(ele, locatedNodes)$;
    $locatdNodes.add(elePos)$;
end
$globalOptimization(locatedNodes, rangePairs)$;
for $map \in receivedMaps$ **do**
    if $overlayVertexNum(globalMap, map) > 3$ **then**
        $globalGrap \Leftarrow Merge(globalMraph, map)$;
    end
end

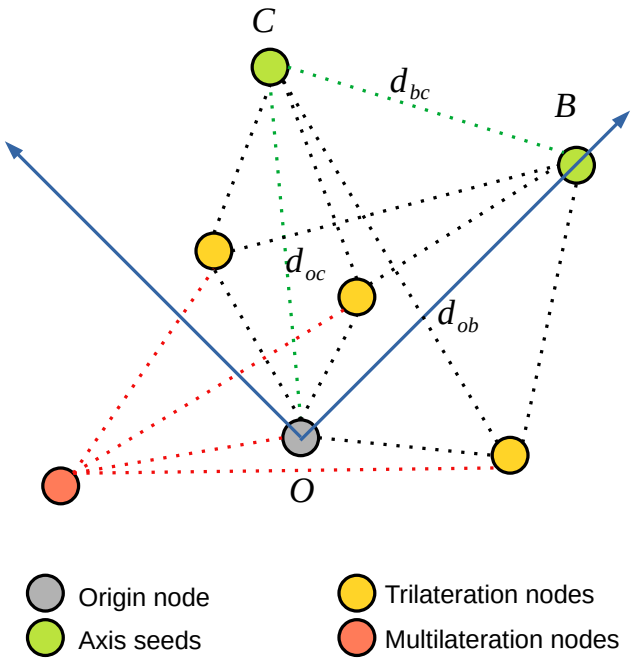


Figure 4.7 Local map construction with nodes in different roles. Origin node and axis seeds (reference nodes) are used to create the coordinate frame. Nodes that have measurements to reference nodes use trilateration to find positions. For the rest, the position is estimated by multilateration to nodes that already localized.

The construction of the local map is divided into three stages. Consider a node i: the first stage is to find reference nodes to create the coordinate system (lines 1-5). Node i uses its position as the origin of the coordinate frame (line 1). i then identifies another node *X_Seed* to define its x axis. The selection of X_Seed is based on three criteria (line 2):

1. the number of common neighbors with node i should be large, which is found by sorting $size(N_i^1 \cap N_j^1)$, where $j \in N_i^1$;
2. the node should be as far as possible from i, which is selected by sorting e_{ij}, where $j \in N_i$, $e_{ij} \in$ rangeMeas;
3. the measurement should as recent as possible, which is selected by sorting the ages of e_{ij}, where $j \in N_i$, $e_{ij} \in$ rangeMeas.

Then node i uses similar criteria to select another node *Y_Seed* to define the y axis. Suppose node i select node j as X_Seed and k as Y_Seed. The mutual distances between i, j, and

k are $[d_{ij}, d_{jk}, d_{ik}]$. i calculates the coordinates of j and k as:

$$\begin{aligned}
\mathbf{X}_i &= (0,0), \\
\mathbf{X}_j &= (d_{ij},0), \\
\mathbf{X}_k &= \left(\frac{d_{ik}^2 + d_{ij}^2 - d_{ik}^2}{2d_{ij}}, \pm\sqrt{d_{ik}^2 - \frac{d_{ik}^2 + d_{ij}^2 - d_{jk}^2}{2d_{ij}}} \right)
\end{aligned} \tag{4.2}$$

which corresponds to line 4 in Alg. 4. Note that we use the positive value for the Y coordinate of $\mathbf{X}_k$. Then X_Seed and Y_Seed are added to *locatedNodes* with their coordinates.

In a second stage, each node localizes the remaining neighbors with respect to the itself and the two reference nodes by trilateration (lines 6-14). Sometimes neighbors do not have ranging measurements to all reference nodes, or the ranging measurements are outdated. In this case, the node cannot perform trilateration and it is added to *suspendSet* (line 12) to be reexamined at a later stage. The nodes with successful trilateration are put into locatedNodes (line 10).

In a third stage, each node attempts to localize the nodes j with $j \in$ suspendSet using the existing ranging measurements using least squares multilateration (lines 15-17) [27]:

$$\mathcal{X}_\mathbf{j}^* = \arg\min_{\mathcal{X}_j} \sum_{k \in locatedNodes} (d_{jk} - \|\mathcal{X}_j - \mathbf{X}_k\|)^2, \tag{4.3}$$

with $\mathcal{X}_j^*$ the computed coordinates of node j. Following this procedure, a node can acquire the coordinates of all neighbors that can be located in the local map. These nodes are added to locatedNodes. However, these coordinates do not consider the error model of the ranging sensor.

UWB sensors have two-way ranging error that depends on many factors [159]. Lederberger et al. [160] introduce a Gaussian process error model for UWB ranging. However, this model requires knowledge of the relative angle between UWB antennas, making it impractical for our system. We apply a least square optimization to the coordinates of all nodes except for the origin and the y coordinate of X_Seed (that are all zero). Equ. 4.4 shows the *globalOptimization* step of line 19 in Alg. 4: all mutual distance measurements for nodes in locatedNodes are used to find the optimal coordinate estimations. The coordinates from the previous stages are used as the initial value for the least square optimization.

$$\begin{aligned} \mathcal{X}^* &= \min_{\mathcal{X}} \sum_{i,j \in locatedNodes} (d_{ij} - \|\mathcal{X}_i - \mathcal{X}_j\|)^2 \\ \mathcal{X}^* &: [x^*_{\text{X\_Seed}}, \mathcal{X}^*_t, ..], t \in locatedNodes \setminus \text{X\_Seed} \end{aligned} \tag{4.4}$$

4.6.2 Global Map Recovery

Once a node i has built its local map M_i, it broadcasts it to its neighbors. When a node i receives a map from a neighbor j, it merges j's map M_j to its local map M_i if the maps share common nodes.

The key to merge different maps is to estimate translation, rotation and reflection with respect to a given axis [4, 5]. In our case, reflection on the x axis might be induced since nodes always select the positive value for the Y coordinate of the Y_Seed node. Therefore, compared to a classical 2D transformation, we add an extra parameter related to reflection to the transformation matrix. Suppose node i has its own map M_i and receives map M_j from node j. The goal is to find the transformation $\mathcal{T}^i_j = [\mathbf{R}, \mathbf{T}, \mathbf{F}]$ from j to i in to place the nodes that only exist in M_j in i's coordinate frame:

$$\begin{aligned} \mathcal{T}^i_j &= [\mathbf{R}, \mathbf{T}, \mathbf{F}] \\ \mathbf{R} &= \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}, \theta \in (-pi, pi] \\ \mathbf{T} &= \begin{bmatrix} t_x \\ t_y \end{bmatrix}, t_x, t_y \in \mathbb{R} \\ \mathbf{F} &= \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}, \gamma = 1 \; or \; -1 \end{aligned} \tag{4.5}$$

where $[\mathbf{R}, \mathbf{T}, \mathbf{F}]$ correspond to the rotation, translation, and reflection matrices, respectively.

A node can start merging two maps if the maps have more than three common vertices (lines 20-24 in Alg. 4). We use $V(M_i \cap M_j)$ to indicate the common vertices between M_i and M_j. We use a least squares optimization to find the optimal transformation $\mathcal{T}^*$:

$$\begin{aligned} \mathcal{T}^* &= \min_{[\mathbf{R},\mathbf{T},\mathbf{F}]} \sum_{t \in V_o} (\mathbf{X}^i_t - (\mathbf{R} \cdot \mathbf{F} \cdot \mathbf{X}^j_t - \mathbf{T}))^2 \\ &\text{where } \; V_o = V(M_j \cap M_i). \end{aligned} \tag{4.6}$$

$\mathbf{X}_t^i$ stands for the coordinates of node t in node i's coordinate frame. With $\mathcal{T}^*$, the node can transform the vertices in $V(M_j \setminus M_i)$ to its coordinate frame to get a merged map M_i^G:

$$\begin{aligned} m_{j\setminus i} &= \{\mathbf{X}_t^i | t \in V(M_j \setminus M_i), \mathbf{X}_t^i = \mathbf{R} \cdot \mathbf{F} \cdot \mathbf{X}_t^j - \mathbf{T}\} \\ M_i^G &= M_i \cup m_{j\setminus i}, \end{aligned} \tag{4.7}$$

where $m_{j\setminus i}$ indicates the transformed map in n_i's coordinate frame consisting of the vertices that are unique to M_j.

4.7 Experiments

4.7.1 Simulations

We perform a set of simulations to assess the scalability of the proposed algorithm. The algorithm is run in a custom simulator written in Python and run on a laptop with 16GB of memory and an Intel i7-6700HQ processor. We simulate different numbers of nodes in an arena with a size of $50m \times 50m$. The communication range between nodes is $5m$. The nodes are randomly distributed in the arena, as shown in Fig. 4.8. We use three different orders of magnitude (10, 100, 1000) for the number of nodes in the system to simulate different deployment densities. We allow a maximum of 50 frames for the TDMA scheduling. The length of each time slot (as t_{slot}) is defined as 3 ms (the time for an UWB raging operation [59]).

Fig. 4.9 shows the scheduling process for a system with a distribution of 100 nodes in Fig. 4.8, meaning there are 100 slots in a frame. Each node starts with a send slots set with only its own slot, and they have an average of 78 candidate slots available to assign. The number of candidate slots decreases quickly over time, which means the slots are assigned. All candidate slots are assigned around the 6th frame and each node is given 27 slots on average. The initial peak in the send slots suggests some form collision. Collisions are resolved in one frame (by releasing slots) and therefore the number of send slots quickly decreases.

The distribution with 1000 nodes is quite dense as shown in Fig. 4.11. Using the same setup for the arena and slot size, we run each experiment 30 times and show the aggregate results in Fig. 4.10 and Table 4.2.

Fig. 4.10 also plots the average number of frames needed to assign all the candidate slots ("frames" in the plot). We can see the increase in the number of frame iterations is fundamentally linear even for an exponential increase in the number of nodes. The average number of send slots increases significantly from 10 to 100 nodes, but due to the increase in density,

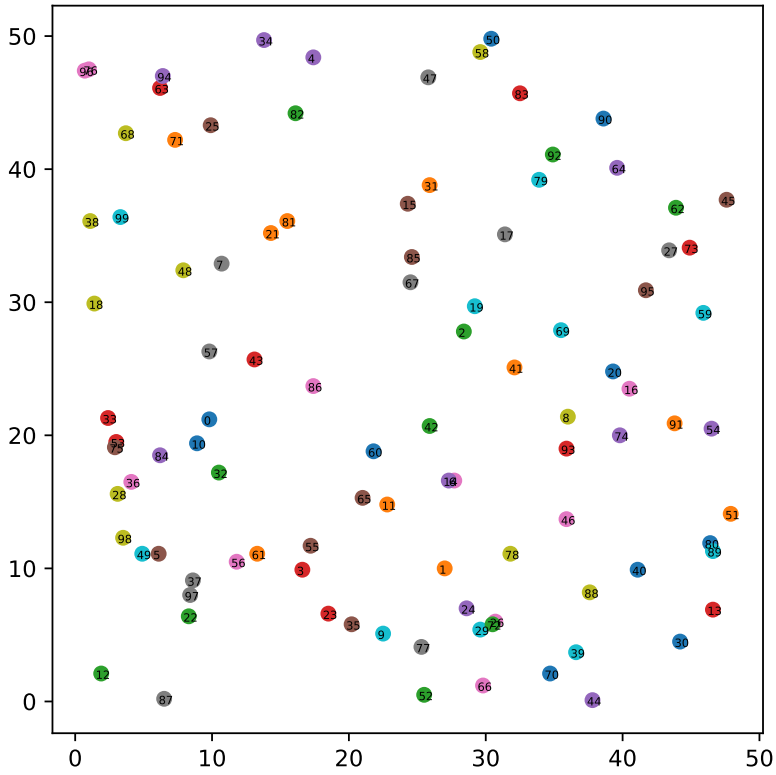


Figure 4.8 Random distribution of 100 nodes in a area of 50m*50m arena

it is basically unchanged from 100 to 1000 nodes.

In Table 4.2, N_Frame is the number of slots in a frame, which has the same value of the number of nodes (n) in the network, and N_Neighbors) is the average number of neighbors of each node. *Density* is the ratio between the "communication area" occupied by all nodes and the total area of the arena [161], as shown in Equ. 4.8. As the size of the arena is fixed, it is linearly related to the number of nodes.

$$Density = \frac{N * \pi R^2}{L^2} \tag{4.8}$$

where $R = 5$ and $L = 50$ in our experiment.

The average (Avg) and standard deviation (Std) for frame iterations (Frames) and resulting send slots (sS) listed in Table 4.2 correspond to Fig. 4.10.

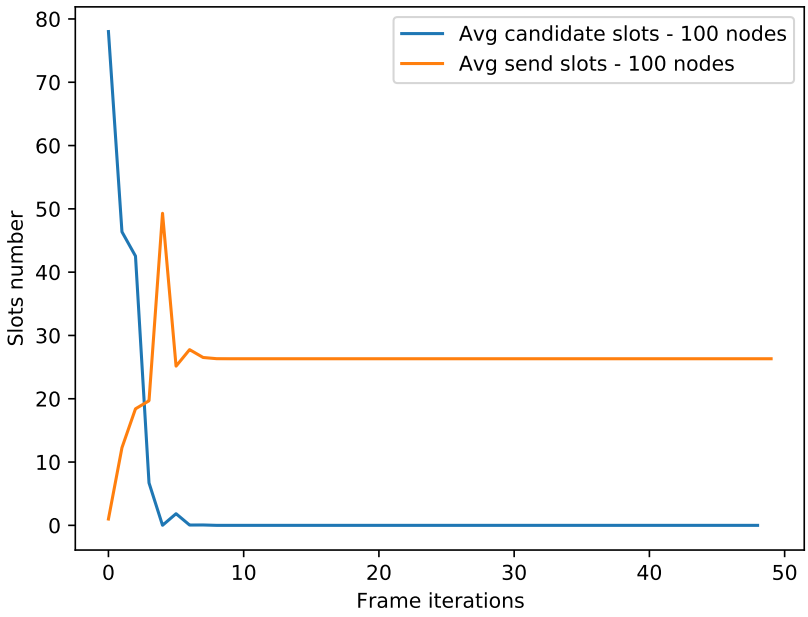


Figure 4.9 Scheduling process of a system with 100 nodes, distributed in an area of 50m*50m. The average number of send slots and candidate slots are plotted. At the start, each node has 1 send slot and an average around 79 candidate slots. Around frame 6, all the candidate slots are assigned and each node got around 27 on average.

Finally, we report some additional measurements:

$$\begin{aligned} \text{T\_Frame} &= t_{slot} \cdot \text{N\_Frame} \\ \text{avg\_T\_Frame} &= 2 \cdot \text{Avg\_Frame} \cdot \text{T\_Frame} \\ \text{N\_}sS &= \frac{\text{Avg\_}sS}{\text{T\_Frame}} \\ \text{LN\_}sS &= (1 + \text{N\_Neighbors}) \cdot \text{N\_}sS \\ \text{Total\_N\_}sS &= \text{N\_}sS \cdot \text{N\_Nodes} \end{aligned} \tag{4.9}$$

T_Frame is the time taken for each frame; Avg_T_Frame is the average time needed for scheduling considering two cycles; N_sS is the average number of slots per node per second, which indicates the number of range each node can measure per second; LN_sS is the total number of slots used by a local neighborhood per second, including the node and its neighbors; Total_N_sS is the total number of slots per second across the whole network, indicating the total number of ranging measurements can be made in the system.

From Table 4.2, we can see the number of frames used in scheduling (Avg_Frames) does not increase significantly between 100 and 1000 nodes. However, the time duration (Avg_T_Frame) is still large compared to a 100 nodes configuration: this result is acceptable considering the extremely high density of devices (see Fig. 4.11 for an intuition).

In terms of network usage, N_sS is equal to $1/t_{slot}$ (=333.3) when there are no neighbors. For the 10 nodes configuration, which is quite sparse, the average number of slots (N_sS) for each node is 263. The slots used by local neighborhoods on average (LN_sS) is 336.4. This is because the average number of neighbors (N_Neighbors) is only 0.28.

For the other configurations, the LN_sS have a similar value. This is due to the fact that some of the slots are used by two-hop neighbors. To find the maximum of Total_N_sS, we give consider the maximum number of nodes N_{max} allowed in the arena without any intersections of their communication ranges, i.e. when each node can make full use of the channel:

$$N_{max} < \frac{L^2}{\pi R^2} = \frac{50 \cdot 50}{\pi \cdot 5 \cdot 5} = 31.8 \tag{4.10}$$

N_{max} is less than the ratio between the area of the arena and the communication area. Therefore, the maximum of Total_N_sS can not be larger than $N_{max} \cdot 1/t_{slot}$, which is 10605. We can see in the configurations of 100 and 1000 nodes, the results are close to full channel usage.

4.7.2 Hardware Setup

We test the system with a physical implementation with 12 nodes, as shown in Fig. 4.12-b. Except for the unique ID of each module, all modules are identical. Each module consists of a Raspberry Pi 3 A+ with a Decawave 1000 [18] UWB module from Pozyx [15] as its ranging and communication sensor. All processing is performed on-board, with the exception of the global map construction, which is done on a separate computer used by the system operator. All communication between nodes is implemented through the UWB channel, while a 802.11 network is used for control and debugging. Please note that since the ceramic antenna of the Pozyx modules is directional, we place the modules upright as shown in Fig. 4.12-a to minimize orientation-related issues.

We conduct 3 sets of experiments to validate the system. To evaluate the performance of TDMA scheduling and rescheduling, we consider three situations: new nodes joining, nodes leaving, and multi-hop scenarios.

Table 4.2 Scalability study

ID	10 nodes	100 nodes	1000 nodes
N_Frame , N_Nodes	10	100	1000
N_Neighbors	0.28	2.29	28.73
Density	0.31	3.14	31.41
Avg_Frames	1.00	6.41	20.76
Std_Frames	0.0	1.25	9.46
Avg_sS	7.89	27.01	28.28
Std_sS	0.89	1.71	8.66
T_Frame	0.03 s	0.30 s	3.0 s
Avg_T_Frame	0.06 s	3.84 s	124.8 s
N_sS	263	90.03	9.43
LN_sS	336.4	296.2	280.3
Total_N_sS	2630.0	9003.33	9433.33

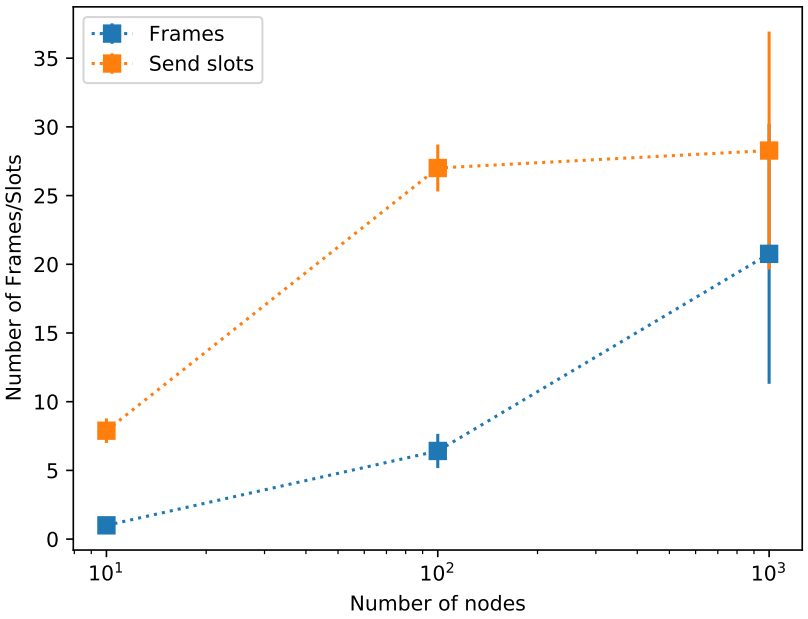


Figure 4.10 Scalability study with 10, 100, and 1000 nodes. Frames stands for the average number of frames used before all the candidate slots are distributed. Send slots shows the average number of result slots each node gets.

To show the fast response of our system to changes in high-traffic conditions, we test the effect of joining and leaving nodes in a static, fully connected scenario, by placing the nodes in proximity and turning on/off the nodes that are joining/leaving.

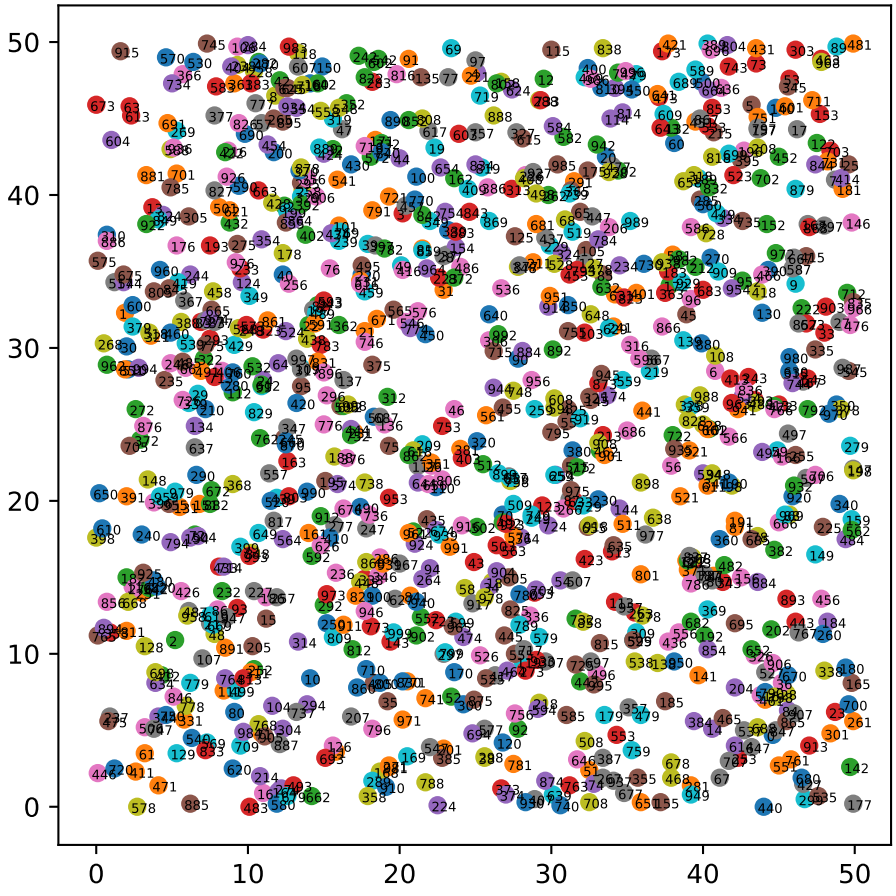


Figure 4.11 Random distribution of 1000 nodes in a area of 50m*50m arena

We also test our algorithm in a dynamic environment by moving the nodes from a fully connected network to a multi-hop network. This result shows the conflict-free scheduling and slot multiplexing usage in a complex network topology.

Finally, we conduct a comprehensive experiment to test the spatial map construction for a dynamic network with mobile devices. This last experiments shows that the system can localize the devices using ranging in the assigned time slots. Each node can construct its own local spatial map and recover the global map using local communication.

4.7.3 New Nodes Joining

Without TDMA (or with Carrier Sense [64]), the system may encounters a large number of collisions since all nodes attempt to achieve high channel usage.

We place twelve modules together on a desk as shown in Fig. 4.12-b. Starting the nodes in

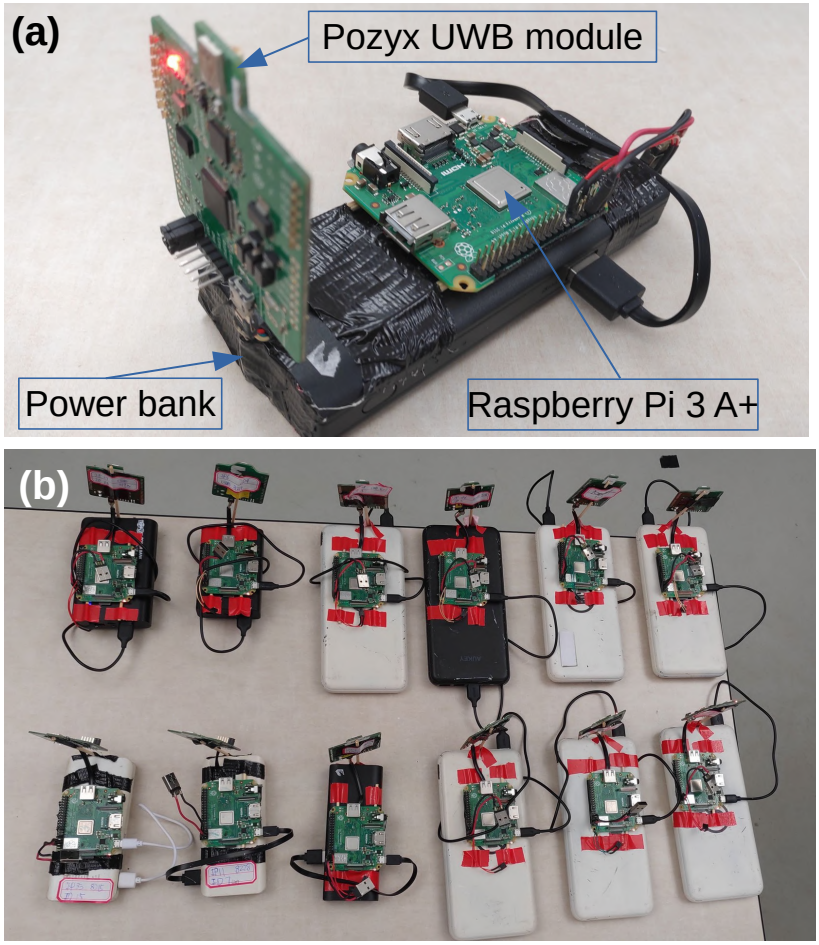


Figure 4.12 The hardware modules used as nodes in our experiments. Each module consists of a battery, a Raspberry Pi 3 A+, and a Pozyx module based on the Decawave 1000 UWB chip. All the processing is done onboard and all the communication between nodes passes through the UWB channel.

sequence, we can emulate new nodes joining the network. We start nodes 1, 3, 4, and 5 at first. After two minutes, we start nodes 6, 8, 12, and 13. Finally, we start the remaining nodes: 7, 9, 10, and 15.

The scheduling process results are shown in Fig. 4.13. The numbers from 0 to 29 on the x axis represent the slot IDs. We use 30 slots in our system (with slot 0 is reserved the processing of TDMA algorithm). We select the 30 to have a more visible scheduling result to show. The system can be viewed as system supporting a maximum of 30 nodes, but only 12 nodes are studied. Each slot had a duration of 50 milliseconds due to firmware limitations of the Pozyx module. Ter Horst et al. [59] show that a 3 ms slot is enough to perform ranging measurements using the same Decawave 1000 UWB chip, but for a different type of module.

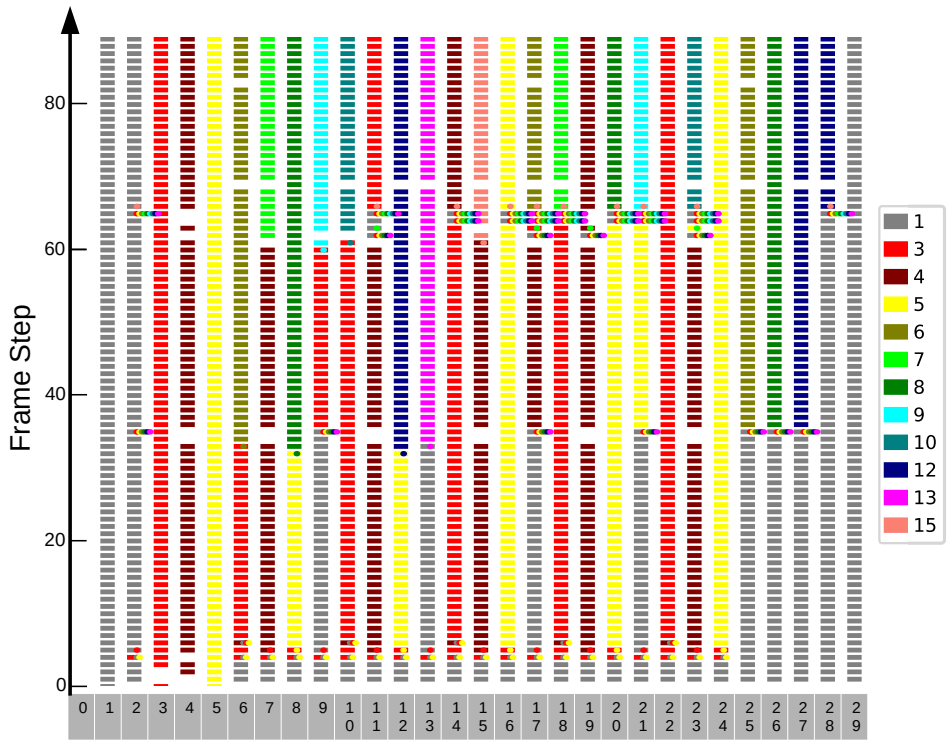


Figure 4.13 TDMA scheduling results for a system with new nodes joining. Nodes group {1, 3, 4, 5}, {6, 8, 12, 13} and {7, 9, 10, 15} are started gradually. This result is from the perspective of node 4, but it is the same for everyone in the network as they are fully connected. The X-axis is the slot ID and the Y-axis is the frame iteration, representing the time. A total of 30 slots are configured. We can see that when a new group of nodes join the system near frame 35 and 53, the system can quickly adapt to the new schedule. All slots are occupied all the time to have full channel usage.

Our system could be set to similar values on different hardware.

Having 30 slots of 50 ms means each frame is 3 seconds due to our organization of two cycles per frame (see Fig. 4.2). On the x axis, we show the 30 slots of the first cycle (cycle A) of the frame as the scheduling is identical for both cycles. Considering 12 nodes and 50 ms slot size, each node can be assigned at least 2 slots per frame (one own slot and one extra slot). Therefore, the lower bound slot use for each node is 4 slots every 3 seconds. The y axis is the frame count, increasing in time, meaning that the plane forms a grid showing the assignment of slots in time with colours assigned according to the assigned node identity. This reporting is a general way to show the scheduling speed, and it is independent from the slot size, which can differ based on the system configuration.

From Fig. 4.13, we can see that slot scheduling can quickly adapt to network changes and obtain a conflict-free schedules with full slot use in two to three frames. During the first 3

frames, almost all slots except slots 3, 4, and 5 are assigned to the node with ID 1 (gray color). This is because when the first 4 nodes are started, they first synchronize. However, since the ID of node 1 is the smallest in the neighborhood, node 1 enters the TDMA scheduling phase directly, while the other three nodes synchronize to node 1 instead.

Therefore, node 1 does not receive any candidate slots or sending slots information from the other nodes and takes all idle slots. Note that node 1 does not occupy slots 3, 4, 5 since its neighbors still broadcast heartbeat messages (i.e. they notify node 1 of their presence) during synchronization. After the slot usage fairness check described in Section 4.5.5, node 1 releases some slots for its neighbors: nodes 3, 4, and 5 can get these extra slots after they are synchronized. One can see that node 3 gets slot {6, 10, 14, 18, 22} as extra slots starting from frame 3. Node 4 and 5 get {7, 11, 15, 19, 23 } and {8, 12, 16, 20, 24 }, respectively. All the remaining slots are occupied by node 1. The four nodes share slots 6 and 25 periodically: this is because they are sibling nodes, and get a shared slot set of {6-25} during scheduling.

Around frame 35, we turn on four additional modules with ID 6, 8, 12, and 13. We can see that once the new nodes are detected, their own slots {6, 8, 12, 13} are released immediately. As the number of neighbors increase, the older nodes notice they are using too many slots during their fairness check and begin to release slots. The change of slot owner is reflected with color changes for columns {9, 17, 21, 25, 26, 27}. Some collisions do occur during scheduling on these slots, which are indicated by the colored dots next to the columns. We can see that the collisions are resolved within five frames (mostly in two frames), leading again to full slot usage. Following a similar process, we turn on the remaining four nodes, which reach a new schedule after 3 frames. The resulting schedule for each node during the three stages of the experiment is shown in Table 4.3. Please note some slots have a short white gap: this is due to the initial synchronization of the newly added nodes.

4.7.4 Nodes Leaving

We tested the system with the same setting as the nodes joining experiment, but starting with all modules turned on, after which we turn off two groups of 4 nodes in sequence. First, nodes with ID 7, 9, 10, 15, followed by nodes 1, 3, 4, 5. The scheduling results are shown in Fig. 4.14, with the same axes and content format as Fig. 4.13.

We can see that the slots are always fully used for the three stages: the system adapts to the new schedule as soon as the nodes detect the departure of a set of nodes. The free slots are allocated by the remaining nodes and reach a conflict-free schedule. The scheduling result of each node in every stage is listed in Table 4.4.

Table 4.3 Slot distribution for new nodes merging scenario

ID	Stage 1	Stage 2	Stage 3
1	1,2,9,13,17,21,25,26,27,28,29	1,2,28,19	1,2,29
3	3, 6, 10, 14, 18, 22	3,9,10,14,18,22	3,11,22
4	4, 7, 11, 15, 19, 23	4,7,11,15,17,19,23	4,14,19
5	5, 8, 12, 16, 20, 24	5,16,20,21,24	5,16,24
6		6,25	6,1,25
8		8,26	8,20,26
12		12,27	12,27,28
13		13	13
7			7,18
9			9,21
10			10,23
15			15

Please note that the white gaps are mainly caused by the nodes leaving the system: we have set a timeout of 3 frames without messages from a node before considering that it has left the neighborhood to avoid rescheduling due to network noise. A short gap can also be caused by synchronization induced by the departure of the node with the smallest ID (which is used as a reference). This happens during the transition from stage 2 to stage 3, when the node 1 is turned off. In this scenario, node 13 took three frames to synchronize with the new reference node 6, leading to the gap for slot 13.

4.7.5 Multi-Hop TDMA Scheduling

We tested the system in the indoor parking lot at Polytechnique Montreal as shown in Fig. 4.15. The parking lot has very thick walls and makes it easy to set up a multi-hop network. We first put all modules together in one parking spot as depicted in Fig. 4.16-a. They form a fully connected network with a TDMA schedule shown at the bottom of Fig. 4.16-a in the column “Stage 1” of table 4.5. We can see they reach a conflict-free schedule.

In a second stage of the experiment, we move the modules to other parking spots, forming a multi-hop network topology as shown in Fig. 4.15. The connectivity graph of the nodes is shown in Fig. 4.16-b with lines between nodes indicating a data and ranging connection.

The scheduling result is displayed in the color block of Fig. 4.16-b, as well as listed in the column “Stage 2” in Tab. 4.5. The slot schedule shown in Fig. 4.16 is plotted using the log of each node, which includes its own time slot and the slot schedule received from its neighbors.

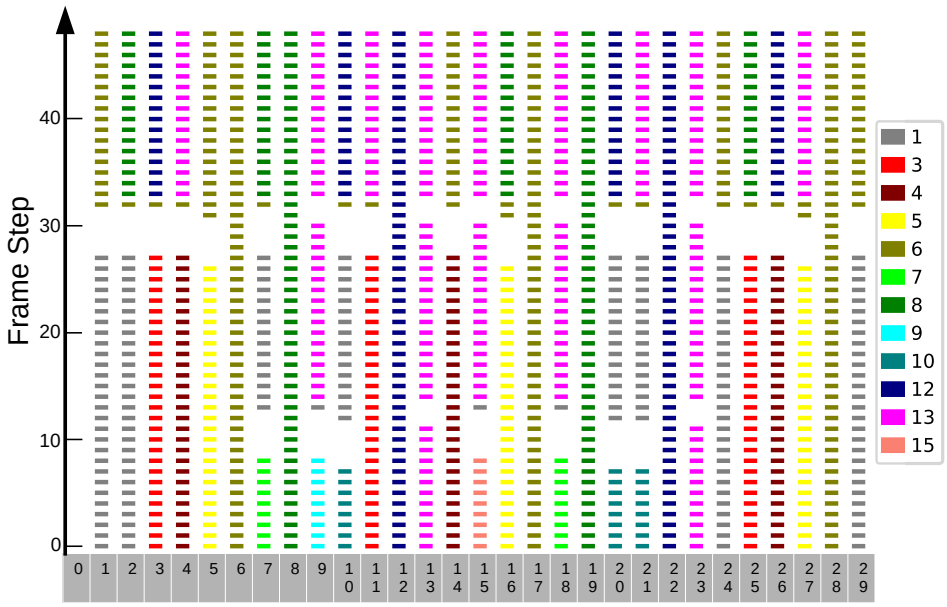


Figure 4.14 Scheduling result of experiment that nodes leave the network. Starting from all modules powered on, we first turn off nodes with the ID of 7, 9, 10, 15 near the frame 10, and then turn off nodes 1, 3, 4, 5 near frame 30. This plot is based on the logs from node 13. We can see when nodes left near frame 10 and 30, the released slots are allocated by the rest nodes and reach a conflict-free schedule.

Figure 4.15 Multi-hop experiment testing field at parking place

Taking the top line as an example, node 10 receives the slot assignment of all its neighbors, namely neighbors 5, 13, 3, 8, 12. Compared with the following row, node 5 only has neighbors 10 and 13. The advantage of our algorithm can be seen from nodes that are three hops away: as an example, nodes 5 and 6 share slot 26 (there are two colors on column 26 of Fig. 4.16-b) without collisions (the nodes are 3 hops apart, see the graph in Fig. 4.16-b).

For this schedule, 21 slots out of 29 are multiplexed (shared among multiple nodes), leading

Table 4.4 Slot schedule for nodes leaving scenario

ID	Stage 1	Stage 2	Stage 3
6	6,17,28	6,17,28	1,5,6,14,17,24,28,29
8	8,19	8,19	2,7,8,16,19,25
12	12,22	12,22	3,10,12,20,22,26
13	13,23	9,13,15,18,23	4,9,11,13,15,18,21,23,27
1	1,2,24,29	1,2,7,10,20,21,24,29	
3	3,11,25	3,11,25	
4	4,14,26	4,14,26	
5	5,16,27	5,16,27	
7	7,18		
9	9		
10	10,20,21		
15	15		

to full channel utilization.

4.7.6 Map Construction

So far, we have proved that our system can effectively perform real-time TDMA scheduling with full channel usage. In this experiment, we explore the relative localization capability of the system.

In a first stage, we put all nodes in a fully-connected grid formation, as shown in Fig. 4.17 and in Fig. 4.18-a. Using all available ranging measurements, the nodes can easily recover their local map. We show the map constructed by 4 randomly selected nodes (all the remaining maps are similar) in Fig. 4.19. We can see that they all have the same relative structure, but with different coordinates as each node uses its own coordinate frame.

When the nodes are moved to the configuration in Fig. 4.18-b, the network topology changes and the nodes rescheduled the TDMA assignment with the same logic as in the previous experiments. For this new configuration, we randomly select 3 nodes on the left side and 3 nodes on the right side to show their local map in Fig. 4.20. We can see they are able to construct the correct relative map. Using the local map from node 5 and 1, node 8 can recover the global map as shown in Fig. 4.21.

4.8 CONCLUSION AND FUTURE WORK

In this paper, we present a fully decentralized system that is able to do accurate localization in a UWB-only network. We propose a novel distributed TDMA algorithm that can allocate

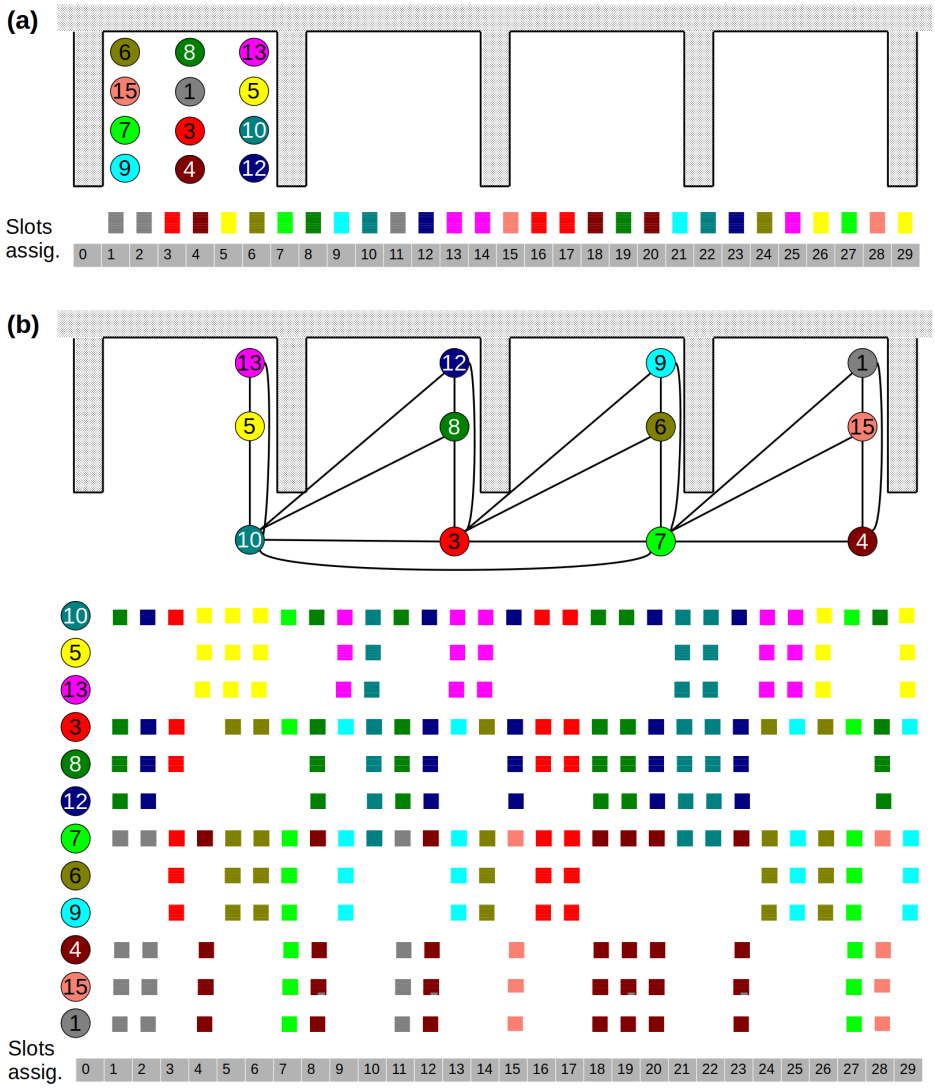


Figure 4.16 Multi-hop network TDMA experiment. Twelve modules are moved from (a) to (b). The slots scheduling is shown with the color block under the graph. In sub-figure (a), all nodes have the same scheduling as they are fully connected. When moving nodes to form the topology in (b), they build a multi-hop network. The slot scheduling of each node is plotted from the log of each node. The log of each one includes its own slot and the slot schedule received from its neighbors. Looking from the point of the slot, any slot that has more than one color means there are more than one node use the time slot. For example, in time slot 1, the green node 8 and gray node 1 use the slot at the same time. From the graph, we can see nodes 8 and 1 are three hops apart. In this schedule, there are 21 slots out of 29 multiplexing used. For each node, there are no more slot can be used.

all the idle slots quickly to have full channel usage for the mobile networks. This high channel usage results in high rate of measurement in the UWB network, leading to high localization accuracy. Fast scheduling also makes the system suitable for a network with changing topol-

Table 4.5 Slot distribution for multi hop network

ID	Slots in Stage 1	Slots in Stage 2
10	10, 22	10,21,22
5	5,26,29	4,5,6,26,29
13	13,14,25	9,13,14,24,25
3	3,16,17	3,16,17
8	8,19	1,8,11,18,19,28
12	12,23	2,12, 15,20,23
7	7,27	7,27
6	6,24	6,7,14,24,26
9	9,21	9,13,25,29
4	4,18,20	4,8,12,18,19,20,23
15	15,28	15,28
1	1,2,11	1,2,11

Figure 4.17 Node distribution at the start of map construction experiments. Nodes are placed in a grid formation occupying an area around 8*12m.

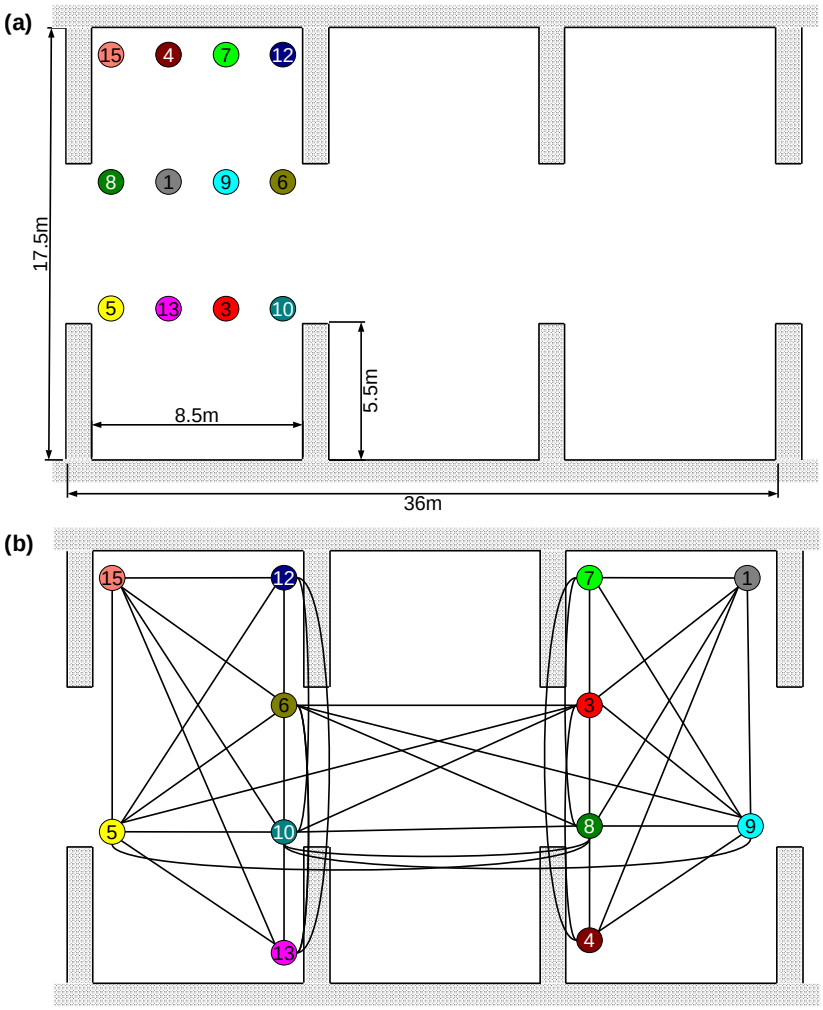


Figure 4.18 The experiment involves 2 stages, corresponding to sub-figure (a) and (b). At the start, all nodes are placed in a grid formation shows in (a). The network in this configuration is a fully connected network. Then they are moved to the configuration (b). The maximum of 3 hops is used in this configuration.

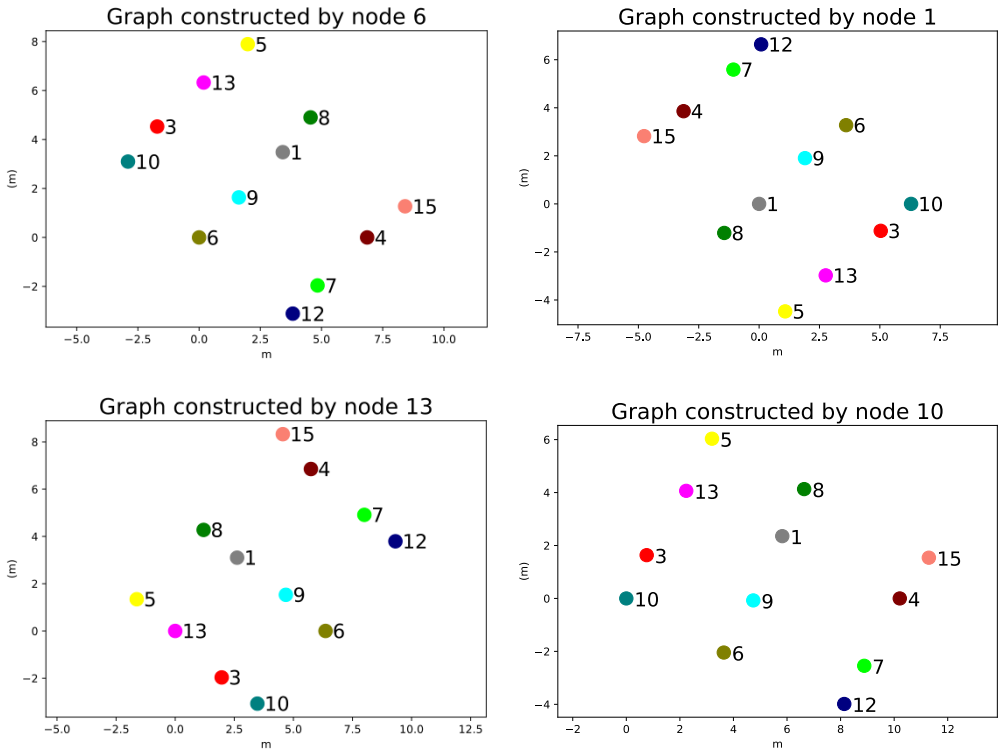


Figure 4.19 Local map construction of nodes 6, 1, 13, and 10. They build their own coordinate system and get the relative localization for other nodes. As they are fully connected, the relative localization is the same, which is also the same for the rest nodes.

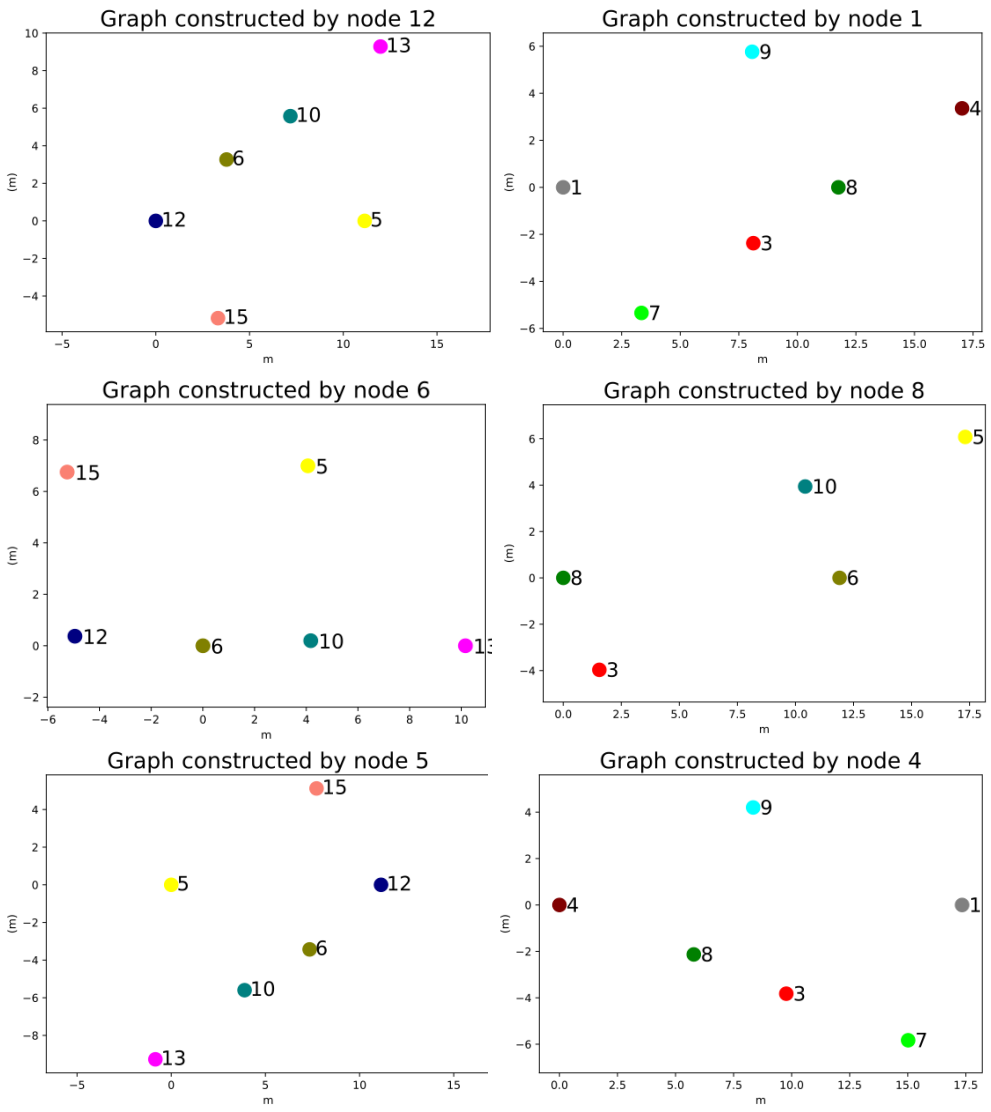


Figure 4.20 Local map construction of nodes 12, 1, 6, 8, 5, and 4 at stage 2. Every node build a relative location for its neighbors. We can see the map shows part of the map in Fig. 4.18-b.

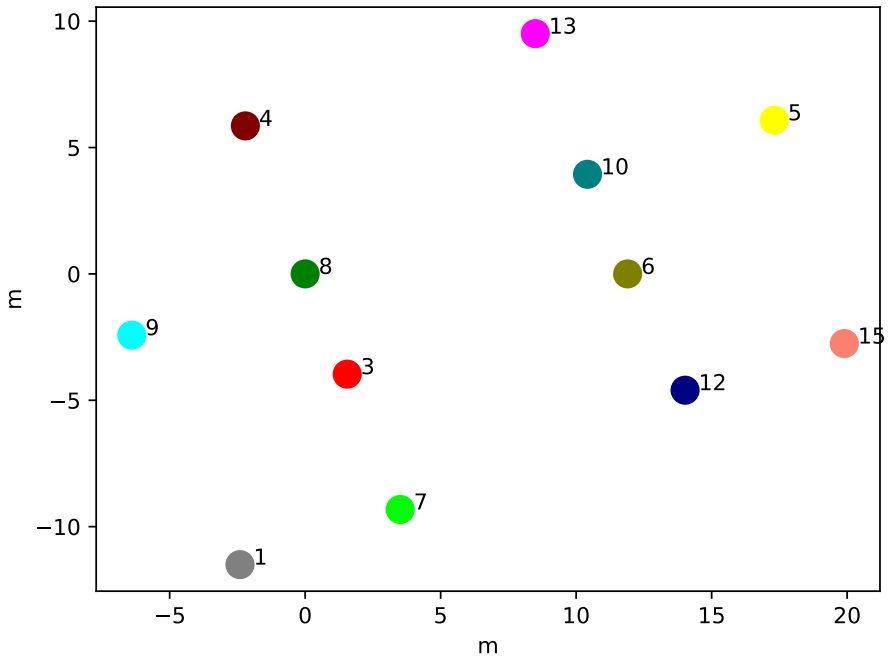


Figure 4.21 Global map recovery for node 8 when receiving local maps from node 5 and 1.

ogy. We also separate the localization in the network into local relative localization and global map merging. Although our system is designed for UWB localization, our TDMA algorithm can also be applied to any networks requiring high channel usage, with static or mobile nodes. One limitation for our system is the fixed size of slots in frame structure initialized at the start, limiting the TDMA scheduling process to the update rate of frame cycle. In future work, we will implement an adaptive frame size to improve the scheduling update rate. Furthermore, to improve the localization and tracking performance, we will fuse measurements from different sensors, such as Inertial Measurement Units (IMUs) or cameras.

CHAPTER 5 ARTICLE 3: ACCURATE POSITION TRACKING WITH A SINGLE UWB ANCHOR

Preface: In last two chapter, we solve the systematic strategy and the hardware challenge of the application of UWB in a multi-robot system. With the position initialized from previous result, we work on the odometry estimation of single robot. To have a minimal system, only IMU sensor is used in this work. We propose a solution allow single robot to locate itself with a ranging source from single one anchor in the environment. This method can be used in low-cost devices with IMU and UWB only, such as IoT devices.

Full Citation: Y. Cao, C. Yang, R. Li, A. Knoll, and G. Beltrame, "Accurate position tracking with a single UWB anchor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2344–2350, IEEE, May 2020.

Personal Contributions: Y. Cao conceived of the presented ideas, developed the theory, designed and implemented the simulations. Y. Cao designed and carried out real robot experiments with support from C. Yang and R. Li. Y. Cao wrote the manuscript with support from G. Beltrame.

Abstract

Accurate localization and tracking are a fundamental requirement for robotic applications. Localization systems like GPS, optical tracking, simultaneous localization and mapping (SLAM) are used for daily life activities, research, and commercial applications. Ultra-wideband (UWB) technology provides another venue to accurately locate devices both indoors and outdoors. In this paper, we study a localization solution with a single UWB anchor, instead of the traditional multi-anchor setup. Besides the challenge of a single UWB ranging source, the only other sensor we require is a low-cost 9 DoF inertial measurement unit (IMU). Under such a configuration, we propose continuous monitoring of UWB range changes to estimate the robot speed when moving on a line. Combining speed estimation with orientation estimation from the IMU sensor, the system becomes temporally observable. We use an Extended Kalman Filter (EKF) to estimate the pose of a robot. With our solution, we can effectively correct the accumulated error and maintain accurate tracking of a moving robot.

5.1 Introduction

Accurate localization and tracking are fundamental services for an autonomous system. Many options are available for localization: GPS for open outdoor areas, motion capture systems in a laboratory, visual systems. However, they are generally limited by the environment or time-consuming and labor-intensive setup work and expensive infrastructure [162]. Ultra-wideband (UWB) technology provides another venue to accurately locate devices both indoors and outdoors. Most available UWB systems are based on multi-anchor arrangements, which need some labor-intensive setup work, like mounting anchors and calibration. Furthermore, it is often difficult to set up such systems outdoors or in an unstructured environment. We believe that a localization system which can accurately track devices without complex setup is highly desirable.

Tracking with a single anchor is attractive because one can easily drop an anchor in the environment as reference. However, it is also quite challenging: a single source of distance information is generally too limited for tracking. Current research in underwater robotics proposes fusing distance to an acoustic anchor with odometry, but they usually rely on very expensive sensors (e.g., high accuracy IMU, doppler anemometer) [79, 163]. Our goal is to enable single anchor localization with low-cost UWB and IMU sensors. Robots and IoT devices can easily be equipped with these two sensors, while velocity sensors (encoders, doppler anemometers, etc.) are much more rare and generally too expensive for IoT devices. Moreover, nowadays UWB is becoming pervasive, being present in the latest Apple iPhone [164] for spatial awareness at the time of writing.

Getting odometry from low-cost IMUs is challenging. Velocity, integrated from acceleration coming from the IMU, drifts quickly and cannot be used for odometry. Unfortunately, velocity is crucial for the observability of mobile robot localization system, especially for single anchor localization [84]. To solve this problem, we propose a novel algorithm to estimate velocity by combining UWB and IMU measurements. An EKF fuses the range, orientation, and speed estimation to estimate the robot pose. Simulations and real-world experiments validate our algorithm. We believe our system can unlock localization of a large number of devices in practical applications.

5.2 Related Work

Using UWB technology to locate devices has recently become popular. Most applications are based on a multi-anchor configuration [54,107,165–168]. We aim at simplifying the infrastructure to a single anchor as reference. Many researchers have studied single anchor localization,

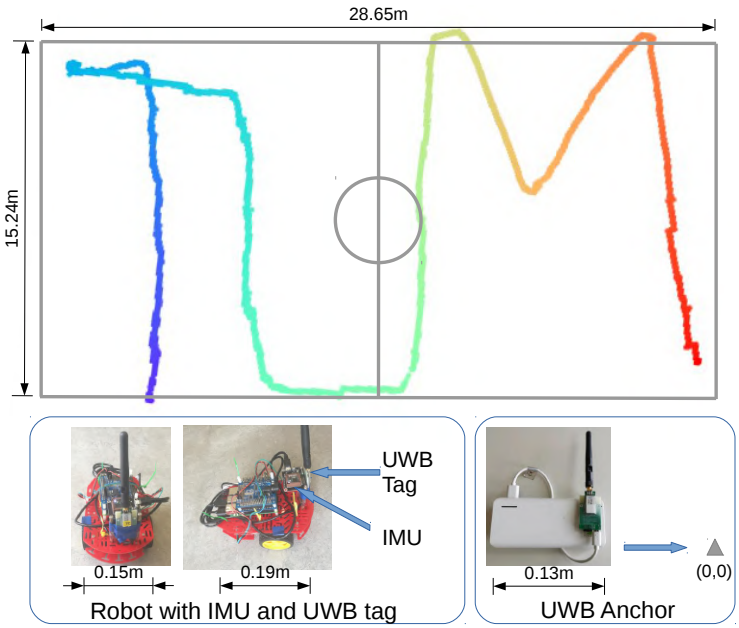


Figure 5.1 Trajectory of a real-world experiment with a differential wheel robot, Duckiebot. The robot only has IMU and UWB sensors. A UWB anchor is placed on (0,0) in the right bottom. There is no encoder or other velocity sensor used in the robot. The robot is controlled manually to moving a "TUM" like trajectory in a basketball pitch in TUM Garching campus.

especially for underwater robotics [79, 80]. Underwater robots usually use acoustic sensors, top-of-the-line IMUs, and expensive doppler sensors. Guo et al. [81] study a cooperative relative localization algorithm. They propose an optimization-based single-beacon localization algorithm to get an initial position for collaborative localization. However, they only observe a sine-like moving pattern and they require a velocity sensor. Similar with a recent work proposed by Nguyen et al. [82], they also use odometry measurements from optical flow sensors. In our study, we only use UWB and a low-cost IMU, dropping the need for a velocity sensor.

To better understand the single anchor localization problem, which is typically non-linear because of distance and angle, an observability study is necessary. Based on the groundwork of Hermann and Krener [83], researchers have studied the observability of range-only localization system, from one fixed anchor [80] to the relative range and bearing measurements between two mobile robots [84]. Bastista et al. [85, 86] used an augmented state to linearize the problem, enabling classical observability analysis methods. A recent study [87] explores the leader-follower experiment for drones with UWB ranging between robots, also with velocity measurement either from the motion capture system or optical flow. However, all these studies assume the velocities are available as a direct measurement, which we do not have.

Although we do not use velocity sensors, the system still needs velocity to become observable. Getting a reliable velocity from a low-cost IMU or UWB is challenging: the integration of acceleration drifts dramatically using low-cost MEMS IMU sensors [88, 89]. For position estimation, IMUs are often combined with other sensor measurements, like GPS, muti-anchor UWB [90], and cameras [91].

One straightforward way to estimate velocity is the distance change from a UWB anchor when the robot is moving along a radical line from the anchor. This situation is rarely lasting in reality, but the range changing pattern can be used as a speed estimator. We propose a method based on simple geometry relations under the assumption that the robot moves at constant velocity. The estimated speed coupled with data from the IMU gyroscope can provide a velocity estimate to keep the system observable. Finally, we use an EKF to fuse range, orientation, and velocity estimation to get the robot pose. The contributions for this paper are:

- a speed estimator using only UWB range information, which changes an unobservable system to observable;
- error analysis for the speed estimator to help design a sensor fusion algorithm;
- a loosely coupled tracking algorithm fusing IMU, UWB, and the proposed speed estimation;
- simulation and real-world experiments to validate our methodology.

5.3 Proposed method

5.3.1 System Definition and Observability Analysis

In this paper, we consider a robot moving on a 2D plane in proximity of a UWB anchor. The robot has a state vector $\mathbf{x} = [x, y, \theta, v, w]$, where x, y are the coordinates of the robot, θ is the heading, and v, w are linear and angular velocities. The system kinematics are described as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ w \\ a \\ b \end{bmatrix} \tag{5.1}$$

where a and b are linear and angular acceleration, respectively. The measurement functions are

$$\boldsymbol{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \sqrt{(x - x_A)^2 + (y - y_A)^2} \\ \theta \end{bmatrix} \tag{5.2}$$

where x_A, y_A are the coordinate of the UWB anchor. $h_1()$ is the range measurement function for the UWB sensor. $h_2()$ is a heading measurement function that takes the output orientation of a complementary filter, which fuses the measurements of the accelerometer, gyroscope, and magnetometer as the heading measurement of the IMUs.

In control theory, the observability of a system refers to the ability to reconstruct its initial state from the control inputs and outputs. For a linear time invariant system, if the observability matrix $[C|CA|...|CA^{n-1}]$ is nonsingular, the system is observable [169]. We refer to an approach by Hermann [83] using differential geometry to analyze the observability of non-linear systems.

To easily compare the impact of velocity, we extend the measurement functions (5.2) to a typical system that has linear and angular velocity measurements like (5.3), similar with [80]. In addition, we define the anchor position as the origin and map distance measurement d to $\frac{d^2}{2}$ to simplify $h_1(\mathbf{x})$ like in [170]. Then we have:

$$\boldsymbol{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ h_3(\mathbf{x}) \\ h_4(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(x^2 + y^2) \\ \theta \\ v \\ w \end{bmatrix} \tag{5.3}$$

We rewrite the model in (5.1) to the following format [83]:

$$\dot{\mathbf{x}} = \sum f^k(\mathbf{x}) u_k$$

with a state $\mathbf{x} = [x, y, \theta, v, w]$ and control input $u = [a, b]$. Then we extract a vector field $\boldsymbol{f}$ of the following functions on the state space:

$$\begin{aligned} f^0(\mathbf{x}) &= \Big[v\cos(\theta), v\sin(\theta), w, 0, 0\Big]^\top \\ f^1(\mathbf{x}) &= \Big[0, 0, 0, 1, 0\Big]^\top \\ f^2(\mathbf{x}) &= \Big[0, 0, 0, 0, 1\Big]^\top \end{aligned}$$

Next we find the Lie derivatives of the observation functions on state space along the vector field $\boldsymbol{f}$. The zero order Lie derivatives are, $L^0h_1 = \frac{1}{2}(x^2+y^2)$; $L^0h_2 = \theta$; $L^0h_3 = v$; and $L^0h_4 = w$, which are same as observation functions. Then we compute the first-order derivative as $L^1_{f^0}h_1 = xv\cos(\theta) + yv\sin(\theta)$; $L^1_{f^0}h_2 = w$; $L^1_{f^1}h_3 = L^1_{f^2}h_4 = 1$, and all the other first-order Lie derivatives, $L^1_{f^0}h_3 = L^1_{f^0}h_4 = L^1_{f^1}h_1 = L^1_{f^1}h_2 = L^1_{f^1}h_4 = L^1_{f^2}h_1 = L^1_{f^2}h_2 = L^1_{f^2}h_3 = 0$.

We write the observation space $\mathcal{G}$ spanned by $L^k_f h_i$, for $k = 0:1; f = 0:2; i = 1:4$. Note that all constant Lie derivatives are eliminated when computing the state derivative as $d\mathcal{G}$ in (5.4). Finally, we compute the state derivatives of space $\mathcal{G}$ and get:

$$\left[d\mathcal{G}\right] = \begin{bmatrix} o_{11} & o_{12} & 0 & 0 & 0 \\ o_{21} & o_{22} & o_{23} & o_{24} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

where $o_{11} = x$, $o_{12} = y$, $o_{21} = v\cos(\theta)$, $o_{22} = v\sin(\theta)$, $o_{23} = -xv\sin(\theta) + yv\cos(\theta)$, $o_{24} = x\cos(\theta) + y\sin(\theta)$, and $x, y \neq 0$ because $(0,0)$ is occupied by the anchor.

$d\mathcal{G}$ is with full rank when $v \neq 0$ and $y/x \neq \tan\theta$, which are reasonable assumptions. If the robot is static, it is difficult to locate the robot just from the range and the orientation. The second condition means that the robot moves along a radial line from the anchor, which is rarely lasting in practice. However, if we do not have velocity measurements, which is the situation we proposed, the fourth row of the state space becomes $\mathbf{0}_{1\times5}$. The dimension of space is reduced to four, and therefore the system does not meet the observability rank condition.

Therefore, velocity is crucial for system observability, and we estimate it from inertial measurements and UWB ranging.

5.3.2 Speed Estimation Model

By observing the range change pattern as a robot moves on a straight line, we propose a speed estimator based on simple geometric relations. As shown in Fig.5.2, three pairs of range and time measurements, (r_0, t_0) , (r_1, t_1) and (r_2, t_2) are given when the robot passes points A, B and C with a constant velocity. $r = OP$ is the virtual distance from the anchor to the motion line. $x = PA$ is the length from starting point A to the virtual intersection P. Based on the Pythagorean theorem, we can get the algebraic solution of the moving speed v:

$$\begin{bmatrix} {r_0}^2 \\ {r_1}^2 \\ {r_2}^2 \end{bmatrix} = \begin{bmatrix} r^2 + x^2 \\ r^2 + (x + v\Delta t_1)^2 \\ r^2 + (x + v\Delta t_1 + v\Delta t_2)^2 \end{bmatrix} \tag{5.5}$$

where

$$\Delta t_1 = t_1 - t_0; \Delta t_2 = t_2 - t_1$$

From these three functions, we can solve r, x and v. As we are interested only in the velocity, we just show:

$$v = \pm\sqrt{\frac{(r_2^2 - r_1^2) - \frac{\Delta t_2}{\Delta t_1} \cdot (r_1^2 - r_0^2)}{\Delta t_1 \Delta t_2 + \Delta t_1^2}}.$$

For simplicity, assuming the ranging measurements have a fixed frequency f, we have $\Delta t_1 = \Delta t_2 = \Delta t = 1/\mathsf{f}$. Then

$$v = \pm\sqrt{\frac{r_2^2 + r_0^2 - 2r_1^2}{2\Delta t^2}} \tag{5.6}$$

The positive value is the current speed (as the kinematics model shows the robot can only move forward). We present a simulation that includes ten stages, with different configurations of linear and angular velocities, as shown in Fig.5.3. The velocities change at the beginning of each stage and remain constant thereafter. The range measurements (cyan) is continuously fed into the estimator. As we can see, the velocity can be correctly provided (green). The changing pattern of the range in each phase suggests the speed value.

The peaks between stages are caused by velocity changes, and should be expected as the changes break the constant velocity assumption. Even if we cannot estimate the velocity

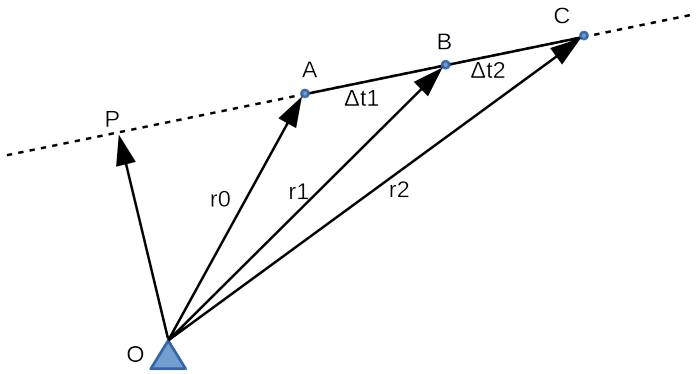


Figure 5.2 An illustration of our speed estimator. Three pairs of range measurements are used to calculate the speed.

correctly when it is suddenly changing, our algorithm does not lose its generality, as constant velocity motion is still the predominant motion in most real-world scenarios. One can estimate the speed over these periods and maintain a correct pose estimate. Furthermore, our sensor fusion algorithm also provides tolerance to velocity changes.

5.3.3 Speed Estimator Error Analysis

UWB ranging is considered as fairly accurate. This is true compared with WIFI or Bluetooth technologies that provide meter accuracy, but UWB can only achieve decimeter accuracy. For instance, DW1000 from Decawave [171] provides the accuracy of ± 10 cm using two-way ranging (TWR) time-of-flight (TOF) protocol, which is still too noisy to calculate the speed from range measurements directly. In this section, we analyze the error propagation for the speed estimator and design the speed estimation algorithm accordingly.

The range measurement model is expressed as $r = R + e$, where the measurement r is the true range R plus some noise e. We represent standard deviation as $\delta *$, e.g., δr for the standard deviation of r.

To determine the properties of δv from three range measurements with deviation δr, we compute the error propagation as in [172] (Ch4). For (5.6). We get:

$$(\delta v)^2 = \left(\frac{\partial v}{\partial r_0}\right)^2 (\delta r_0)^2 + \left(\frac{\partial v}{\partial r_1}\right)^2 (\delta r_1)^2 + \left(\frac{\partial v}{\partial r_2}\right)^2 (\delta r_2)^2$$

this can be rewritten as:

$$\begin{aligned}(\delta v)^2 = &\left(\frac{\sqrt{2} r_0 \Delta t}{\sqrt{r_2^2 + r_0^2 - 2r_1^2}}\right)^2 (\delta r_0)^2 \\ &+ \left(\frac{\sqrt{2} r_2 \Delta t}{\sqrt{r_2^2 + r_0^2 - 2r_1^2}}\right)^2 (\delta r_2)^2 + \left(2\frac{\sqrt{2} r_1 \Delta t}{\sqrt{r_2^2 + r_0^2 - 2r_1^2}}\right)^2 (\delta r_1)^2\end{aligned}$$

Since δr_0, δr_1, and δr_2 are measurements from the same sensor, we have $\delta r_0 = \delta r_1 = \delta r_2 = \delta r$, which is 0.2 for UWB sensors in our situation. Δt is a constant variable depending on the ranging update rate. Then we simplify the equation above as:

$$(\delta v)^2 = \frac{(2r_0^2 + 2r_2^2 + 8r_1^2)\Delta t^2}{r_2^2 + r_0^2 - 2r_1^2} (\delta r)^2$$

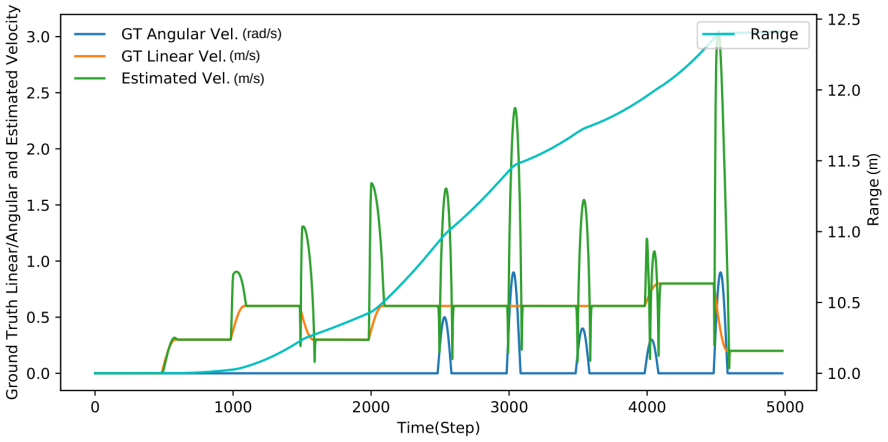


Figure 5.3 A simulation of speed estimation using noise-free ranging measurements to one anchor. The speed can be tracked correctly during constant velocity phases. Speed estimation produces erroneous peaks during speed change time, which can be avoided by filtering.

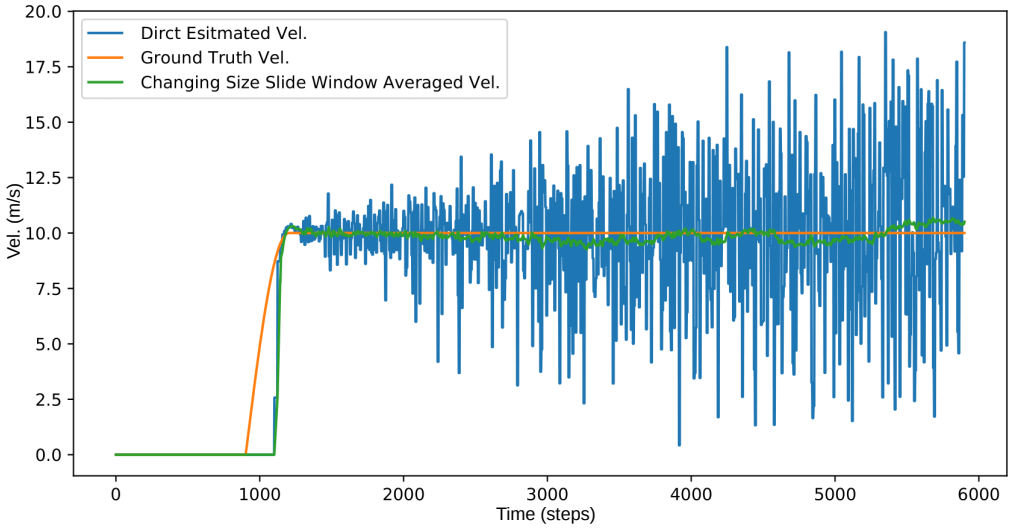


Figure 5.4 The velocity amplitude is estimated only from range measurements. The robot moves from the point (10,0) to (10, 250) at 10.0 m/s with a step of 0.01s. The anchor is located at (0,0). The deviation of the direct speed estimation (blue) increases with the range. Using a variable window size average filter, our speed estimator (green) gives the correct speed.

namely,

$$(\delta v)^2 = \left(2 + \frac{10r_1^2}{r_2^2 + r_0^2 - 2r_1^2}\right)(\delta r)^2 \Delta t^2 \tag{5.7}$$

From above equation, we can see if three measurements, r_0, r_1, and r_2, are similar, the denominator $(r_2^2 + r_0^2 - 2r_1^2)$ becomes small, and then the error becomes very large, a condition we want to avoid. Similar r_0, r_1, and r_2 can be caused by an excessively fast sample rate or by very slow motion.

To avoid the arrival of similar ranging measurements, we update the speed based on the ranging change, instead of updating at regular intervals as usual. In other words, when the range measurement difference exceeds a given threshold, the estimator is triggered to compute the speed. This way, the noise in the estimation can be effectively eliminated. Note that the update rate depends on the speed of the robot and also the direction of motion. If the robot moves quickly and on a radial line from the anchor, the range measurement difference quickly reaches the threshold and the update rate is high, and vice versa.

Furthermore, we can also see the standard deviation of the speed is positively correlated with r_1 in the numerator in (5.7). The value of r_1 is the intermediate measurement of the distance from the robot to the anchor. As Fig.5.4 shows, the deviation of the speed estimation (blue) increases as the range measurement increases. Thus, if the robot is very far away from the anchor, the standard deviation becomes large, leading to noisy speed estimation.

To solve this problem, we implement a variable window size filter, based on the distance from the robot to the anchor, to smooth the speed estimation. As Fig.5.4 shows, the deviation of direct speed estimation increases as the range increases. However, our variable window speed estimation can track the actual speed well.

5.3.4 Sensor Fusion System

The block diagram in Fig.5.5 shows our localization system. As mentioned above, the system has two kinds of sensors: UWB sensors for range measurement and low-cost IMUs, with gyroscopes, accelerators, and magnetometers, for orientation. The UWB range measurement is used in two pipelines: first, it goes to the EKF; at the same time, it is fed to a Kalman filter and then into the speed estimator. The speed estimation output goes into the EKF (red line). The robot heading is estimated by a complementary filter [173] that provides a quaternion estimation of the orientation using an algebraic solution from the observation of gravity and magnetic field. Finally, the range measurements, robot heading, and speed estimation are fed into the EKF to estimate the robot pose.

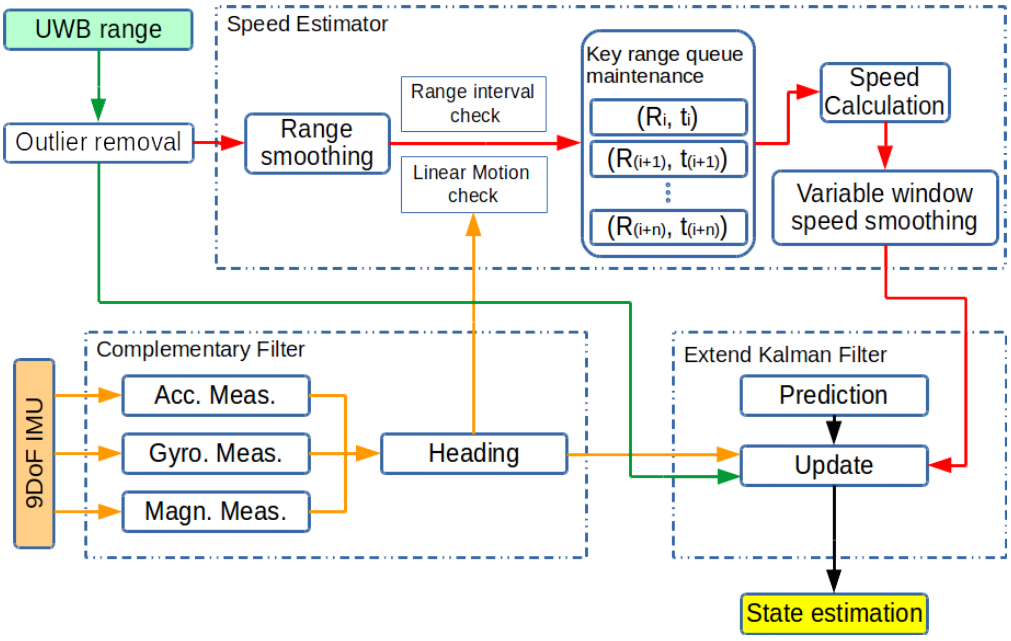


Figure 5.5 Sensor fusion system

Algorithm 5 outlines the approach in detail. Sensor readings from the IMU are fed into a complementary filter to obtain accurate heading in a non-magnetic field inferred environment, as shown in line 10. The UWB range and the computed heading are fed into an EKF to do a classical state estimation for position, heading, linear and angular velocities (lines 11-12 in Alg. 5).

The novelty of our system is the speed estimation, which corrects the velocity estimation in the EKF. As explained above, the UWB range measurements are fed into a separate Kalman filter to obtain smooth range measurements as $sRange$ (lines 13-14 in Alg.5). When a change in $sRange$ is greater than the threshold $TH_{distance}$ and the robot moves in a linear trajectory, the speed estimator will calculate the speed. This speed is then used to correct the velocity estimated from the EKF (lines 15-19). A variable window size filter is applied to get a smooth speed estimate result (lines 6-7) as discussed in 5.3.3.

5.4 Experiments

We validate our algorithm through simulations and real robot experiments. We used two types of robots: a DJI M100 quadcopter and a Duckiebot [174]. Robots are equipped with Decawave DW1000 [171] based UWB modules [175]. The quadcopter experiment specifically validated our speed estimation algorithm and gave quantitative localization error based on GPS. The ground mobile robot experiment shows that our algorithm can track very simple

Algorithm 5: State estimation algorithm from single UWB anchor.

input : $time, range_t, \boldsymbol{acc}_t, \boldsymbol{gyro}_t, \boldsymbol{magn}_t$
output: $\mathbf{x}_t$

$kRPs$ = ([range,timestamp],...) # keyRangePairs
Function `VelEstimator(`$range, time$`):`
    newKeyPair = $[range, time]$
    $kRPs.append(newKeyPair)$
    $velList \leftarrow calculate\_velocity(kRPs_{i-2}, kRPs_{i-1}, kRPs_i)$;
    $windowLen = range * ratio$
    $vel = average(velList[windowLen])$
    return vel
while *True* **do**
    $heading_t = complementaryFilter(\boldsymbol{acc}_t, \boldsymbol{gyro}_t, \boldsymbol{magn}_t)$
    $\hat{\mathbf{x}}_t$ = EKF.$predict()$
    $\mathbf{x}_t$ = EKF.$update(range_t, heading_t)$
    $\hat{sRange}$ = KF.$predict()$
    $sRange$ = KF.$update(range_t)$
    if $|\boldsymbol{gyro}_t| < TH_{linear\_motion}$ *and* $(fdRange - range_i) > TH_{distance}$ **then**
        $i = i + 1$
        $range_i = sRange$
        $vel = VelEstimator(fdRange, time)$
        $\mathbf{x}_t.vel = w * \mathbf{x}_t.vel + (1 - w) * vel$
    end
end

robots even in complex trajectories. From the error summary in Table 5.1, we can see our method has improved the accuracy by a factor 2 in simulations and real robot experiments. Please note we use RMSE for simulation and absolute trajectory error (ATE) for real robots experiments because we have exact ground truth from simulations, but not for real robot experiments.

5.4.1 Simulation

In our simulations, we generate a random trajectory with a differential wheel robot kinematics model [176]. There are five stages of motion, with different headings and speed settings. The anchor is set at position (0,0) and the robot starts at the point (10,0). The range measurement is corrupted by white Gaussian noise with 0.2 m standard derivation, which is similar to the actual UWB measurement derivation. The noise added to orientation is with a deviation of 0.1 rad.

As the top of Fig. 5.7 shows, our method (green) can track the ground truth velocity (gray) most of the time, which results in an accurate trajectory in Fig.5.6. However, the vanilla EKF model (red) cannot recover from the drift errors accumulated during the first stage. This proves that our algorithm can correct the accumulated error as long as the speed estimator gives accurate estimations. More specifically, compared with the bottom figure in Fig. 5.7, the RMSE of our method drops rapidly (starts from 1200 steps) when the speed estimation is available (around 1200 steps). We reduce the RMSE of the vanilla EKF by more than 70% and achieve the accuracy of 0.48 m, which is impressive given the limited information.

5.4.2 Speed Estimation for Flying Robot

We have used a quadcopter (DJI M100) to validate our tracking and speed estimation in the real world, which also illustrates the potential use for 3D applications as well. We programmed a triangle trajectory with a speed parameter of 2 m/s for the M100. First, we compare the estimated results with the velocity feedback from DJI_SDK software. Fig.5.9 shows our algorithm can give correct speed estimation (red) based on range measurements (cyan). Then we calculate ATE between our estimated trajectory and GPS trajectory. As Fig.5.8 and table 5.1 show, the trajectory from our algorithm is much closer than that of the

Table 5.1 Error comparison between EKF with or without speed estimator.

Error (m)	Simulation (RMSE)	Drone Exp. (ATE)
Without speed estimator	1.73	2.81
With speed estimator	0.48	1.05

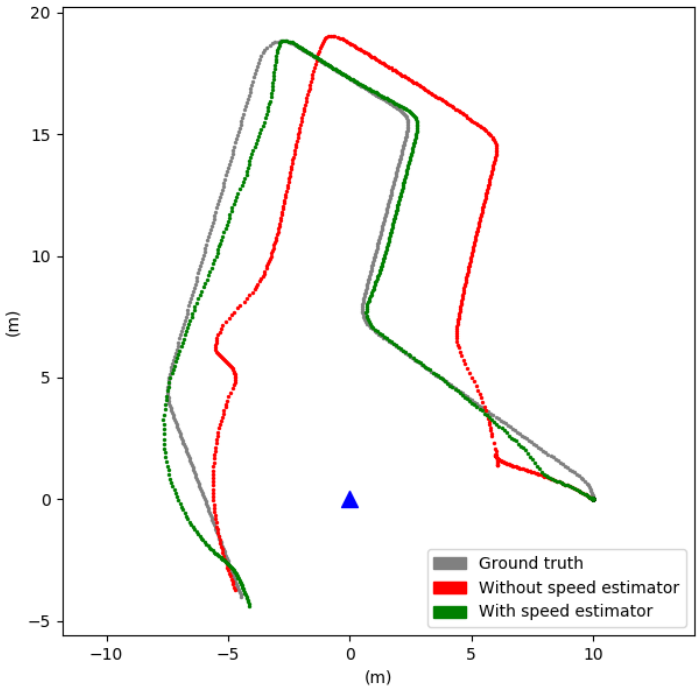


Figure 5.6 The trajectories of EKF with our speed estimator (proposed method, in green) and without speed estimator (vanilla EKF, in red), refer to the ground truth trajectory (gray).

EKF without our speed estimator. With only one anchor setup, we get around 1 m ATE error, which is much higher than the vanilla EKF with the accuracy of 2.8 m. Please also note that the DJI velocity feedback in Fig.5.9 only works as reference, instead of ground truth: for the first ten seconds, the robot is hovering but the speed indicated from DJI_SDK is around 0.3 m/s.

5.4.3 Tracking of Ground Mobile Robot

The ground robot platform is a simple indoor robot, the Duckiebot [174]. Our version of the Duckiebot has an RPi2 on-board computer. We add a 9DOF IMUs sensor and a UWB module to the robot, as shown in Fig.5.1. The robot does not have wheel encoders to get the speed or displacement. We manually control the robot to run a university logo trajectory (TUM) in an outside basketball court. We use our localization system to track the robot pose. Fig.5.1 shows that the algorithm can track the trajectory correctly. This experiment shows a qualitative evaluation, indicating that our algorithm can be easily applied to simple robots and IoT devices.

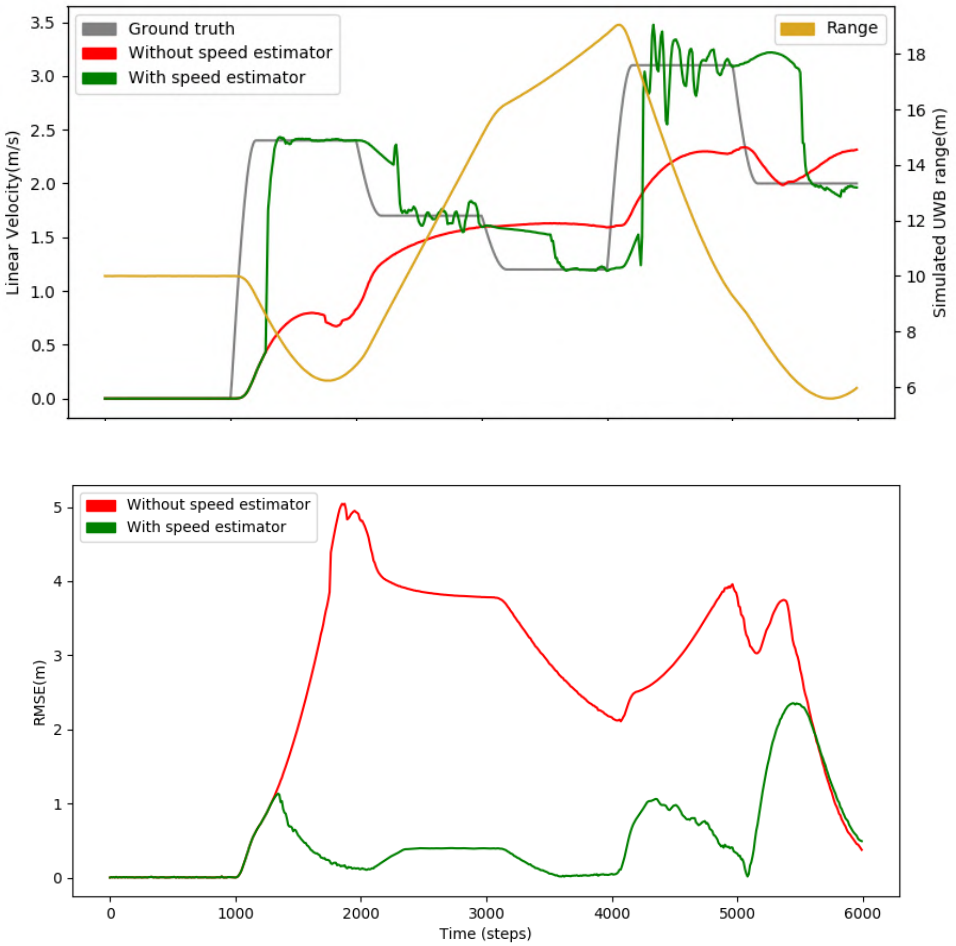


Figure 5.7 The top figure shows the velocity estimations of our method and vanilla EKF. We can see that although there are some delays, our algorithm can track the actual speed most times. However, the EKF without speed estimator can only catch up the trend of speed. The bottom figure is the comparison of RMSE. The error of our algorithm drops rapidly when the correct speed estimation is available.

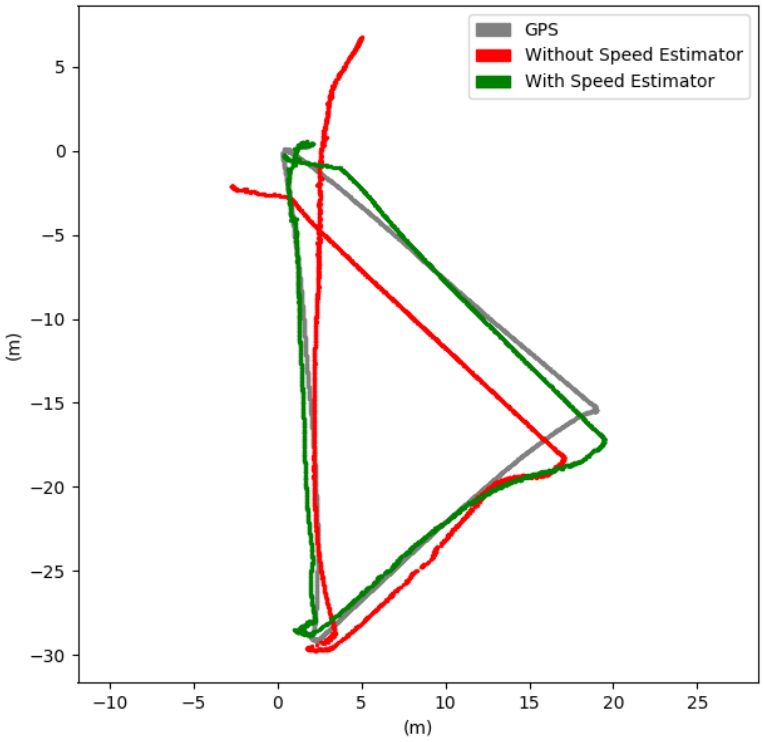


Figure 5.8 Experiments with a DJI M100 quadcopter. The robot is programmed to fly in a triangle trajectory. The anchor is placed on the ground. Our trajectory is much closer to the GPS trajectory than the vanilla EKF without speed estimator.

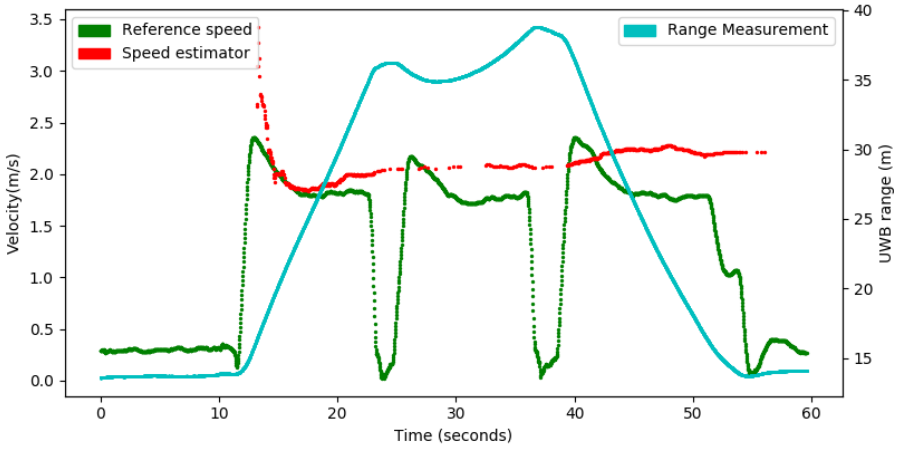


Figure 5.9 In the quadcopter experiment, we can estimate the speed (red) simply by using the range measurement (cyan). The reference from the DJI_SDK is plotted in green.

5.5 Conclusion and Discussion

In this paper, we propose a localization algorithm for robots that are equipped with low-cost IMUs and UWB sensors in an environment configured with only a single UWB anchor. We estimate the speed from UWB range changes, which makes the system temporally observable. Our algorithm effectively reduces the accumulated errors by 60%. With this algorithm, a large number of devices can be localized, including IoT devices or cellphones with IMU and UWB sensors.

Erratum: In page 89, paragraph 2, the second sentence "For a linear time invariant system, if the observability matrix $[C|CA|...|CA^{n-1}]$ is nonsingular, the system is observable [169]." is replaced by "For a linear time invariant system, if the observability matrix $[C|CA|...|CA^{n-1}]$ is full rank, the system is observable [169].".

CHAPTER 6 ARTICLE 4: VIR-SLAM: VISUAL, INERTIAL, AND RANGING SLAM FOR SINGLE AND MULTI-ROBOT SYSTEMS

Preface: In this chapter, we propose a system for robots equipped with IMU, UWB, and a monocular camera. With the same goal with last chapter, improve the odometry accuracy, this chapter builds on more sophistic method of SLAM. This chapter propose a different strategy to integrate the UWB ranging measurements into a SLAM frame, which achieve great improvement of odometry accuracy. Furthermore, the UWB ranging can assist the multi-robot system to acquire relative information to merge their trajectories and maps.
Authors: Yanjun Cao and Giovanni Beltrame
Submitted to: Autonomous Robots, Springer
Personal Contributions: Y. Cao conceived of the presented ideas, developed the theory, designed and carried out the simulations and the real robots experiments. Y. Cao wrote the manuscript with support from G. Beltrame.

Abstract

Monocular cameras coupled with inertial measurements generally give high performance visual inertial odometry. However, drift can be significant with long trajectories, especially when the environment is visually challenging. In this paper, we propose a system that leverages Ultra-WideBand (UWB) ranging with one static anchor placed in the environment to correct the accumulated error whenever the anchor is visible. We also use this setup for collaborative SLAM: different robots use mutual ranging (when available) and the common anchor to estimate the transformation between each other, facilitating map fusion. Our system consists of two modules: a double layer ranging, visual, and inertial odometry for single robots, and a transformation estimation module for collaborative SLAM. We test our system on public datasets by simulating UWB measurements as well as on real robots in different environments. Experiments validate our system and show our method can outperform pure visual-inertial odometry by more than 20%, and in visually challenging environments, our method works even when the visual-inertial pipeline has significant drift. Furthermore, we can compute the inter-robot transformation matrices for collaborative SLAM at almost no extra computation cost.

6.1 INTRODUCTION

Robot localization is a fundamental topic in any mobile robot application. Recent advances in robot hardware and software have boosted the opportunity and demand for multi-robot systems for their inherent benefits, such as high efficiency and robustness.

With a monocular camera and low-cost inertial measurement units, Visual Inertial Odometry (VIO) is accepted as the minimal sensor configuration for single robots state estimation and navigation considering size, wight, power and cost [20]. Recent technical advances [21–23] make VIO more and more robust and stable in many conditions. However, the drift caused by accumulated error is still hard to control without loop closures. Although loop closure is a natural part for SLAM system, the requirements to close the loop rely much on the trajectory and environment. For example, generating high-quality closures requires revisiting the same location with a similar viewpoint. Furthermore, perception outliers caused by illumination, self-similar environments, etc. are challenging for loop closure. In addition, as the nature of loop closure is to distribute the accumulated error alone the trajectory, proper maintenance of the history key frames is needed to recovery accurate trajectory [25].

In this paper, we use a single extra sensor, a static Ultra-WideBand (UWB) anchor, to improve the performance of robot localization. UWB technology has attracted a lot of attention recent years for its accurate ranging performance and long-distance support. For example, the latest Apple IPhone at the time of writing is equipped with a UWB chip (actually, it includes all sensors needed in this paper). Most available UWB systems (e.g. [177]) use several (at least four for 3D and three for 2D) calibrated anchors as a Global Positioning System (GPS) for specific areas. This type of infrastructure is not applicable for the exploration in unknown environments, which is one of the primary objectives of SLAM. Therefore, we design our system to rely only on one anchor, which can be dropped off at any moment, even in an unknown location, by a robot during its mission. Our experiments show that this one anchor can improve the localization accuracy significantly.

This setup has another significant benefit for multi-robot SLAM. One issue in multi-robot SLAM is the estimation of relative transformation matrix between robots [114]. Most available works rely on common features to extract the relative pose of robots, which is a type of inter-robot loop closure, which leads to similar constraints as single robot loop closures. An additional challenge for inter-robot loop closure is the need to exchange the information required for loop closure among all robots, which can be significant.

In our system, we can solve this challenge using the UWB sensor. When any two robots move into their respective UWB ranging radii, they can exchange their anchor measurements and do

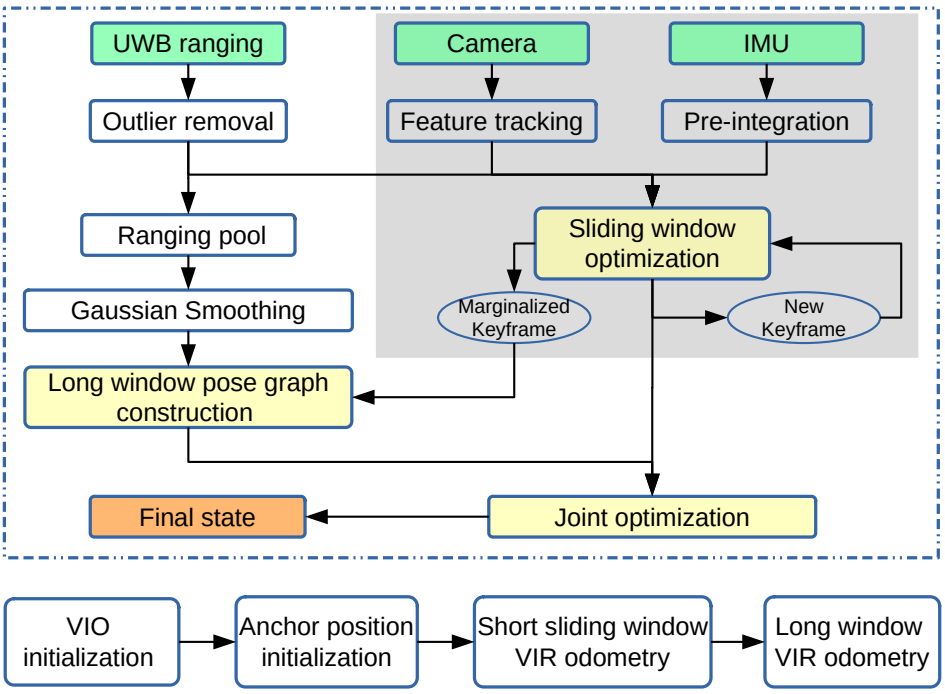


Figure 6.1 System overview. The system is built on a state-of-the-art VIO algorithm (shaded in gray). We apply the UWB sensor to get drift-free odometry with the help of a single anchor placed arbitrarily in the environment. More than directly integrating the UWB ranging measurement into the *Sliding window optimization*, we also collect the *Marginalized key frame* into the *Long window pose graph construction* for a *Joint optimization*. The running process of the system is shown at the bottom. After *VIO initialization*, robots start the *Anchor position initialization* and then execute the *VIR odometry* or *Long window VIR Odometry* when the long window pose graph is available.

mutual ranging. Using this information, the robots can estimate their respective coordinate system transformations. Having the transformation matrix, a robot can correctly project all the information from its neighbors onto its own frame. This solution greatly simplifies multi-robot SLAM, minimizing the need for inter-robots loop closure, which requires the exchange of feature databases, the identification of loop closures and distributed pose graph optimization.

The last but not the least reason for choosing UWB is its inherent advantage in data association. When a range measurement is received, the identity of the sender is recognized without extra effort, which can be of great assistance for multi-robot systems. Moreover, UWB can provide modest communication bandwidth while ranging. For example, the state estimation of poses can be carried by UWB packets while performing ranging. In the future, we believe a monocular camera, inertial measurement units (IMU), and UWB will be a standard minimal configuration for multi-robot systems.

6.2 RELATED WORK

6.2.1 Single Robot SLAM with Visual, Inertial and Ranging Measurements

SLAM has been the subject of intense research for more than thirty years [94]. Monocular visual-inertial odometry is a popular choice as it provides good state estimation performance with a minimal sensor configuration. Although state-of-the-art VIO algorithms (e.g. SVO [22], VINS-Mono [23], DSO [105]) can reach very high accuracy in relative translation and orientation, the accumulated drift can still be an issue: any small orientation error can lead to large end-point error. Our system leverages UWB ranging measurements to correct the accumulated error. We developed our system based on VINS-Mono [23], which is a robust and versatile state estimator which uses a sliding window tightly coupled nonlinear optimization for visual and IMU measurements.

UWB technology, as a localization solution on its own, has attracted a lot of attention in recent years both in research and industry for its decimeter localization accuracy. However, most results are based on a well-calibrated multi-anchor setup [54, 106–109], which is not applicable for navigation in unexplored, unstructured environments. Single anchor setup is desired as the easy deployment [29]. Wang et al. propose a system using a monocular camera, IMU and UWB to bypass the complexity of loop closure [110] . However, they still use multiple preconfigured UWB anchors. Their UWB module provides coarse drift-free global position and VIO identifies the local trajectory. On the contrary, in this paper we use only one anchor, placed in an unspecified location. [111] present an idea similar to to ours: they start with one UWB anchor and keep dropping anchors from a moving robot. Unfortunately, their experimental results are available only in simulation for one sequence of the EuRoC dataset [112], with five simulated anchors. A closely related recently published work [113] combines monocular camera with UWB ranging to a single anchor. The system is developed based on ORB-SLAM [25] and use UWB ranges to estimate the scale after recording the first batch of data. In our case, we also integrate the ubiquitous IMU and effectively fuse accurate VIO and range constraints along the robot's trajectory.

6.2.2 Multi-Robot SLAM

Multi-robot SLAM has gained recent attention for the increased viability and accessibility of multi-robot systems. A review work [114] gives a comprehensive survey of multi-robot SLAM and points out one key issue: relative pose estimation. Most current multi-robot SLAM systems solve this issue by analyzing inter-robot loop closures, either in centralized [115] or distributed [116, 117] fashion. The distributed approach is more robust, but it is harder to implement in practice: robots need to exchange map data to get the feature database for

future loop closures, and distributed optimization usually requires additional communication and computation. Ranging measurements can assist in relative pose estimation. Trawny et al. provide theoretical proofs and simulations that show how six range measurements can be used to get the transformation matrix between two robots [6]. [57] adapt it to 4DOF relative pose estimation with a UWB setup, and use it for merging maps for VR applications. Both methods require mutual ranging measurements over long trajectories. In our solution, robots can estimate the transformation matrix as soon as they can get two measurements from their neighbors, which meets the requirement of real-time transformation estimation during robot rendezvous. These two methods [6, 57] represent a good solution when no common anchor is present (or is not measured) and can be combined with inter-robot loop closures to improve the transformation results. Recent work [58] presents a decentralized Visual-Inertial-UWB fusion for relative state estimation: the authors combine VIO, UWB and vision detectors of YOLOv3 [118] to track the relative state of neighbors.

In practice, many works in literature have explored the combination of multiple UWB anchors with vision and IMU. In this paper, we focus on the use of a single anchor in an arbitrary location, which is trivial to deploy as a beacon in real exploration tasks. We propose a double layer sliding window technique to combine VIO with UWB ranging, which produces drift-free state estimation by leveraging VIO for its accurate short time relative pose estimation, and range constraints for longer trajectories. Moreover, recorded anchor ranges can help robots find the transformation matrix efficiently with only two range measurements in multi-robot settings. Our contributions can be summarized as:

- a UWB-aided SLAM system for single robots which outperforms pure VIO and has drift-free odometry estimation;
- a double layer sliding window algorithm that combines relative pose estimation from VIO and UWB ranging constraints;
- an efficient method to estimate the transformation matrix between multiple robots.

6.3 SYSTEM DESIGN

In this section, we explain the preliminaries and symbols used in this paper. We then detail the formulation of our ranging-aided visual inertial SLAM, focusing on the cost factors of the optimization problem. We also explain the techniques to tackle the noise of UWB measurements. Finally, we introduce our solution for the estimation of the transformation matrix between robots.

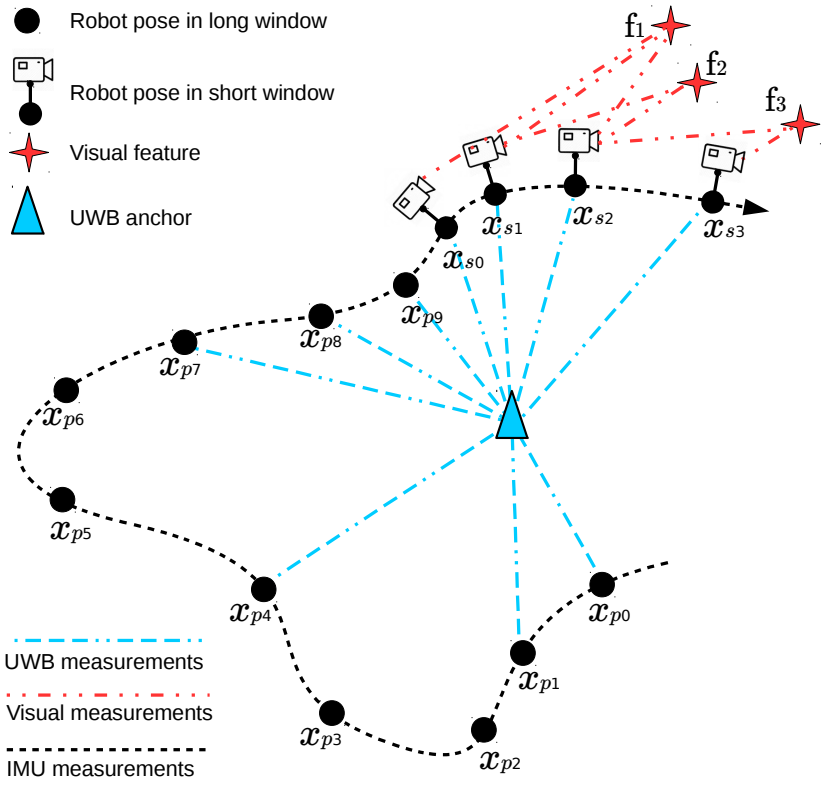


Figure 6.2 Illustration of the scenario of VIR-SLAM, with a single anchor setup. Measurements from sensors are shown with different lines. Robot poses with the camera and IMU measurements belong to the short sliding window. All poses associated with UWB ranging to the anchor are kept for the long sliding window.

6.3.1 Single Robot Estimation Preliminaries

We assume a robot carries three kinds of sensors (a monocular camera, an IMU and UWB module), and moves in a 3D environment, as shown in Fig. 6.2. A UWB anchor is placed in the environment with an unknown initial position. We define the world frame of the robot i as $[\ ]^{iw}$, which is usually aligned with the first camera frame when the robot starts its mission. The position of the anchor is expressed in the robot world frame, denoted by $\mathbf{P}_A^{iw}$. We use $[\ ]^{ib}$ to indicate the body frame of robot i, and similarly $[\ ]^{ic}$ for the camera frame. Note that we do not define the UWB sensor frame because it is a scalar measurement. The UWB ranging measurement is transferred to body frame by considering the 3D offset of the UWB antenna in the body frame.

Classical VIO proposes an optimization formulation of states over a sliding window with size n as:

$$\begin{aligned} \mathcal{X} &= [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, l_0, l_1, \ldots, l_m] \\ \mathbf{x}_k &= [\mathbf{p}_{b_k}^{iw}, \mathbf{v}_{b_k}^{iw}, \mathbf{q}_{b_k}^{iw}, \mathbf{b}_{a_k}^{ib}, \mathbf{b}_{w_k}^{ib}] \qquad k \in [0, n] \end{aligned} \tag{6.1}$$

which includes the state $\mathbf{x}$ for all n frames and the visual feature inverse depth l. The kth frame state $\mathbf{x}_k$ includes the position $\mathbf{p}_{b_k}^{iw}$, velocity $\mathbf{v}_{b_k}^{iw}$, and orientation in quaternions $\mathbf{q}_{b_k}^{iw}$ for robot i in its world frame, plus accelerometer bias $\mathbf{b}_{a_k}^{ib}$ and gyroscope bias $\mathbf{b}_{w_k}^{ib}$ in its body frame. l_i is the inverse depth of the ith feature among m features from the visual observations over the sliding window. If the sliding window includes all the camera frames since the start of the mission, the optimization becomes a full smoothing estimation. Although full smoothing offers the best accuracy, it is not scalable in reality, so we follow the key-frame approach that discards similar frames while not losing tracking [25,101]. However, as the trajectory becomes longer, the size of state keeps growing. For this reason, the old key frames are marginalized into a prior factor in the optimization [23,104]. The drift from visual inertial odometry is still hard to avoid, and pose graph optimization with loop closure becomes necessary to correct the accumulating error.

To obtain an accurate localization system while avoiding to rely on loop closures, we design a SLAM system that uses a novel double layer sliding window structure, with an implementation based on VINS-Mono [23] (shown by a gray area in Fig. 6.1).

6.3.2 Double Layer Tightly Coupled VIR Optimization

We propose a double layer sliding window tightly coupled SLAM optimization by considering following aspects: high accuracy relative pose estimation from VIO, less accurate but absolute

UWB measurements, and the computation cost for these factors. As shown in Fig. 6.1, the ranging measurements are fed into the *Sliding window optimization* in the standard VIO optimization. As the number of variables to optimize increase greatly with the number of key frames in the sliding window, the computational resources limit the choice of sliding window size. We keep the same size as the standard VIO sliding window for the visual and inertial measurement optimization. However, we apply an extra UWB constrain to the *Marginalized key frames* to leverage the distance constraint along the trajectory, as shown in Fig. 6.1. A *joint optimization* combining standard *sliding window optimization* [178] and the proposed *long window pose graph* compute the final odometry estimation.

Three kinds of variables are involved in our two sliding windows optimization:

$$\mathcal{X} = [\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_s, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, l_0, l_1, \ldots, l_m]$$

A short window Ψ, the same as the classical visual SLAM sliding window with size n from Equ.6.1. The variables in this window include $\mathbf{x}_k, k \in [0, n]$ and $[l_0, l_1, \ldots, l_m]$, which have the same meaning as the variables in Equ. 6.1.

$$\mathbf{x}_k = [\mathbf{p}_{b_k}^{iw}, \mathbf{v}_{b_k}^{iw}, \mathbf{q}_{b_k}^{iw}, \mathbf{b}_{a_k}^{ib}, \mathbf{b}_{w_k}^{ib}] \;\; k \in \Psi[0, n]$$

The novelty in our system lies in the addition of another long sliding window Ω, which carries state $\mathbf{w}_t$:

$$\mathbf{w}_t = [\mathbf{p}_{b_t}^{iw}] \;\; t \in \Omega[0, s]$$

The size of the long sliding window is s, which is much larger than n. The state in this window only contains the robot position $\mathbf{p}_b^{iw}$ in the robot world frame. Fig.6.3 shows the factor graph of our system, corresponding to the scenario of Fig. 6.2. The short window (in the gray area) includes the state from the camera and IMU measurements. The long window (orange area) contains the state from the UWB measurements. Following the state definition, we formulate a full nonlinear optimization problem as:

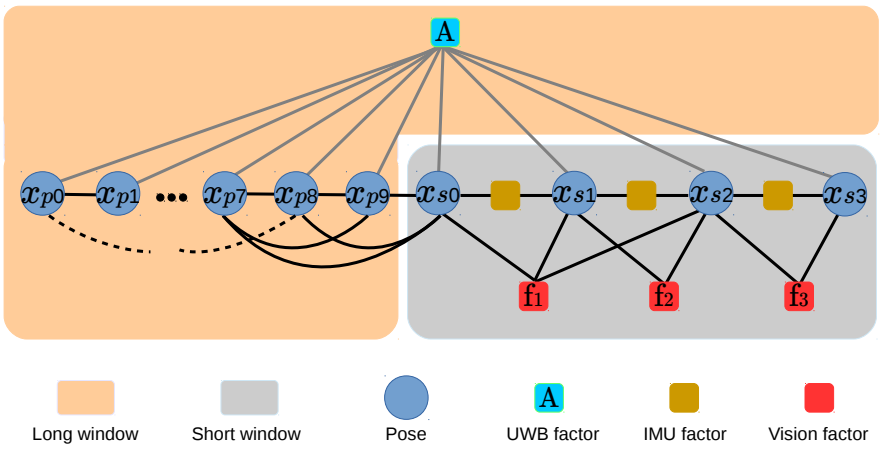


Figure 6.3 Factor graph for the scenario in Fig. 6.2. The factor graph includes poses, UWB factor, IMU factors and visual factors. The edges are the error. We have a UWB factor for the poses with range measurements. Further, smooth edges (curves) are added between poses.

$$\min_{\mathcal{X}} \Bigg\{ \underbrace{\sum_{t\in\Omega} \|\mathbf{r}_{\mathcal{U}}(\hat{\mathbf{z}}_t, \mathcal{X})\|^2_{\mathbf{P}^{iw}_{b_t}}}_{UWB\ factor} + \underbrace{\sum_{k\in\Psi} \|\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X})\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}}}_{IMU\ factor} + \underbrace{\sum_{(l,j)\in\mathcal{C}} \rho(\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_l, \mathcal{X})\|^2_{\mathbf{P}^{c_j}_l})}_{Vision\ factor} \Bigg\} \tag{6.2}$$

This nonlinear optimization problem considers three factors, corresponding to the UWB factor, the IMU factor, and the vision factor. The UWB residuals are calculated for the long window Ω, while the IMU and visual residuals for the short window Ψ.

UWB Factor

UWB localization uses different protocols: time of arrival (TOA), time difference of arrival (TDoA), and two-way ranging (TWR). In our system, we use TWR, which measures the distance between two transceivers by sending a packet back and forth. Although the number of nodes supported is limited because of the shared UWB communication medium, TWR can be used without device synchronization, which makes the protocol widely used. Since we are not considering hundreds of robots communicating simultaneously and we would like to avoid synchronization (which can be difficult to implement in distributed multi-robot applications), we choose TWR as our ranging mode. Similar to [106, 179], we model the

ranging measurement of UWB modules as:

$$\hat{d} = d + f(d) + e \tag{6.3}$$

where $\hat{d}$ is the UWB measurement, d is true distance, $f(d)$ is distance based bias, and e is the error following a Gaussian distribution $N(0, \sigma^2)$. Considering the properties of UWB measurements, we apply an outlier rejection and smoothing technique before using them, (see Section 6.3.2). With this UWB model, we define the UWB factor in Equ. 6.2 as:

$$\begin{aligned} \mathbf{r}_{\mathcal{U}}(\hat{\mathbf{z}}_t, \mathcal{X}) = & \underbrace{\gamma_r \cdot (\|\mathbf{p}_{b_t}^{iw} - \mathbf{P}_A^{iw}\| - \hat{d}_t)}_{UWB\ ranging\ residual} \\ & \underbrace{+ \gamma_s \cdot (\sum_{j \in (t, t+s]} \{\mathbf{p}_{b_j}^{iw} - \mathbf{p}_{b_t}^{iw}\} - \tilde{\mathbf{z}}_{b_t}^{b_j})}_{Relative\ transformation\ residual} \end{aligned} \tag{6.4}$$

The UWB factor above includes two residuals: a ranging measurement residual and a virtual relative transformation measurement residual, with weights γ_r and γ_s, respectively. $\hat{d}_t$ are the UWB ranging measurements, which are compared with the predicted distance from the robot body frame t to the anchor $\mathbf{P}_A^{iw}$ in the world frame. To avoid the ranging factor excessively affecting the optimizer and breaking the relative pose property from visual-inertial estimation, we introduce a virtual relative transformation measurement $\tilde{\mathbf{z}}_{b_t}^{b_j}$ between frames t and $j \in (t, t+s]$, which is extracted from the short sliding window estimation result. We select $s = 3$ in our experiments to create two consecutive links between poses, as can be seen for example with pose x_{p7} in Fig. 6.3.

UWB Ranging Integration

Although the UWB measurement model in Equ. 6.3 holds true in theory, [180] pointed out that real-world environment is much more complicated. Barral et al. have tested the systematic bias in a semi-open environment (a sports hall) and showed the multipath impact on the bias. In a more complicated environment with walls, ceilings, obstacles that partially absorbs the transmit power, or even not in line of sight, large estimation errors can occur [181]. Therefore, we adhere to the basic distance-related bias model, but combine outlier rejection and smoothing techniques to reduce the measurement noise.

To calibrate the systematic bias $f(d)$, we first collect several datasets within (0,40 m) in the atrium of our building (the configuration can be partially seen in Fig. 6.10). One of the sequences is shown in Fig. 6.4. The ground truth distances are measured by laser range finder in the static periods, corresponding to the horizontal segment in Fig. 6.4. We used an

exponential model regression to find out the bias relation, similar to [106].

Many outliers can be encountered indoors, as the magnified part in Fig. 6.4 shows (inside the orange box). This is very likely due to multipath or two-ray ground reflection [182]. To remove these outliers, we propose an outlier rejection algorithm inspired by [183]. Considering the multi-path phenomenon will only produce outliers with higher value than the true distance, we track the lower limit of the most recent measurements in a range buffer. By applying an acceptable bound considering the maximum motion of the node, we can reject the outliers. Unlike the algorithm in [183], where the authors use a detailed motion model of the sensor to select two valid previous measurements, our algorithm only needs a basic estimation of the maximum speed. Fig. 6.5 shows the result of applying the outlier rejection module for the noisy period in Fig. 6.4 (from 162s to 180s). We can see the accepted measurements (orange dots) contains only the bottom envelope of the signal: peaks and large measurements are rejected as outliers.

Another technique we use to reduce the error in this pre-processing stage is through smoothing. We apply Gaussian smoothing [184] to all accepted measurements for the duration of the long window. The smoothing result can be seen in Fig. 6.5. For the gap in the measurement timeline, we interpolate the points based on the adjacent ends if they are close in time. The smoothed value corresponding to the key frame timestamp is used to create the UWB factors in the long window.

IMU Factor

IMU measurements are critical for monocular visual odometry. As the frequency of the IMU is usually higher than the camera image frame rate, the IMU measurements are preintegrated between two consecutive image frames [21]. By referring to the last body frame motion, this technique avoids repeating the IMU reintegration and reduces computation during optimization. Forster et al. extend this approach to manifold structures of the rotation group $\mathfrak{S}o3$ for higher accuracy and robustness [22]. We follow the same method as [23] in quaternion form.

The preintegrated IMU measurements between two consecutive frames of b_k and b_{k+1} referring

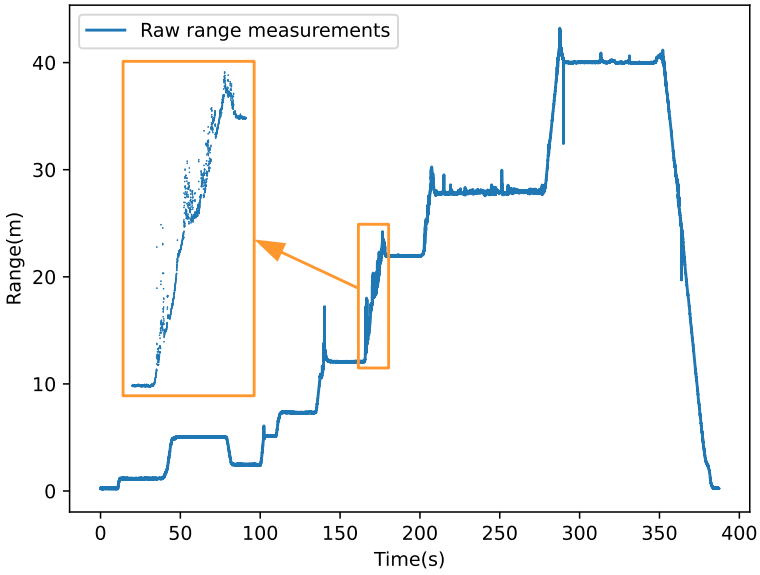


Figure 6.4 A sequence of raw ranging measurement to estimate the distance based UWB ranging bias. A laser range finder is used to find the ground truth distance for the static periods (horizontal segments). This figure shows an abnormal measurement happened in the magnified period. This can happen in complicated indoor environment with multipath radio wave propagation.

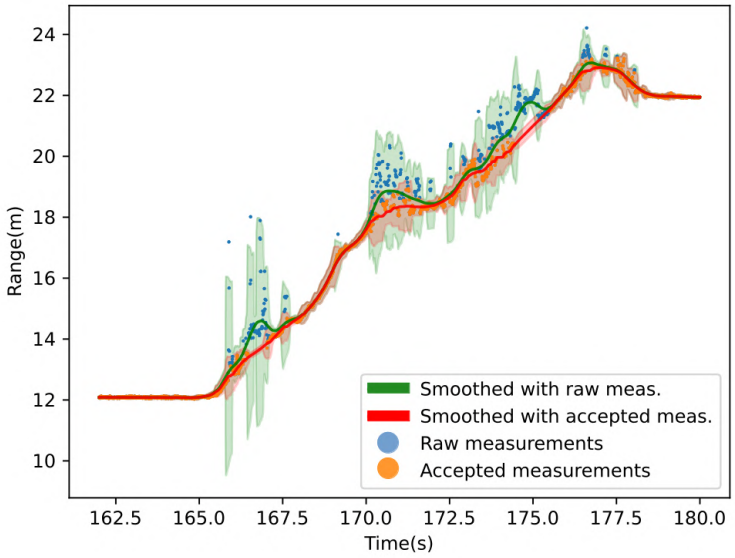


Figure 6.5 An example of outlier rejection and smoothing pre-processing for the measurements in the abnormal period in Fig. 6.4. Filtered by the outlier rejection module, the accepted values are the value that is small, considering the fact that multipath causes bigger measurement than the true distance. Then we smooth the accepted measurements with a Gaussian kernel before applying them to the UWB factors. Shaded areas are $\pm 3\sigma$ for the raw measurement (green) or the accepted measurements (orange).

to frame b_k can be expressed as:

$$\begin{aligned}
\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \iint_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}\left(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}\right) dt^2 \\
\boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}\left(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}\right) dt \\
\boldsymbol{\gamma}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}\left(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t}\right)\boldsymbol{\gamma}_t^{b_k} dt
\end{aligned}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$\hat{\mathbf{a}}_t$ and $\hat{\boldsymbol{\omega}}_t$ are the accelerometer and gyroscope measurement vectors, respectively. These three formulas correspond to relative the motion changes of position, velocity and orientation to the local body frame of b_k.

The IMU factor is the residual between predicted motion and the preintegrated results referring to the body frame:

$$\mathbf{r}_{\mathcal{B}}\left(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}\right) = \begin{bmatrix} \delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\beta}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\theta}_{b_{k+1}}^{b_k} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \mathbf{p}^{b_k} - \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \mathbf{v}^{b_k} - \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \boldsymbol{\theta}^{b_k} \otimes (\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k})^{-1} \\ \mathbf{b}_w^{b_{k+1}} - \mathbf{b}_w^{b_k} \\ \mathbf{b}_a^{b_{k+1}} - \mathbf{b}_a^{b_k} \end{bmatrix}$$

where $\mathbf{p}^{b_k} = \mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k\right)$, $\mathbf{v}^{b_k} = \mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w\right)$ and $\boldsymbol{\theta}^{b_k} = \mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^w$ are the estimated position, velocity and orientation referred to the local body frame of b_k (see appendix in [23] for details).

Vision Factor

The vision factor consists of the reprojection error for the tracked features. We compare the reprojection of all features in the current frame with their first observations. The visual residual is as

$$\mathbf{r}_{\mathcal{C}}\left(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}\right) = \|\mathbf{u}_{l_k}^{c_j} - \pi(\mathbf{R}_{c_i}^{c_j}, \mathbf{T}_{c_i}^{c_j}, \mathbf{P}_{l_k}^{c_i})\|$$

where $\mathbf{u}_{l_k}^{c_j}$ is the coordinate of features l_k in the image of the camera frame j, $\pi()$ is the projection that converts homogeneous coordinates into image coordinates, $\mathbf{R}_{c_i}^{c_j}, \mathbf{T}_{c_i}^{c_j}$ represent

the frame transformations (rotation and translation) from the camera frame i to j, which are inferred from state poses, and $\mathbf{P}_{l_k}^{c_i}$ is the 3D position of the kth feature in the first observation frame i. The vision factor iterates through all the frames and all the tracked features in the estimated state.

Anchor Position Estimation

In the above discussion, we assume that the anchor coordinates are available to the optimizer. As the anchor position is initially unknown, it needs to be estimated. Therefore, the state vector to estimate at the start is $\mathcal{X} = [\mathbf{P}_A^{iw}, \mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_s, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, l_0, l_1, \ldots, l_m]$. The cost function and factors are kept unchanged. The optimization result of $\mathbf{P}_A^{iw}$ after the initialization stage is saved as a fixed value. We fixed the anchor position with two considerations: 1. The anchor is static in practice, and 2. usually, the initialization phase can be controlled and have the robot move in proximity of the anchor. Although the distance measurement error is not correlated with the distance value, the estimation of the anchor position depends on the distance. In other words, the ratio of trajectory length with the distance affects the estimation. So we treat the initial estimate as fixed value after the initialization phase. A time-varying wight as in [113] can also be applied to get stable anchor pose values.

6.3.3 Distributed Collaborative SLAM

Another obvious benefit of our ranging-aided system is that, with a common anchor, robots can directly estimate inter-robot transformations when they rendezvous. Robots simply need to send their current position and anchor position while ranging their neighbors. After receiving this information twice, a robot can calculate the transformation matrix between itself and the sender. Once the transformation matrix is correctly estimated, all information received from neighbors can be correctly placed in the robot's frame.

Estimating the transformation between robots is a critical requirement for multi-robot systems. We mark the transformation of coordinate systems from robot i to j as $\mathbf{T}_i^j = [\mathbf{R}_i^j, \mathbf{t}_i^j]$, where $\mathbf{R}_i^j$ is 3×3 matrix representing rotation and $\mathbf{t}_i^j$ is 3×1 vector of translation. With the help of accelerometers and gyroscopes, we can establish the direction of gravity and define the same z axis for all robots. VINS-Mono [23] uses the same strategy: the z axis is aligned with the opposite of gravity when creating the coordinate system. Therefore, only the yaw angle θ and 3D offset $\mathbf{t}_i^j$ between two coordinate systems need to be estimated. The transformation $\mathbf{T}_i^j$ consists of:

$$\mathbf{R}_i^j = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{t}_i^j = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Using the estimation of the anchor position for two robots, we can easily get $t_z = (\mathbf{P}_A^{jw} - \mathbf{P}_A^{iw})_z$, which is the projection of the difference vector on z axis. This leaves three parameters (θ, t_x, t_z) to be estimated.

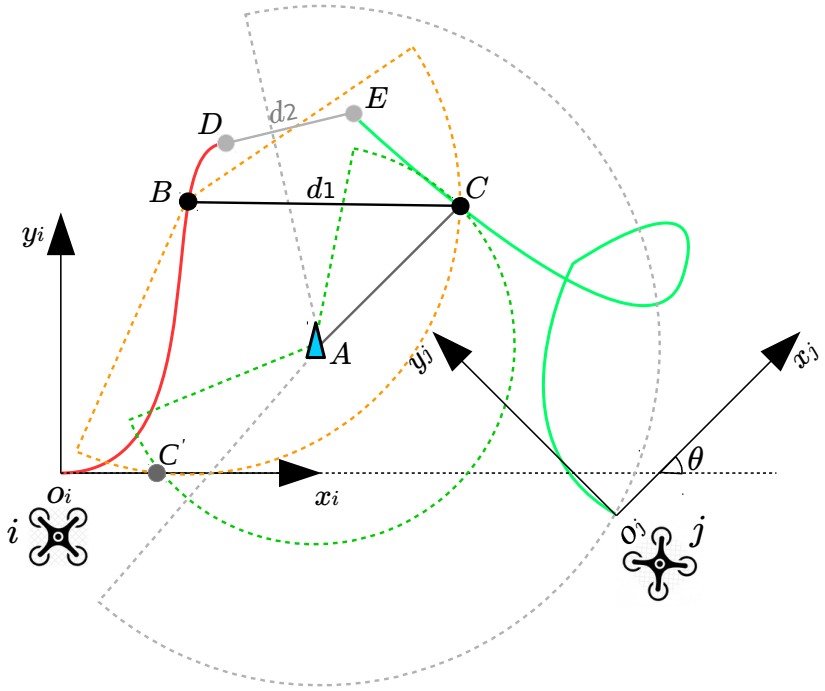


Figure 6.6 Transformation matrix estimation for multiple robots scenario.

Fig.6.6 shows how the remaining parameters are estimated. Let us assume two robots i and j moving independently in 2D for simplicity (without loss of generality, as we already know the z axis offset t_z). Both robots have their own coordinate systems, $o_i x_i y_i$ and $o_j x_j y_j$. They also have the anchor A position after initialization, and have their own trajectory tracked in their respective coordinate systems. When robot i passes point B and robot j passes point C, they enter in communication range and they can both obtain a reciprocal distance measurement d_1. Simultaneously, they exchange their current position and A's position (which are in their own coordinate system).

We take the view of robot i to illustrate our solution. When robot i receives the position of the anchor and the current position of robot j (point C) expressed in j's frame, $\mathbf{P}_{b_A}^{jw}$ and $\mathbf{P}_{b_C}^{jw}$, respectively. Robot i then knows the origin of $o_j x_j y_j$ must be on the gray circle around

the anchor with radius $|Ao_j|$ (known from $\mathbf{P}_A^{jw}$). In addition, robot i calculates $|AC|$ from $\mathbf{P}_A^{jw}$ and $\mathbf{P}_{b_C}^{jw}$ to find robot j's position C must lie on a green circle around A with radius $|AC|$. As we know, the distance between point B and C is the range measurement d_1, and point C also must lie on the orange circle around its current position B with radius d_1.

Therefore, the current position lies at one of the intersections between the green and orange circles, C and C'. To find which intersection is the correct one, we take an additional measurement d_2 between two following points D and E. Following the same procedure, we can find the candidates of E. For clarity, we did not draw these circles for D and E. By comparing the relation between E, E', C and C' with the true motion from C to E, we can find C and determine the transformation $\mathbf{T}_j^i$. This is the same as solving:

$$\begin{cases} \mathbf{P}_{b_A}^{jw} = & \mathbf{T}_i^j \cdot \mathbf{P}_{b_A}^{iw} \\ d_1 = & \|\mathbf{P}_{b_B}^{jw} - \mathbf{T}_i^j \cdot \mathbf{P}_{b_C}^{iw}\| \\ d_2 = & \|\mathbf{P}_{b_D}^{jw} - \mathbf{T}_i^j \cdot \mathbf{P}_{b_E}^{iw}\| \end{cases} \tag{6.5}$$

where $\mathbf{P}_{b_Q}^{rw}$ represent the 3D position vector of points Q in world coordinate frame of robot r.

6.4 EXPERIMENTS

We validate our algorithms using public datasets and real hardware experiments. We compare our system with and without the long window optimization, marked as *VIR Full* and *VIR w/o LW*. We also show the result of pure VIO VINS-Mono[1] for reference. The result is calculated using the TUM evaluation tool[2]. We compute the absolute trajectory error (ATE) when ground truth is available. For our large area experiments we do not have ground truth, and we start and end the experiment in the same location and then calculate the start-to-end error.

6.4.1 EuRoC Dataset Experiments

We test our single robot tracking system on the EuRoC [112] dataset. We simulate the UWB ranging measurements from ground truth data. The static anchor is assumed in the origin of the frame created during robot initialization. We add Gaussian white noise $\mathcal{N}(0, 0.03)$ to model the error of our UWB sensor. For our system, we tested with a short window size of 10 and a long window size of 100. The sliding window size of VINS-Mono is also set to 10.

[1]https://github.com/HKUST-Aerial-Robotics/VINS-Mono
[2]https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

Table 6.1 shows the average translation error (ATE) results of *VIR Full*, VIR without long window (*VIR w/o LW*), and *VINS-Mono*. We can see full version of VIR outperform the method without the long sliding window and VINS by a large amount in sequence 1, 2, and 4. The difference between *VIR w/o LW* and *VINS-Mono* is not extreme in sequences 4 and 5. As an example, we show the trajectory of the comparison for the EuRoC MH-05 sequence in Fig.6.7. Although our method does not have big difference in terms of ATE, which is calculated by aligning trajectories, our method is closer to the ground truth in red. From the ellipse, we can see our estimation can correct the error even when drifting.

6.4.2 Single Robot Experiments

We also test our system with a Spiri robot [185] with Decawave DW1000 [171]-based UWB sensors [175], shown in Fig. 6.8. The system has a D435 Realsense camera but we use it as a monocular camera. The IMU measurements come from the Pixracer flight controller, and the robot carries an Nvidia TX1 as an on-board computer. We test the system in our lab with an OptiTrack motion capture system as ground truth.

We manually control the robot and collect two sequences: the results are listed in Table 6.1. Full VIR achieves better accuracy than the version without the long window, which is also much better than VINS. To clearly see the difference, we show the trajectories using the same origin in Fig. 6.9. This is different from aligning the two trajectories to compute the ATE, and as Fig. 6.9 shows, our estimator has less accumulated error than VINS-Mono.

We also test our system performance in a large atrium. The environment in the atrium is quite challenging: featureless walls and ground, low light conditions, glass walls with reflections, etc. As shown in Fig. 6.10-(a), the reflection of lights and structures can easily be regarded as static features, which result in large reprojection errors in visual tracking. In (b), the lack of features in the close range also brings a challenge to visual SLAM. We move the robot

Table 6.1 Error comparison between full version of VIR, VIR without the long sliding window, and VINS-Mono.

ATE Error (m)	VIR Full	VIR w/o LW	VINS-Mono
EuRoC MH_01	**0.110**	0.155	0.188
EuRoC MH_02	**0.150**	0.223	0.240
EuRoC MH_03	0.364	**0.263**	0.271
EuRoC MH_04	**0.294**	0.404	0.393
EuRoC MH_05	0.397	**0.388**	0.392
Lab Seq1	**0.195**	0.257	0.278
Lab Seq2	**0.180**	0.220	0.239

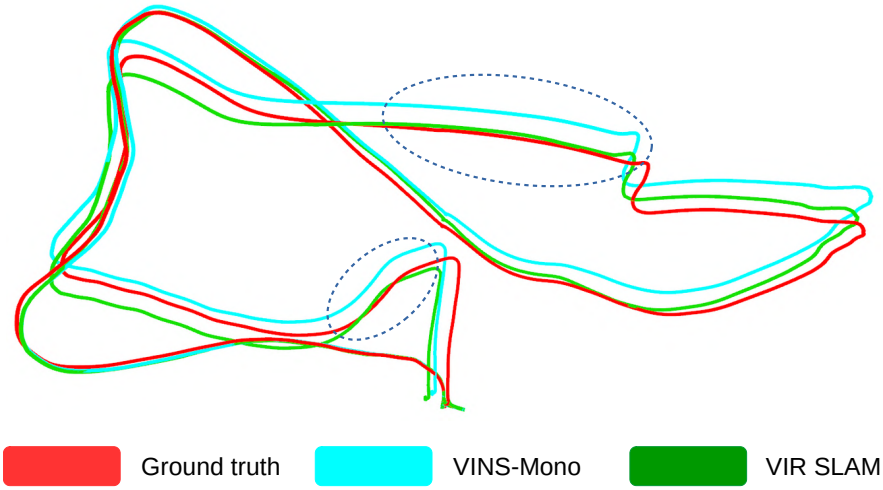


Figure 6.7 VIR results in EuRoC MH05 dataset. ATE error of our method is 0.291m, compared to VINS-Mono (0.388m). We set our method is closer to the ground truth in most sections, which testify the capability to correct the drift error.

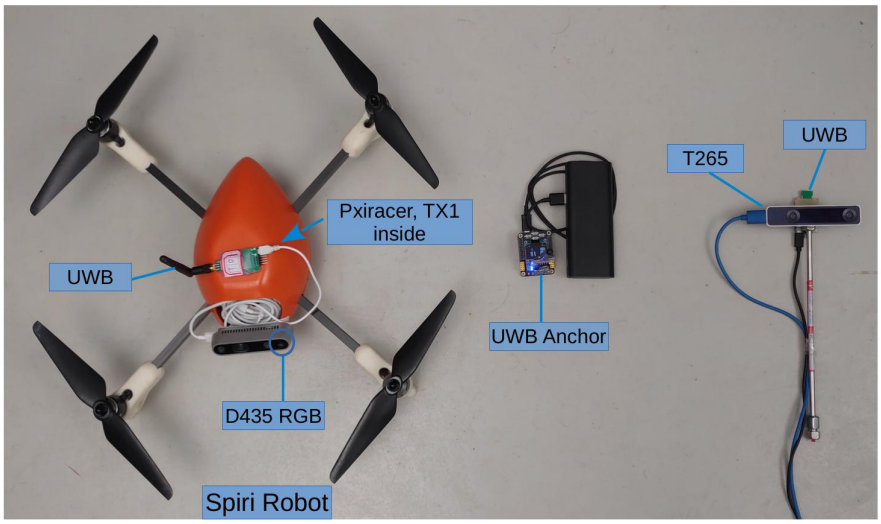


Figure 6.8 For single robot experiments, we use the Spiri robot (w/ Pixracer and NVIDIA TX1). We add a Realsense D435 camera (although we only use RGB) and UWB sensor module. The IMU measurements are obtained from the Pixracer. The static is in the middle, while the Realsense T265 on the right simulates a second robot in our multi-robot experiment.

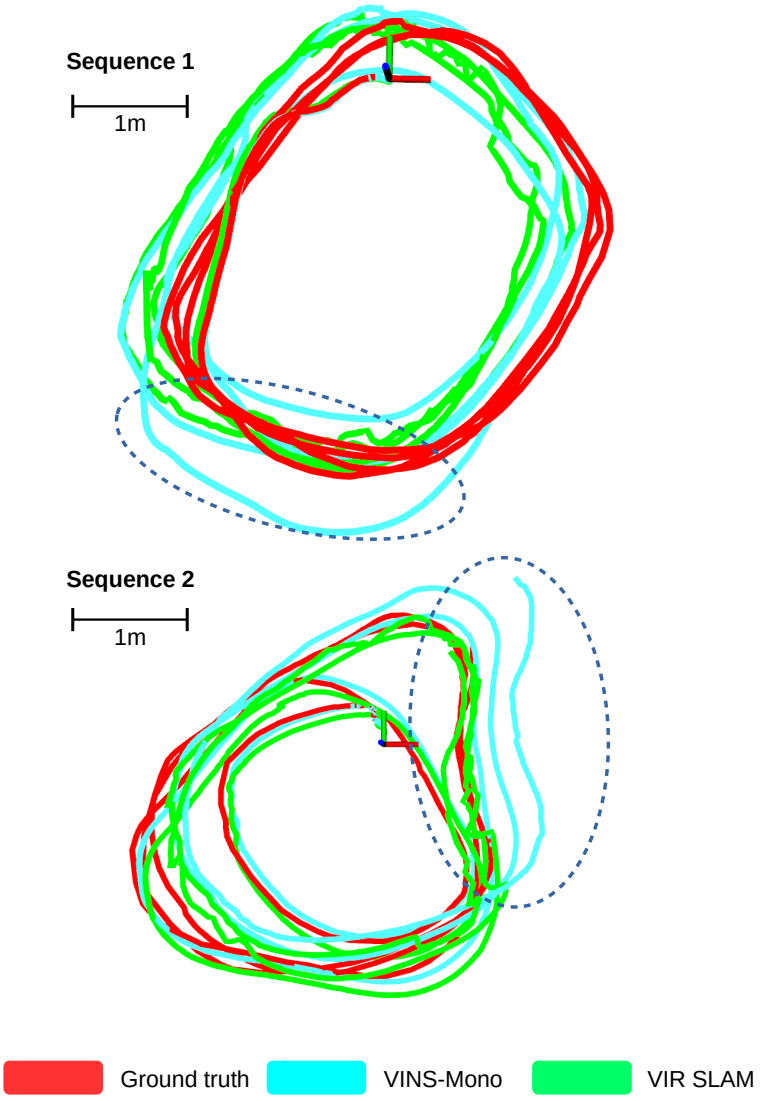


Figure 6.9 VIR indoor experiments with OptiTrack ground truth. The top and bottom figures are two sequences. All trajectories start at the origin and are not aligned. Our estimator does not present excessive drift.

around in the atrium and make sure the robot ends at the same point as it starts. Then we compare the start-to-end error. As Fig. 6.11 and Table 6.2 shows, the VINS-Mono estimation has significant drift, more than 5.4 m. Although the loop closure version can correct the drift as the loop closure is detected, there is still a clear error in the trajectory. However, our VIR SLAM works correctly without any loop closure. The start and end point have a small translation error (0.148 m) in 2D, but our system introduces a bigger accumulated error on the z axis. We believe this is due to the anchor being close to the horizontal plane of the robot, and the small difference in measurement does not help correct the z coordinate. We manually checked the UWB ranging information along the top and bottom edges and we can confirm that our estimation is at the right position on the boundaries.

Figure 6.10 Two snapshots illustrate the visually challenging environment in our atrium experiment. The reflection of light and structure in (a) can easily be regarded as static features (red dots in yellow ellipses), thereby increasing the reprojection error. In (b), the lack of features in the close range also brings a challenge to visual SLAM.

Table 6.2 Start-to-end Error Comparison

	End point	2D Error	3D Error
VINS	(4.29, 3.35, 0.52)	5.439	5.465
VINS-LC	(0.18, 0.43, -0.38)	0.464	**0.602**
VIR	(-0.04, 0.14, 0.84)	**0.148**	0.853

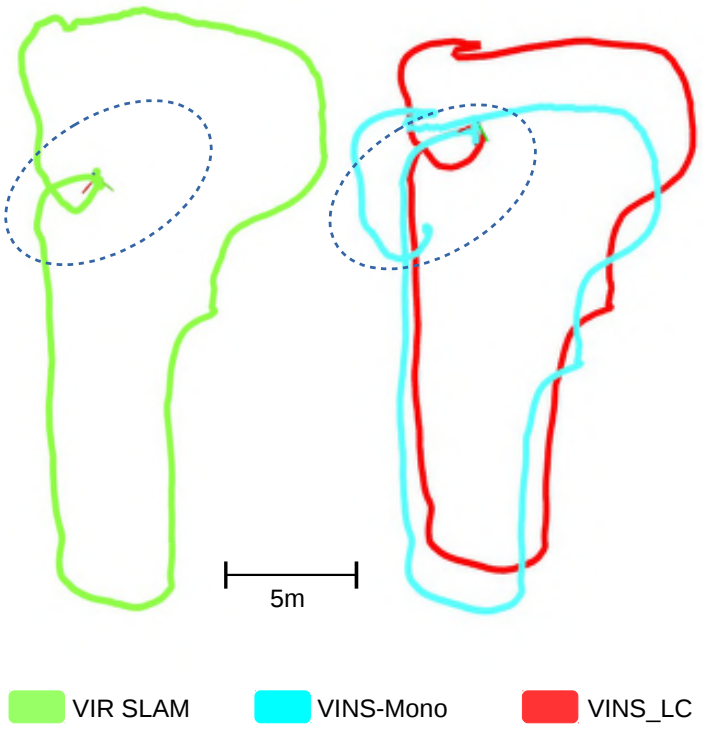


Figure 6.11 VIR results in the atrium, which is very visually challenging. The robot starts and ends at the same point and we compare the start-to-end error. As the figure shows, the start and end point of VIR-SLAM are overlapped. Although the VINS-Mono with loop closure (VINS_LC) can close the loop, it does not eliminate the accumulated error from the VIO.

6.4.3 Multi-Robot Experiments

We also test our multi-robot technique, as shown in Fig. 6.12. We manually and independently control two robots to make a trajectory similar to the "MIST", the name of our lab. We place one static UWB anchor in the environment. The first robot is imply a Realsense T265 and a UWB module, show in Fig. 6.8 (right). We move it to form an "M" shape. The other robot, Spiri, moves along a "IST" and it controlled independently. Their trajectories in each robot's frame are shown at the bottom of Fig. 6.12. With simply two inter-robot measurements (with data exchange), robot 1 can estimate the transformation of robot 2 and map robot 2's trajectory in its own frame as shown at the top of Fig. 6.12.

6.5 CONCLUSIONS AND DISCUSSIONS

In this paper, we propose VIR-SLAM, a novel SLAM paradigm combining vision, inertial, and UWB sensors. By arbitrarily setting up a static UWB anchor in the environment, robots can have drift-free state estimation and collaboration on SLAM. Our solution combines the accurate relative pose estimation from VIO and enhances it using ranging to correct the accumulated error. As our experiments show, the introduced static anchor can help correct the drift effectively to improve localization accuracy. We also show an example with two independent robots mapping each other's trajectory to their own frame after obtaining two range measurements. This technique allows the robots to find the inter-robot transformation, which is extremely useful for multi-robot SLAM.

As we know, UWB works well under LOS conditions, so the system is limited by the visibility and maximum range of the UWB anchor. When considering the level of a multi-robot system, it is interesting to determine where to drop a UWB anchor. In future work, we will complete the work for a multi-robot exploration scenario, also integrating loop closure for a more robust system.

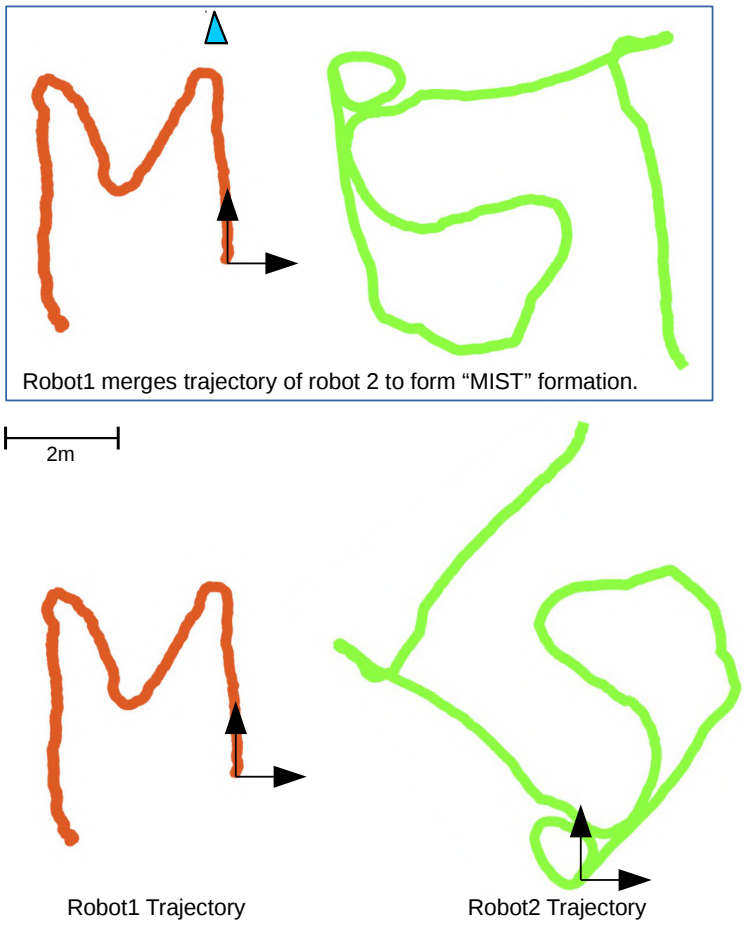


Figure 6.12 Two robots are manually controlled independently. The first robot with a Realsense T265 and a UWB module draws an "M" in its own coordinate frame. The other robot, carrying a Realsense D435, a UWB module and a Pixracer flight controller, draws "IST" in its frame. One static UWB anchor is placed in the environment (the blue triangle shown in Robot1 frame). Their trajectories are shown at the bottom. With two inter-robot communications and measurements, robot 1 can estimate the transformation matrix from robot 2, and then map robot 2's trajectory in its own frame as the top picture shows, forming "MIST" (our lab).

CHAPTER 7 GENERAL DISCUSSION

This chapter discusses the research work presented in the previous Chapters 3– 6 and highlights the findings of the experiments conducted. Table 7.1 wraps up this chapter in a summary of the main contributions and their impacts.

7.1 On the Multi-Robot System

Although there exist some multi-robot systems and robot swarms on the market, such as light shows with hundreds of drones, they usually rely on GPS and powerful central servers. These requirements limit the applications in unstructured GPS denied environments. We aim at multi-robot systems that are able to help humans in performing dangerous or strenuous tasks, therefore they have to be robust and flexible. The systems shown in Chapter 3 and Chapter 4 are fully distributed and without relying on any central controller or other infrastructure.

To work collaboratively, it is important that the robots in the system are coordinated. The work in Chapter 3 takes the advantage of multi-robot programming language, Buzz [139], and realizes the consensus on a coordinate frame of the system by bidding the reference robots. Reference robots create a coordinate frame and broadcast to the whole system. With a common coordination frame, robots can directly apply the information received from neighbor robots. Combining with the relative odometry estimation methods of Chapter 5 and Chapter 6, each robot can acquire its absolute location within the multi-robot systems. Through broadcasting and receiving, they can perform collaborative localization. In Chapter 4, another direction of multi-robot localization is proposed in a local to global manner. Each robot creates a local coordinate system and locates its neighbors in its own frame, which guarantees an up-to-date local map. By exchanging local maps with neighbors, all robots can recover a global map. The transformation matrices between neighbors are estimated by a least square optimization when they have common neighbors located in their local map. We also show a solution integrated with SLAM in Chapter6. Robots leverage a common anchor to estimate the transformation matrix with mutual ranging measurements. Having the transformation matrices, a robot can correctly project the information received from its neighbors on its own frame even when the common coordinate has not been created yet.

The simulation results in Chapter 3 show that the distributed localization system can build a common coordinate frame in ARGoS simulator [147]. After the initial stage, robots get their absolute coordinates, which can work as starting points for their odometry estimation.

The robots use EKF to track their positions with inputs of odometry and inter robot ranging measurements. In Chapter 4, we perform experiments using real hardware of UWB sensors as ranging sources. Each node estimates the map locally and then broadcasts the constructed local map. Experiments show that each node can create a local map and merge the maps received from neighbors into a global map. Experiments combining IMU, UWB, and cameras for multi-robot systems are shown in Chapter 6. The transformation matrices are estimated for two robots with records of a common anchor. The estimated transformation matrices can help the robots merge their neighbors' trajectories into their own frame.

These three strategies in Chapters 3, 4, and 6 provide different solutions to different situations. To be robust and with great resilience, the system can combine three methods together. For example, the system maintains the common coordinate frame, and meanwhile each robot estimates relative transformation matrices based on its measurements. The relative transformations help the system monitor the accumulated error and trigger the reconfiguration stage if the error passes the threshold. Furthermore, the loop closure of multi-robot SLAM systems can also be added to improve accuracy.

7.2 On the Application of UWB

UWB technology has attracted great attention in recent years for its accurate ranging performance. For example, the latest Apple iPhone at the time of writing is equipped with a UWB chip (actually, it includes all sensors needed in this dissertation). UWB is a promising technology to assist multi-robot systems. Ranging and communication allow robots in the system to have collaborative localization with relying on only the UWB medium. TWR [18] is a flexible ranging solution used in UWB technology, enabling arbitrary pairs of nodes to perform distance measurements without the requirement of synchronization. The key issue with TWR is the access control to a shared medium, i.e., the UWB communication channel. Ranging using TWR takes significantly longer than simply broadcasting a message, therefore requiring an appropriate medium access control to coordinate the measurements across all devices at a given location. The contention-based TDMA [65] cannot apply as the potential of collisions is very high. Another major category of distributed TDMA algorithms is based on request and response negotiations [62, 70]. However, these techniques usually require several iterations to converge, which makes them difficult to apply in dynamic networks where the topology changes frequently.

One of the main contributions is the novel TDMA algorithm proposed in Chapter 4. This algorithm eliminates the limitations, such as high conflicts rate and low channel usage, when applying UWB to a fully distributed system. Compared to the available TDMA techniques

for communication networks, our applications ask for several additional challenges: 1) the maximization of UWB channel usage to increase the localization frequency; 2) rapid time slot scheduling to account for dynamic topologies; 3) decentralization to avoid the need for fixed infrastructure. We address these challenges with a novel distributed TDMA for the UWB localization system. By applying set theory, this TDMA algorithm allows the system to reach a conflict-free schedule quickly. In addition, the system maintains a full usage of the shared UWB medium in any sub-graph, which means more measurements over the system. With this system, the algorithms in Chapters 3, 5, and 6 can work without worries about the collision of ranging measurements. The nearly full usage of the UWB medium allows the robots to have a maximum number of ranging measurements for accurate localization.

Chapter 4 shows a campaign of experiments, both in simulations and with real hardware. The simulations prove the scalability of the proposed algorithm. In the experiment of 1000 nodes, which is a highly dense configuration, the system can still reach a conflict-free schedule with around 20 frames. The real hardware experiments with UWB modules verify the performance in dynamic environments, considering new nodes joining, nodes leaving, and for a multi-hop network. The distributed TDMA algorithm is proved to be able to schedule the time slots in a dynamic environment, with full usage of the channel. With this technique, the applications of UWB ranging in dynamic mobile distributed systems become possible. This fills the gap to adapt the simulation in Chapter 3 to real systems and paves the road for the other chapters. Although the system is designed for UWB localization, this TDMA algorithm applies to any wireless networks that require high channel usage for static or mobile nodes.

7.3 On the Single Robot Odometry

Accurate odometry estimation of each robot is important to a cooperative system since robots keep exchanging their estimations with neighbors for better estimation. UWB sensors, as additional sensors available in our system, can be applied to assist the odometry estimation. We address the odometry estimation for two kinds of configurations: IMU and UWB, without or with an extra monocular camera. IMU and UWB are relatively cheap, and it is trivial to equip all robots with them. Cameras can be necessary for some robots designed for special functions, such as object detection and place recognition. We propose odometry estimation algorithms for these two kinds of minimal configurations. Starting with minimal configurations, we can understand the limits of the systems, which are important for further research.

In Chapter 5, we investigate the performance of localization and tracking with a single UWB ranging source to a static anchor and a low-cost 9 DoF IMU. Many researchers have

studied single anchor localization, especially for underwater robots [79, 80]. Underwater robots usually use acoustic sensors, top-of-the-line IMUs, and expensive doppler sensors. For some research with ground robots or drones [81, 82], velocity sensors, such as wheel encoders or optical flow sensors, are used. However, in our first configuration, we only use UWB and a low-cost IMU, dropping the need for a velocity sensor. Chapter 5 proposes an algorithm using the changes of UWB range measurements to estimate the speed of the robot when it moves in line. Combining the speed estimation with orientation estimation of the IMU sensor, the system becomes temporally observable. Experimental results of both simulations and hardware are shown in Chapter 5. This method improves the positioning accuracy by three times. This solution has a major impact as it enables the tracking of simple robots in the system, as well as any low-cost IoT devices with only IMU and UWB.

VIO leverages an IMU and a monocular camera to estimate odometry, which is proved to have good tracking results [20]. However, drifts can still be significant for long trajectories, especially when the environment is visually challenging. Any small orientation error can lead to a large endpoint error. In Chapter 6, a visual, inertial, ranging solution is proposed by leveraging the UWB range measurements to a static anchor. We design the SLAM optimization solution by considering the following aspects: 1) accurate relative pose estimation from VIO; b) less accurate but absolute UWB range measurements; and c) the computation cost. A double layer sliding window technique is used to combine a standard VIO optimization with UWB measurements. The marginalized key frames of the visual-inertial optimization window are kept in a second layer sliding window enhanced by ranging factors. The system is tested on public datasets as well as on real robots. Experiments show that our method can outperform state-of-the-art VIO by more than 20%. The joint optimization can effectively correct the accumulated error whenever the ranging measurements are available. For visually challenging environments, our method works even the visual-inertial odometry has significant drift.

Table 7.1 Summary of the contributions and impact of the dissertation.

Ch.	Ref.	Contribution	Impact
3	[27]	A dynamic localization system for multi-robot systems based on the range-only information.	Providing researchers with a strategy of localization for multi-robot systems at the swarm level.
		A distributed algorithm that generates a common coordinate system for a swarm and a reconfiguration function to mitigate accumulated error.	Allowing researchers to have a consensus coordinate frame in multi-robot systems, which paves the way for the cooperation between agents.
		Simulations using Buzz and ARGoS demonstrating the localization system of self-organized swarms.	Providing researchers an example of the localization system with advanced tools.
4	–	A scalable decentralized system for relative localization in mobile ad hoc networks.	Allow researchers to have a distributed localization system for UWB networks.
		A novel TDMA algorithm that can quickly schedule the use of UWB medium without collisions and maximize channel usage.	Providing a TDMA algorithm for static or mobile networks requiring full usage.
		An algorithm for individual robots to construct local maps and recover the global map.	Enabling the localization service of an infrastructure-free UWB network.
5	[29]	A speed estimator using only UWB range information.	Enhance the information that can be extracted from only ranging measurements.
		Error analysis for the speed estimator to help design a sensor fusion algorithm.	Unveiling error source to evaluate speed from ranging.
		A loosely coupled tracking algorithm fusing IMU, UWB, using the proposed speed estimation.	Push the limit of localization and tracking for devices equipped with just IMU and UWB.
6	–	A UWB-aided SLAM system for individual robots, which can effectively correct drifts.	Providing researchers and designers with a system to reduce the accumulated error in odometry.
		A double layer sliding window algorithm that combines relative pose estimation from VIO and UWB ranging.	Insight into integration visual, inertial, and ranging measurement for accurate odometry.

CHAPTER 8 CONCLUSIONS

This dissertation started with the aim of localization and tracking multiple robots, or robot swarms, in unknown GPS-denied environments for real-world applications. The research objectives identified in Chapter 1 were accomplished through the four research articles presented from Chapter 3 to Chapter 6, from macroscopic level to individuals, and verified in simulations and real hardware. In particular, a system strategy for a swarm of robots to generate a common coordinate system was proposed in Chapter 3; Chapter 4 solved a critical challenge of the access control of shared medium when applying the strategy into real hardware with a novel TDMA; the odometry for single robot tracking was addressed in two different configurations, IMU and UWB only in Chapter 5 and with an extra monocular camera in Chapter 6.

The discussion performed in the previous chapter highlighted the findings of these works and their impact. The significance of this research endeavor shines in these main outcomes:

- A localization strategy for multi-robot systems with robots equipped with range sensors. The strategy can dynamically achieve a consensus on a coordinate frame and locate robots in it. The coordinate system can reconfigure depending on the accumulated error and robot locations. This strategy plays an important role in coordination when there are numerous robots. The strategy has the potential to promote research of long-term autonomy for a self-organized system.

- A novel distributed TDMA algorithm designed for MAC of a distributed UWB network for localization. The TDMA algorithm can quickly schedule the use of UWB medium without collisions and maximize the network usage, which results in a high measurement update rate. The application of the TDMA algorithm is not limited to the UWB network, any wireless network that requires high channel usage and has frequently changed topology can use this TDMA algorithm.

- A novel solution to tracking robots using only IMU and UWB. A speed estimator from ranging to a static anchor enables the system to be observable. More than the application of robotics, any device has IMU and UWB can apply this algorithm, especially considering the low-cost IoT devices.

- A SLAM system combines vision, IMU, and ranging. With a novel double layer sliding window algorithm, the SLAM system can effectively reduce the accumulated odometry

error. This system can be used to assist VIO to get drift-free odometry estimation. With more and more applications of UWB, this system can enhance the localization service for any user.

8.1 Open Questions

We have proposed a system facilitating the localization service of multi-robot systems in different aspects. As the research advancing, a number of new questions are often discovered on the surface. For a reliable localization system using UWB, two questions that appeared but do not have been answered may affect our system.

- The property of UWB signal propagation. The UWB technology becomes popular with its accurate ranging performance. However, this result usually comes from the comparison between UWB with other EM-based techniques, such as Wifi, Bluetooth. Although UWB does not have a strict constraint on LOS when communicating, NLOS cause noise for the ranging measurements [180]. In a complex environment, multipath and two-ray ground reflection [182] can introduce a lot of outliers. Furthermore, to keep a small size, the ceramic antenna is used widely. However, the directional effect of the ceramic antenna causes large systematic bias when ranging. Anton and Raffaello [160] use the Gaussian process to compensate for the error based on known relative orientation, which is not easy to apply to distributed systems. How to have stable ranging measurements is still hard to tell.
- The resilience for the system to work robustly long-term autonomy. The system is designed to achieve long-term autonomy in unstructured environments. In our system, the self-organization and reconfiguration structure guarantee their inter-robot relation and the odometry provides continuous tracking. Another important factor of long-term autonomy is self-healing. If the system fails, how to retrieve from the previous status and maintain the goal of the system are still an open question.

8.2 Future Ventures

With regard to future work, several known features in the system that can be improved. One known limitation for the TDMA algorithm is the fixed size of slots in the frame structure initialized at the start. Although the local update rate is not affected, the fixed frame size constrains the TDMA scheduling process to the frame cycle. Designing an adaptive frame size to improve the scheduling update rate is an interesting work to enhance the algorithm. Furthermore, to improve the localization and tracking performance, we will integrate the loop

closure function to improve the robustness, both in local and inter-robot. Another point that we have not addressed is the map fusion. The robots in the system should be able to fuse their maps for cooperation. At last, the eventual goal is to have a long-term autonomy test with the full system in a GPS denied environment. Combining the odometry estimation for single robots, the loop closure, map fusion, and the multi-robot coordinate system, a fully autonomous exploration would be a good milestone for multi-robot localization.

REFERENCES

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of science robotics," *Science Robotics*, vol. 3, no. 14, 2018.

[2] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, Jul. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606003227

[3] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[4] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, Jan. 2016. [Online]. Available: http://doi.wiley.com/10.1002/rob.21620

[5] A. Cornejo and R. Nagpal, "Distributed range-based relative localization of robot swarms," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 91–107.

[6] N. Trawny, X. S. Zhou, and S. I. Roumeliotis, "3d relative pose estimation from six distances." in *Robotics: Science and Systems*. Citeseer, Jun. 2009.

[7] G. M. Mendoza-Silva, J. Torres-Sospedra, and J. Huerta, "A meta-review of indoor positioning systems," *Sensors*, vol. 19, no. 20, p. 4507, Jan. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/20/4507

[8] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, 2013, pp. 71–84.

[9] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: indoor location sensing using active rfid," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).* IEEE, 2003, pp. 407–415.

[10] "Bluetooth 5.1 can achieve 1cm accuracy thanks to 1888 tech," Jan. 2019. [Online]. Available: https://www.cbronline.com/news/bluetooth-5-1

[11] M. Cominelli, P. Patras, and F. Gringoli, "Dead on arrival: An empirical study of the bluetooth 5.1 positioning system," in *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization - WiNTECH '19*. Los Cabos, Mexico: ACM Press, 2019, pp. 13–20. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3349623.3355475

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

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

[14] M. Hamer, "Scalable localization and coordination of robot swarms," Ph.D. dissertation, ETH Zurich, 2019. [Online]. Available: http://hdl.handle.net/20.500.11850/382258

[15] "Pozyx - centimeter positioning for Arduino." [Online]. Available: https://www.pozyx.io

[16] D. Zito and D. Morche, "UWB radios — the maturity age?" in *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, Jun. 2016, pp. 1–4.

[17] Q. Shi, X. Cui, S. Zhao, S. Xu, and M. Lu, "Blas: Broadcast relative localization and clock synchronization for dynamic dense multi-agent systems," *IEEE Transactions on Aerospace and Electronic Systems*, 2020.

[18] V. DW1000 User Manual, "2.11.(decawave, 2017)," 2017.

[19] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[20] 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.

[21] T. Lupton and S. Sukkarieh, "Preintegration_visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[22] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[23] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[24] 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*, vol. 5, no. 2, pp. 1656–1663, 2020.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[26] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling perceptual aliasing in slam via discrete–continuous graphical models," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1232–1239, 2019.

[27] Y. Cao, M. Li, I. Švogor, S. Wei, and G. Beltrame, "Dynamic range-only localization for multi-robot systems," *IEEE access*, vol. 6, pp. 46 527–46 537, 2018.

[28] Y. Cao, D. St-Onge, and G. Beltrame, "Collaborative localization and tracking with minimal infrastructure," in *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2020, pp. 114–117.

[29] Y. Cao, C. Yang, R. Li, A. Knoll, and G. Beltrame, "Accurate position tracking with a single UWB anchor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020, pp. 2344–2350.

[30] J. Borenstein, H. R. Everett, and L. Feng, "Where am I? sensors and methods for mobile robot positioning," *University of Michigan*, vol. 119, no. 120, p. 27, 1996.

[31] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[32] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[33] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

[34] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[35] N. Point, "Optitrack," *Natural Point, Inc.,[Online]. Available: http://www. naturalpoint. com/optitrack/.[Accessed 22 2 2014]*, 2011.

[36] M. S. Couceiro, "An overview of swarm robotics for search and rescue applications," *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, p. 345, 2015.

[37] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013. [Online]. Available: https://link.springer.com/article/10.1007/s11721-012-0075-2

[38] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[39] I. Amundson and X. Koutsoukos, "A survey on localization for mobile wireless sensor networks," *Mobile entity localization and tracking in GPS-less environnments*, pp. 235–254, 2009.

[40] "LoRa Alliance®," 2020. [Online]. Available: https://lora-alliance.org/

[41] J. Liu and Y. Zhang, "Error control in distributed node self-localization," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, p. 162587, Dec. 2007. [Online]. Available: https://asp-eurasipjournals.springeropen.com/articles/10.1155/2008/162587

[42] A. Paul and T. Sato, "Localization in wireless sensor networks: A survey on algorithms, measurement techniques, applications and challenges," *Journal of Sensor and Actuator Networks*, vol. 6, no. 4, p. 24, Oct. 2017. [Online]. Available: http://www.mdpi.com/2224-2708/6/4/24

[43] H. Yang and Y. Gao, "GPS satellite orbit prediction at user end for real-time PPP system," *Sensors*, vol. 17, no. 9, p. 1981, Aug. 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/9/1981

[44] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 340–341.

[45] L. Fang, W. Du, and P. Ning, "A beacon-less location discovery scheme for wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, Mar. 2005, pp. 161–171 vol. 1.

[46] E. Liao, G. Hollinger, J. Djugash, and S. Singh, "Preliminary results in tracking mobile targets using range sensors from multiple robots," *Distributed Autonomous Robotic Systems 7*, pp. 125–134, 2006.

[47] W. Zhang, J. A. Djugash, and S. Singh, "Parrots: A range measuring sensor network," Carnegie Mellon University, Tech. Rep., 2006. [Online]. Available: http://repository.cmu.edu/robotics/966/

[48] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 1250–1257.

[49] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert, "Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference," *IEEE Control Systems*, vol. 36, no. 2, pp. 41–74, 2016.

[50] A. R. Shirazi and Y. Jin, "A strategy for self-organized coordinated motion of a swarm of minimalist robots," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 326–338, 2017.

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

[52] P. Withington and L. W. Fullerton, "An impulse radio communications system," in *Ultra-Wideband, Short-Pulse Electromagnetics*. Springer, 1993, pp. 113–120.

[53] F. Mazhar, M. G. Khan, and B. Sällberg, "Precise indoor positioning using uwb: a review of methods, algorithms and implementations," *Wireless Personal Communications*, vol. 97, no. 3, pp. 4467–4491, 2017.

[54] A. Prorok and A. Martinoli, "Accurate indoor localization with ultra-wideband using spatial models and collaboration," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 547–568, 2014.

[55] J. Li, Y. Bi, K. Li, K. Wang, F. Lin, and B. M. Chen, "Accurate 3d localization for MAV swarms by UWB and IMU fusion," *arXiv:1807.10913 [cs]*, Jul. 2018.

[56] 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*, pp. 1–14, 2019.

[57] F. M. Martel, J. Sidorenko, C. Bodensteiner, M. Arens, and U. Hugentobler, "Unique 4-DOF relative pose estimation with six distances for UWB/V-SLAM-based devices," *Sensors*, vol. 19, no. 20, p. 4366, Oct. 2019.

[58] 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)*, May 2020, p. 7.

[59] T. ter Horst, "Ultra-wideband communication and relative localisation for swarming robots," Master's thesis, Delft University of Technology, 2019.

[60] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter, "Analysis of the scalability of uwb indoor localization solutions for high user densities," *Sensors (Basel, Switzerland)*, vol. 18, no. 6, Jun. 2018.

[61] N. Macoir, M. Ridolfi, J. Rossey, I. Moerman, and E. De Poorter, "Mac protocol for supporting multiple roaming users in mult-cell uwb localization networks," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 588–599.

[62] J. Zhu and S. Kia, "A SPIN-based dynamic TDMA communication for an UWB-based infrastructure-free cooperative navigation," *IEEE Sensors Letters*, pp. 1–1, 2020.

[63] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: self-organizing desynchronization and tdma on wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 11–20.

[64] A. Astrin, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, Sep. 2011.

[65] N. Abramson, "The ALOHA system: another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*, ser. AFIPS '70 (Fall). Houston, Texas: Association for Computing Machinery, Nov. 1970, pp. 281–285.

[66] "ALOHA packet system with and without slots and capture | ACM SIGCOMM Computer Communication Review." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1024916.1024920

[67] J. Yeo, H. Lee, and S. Kim, "An efficient broadcast scheduling algorithm for tdma ad-hoc networks," *Computers & operations research*, vol. 29, no. 13, pp. 1793–1806, 2002.

[68] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 15, no. 2, p. 11, 1997.

[69] S. Ramanathan, "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," in *Proceedings of INFOCOM '97*, vol. 2, Apr. 1997, pp. 900–907 vol.2.

[70] C. Zhu and M. S. Corson, "A five-phase reservation protocol (fprp) for mobile ad hoc networks," *Wireless networks*, vol. 7, no. 4, pp. 371–384, 2001.

[71] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized tdma scheduling for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, Oct. 2009.

[72] B. Dezfouli, "DICSA: Distributed and concurrent link scheduling algorithm for data gathering in wireless sensor networks," *Ad Hoc Networks*, p. 18, 2015.

[73] Y. Xu, K.-W. Chin, and S. Soh, "A novel distributed pseudo-tdma channel access protocol for multi-transmit-receive wireless mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2531–2542, 2017.

[74] C. D. Young, "USAP: a unifying dynamic distributed multichannel TDMA slot assignment protocol," in *MILCOM '96, Conference Proceedings, IEEE Military Communications Conference, 1996*, vol. 1, Oct. 1996, pp. 235–239 vol.1.

[75] H. A. Omar, W. Zhuang, and L. Li, "VeMAC: A tdma-based mac protocol for reliable broadcast in VANETs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1724–1736, Sep. 2013.

[76] C. Young, "USAP multiple access: dynamic resource allocation for mobile multihop multichannel wireless networking," in *MILCOM 1999. IEEE Military Communications. Conference Proceedings (Cat. No.99CH36341)*, vol. 1, Oct. 1999, pp. 271–275 vol.1.

[77] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio, "Dynamic TDMA slot assignment in ad hoc networks," in *17th International Conference on Advanced Information Networking and Applications, 2003. AINA 2003.*, Mar. 2003, pp. 330–335.

[78] S. Cao and V. C. Lee, "A novel adaptive tdma-based mac protocol for vanets," *IEEE Communications Letters*, vol. 22, no. 3, pp. 614–617, 2017.

[79] B. Ferreira, A. Matos, and N. Cruz, "Single beacon navigation: Localization and control of the MARES AUV," in *OCEANS 2010 MTS/IEEE SEATTLE*, 2010, pp. 1–9.

[80] A. Ross and J. Jouffroy, "Remarks on the observability of single beacon underwater navigation," in *Proc. Intl. Symp. Unmanned Unteth. Subm. Tech*, 2005.

[81] 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*, vol. 9, no. 3, pp. 169–186, 2017.

[82] T.-M. Nguyen, Z. Qiu, M. Cao, T. H. Nguyen, and L. Xie, "Single landmark distance-based navigation," *IEEE Transactions on Control Systems Technology*, pp. 1–8, 2019.

[83] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on automatic control*, vol. 22, no. 5, pp. 728–740, 1977.

[84] A. Martinelli and R. Siegwart, "Observability analysis for mobile robot localization," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1471–1476.

[85] P. Batista, C. Silvestre, and P. Oliveira, "Single range aided navigation and source localization: Observability and filter design," *Systems & Control Letters*, vol. 60, no. 8, pp. 665–673, 2011.

[86] G. Indiveri, D. De Palma, and G. Parlangeli, "Single range localization in 3-d: Observability and robustness issues," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1853–1860, 2016.

[87] 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*, pp. 1–27, 2019.

[88] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1, pp. 1–153, 2017.

[89] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep., 2007.

[90] J. D. Hol, F. Dijkstra, H. Luinge, and T. B. Schon, "Tightly coupled UWB/IMU pose estimation," in *2009 IEEE International Conference on Ultra-Wideband*. IEEE, 2009, pp. 688–692.

[91] J. D. Hol, "Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and gps," Ph.D. dissertation, Linköping University Electronic Press, 2011.

[92] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[93] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[94] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[95] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792.

[96] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[97] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[98] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[99] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[100] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[101] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[102] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose slam," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 78–93, 2009.

[103] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2088–2095.

[104] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[105] ——, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[106] J. González, J.-L. Blanco, C. Galindo, A. Ortiz-de Galisteo, J.-A. Fernández-Madrigal, F. A. Moreno, and J. L. Martínez, "Mobile robot localization based on ultra-wide-band ranging: A particle filter approach," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 496–507, 2009.

[107] 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)*. Seattle, WA, USA: IEEE, May 2015, pp. 1730–1736.

[108] X. Fang, C. Wang, T.-M. Nguyen, and L. Xie, "Graph optimization approach to localization with range measurements," *arXiv:1802.10276 [cs]*, Feb. 2018.

[109] J. Tiemann, A. Ramsey, and C. Wietfeld, "Enhanced UAV indoor navigation through SLAM-augmented UWB localization," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. Kansas City, MO: IEEE, May 2018, pp. 1–6.

[110] C. Wang, H. Zhang, T.-M. Nguyen, and L. Xie, "Ultra-wideband aided fast localization and mapping system," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1602–1609.

[111] Q. Shi, X. Cui, W. Li, Y. Xia, and M. Lu, "Visual-uwb navigation system for unknown environments," in *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+)*, 2019, p. 11.

[112] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, Sep. 2016.

[113] T. H. Nguyen, T.-M. Nguyen, and L. Xie, "Tightly-coupled single-anchor ultra-wideband-aided monocular visual odometry system," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020.

[114] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[115] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019.

[116] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017.

[117] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2916–2923.

[118] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[119] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[120] V.-L. Dang, B.-S. Le, T.-T. Bui, H.-T. Huynh, and C.-K. Pham, "A decentralized localization scheme for swarm robotics based on coordinate geometry and distributed gradient descent," in *MATEC Web of Conferences*, vol. 54. EDP Sciences, 2016.

[121] P. Skrzypczyński, "Mobile robot localization: Where we are and what are the challenges?" in *International Conference Automation*. Springer, 2017, pp. 249–267.

[122] G. A. Korikar, S. K. Katragadda, S. Kesarla, J. M. Conrad, and A. F. Browne, "A survey on robot localization in extraterrestrial environments," in *SoutheastCon, 2016*. IEEE, 2016, pp. 1–7.

[123] I. Conduraru, I. Doroftei *et al.*, "Localization methods for mobile robots-a review," in *Advanced Materials Research*, vol. 837. Trans Tech Publ, 2014, pp. 561–566.

[124] J. Torres-Solis, T. H. Falk, and T. Chau, "A review of indoor localization technologies: towards navigational assistance for topographical disorientation," in *Ambient Intelligence*. InTech, 2010.

[125] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4193–4198.

[126] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "Swistrack-a flexible open source tracking software for multi-agent systems," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 4004–4010.

[127] V. Peak, "Vicon motion capture system," 2005.

[128] I. Guvenc, S. Gezici, and Z. Sahinoglu, "Ultra-wideband range estimation: Theoretical limits and practical algorithms," in *Ultra-Wideband, 2008. ICUWB 2008. IEEE International Conference on*, vol. 3. IEEE, 2008, pp. 93–96.

[129] G. A. Hollinger, J. Djugash, and S. Singh, "Target tracking without line of sight using range from radio," *Autonomous Robots*, vol. 32, no. 1, pp. 1–14, 2012.

[130] A. Prorok and A. Martinoli, "A reciprocal sampling algorithm for lightweight distributed multi-robot localization," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3241–3247.

[131] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost collaborative localization for large-scale multi-robot systems," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. Ieee, 2012, pp. 4236–4241.

[132] A. Bahr, J. J. Leonard, and M. F. Fallon, "Cooperative localization for autonomous underwater vehicles," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 714–728, 2009.

[133] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[134] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2690–2695 vol.3.

[135] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "Collaborative multi-robot localization," in *Mustererkennung 1999*, ser. Informatik aktuell. Springer, Berlin, Heidelberg, 1999, pp. 15–26.

[136] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost multi-robot localization," in *Redundancy in Robot Manipulators and Multi-Robot Systems*. Springer, 2013, pp. 15–33.

[137] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized collaborative localization for sparsely communicating robots." in *Robotics: Science and Systems*, 2016.

[138] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) on 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 287–294.

[139] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 3794–3800.

[140] G. MacGougan, K. O'Keefe, and R. Klukas, "Ultra-wideband ranging precision and accuracy," *Measurement Science and Technology*, vol. 20, no. 9, p. 095105, 2009.

[141] H. Benzerrouk, A. Nebylov, S. Hassen, and P. Closas, "Cubature and Gauss-Hermite based Kalman filters applied to unmanned aerial vehicle attitude estimation problem," *Journal of the Moscow Aviation Institute*, vol. 20, no. 5, 2013.

[142] P. Müller and R. Piché, "Statistical trilateration with skew-t errors," in *Localization and GNSS (ICL-GNSS), 2015 International Conference on*. IEEE, 2015, pp. 1–6.

[143] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of basic engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[144] R. S. Bucy, "Linear and nonlinear filtering," *Proceedings of the IEEE*, vol. 58, no. 6, pp. 854–864, 1970.

[145] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[146] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic Press, 1979.

[147] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[148] Y. Xiong, N. Wu, Y. Shen, and M. Z. Win, "Cooperative network synchronization: Asymptotic analysis," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 757–772, 2018.

[149] W. Yuan, N. Wu, B. Etzlinger, H. Wang, and J. Kuang, "Cooperative joint localization and clock synchronization based on Gaussian message passing in asynchronous wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7258–7273, 2016.

[150] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017.

[151] Z. Yang and Y. Liu, "Quality of trilateration: Confidence based iterative localization," in *2008 The 28th International Conference on Distributed Computing Systems*, Jun. 2008.

[152] S. Hadzic and J. Rodriguez, "Utility based node selection scheme for cooperative localization," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, Sep. 2011, pp. 1–6.

[153] S. Han, S. Lee, S. Lee, J. Park, and S. Park, "Node distribution-based localization for large-scale wireless sensor networks," *Wireless Networks*, vol. 16, no. 5, pp. 1389–1406, Jul. 2010.

[154] M. Hadded, P. Muhlethaler, A. Laouiti, R. Zagrouba, and L. A. Saidane, "TDMA-based mac protocols for vehicular ad hoc networks: A survey, qualitative analysis and open research issues," *IEEE COMMUNICATION SURVEYS AND TUTORIALS*, vol. 17, no. 4, p. 37, 2015.

[155] L.-A. Phan, T. Kim, T. Kim, J. Lee, and J.-H. Ham, "Performance analysis of time synchronization protocols in wireless sensor networks," *Sensors*, vol. 19, no. 13, p. 3020, Jul. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/13/3020

[156] L. Van Hoesel and P. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks," in *1st Int. Workshop on Networked Sensing Systems (INSS 2004)*, 2004.

[157] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. Los Angeles, California, USA: Association for Computing Machinery, Nov. 2003, pp. 181–192.

[158] A. K. Paul and T. Sato, "Localization in wireless sensor networks: A survey on algorithms, measurement techniques, applications and challenges," *Journal of Sensor and Actuator Networks*, vol. 6, no. 4, p. 24, 2017.

[159] D. Ltd, "Aps011 application note: Sources of error in dw1000 based two-way ranging (twr) schemes," 2014.

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

[161] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 287–294.

[162] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, S. Sen, and V. Handziski, "Microsoft indoor localization competition: Experiences and lessons learned," *Get Mobile*, vol. 18, no. 4, pp. 24–31, 2015.

[163] J. O. Reis, P. T. M. Batista, P. Oliveira, and C. Silvestre, "Source localization based on acoustic single direction measurements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 6, pp. 2837–2852, 2018.

[164] "iphone 11 - technical specifications," 2019. [Online]. Available: https://www.apple.com/ca/iphone-11/specs

[165] G. Miraglia, K. N. Maleki, and L. R. Hook, "Comparison of two sensor data fusion methods in a tightly coupled UWB/IMU 3-d localization system," in *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2017, pp. 611–618.

[166] L. Yao, Y.-W. A. Wu, L. Yao, and Z. Z. Liao, "An integrated IMU and UWB sensor based indoor positioning system," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2017, pp. 1–8.

[167] C. Gentner and M. Ulmschneider, "Simultaneous localization and mapping for pedestrians using low-cost ultra-wideband system and gyroscope," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2017, pp. 1–8.

[168] T. M. Nguyen, A. H. Zaini, K. Guo, and L. Xie, "An ultra-wideband-based multi-UAV localization system in GPS-denied environments," in *2016 International Micro Air Vehicles Conference*, 2016.

[169] R. C. Dorf and R. H. Bishop, *Modern control systems*. Pearson, 2011.

[170] Q. Shi, X. Cui, S. Zhao, J. Wen, and M. Lu, "Range-only collaborative localization for ground vehicles," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation*, 2019, pp. 2063–2077.

[171] Decaware, *DW1000 Radio IC*, 2014. [Online]. Available: https://www.decawave.com/product/dw1000-radio-ic/

[172] G. Bohm and G. Zech, *Introduction to statistics and data analysis for physicists*. Desy Hamburg, 2010, vol. 1.

[173] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[174] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, and Y. Fang, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 20–25.

[175] YCHIOT, "YCHIOT Wenzhou Yanchuang IOT Technology," 2019. [Online]. Available: https://www.ychiot.com/

[176] L. Feng, H. R. Everett, and J. Borenstein, "Where am I?: Sensors and methods for autonomous mobile robot positioning," 1994.

[177] Bitcraze, "Loco positioning system | Bitcraze," 2020. [Online]. Available: https://www.bitcraze.io/loco-pos-system/

[178] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364906072768

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

[180] V. Barral, P. Suárez-Casal, C. Escudero, and J. García-Naya, "Assessment of uwb ranging bias in multipath environments," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcala de Henares, Spain*, 2016, pp. 4–7.

[181] V. Savic, E. G. Larsson, J. Ferrer-Coll, and P. Stenumgaard, "Kernel methods for accurate UWB-based ranging with reduced complexity," *IEEE Transactions on Wireless Communications*, vol. 15, no. 3, pp. 1783–1793, Mar. 2016.

[182] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[183] V. Mai, M. Kamel, M. Krebs, A. Schaffner, D. Meier, L. Paull, and R. Siegwart, "Local positioning system using uwb range measurements for an unmanned blimp," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2971–2978, 2018.

[184] A. Witkin, "Scale-space filtering, in processing of ijcai," 1983.

[185] Pleiades, *Spiri*, 2020. [Online]. Available: https://pleiadesrobotics.com/products/spiri-mu/