UNIVERSITY
OF TURKU

COLLABORATIVE AUTONOMY IN HETEROGENEOUS MULTI-ROBOT SYSTEMS

Jorge Peña Queralta

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
PUBLICATION SERIES INFORMATION SHOULD GO HERE

University of Turku

Faculty of Technology
Department of Computing
Robotics and Autonomous Systems
Doctoral Programme in Technology

Supervised by

Associate Professor, Tomi Westerlund
University of Turku

Professor, Zhuo Zou
Fudan University

Reviewed by

Reader, Simon Watson
The University of Manchester

Professor, Raivo Sell
Tallinn University of Technology

Opponent

Associate Professor, Laura Ruotsalainen
University of Helsinki

Serial F 18
ISBN 978-951-29-9116-7 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Printhouse, Turku, Finland, 2022

To my parents Paula and Víctor, and my sister Laura...
...and to Lei

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Robotics and Autonomous Systems
PEÑA QUERALTA, JORGE: Collaborative Autonomy in Heterogeneous Multi-Robot Systems
Doctoral dissertation, 227 pp.
Doctoral Programme in Technology
November 2022

ABSTRACT

As autonomous mobile robots become increasingly connected and widely deployed in different domains, managing multiple robots and their interaction is key to the future of ubiquitous autonomous systems. Indeed, robots are not individual entities anymore. Instead, many robots today are deployed as part of larger fleets or in teams. The benefits of multi-robot collaboration, specially in heterogeneous groups, are multiple. Significantly higher degrees of situational awareness and understanding of their environment can be achieved when robots with different operational capabilities are deployed together. Examples of this include the Perseverance rover and the Ingenuity helicopter that NASA has deployed in Mars, or the highly heterogeneous robot teams that explored caves and other complex environments during the last DARPA Sub-T competition.

This thesis delves into the wide topic of *collaborative autonomy* in multi-robot systems, encompassing some of the key elements required for achieving robust collaboration: solving collaborative decision-making problems; securing their operation, management and interaction; providing means for autonomous coordination in space and accurate global or relative state estimation; and achieving collaborative situational awareness through distributed perception and cooperative planning. The thesis covers novel formation control algorithms, and new ways to achieve accurate absolute or relative localization within multi-robot systems. It also explores the potential of distributed ledger technologies as an underlying framework to achieve collaborative decision-making in distributed robotic systems.

Throughout the thesis, I introduce novel approaches to utilizing cryptographic elements and blockchain technology for securing the operation of autonomous robots, showing that sensor data and mission instructions can be validated in an end-to-end manner. I then shift the focus to localization and coordination, studying ultra-wideband (UWB) radios and their potential. I show how UWB-based ranging and localization can enable aerial robots to operate in GNSS-denied environments, with a study of the constraints and limitations. I also study the potential of UWB-based relative localization between aerial and ground robots for more accurate positioning in areas where GNSS signals degrade. In terms of coordination, I introduce two new algorithms for formation control that require zero to minimal communication, if enough degree of awareness of neighbor robots is available. These algorithms are validated in simulation and real-world experiments. The thesis concludes with the integration of a new approach to cooperative path planning algorithms and UWB-based relative localization for dense scene reconstruction using lidar and vision sensors in ground and aerial robots.

KEYWORDS: robotics, multi-robot systems, collaborative autonomy, edge computing, blockchain, distributed ledger technologies, ultra-wideband localization, sensor fusion

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Robotiikka ja autonomiset järjestelmät
PEÑA QUERALTA, JORGE: Collaborative Autonomy in Heterogeneous Multi-Robot Systems
Väitöskirja, 227 s.
Teknologian tohtoriohjelma
Marraskuu 2022

TIIVISTELMÄ

Robotit eivät enää ole yksittäisiä kokonaisuuksia vaan ovat nykyään käytössä osana suurempia kokonaisuuksia tai toimivat ryhmissä. Siten useiden robottien ja niiden vuorovaikutuksen hallinta on avainasemassa kaikkialla läsnä olevien autonomisten järjestelmien tulevaisuuden kannalta. Usean robotin yhteistyön edut, erityisesti heterogeenisissä ryhmissä, ovat moninaiset. Voimme esimerkiksi saavuttaa huomattavasti korkeamman tilannetietoisuuden ja ymmärryksen ympäristöstä, kun eri toimintakykyisiä robotteja käytetään yhdessä. Esimerkkejä tästä ovat Perseverance-mönkijä ja Ingenuity-helikopteri, jotka NASA on käyttänyt Marsissa, tai erittäin heterogeeniset robottiryhmät, jotka tutkivat luolia ja muita monimutkaisia ympäristöjä viime DARPA Sub-T -kilpailun aikana.

Tämä väitöskirja perehtyy laajaan aihealueeseen monirobottijärjestelmien yhteistyön autonomiassa käsitellet seuraavia keskeisiä elementtejä vahvan yhteistyön saavuttamiseksi: yhteistyöhön perustuvien päätöksenteko-ongelmien ratkaiseminen; niiden toiminnan, hallinnan ja vuorovaikutuksen turvaaminen; välineiden tarjoaminen autonomiseen koordinointiin tilassa ja tarkan globaalin tai suhteellisen tilan estimointiin; ja yhteistyöhön perustuvan tilannetietoisuuden saavuttaminen hajautetun havainnon ja yhteistyön suunnittelun avulla. Väitöskirja esittelee uusia parvimuodostelmien ohjausalgoritmeja ja uusia tapoja saavuttaa tarkka absoluuttinen tai suhteellinen lokalisointi monirobottijärjestelmissä. Väitöskirjassa tutkitaan myös hajautettujen tilikirjatekniikoiden mahdollisuuksia taustalla olevana teknologiana yhteistyöhön perustuvan päätöksenteon saavuttamiseksi hajautetuissa robottijärjestelmissä.

Väitöskirjassa esittelen uusia lähestymistapoja kryptografisten elementtien ja tilikirjateknologian hyödyntämiseen autonomisten robottien toiminnan turvaamisessa validoimaan anturidataa ja tehtäväohjeita kommunikaatioketjun päästä päähän. Sitten siirrän painopisteen lokalisointiin ja koordinaatioon, tutkien ultralaajakaistaradioita (UWB) ja niiden mahdollisuuksia. Näytän, kuinka UWB-pohjainen etäisyyden arviointi ja lokalisointi voivat mahdollistaa ilmarobottien toiminnan ilman satelliittipohjaista navigointia (GNSS) huomioiden myös UWB-teknologian tuomia rajoituksia. Tutkin myös ilmassa ja maassa liikkuvien robottien välisen UWB-pohjaisen suhteellisen lokalisoinnin mahdollisuuksia tarkempaan paikannukseen alueilla, joilla GNSS-signaalit heikkenevät. Koordinoinnin osalta esittelen kaksi uutta parviohjausalgoritmia, jotka eivät vaadi yhtään tai vain minimaalisesti viestintää. Nämä algoritmit validoidaan simulaatioissa ja kenttäkokeissa. Väitöskirja päättyy uuden lähestymistavan integrointiin yhteistoiminnallisiin reittien suunnittelualgoritmeihin ja UWB-pohjaiseen suhteelliseen lokalisointiin käyttämällä lidar- ja näköantureita maa- ja ilmarobotteissa.

ASIASANAT: robotiikka, monirobottijärjestelmät, reunalaskenta, UWB-lokalisointi

Acknowledgements

I cannot thank enough my supervisor, and friend, Tomi Westerlund, for the all the flexibility and support, with whom I have developed both professionally and personally. Only working with Tomi I could have been able to enjoy the work as much as I have, and to work on so many interesting topics. I would not have had this many opportunities anywhere else. I am looking forward for what we can do together.

My biggest thanks have to go to my partner Lei, who was able to cope with me over time and distance. I am extremely lucky to have you by my side. The pandemic kept us apart for over two years, but we somehow managed to get out of it even stronger. There are still plenty of challenges ahead, but I am looking forward to how we can shape our future. Beyond the personal dimension, you have been the best listener, and given a number of advice that have shaped different parts of this thesis in both form and substance.

This journey would have never been the same without the increasing number of colleagues and friends that have joined our research group over the past years. My thanks go to Qingqing, who has been part of the evolution of the group and has been a great colleague and a better friend during this journey. Xianjia is an irreplaceable friend, thanks to you and Yuki for all the adventures together; you are now also a key person in our lab. Thank you Paola and Salma for all the fun both in and out of work. And thank you to everyone else that joined me in this journey and has helped build a great team at TIERS: Farhad, Sahar, Daniel, Marius, Jiaqiang and Iacopo. I cannot forget those that have been there earlier: Tuan, without whom the first year in Turku would have been very different, Sebastian, Anum, Victor, Reza, and Maxi; and the master students that I have had the pleasure to supervise or work closely with: Henrique, Carmen, Cassandra, Yuhong, Shule, Ha Sier, Phuoc, Eetu and Myrthe. I cannot leave out my colleagues Jani and Rameez, thank you for all those coffe times together and the board game evenings with the most extravagant deliberations.

I also want to thank the collaborators that have helped and guided me, and with whom I have had the pleasure to delve into interesting technical and personal discussions. My thanks go to Jenni Raitoharju, David Hätsbacka, Wenshuai Zhao, Fabrizio Schiano, Eduardo Castelló Ferrer, Matti Hämäläinen, Kostantin Mikhaylov and Juha Röning. I would also like to mention and thank my research director and close colleagues in Turku: Hanu Tenhunen, Juha Plosila, Jukka Heikkonen, Paavo and Hashem. Special thanks go to Harry Edelman for the countless hours of joint work, jokes, travels, and definitely some of the most interesting conversations.

While my doctoral research has been in Finland, this journey started with my master studies in Fudan University, in China, where I have continued to visit whenever it has been possible. I would like to thank Zhuo, Alexa and Lirong for their support during this time.

I cannot leave out of these acknowledgments the thesis reviewers and pre-examiners, who have provided valuable advice and suggestions on how to improve this thesis. My thanks go to Lecturer Simon Watson and Professor Raivo Sell for their thorough reviews and comments. I also would like to mention that this research has been supported by the University of Turku Graduate School, by the Nokia Foundation and by the Finnish Foundation for Technology Promotion (TES).

I would like to close this with words for my family. My parents Paula and Víctor and my sister Laura have always been by my side and I would be nowhere near where I am without their support and encouragement. The last few years have kept us farther away from each other, but you kept supporting me, believing in me and always shaping my path forward.

Turku, November 23rd, 2022
Jorge Peña Queralta

JORGE PEÑA QUERALTA

Jorge Peña Queralta received two B.Sc. degrees in mathematics and physics engineering from UPC BarcelonaTech, Spain, in 2016, a M.Sc. (Tech.) degree in Information and Communication Science and Technology from the University of Turku, Finland, and a M. Eng. degree in Electronics and Communication Engineering from Fudan University, China, in 2018. Since 2018, he has been a researcher at the Turku Intelligent Embedded and Robotic Systems (TIERS) Lab, Faculty of Technology, University of Turku, and a doctoral candidate since 2019. His research interests include multi-robot systems, collaborative autonomy, distributed perception, aerial robotics, and edge computing.

Table of Contents

Abbreviations

AI	Artificial intelligence
AIoT	AI in the internet of things
ASRE	Application-specific resource ensemble
BVLOS	Beyond visual line of sight
CNN	Convolutional neural network
DAG	Directed acyclic graph
DDS	Data distribution service
DL	Deep learning
DLT	Distributed ledger technology
DMTSP	Dubins traveling salesman problem
DMTSPN	Dubins traveling salesman problem with neighborhoods
DRL	Deep reinforcement learning
EKF	Extended Kalman filter
EVM	Ethereum virtual machine
FMU	Flight management unit
FoV	Field of view
GNSS	Global navigation satellite system
GPS	Global positioning system
FPS	Frames per second
FPGA	Field programmable gate array
IMU	Inertial measurement unit
IPP	Informative path planning
IoT	Internet of things
LIO	Lidar-inertial odometry
LOS	Line-of-sight
LPWAN	Low-power wide-area network
MANET	Mobile ad-hoc network
MAS	Multi-agent system
MAV	Micro-aerial vehicle
MEC	Multi-access edge computing (*alt.* mobile edge computing)
ML	Machine learning
MOCAP	Motion capture
MPC	Model predictive control

MRS	Multi-robot system
MTSP	Multiple traveling salesman problem
NBV	Next best view
NDT	Normal distribution transform
NLOS	Non-line-of-sight
PBFT	Practical byzantine fault tolerance
PoA	Proof of authority
PoS	Proof of stake
PoW	Proof of work
QoS	Quality of service
QoR	Quality of results
QR	Quick response (code)
RNA	Radio access network
RGB-D	RGB and depth
RF	Radio frequency
RL	Reinforcement learning
RM	Resource manager
ROS	Robot operating system
RSSI	Received signal strength indicator
RTLS	Real-time localization system
SAR	Search and rescue
SDPG	Spiral directed path graph
SfM	Structure from motion
SIFT	Scale invariant feature transform
SLAM	Simultaneous localization and mapping
SwaaS	Swarm-as-a-service
TDoA	Time-difference of arrival
ToA	Time of arrival
ToF	Time of flight
UART	Universal asynchronous receiver-transmitter
UAV	Unmanned aerial vehicle
UGV	Unmanned ground vehicle
USV	Unmanned surface vehicle
UUV	Unmanned underwater vehicle
UWB	Ultra-wideband
VHDL	VHSIC hardware description language
VHSIC	Very high speed integrated circuit
VIO	Visual-inertial odometry
VLOS	Visual line of sight
VO	Visual odometry
VSLAM	Visual SLAM
VTOL	Vertical take-off and landing

List of Original Publications

This dissertation is based on the following original publications:

I	**Jorge Peña Queralta**, Li Qingqing, Eduardo Castelló Ferrer, Tomi Westerlund, ”Secure Encoded Instruction Graphs for End-to-End Data Validation in Autonomous Robots”, IEEE Internet of Things Journal, IEEE, 2022.
II	**Jorge Peña Queralta**, Li Qingqing, Fabrizio Schiano, Tomi Westerlund, ”VIO-UWB-Based Collaborative Localization and Dense Scene Reconstruction within Heterogeneous Multi-Robot Systems”, IEEE International Conference on Advanced Robotics and Mechatronics, IEEE, 2022.
III	**Jorge Peña Queralta**, Carmen Martínez Almansa, Fabrizio Schiano, Dario Floreano, Tomi Westerlund, ”UWB-based System for UAV Localization in GNSS-Denied Environments: Characterization and Dataset”, IEEE/ RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020.
IV	**Jorge Peña Queralta**, Li Qingqing, Tuan Nguyen Gia, Zhuo Zou, Hannu Tenhunen and Tomi Westerlund, ”Distributed Progressive Formation Control with One-Way Communication for Multi-Agent Systems”, IEEE Symposium Series on Computational Intelligence, IEEE, 2019.
V	**Jorge Peña Queralta**, Cassandra McCord, Tuan Nguyen Gia, Hannu Tenhunen and Tomi Westerlund, ”Communication-free and Index-free Distributed Formation Control Algorithm for Multi-robot Systems”, Procedia Computer Science, Elsevier, 2019. Presented at the 10th International Conference on Ambient Systems, Networks and Technologies (ANT).
VI	Cassandra McCord, **Jorge Peña Queralta**, Tuan Nguyen Gia and Tomi Westerlund, ”Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of UAVs in ROS/Gazebo with

RotorS", European Conference on Mobile Robots (ECMR), IEEE, 2019.

VII **Jorge Peña Queralta**, Li Qingqing, Tuan Nguyen Gia, Hong-Linh Truong, Tomi Westerlund, "End-to-End Design for Self-Reconfigurable Heterogeneous Robotic Swarms", International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2020.

VIII **Jorge Peña Queralta**, Li Qingqing, Zhuo Zou, and Tomi Westerlund, "Enhancing Autonomy with Blockchain and Multi-Access Edge Computing in Distributed Robotic Systems", The Fifth International Conference on Fog and Mobile Edge Computing (FMEC 2020), IEEE, 2020.

IX **Jorge Peña Queralta**, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, Tomi Westerlund, "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception and Active Vision", IEEE Access, IEEE, 2020.

X Yu Xianjia, Li Qingqing, **Jorge Peña Queralta**, Jukka Heikkonen, Tomi Westerlund, "Cooperative UWB-Based Localization for Outdoors Positioning and Navigation of UAVs aided by Ground Robots", IEEE International Conference on Autonomous Systems (ICAS), IEEE, 2021.

XI Li Qingqing, **Jorge Peña Queralt**a, Tuan Nguyen Gia, Tomi Westerlund, "Offloading Monocular Visual Odometry with Edge Computing: Optimizing Image Quality in Multi-Robot Systems", The 5th International Conference on Systems, Control and Communications, ACM, 2019.

XII **Jorge Peña Queralta** and Tomi Westerlund, "Blockchain for Mobile Edge Computing: Consensus Mechanisms and Scalability", Mobile Edge Computing (Book Chapter), Springer (2021).

The following related publications are not directly included in this thesis:

XIII **Jorge Peña Queralta**, Tuan Nguyen Gia, Hannu Tenhunen and Tomi Westerlund, "Collaborative Mapping with IoE-based Heterogeneous Vehicles for Enhanced Situational Awareness", IEEE Sensors Applications Symposium (SAS), IEEE, 2019.

XIV **Jorge Peña Queralta**, Tuan Nguyen Gia, Zhuo Zou, Hannu Tenhunen, Tomi Westerlund, ”Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN”, Procedia Computer Science, Elsevier, 2019. Presented at the 14th International Conference on Future Networks and Communications (FNC).

XV **Jorge Peña Queralta**, Tuan Nguyen Gia, Hannu Tenhunen and Tomi Westerlund, ”Edge-AI in LoRa-based Health Monitoring: Fall Detection System with Fog Computing and LSTM Recurrent Neural Networks”, 42nd International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2019.

XVI **Jorge Peña Queralta**, Fu Yuhong, Lassi Salomaa, Li Qingqing, Tuan Nguyen Gia, Zhuo Zou, Hannu Tenhunen, and Tomi Westerlund, ”FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS”, Proceedings of the IEEE Sensors Conference, IEEE, 2019.

XVII Tuan Nguyen Gia, **Jorge Peña Queralta**, Tomi Westerlund, ”Exploiting LoRa, edge, and fog computing for traffic monitoring in smart cities”, LPWAN Technologies for IoT and M2M Applications (book chapter), Elsevier, 2020.

XVIII Li Qingqing, **Jorge Peña Queralta**, Tuan Nguyen Gia, Zhuo Zou, Tomi Westerlund, ”Multi Sensor Fusion for Navigation and Mapping in Autonomous Vehicles: Accurate Localization in Urban Environments”, Unmanned Systems (2020). Presented at the 9th IEEE International Conference on Cybernetics and Intelligent Systems and the 9th IEEE International Conference on Robotics, Automation and Mechatronics (CIS-RAM).

XIX Carmen Martínez Almansa, Wang Shule, **Jorge Peña Queralta**, Tomi Westerlund, ”Autocalibration of a Mobile UWB Localization System for Ad-Hoc Multi-Robot Deployments in GNSS-Denied Environments”, International Conference on Localization and GNSS, CEUR WS Proceedings, 2020.

XX Anum Nawaz, **Jorge Peña Queralta**, Jixin Guan, Muhammad Awais, Tuan Nguyen Gia, Ali Kashif, Haibin Kan, Tomi Westerlund, ”Edge Computing to Secure IoT Data Ownership and Trade with the Ethereum Blockchain”, Sensors, MDPI, 2020.

XXI	Wang Shule, Carmen Martínez Almansa, **Jorge Peña Queralta**, Zhuo Zou, Tomi Westerlund, ”UWB-Based Localization for Multi-UAV Systems and Collaborative Heterogeneous Multi-Robot Systems: a Survey”, Procedia Computer Science, Elsevier, 2020. Presented at the 15th International Conference on Future Networks and Communications.
XXII	Wenshuai Zhao, **Jorge Peña Queralta**, Li Qingqing, Tomi Westerlund, ”Towards Closing the Sim-to-Real Gap in Collaborative Multi-Robot Deep Reinforcement Learning”, 5th International Conference on Robotics and Automation Engineering, IEEE, 2020.
XXIII	Wenshuai Zhao, **Jorge Peña Queralta**, Li Qingqing, Tomi Westerlund, ”Ubiquitous Distributed Deep Reinforcement Learning at the Edge: Analyzing Byzantine Agents in Discrete Action Spaces”, The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN), Elsevier, 2020.
XXIV	Wenshuai Zhao, **Jorge Peña Queralta**, Tomi Westerlund, ”Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”, IEEE Symposium Series on Computational Intelligence, IEEE, 2020.
XXV	Li Qingqing, Jussi Taipalmaa, **Jorge Peña Queralta**, Tuan Nguyen Gia, Moncef Gabbouj, Hannu Tenhunen, Jenni Raitoharju, Tomi Westerlund, ”Towards Active Vision with UAVs in Marine Search and Rescue: Analyzing Human Detection at Variable Altitudes”, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, 2020.
XXVI	Yu Xianjia, **Jorge Peña Queralta**, Jukka Heikkonen, Tomi Westerlund, ”Federated Learning in Robotic and Autonomous Systems”, Procedia Computer Science, Elsevier, 2021. Presented at the 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC).

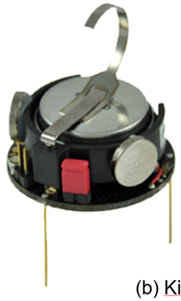
1 Introduction

The robotics field has seen unprecedented evolution over the past decade, with robots becoming increasingly intelligent [27]. This has happened in par with the boom of deep learning and AI in general [28; 29]. Indeed, ubiquitous autonomous entities are set to be the centerpiece behind the dynamic smart cities of the future, bringing novel solutions to mobility, transportation and service creation [30]. For many, the robotics revolution has started. Autonomous robots are already becoming an important facet of the Industrial Internet of Things (IIoT), with collaborative robots and mobile robotic solutions powering more dynamic, flexible and reconfigurable industrial production lines [31; 32; 33]. At the same time, robots are no longer isolated entities that operate a very concrete task, but are more intelligent and programmable, more connected, and more collaborative than ever before. Multi-robot systems are being deployed across an ample range of domains, from factory automation in the logistics sector to the management and control of hazardous materials, as well as in multiple civilian domains [9]. Monitoring and inspection in multiple industrial settings are growing areas of application for both ground and aerial robots [34]. Despite the rapid advances in the robotics field in recent years, and the shift from robot units to robot fleets, these increasingly autonomous entities are often managed individually, and mostly cooperate within homogeneous and well-defined teams. Challenges also arise when accounting for constrained connectivity and limited computing resources for intensive deep learning (DL) algorithms in many scenarios [27].

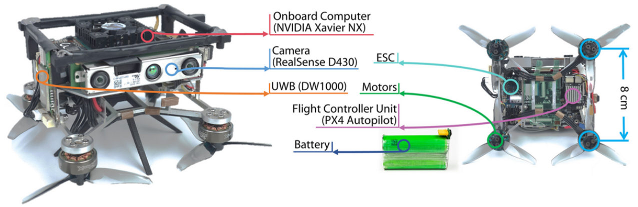
This thesis explores different aspects of the design of multi-robot systems from a system-level and algorithmic perspective. The literature in this area has been traditionally focused towards autonomous ground vehicles or unmanned ground vehicles (UGVs), with applications mainly in the industrial domains. However, recent years have seen impactful solutions and multi-robot systems moving out of the lab formed by aerial robots or unmanned aerial vehicles (UAVs) [35] and heterogeneous robot teams [36]. A selection of robot swarm platforms relevant within a research and education context is shown in Fig. 1.

A recurrent use case for robotics research, and multi-robot systems (MRS) research in particular, is search and rescue (SAR) robotics, among other civil application. The utilization of multi-robot systems within civil applications presents additional challenges owing to the interaction with humans and their deployment in potentially unknown environments [37; 38; 39]. Among civil applications, search and

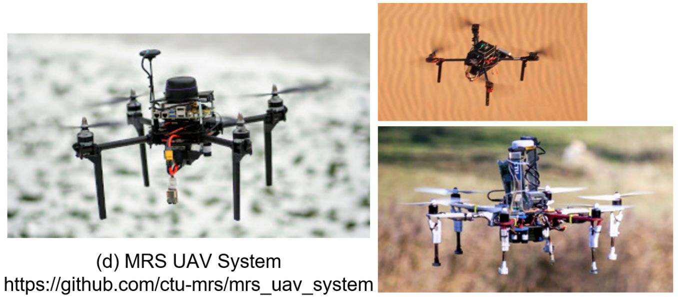
(a) Bitcraze's crazyflie
https://www.bitcraze.io/products/crazyflie-2-1/



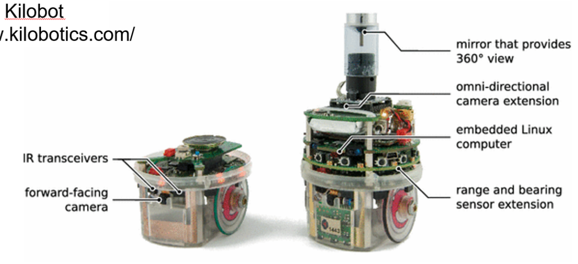
(b) Kilobot
https://www.kilobotics.com/


Onboard Computer (NVIDIA Xavier NX)
Camera (RealSense D430)
UWB (DW1000)
ESC
Motors
Flight Controller Unit (PX4 Autopilot)
Battery
8 cm

(b) Micro-drone swarm unit, published in:
X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, F. Gao,
Swarm of micro flying robots in the wild. Sci. Robot. 7, eabm5954 (2022).



(d) MRS UAV System
https://github.com/ctu-mrs/mrs_uav_system


mirror that provides 360° view
omni-directional camera extension
embedded Linux computer
range and bearing sensor extension
IR transceivers
forward-facing camera

(e) E-Puck 2
https://www.gctronic.com/doc/index.php/e-puck2

Figure 1. Illustration of different robotic platforms typically used in research related to multi-robot systems and swarm robotics.

rescue operations present a key scenario where autonomous robots have the potential to save lives by enabling faster response time [40; 41], supporting in hazardous environments [42; 43; 44], or providing real-time mapping and monitoring of the area where an incident has occurred [45; 46], among other possibilities. A significant part of the research in the thesis has been carried out within the umbrella of SAR robotics projects.

When discussing recent advances in robotic intelligence, it is worth mentioning a set of major events featuring state-of-the-art autonomous robotics solutions: the DARPA challenges, organized by the U.S. Defense Advanced Research Projects Agency (DARPA). Humanoid robots [47] and human-robot interaction and coordination strategies [48] for SAR operations were presented in the 2013-2015 DARPA Robotics Challenge. The DARPA Subterranean (SubT) Challenge, running in 2018-2021, shifted the focus towards underground MRS for SAR operations, with ground robots and UAVs collaborating in the tasks [49]. This challenge has demonstrated the versatility and significant increase of flexibility of heterogenous MRS [50], with robust UAV flight in inherently constrained environments [51], and ground robots with different locomotion modalities able of navigating complex environments and long-term autonomy [52]. In 2020, due to the Covid-19 pandemic, the challenge moved to a fully virtual edition with realistic simulation-based environments [53]. The winning team during the Final Event, CERBERUS, has presented their approach in [36]. The system deployed by the team was highly heterogeneous, with legged robots, different types of multi-rotor aerial robots and wheeled robots.

1.1 Motivation and objectives

The objective of this thesis is to advance in practical and theoretical aspects of multi-robot systems. To that end, the thesis explores multiple aspects involved in the design of multi-robot systems. In the different areas, the focus is consistently on robustness, reconfigurability, or in enabling operations in more complex situations (e.g., localization methods and collaborative sensing approaches for GNSS-denied environments, coordination with minimal to no communication, or architectures for self-reconfiguration).

The following are the key research questions that this thesis aims to answer throughout the different chapters, from more general to more specific concepts:

1. From a high-level architectural point of view, we study how can large-scale multi-robot systems or robot swarms leverage the increasing levels of connectivity present in today's hardware and software systems. To this end, we discuss what is the role that cloud and edge computing can play in building towards more resilient robotic systems. In addition to these, we also briefly explore what is the potential of specialized hardware for accelerating workloads while enabling deterministic and real-time applications.

2. With distributed ledger technologies (DLTs) identified as a promising platform for distributed decision-making and management of robotic systems from edge to cloud, we then move the focus towards exploring their potential. From traditional blockchains to newer, more scalable, and parallelizable solutions, different consensus algorithms naturally suit different use cases. From a general point of view, we delve into how to leverage DLTs for designing a framework for trustable collaboration in heterogeneous multi-robot systems. Connecting to the previous questions, we can also ask how to go beyond distributed resource orchestration at the edge towards seamless collaboration with distributed role allocation, multi-modal sensor fusion at the edge and collaborative sensing. A key concept here is also the definition of the concept of trust between autonomous agents, whether they are robots or other connected systems (e.g., connected infrastructure and sensors in the IIoT).

3. While DLTs enable secure and trustable collaboration, mainly from the perspective of a networked system, they do not a priori provide an out-of-the-box methodology for securing the interaction of robots with their environment. Partly inspired by some of the mechanisms utilized in DLTs, we ask whether cryptographic hashes can be effectively utilized to encode instructions for autonomous robots in a way that they serve to validate their correct operation or interaction with their environment. This covers the definition in which expected sensor inputs can be encoded in a way that they remain reproducible for robots under a predefined level of uncertainty once they are deployed to their missions. We also explore the potential of such an approach for managing multi-robot cooperation in a secure manner.

4. The first part of the thesis focuses on high-level design, while the latter part deals with specific algorithms. While the open questions in the topics introduced above are very wide, the last chapters shift the focus to specific algorithmic design problems. This starts with the problem of localization. To achieve robust spatial coordination in a multi-robot system, and to be able to deploy such a system in GNSS-denied environments, we study what is the potential of wireless ranging based on ultra-wideband (UWB) transmissions for global or relative positioning. We look into whether out-of-the-box systems are suitable for agile robots with low-latency control requirements such as aerial drones, and whether ground-to-air relative positioning between different robots outperforms GNSS solutions in urban environments where GNSS signals degrade.

5. We then explore what type of spatial coordination, or formation control, algorithms can be robust to loss of most or all communication. To this end, we study two different systems. First, we analyze whether communication-free and anonymous spatial coordination is possible based on mutual sensing

only, and what are the limitations. Second, we ask whether more generalizable spatial distributions can be achieved introducing one-way communication but only awareness of other robots within one's field-of-view.

6. Finally, we seek to combine graph theory typically applied to formation control algorithms, connectivity and rigidity analysis, with UWB relative localization for collaborative sensing in heterogeneous multi-robot systems. To this end, we explore whether path planning for a collaborative scene reconstruction mission can be defined in a way that ground robots maintain line-of-sight with aerial robots assuming limited field-of-view, which would lead to more efficient collaborative autonomy. We then analyze whether the relative localization systems developed during this thesis can be used for combining data from different robots and different sensors to build a three-dimensional map of the environment and, if possible, without further optimization.

1.2 Multi-robot coordination algorithms

This section, adapted from [9], introduces basic concepts related to algorithmic design for multi-robot system from a coordination and planning perspective. We describe the main algorithms required for multi-robot coordination and planning in collaborative applications. These are key enablers of MRS capabilities in terms of exploration and navigation over large areas. We discuss this mainly from the point of view of cooperative multi-robot systems. The literature in multi-robot cooperative exploration or collaborative sensing contains mostly generic approaches that consider multiple applications. The main problems discussed in this section are the following:
- *Multi-robot task allocation*: distribution of tasks and objectives among the robots (e.g., areas to be searched, or positions to be occupied to ensure connectivity among the robots and with the base station)
- *Path planning and area coverage*: global path planning covers area coverage (generation of paths to entirely analyze a given area) and area partition (dividing the area between multiple robots). Local planning and deals mainly with obstacle and collision avoidance, incorporating robot dynamics.
- *Area exploration*: coverage and mapping algorithms (or discover/ search for specific objects) in potentially unknown environments.
- *Centralized multi-robot planning*: decision-making on the actions of multiple robots by either gathering and processing data in a single node, from which decisions are distributed to others, or by achieving consensus through communication (often requiring agents to be aware of all others, and stable communication).
- *Distributed multi-robot planning*: algorithms enabling agents to make independent decisions individually or in subsets based only on their own data or data shared by

their neighbors. These do not necessarily need agents to be aware of the existence or state of all other agents in the system.

1.2.1 Multi-robot task allocation

A comparative study on task allocation algorithms for multi-robot exploration was carried out by Faigl et al. in [54], considering five distinct strategies: greedy assignment, iterative assignment, Hungarian assignment, multiple traveling salesman assignment, and MinPos. However, most of these approaches are often centralized from the decision-making point of view, even if they are implemented in a distributed manner. Others, such as MinPos, shift between the two modalities depending on the availability of communication. Successive works have been presenting more decentralized methods. Decentralized task allocation algorithms for autonomous robots are very often based on market-based approaches and auction mechanisms to achieve consensus among the agents [55; 56; 57; 58]. Both of this approaches have been extensively studied for the past two decades within the multi-robot and multi-agent systems communities [59; 60]. Bio-inspired algorithms have also been widely studied within the multi-robot and swarm robotics domains. For instance, in [61], Kurdi et al. present a task allocation algorithm for multi-UAV search and rescue systems inspired by locust insects. Active perception techniques have also been incorporated in multi-robot planning algorithms in existing works [62; 63].

Through this thesis, we explore task allocation from a general perspective when we study the application of distributed ledger technologies for achieving consensus within multi-robot systems. In terms of role allocation from a spatial coordination perspective, we also study different formation control algorithms, which are then put in the context of the relevant literature in the corresponding chapter.

1.2.2 Area coverage and path planning

A common problem that multi-robot systems solve is mapping or inspection of a given area. For this purpose, path planning algorithms can be part of area coverage algorithms or implemented separately for robots to cover their assigned areas individually. In any case, when area coverage algorithms consider path planning, it is often from a global point of view, leaving the local planning to the individual agents. A detailed description of path planning algorithms including approaches of linear programming, control theory, multi-objective optimization models, probabilistic models, and meta-heuristic models for different types of UAVs is available in [64]. While some of these algorithms are generic and only take into account the origin and objective position, together with obstacle positions, others also consider the dynamics of the vehicles and constraints that these naturally impose in local curvatures, such as Dubin curves [64].

Area coverage and path planning algorithms take into account mainly the shape of the objective area to be surveyed. Nonetheless, a number of other variables are also considered in more complex algorithms, such as energy consumption, range of communication and bandwidth, environmental conditions, or the probability of failure. This data is not necessarily available a priori, and therefore it is also in the interest of the robots to collect data affecting the planning outcome while operating. The problem of maximizing the utility of data collection is called the informative path planning (IPP) problem [65]. IPP approaches have been shown to outperform more traditional planning algorithms such as greedy algorithms and genetic algorithms [66].

The specific dynamics and capabilities of the robots being used can also be utilized to optimize the performance of the area coverage, for example when comparing the maneuverability of quadrotors and fixed-wing UAVs. Cabreira et al. have presented algorithms for coverage path planning with UAVs [67].

Area coverage algorithms can be broadly classified in terms of the assumptions they make on the geometry of the area to be covered. The most basic approaches consider only convex and joint areas [68], for which paths can be efficiently generated based on area decomposition algorithms [69; 70].

Recent works have considered more complex environments. For instance, in [71], Xie et al. presented a path planning algorithm for UAVs covering disjoint convex regions. The authors' method considered an integration of both coverage path planning and the traveling salesman problem. In order to account for scalability and real-time execution, two approaches were presented: a near-optimal solution based on dynamic programming, and a heuristic approach able to efficiently generate high-quality paths, both tested under simulation environments. Also aiming at disjoint but convex areas, Vazquez et al. proposed a similar method that separates the optimization of the order in which the different areas were visited and the path generation for each of them [72]. Both of this cases, however, provide solutions for individual UAVs.

In the presence of known obstacles inside the objective area, the search area can be considered non-convex [73]. However, non-convex approaches can often be applied to more general environments. More realistic scenarios often require the exploration of disjoint areas [72].

The problem of disjoint area search can be formulated as a multi-objective optimization problem, where each of the joint subareas can be considered a single objective in the path planning object. This leads to the differentiation between multi-agent single-objective planning and multi-agent multi-objective optimization and planning. The former case is not necessarily a subset of the latter, as it also includes use cases such as collaborative transportation, applicable in emergency scenarios, or can provide higher degrees of fault-tolerance and robustness against the loss of agents. The latter case, nonetheless, is more significant within the scope of this section as multi-agent multi-objective optimization algorithms enable more efficient search in complex environments with distributed systems [74; 75]. Finally, the most comprehensive approaches also account for the existence of unknown environments in the areas to be searched, with the existence of potential obstacles that are a priori unknown. When multi-robot systems are utilized, the increased number of variables already involved in single-agent planning increase the complexity of the optimization problems while at the same time bring new possibilities to more efficient area coverage. For instance, energy awareness among the agents could enable robots with less operational time to survey areas near the deployment point, while other robots can be put in charge of farther zones. The communication system being utilized and strategies for connectivity maintenance play a more important role in multi-robot systems. If the algorithms are implemented in a distributed manner or the robots rely on online path planning, then the paths themselves must ensure robust connectivity enabling proper operation of the system. Security within the communication, while also important in the single-agent case, plays again a more critical role when multiple robots communicate among themselves in order to take cooperative decisions in real-time. Furthermore, the optimization problems upon which multi-robot area coverage algorithms build are known to belong to the NP-hard class of non-deterministic polynomial time algorithms [76]. Therefore, part of the existing research has focused towards probabilistic approaches.

1.2.3 Planning for different robots (U*X*Vs)

Mobile robots operating on different mediums necessarily have different constraints and a variable number of degrees of freedom. For local path planning, a key aspect to consider when designing control systems is the holonomic nature of the robot. In a holonomic robot, the number of controllable degrees of freedom is equal to the number of degrees of freedom defining the robot's state. In practice, most robots are non-holonomic, with some having significant limitations to their local motion such as fixed-wing UAVs [77], or unmanned surface vessels (USVs) [78]. However, quadrotor UAVs, which have gained considerable momentum owing to their flexibility and relatively simple control, can be considered holonomic [79]. Ground robots equipped with omniwheel mechanisms and able of omnidirectional motion can be also considered holonomic if they operate on favorable surfaces [80].

Multiple works have been devoted to reviewing the different path planning strategies for robots or unmanned vehicles across different mediums: aerial robots [64], surface robots [81], underwater robots [82; 83], and ground robots for urban [84], or wilderness [85] environments. From these works, we have summarized the main constraints to be considered in path planning algorithms in Fig. 2.

The main limitations in robot navigation, and therefore path planning, in different mediums can be roughly characterized by: (i) dynamic environments and move-

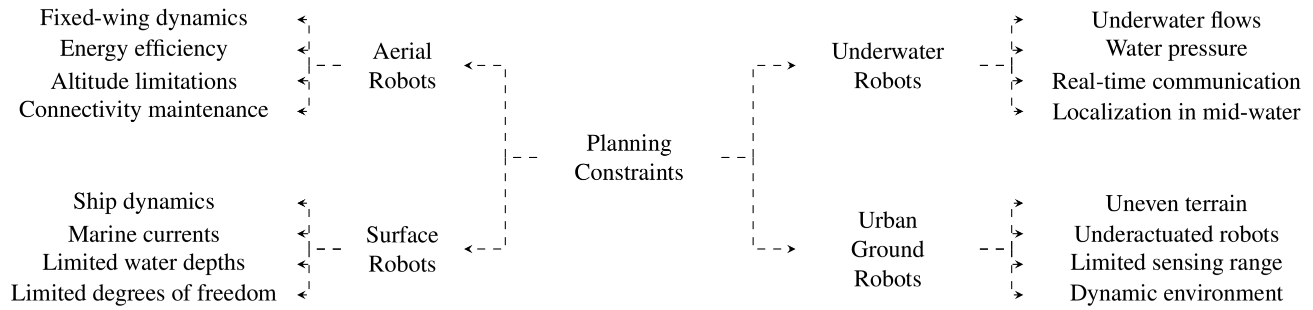


Figure 2. Main path planning constraints that autonomous robots in different domains need to account for. Some of these aspects are common across the different types of robots, such as energy efficiency and inherent constraints from the robots' dynamics, but become more predominant in UAVs and USVs, for instance.

ment limitations in ground robots; (ii) energy efficiency, situational awareness, and weather conditions in aerial robots; (iii) underactuation and environmental effects in surface robots, with currents, winds and water depth constraints; and (iv) localization and communication in underwater robots.

1.2.4 Multi-robot path planning

Research in the field of multi-robot path planning has been ongoing for over two decades. An early approach to multi-robot cooperation was presented in [86] in 1995, where the authors introduced an incremental plan-merging approach that defined a global plan shared among the robots. A relatively simple yet effective mechanism was utilized to maintain a consistent global plan: the robots would ask others for the right to plan for themselves and update the global plan accordingly one by one. This approach, while distributed, would not match the real-time needs and standards of today, nor does it exploit parallel operations at the robots during the distributed planning.

In [69], an early generalization of previous algorithms towards nonconvex and nonsimply connected areas was presented, enabling deployment in more realistic scenarios. The advances since then have been significant in multiple directions. With the idea or providing fault-tolerant systems, in [68] the authors introduced a reconfiguration process that would account in real-time for malfunctioning or missing agents, and adjust the paths of remaining agents accordingly. Considering the need of inter-robot communication for aggregating and merging data, a cooperative approach to multi-robot exploration that considers the range limitations of the communication system between robots was introduced in [87]. Non-polygonal area partitioning methods have also been proposed. Covering the topics of connectivity maintenance and IPP, a multi-robot IPP approach to managing continuous connectivity constraints appears in [88].

Existing approaches often differentiate between area coverage and area exploration. In area coverage algorithms, algorithms focus on optimally planning paths for traversing a known area, or dividing a known area among multiple agents to optimize the time it takes to analyze it. Area exploration algorithms focus instead on the coverage and mapping of potentially unknown environments. The two terms, however, are often used interchangeably in the literature. An overview and comparison of multi-robot area exploration algorithms is available in [89].

A subset of multi-robot path planning algorithms are formation control algorithms. Formation control or pattern formation algorithms are those that define spatial configurations in multi-robot systems [90]. Most formation control algorithms for multi-agent systems can be roughly classified in three categories from the point of view of the variables that are measured and actively controlled by each of the agents: position-based control, displacement-based control, and distance or bearing-based control [90].

1.2.5 Planning in heterogeneous multi-robot systems

Most existing approaches for multi-robot exploration or area coverage either assume that all agents share similar operational capabilities, or that the characteristics of the different agents are known a priori. Pre-defined optimization objectives are also used in this thesis to plan the paths of UAVs that collaborate with UGVs in the last chapter. In the following we point to some works introducing different coordination approaches.

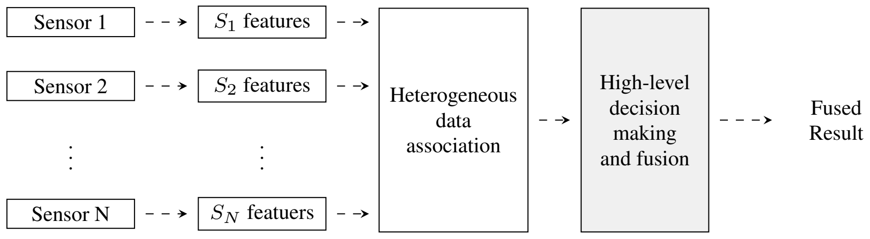
In building towards more robust and resilient multi-robot planning algorithms, Mueke et al. introduced a system-level approach for decentralized coordination of heterogeneous multi-robot systems [91]. The literature in this area is however scarce, with the majority of the approaches in the literature and in international projects involving a priori assignments. Indeed, in existing systems that are formed up by heterogeneous robots, the way in which they are meant to cooperate is often predefined. From a general perspective, an extensive review on control strategies for collaborative area coverage in heterogeneous multi-robot systems was recently presented by Abbasi [92]. Another source of information with a detailed survey on cooperative heterogeneous multi-robot systems is the work Rizk et al. available in [93].

1.2.6 Sensor fusion and multi-robot perception

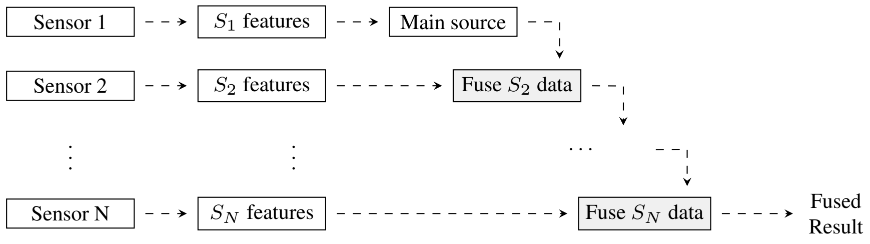
Towards the end of this thesis, we explore the topic of multi-robot sensor fusion. We do so from the perspective of combining relative localization with three-dimensional pointcloud merging from different types of sensors (lidars and stereo cameras) in different robots. However, the area of sensor fusion and multi-robot perception is an active research topic, and thus we introduce here some key concepts.

Multi-modal information fusion aims at combining data from a multiple sources, e.g., images and LiDAR. Information fusion techniques have been actively researched for decades and there is a myriad of different ways to approach the problem. The approaches can be roughly divided into techniques fusing information on raw data/input level, on feature/intermediate level, or on decision/output level [96]. An overview of the main data fusion approaches in multi-modal scenarios is illustrated in Fig. 3.

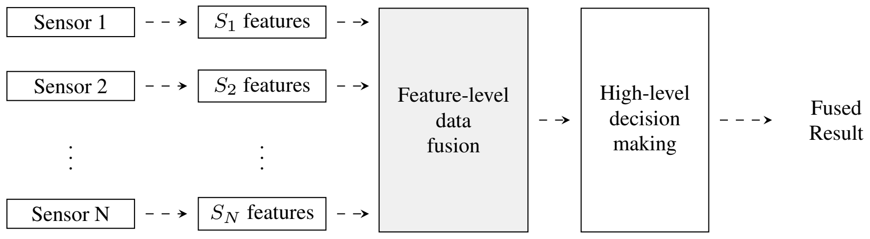
Some of the main challenges include *representation*, i.e., how to represent multi-modal data taking into account complementarity and redundancy of multiple modalities, *translation*, i.e., how to map the data from different modalities to a joint space, *alignment*, i.e., how to understand the relations of the elements of data from different modalities, for example, which parts of the data describe the same object in an image and in a point-cloud produced by LiDAR, *fusion*, i.e., how to combine the information to form a prediction, and *co-learning*, i.e., how to transfer knowledge between



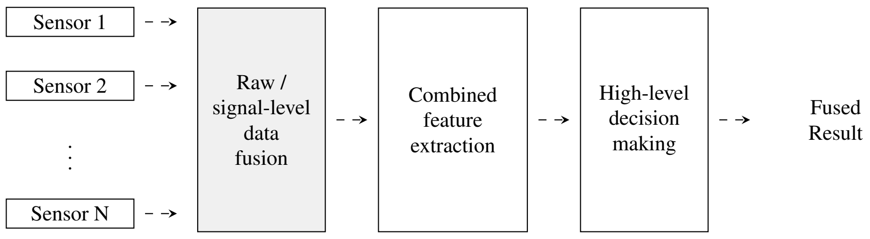
(a) Data integration (parallel processing of modalities)



(b) Sequential processing of modalities, from higher to lower confidence or quality sources.



(c) True fusion using features (high-level features or multivariate features).


Sensor 1
Sensor 2
Sensor N
Raw / signal-level data fusion
Combined feature extraction
High-level decision making
Fused Result

(d) True fusion with minimal reduction.

Figure 3. Different multi-modal data fusion approaches: (a) parallel data integration with high-level decision making, (b) sequential processing of modalities when different modalities have difference confidence or quality levels, (c) true fusion with high-level features or with multivariate features, and (d) true fusion with minimal reduction [94; 95]. In gray, we highlight the stage in which the fusion happens.

the modalities, which may be needed, for example, when one the modalities is not properly annotated [97]. The main challenges related to multi-modal data are listed in Table 1.

In research years, also the information fusion techniques have focused more and more on big data and deep learning. Typical deep learning data fusion techniques have some layers specific to each data source and the features can be then combined before the final layers or processed separately all the way to the network output, while the representations are coordinated through a constraint such as a similarity distance [98; 97].

To design efficient collaboration within multi-robot systems, and to achieve collaborative perception or multi-robot situational awareness, there should be also data fusion from sensors in different robots. For example, an object seen from two different angles could be potentially recognized with a higher accuracy. The sensors carried by different robots may be the same, typically cameras, or different as the presence of multiple agents makes it possible to distribute some of the sensors' weight between the agents, which is important especially in UAV applications. The goal is that the perception the agents have of their environment is based on aggregating information from multiple sources and the agents share data steadily among themselves or to a centralized control station (e.g., through the cloud).

The challenges described above for single-robot multi-sensor fusion systems are further complicated by the fact that the data to be fused is located in different physical locations and the sensors are now moving with respect to each other. Some of the challenges that need to be solved are where to perform data fusion, how to evaluate whether different agents are observing the same objects or not, or how to rank observations from different agents. For many of the challenges, there are no efficient solutions yet.

1.2.7 Shared autonomy

In multi-robot systems and robots involving complex manipulation (e.g., mobile manipulators) or with a high number of degrees of freedom, such as humanoids, the concept of shared autonomy gains importance. Shared autonomy refers to the autonomous control of the majority of degrees of freedom in a system, while designing a control interface for human operators to control a reduced number of parameters defining the global behavior of the system [99].

For instance, in [100] the authors describe the design principles followed in the DARPA Robotics Challenge to give the operators of a humanoid robot enough situational awareness while simplifying the actual control of the robot via predefined task sequences. This results in automated perception and control, avoiding catastrophic errors due to and exceeding amount of unnecessary information overwhelming the operator, while still enabling timely reaction and operation flexibility for unknown

Table 1. Main challenges in multi-modal and multi-source data fusion

Challenge	**Description**
Noisy data	Different data sources suffer from different types and magnitudes of noise. A heterogeneous set of data sources naturally comes with heterogeneous sources of noise, from calibration errors to thermal noise.
Unbalanced data	Having different data sources often involves data with different characteristics in terms of quality and confidence, but also in terms of spatial and temporal resolution.
Conflicting data	Data from different sources might yield conflicting features. For example, in the case of autonomous robots, different types of sensors (visual sensors, laser rangefinders or radars) might detect obstacles at different distances. Missing data over a certain time interval from one of the sources might also affect the data fusion.

environments. In [101], a semi-autonomous trajectory generation system for mobile multi-robot systems with integral haptic shared control was presented. The main novelty was not only in providing an interface for controlling a system with a high number of degrees of freedom through a reduced number of parameters defining the path shape, but also in providing real-time haptic feedback. The feedback provides information about the mismatch between the operator's input and the actual behavior of the robots, which is affected by an algorithm performing autonomous collision avoidance, path regularity checks, and attraction to points of interest. The authors performed experiments with a UAV, where different levels of control were given to the operator. Other works by the same authors defining multi-UAV shared control strategies are available in [102; 103].

1.2.8 Communication

Communication plays a vital role in an multi-robot systems due to the need of coordination and information sharing necessary to carry out collaborative tasks. A mobile ad-hoc network (MANET) is often formed for wireless communication and routing messages between the robots. Owing to the changing characteristics in terms or wireless transmission in different physical mediums, different communication technologies are utilized for various types of robots. An overview of the main MRS communication technologies is available in [104].

Collaborative MRS need to be able to communicate to keep coordinated, but also need to be aware of each other's position in order to make the most out of the shared data [105; 13]. Situated communication refers to wireless communication technologies that enable simultaneous data transfer while locating the data source [106]. Ubiquitous wireless technologies such as WiFi and Bluetooth have been exploited to enable localization [107; 108; 109; 110; 111; 112; 113]. These approaches have been traditionally based on the received signal strength indicator (RSSI) and the utilization of either Bluetooth beacons in known locations [110; 111; 112], or radio maps that define the strength of the signal of different access points over a predefined and surveyed area [107; 109]. More recently, other approaches rely on angle-of-arrival [108], now built-in in Bluetooth 5.1 devices [114]. Ultra-wideband (UWB) technology has emerged as a more accurate and less prone to interference alternative to Wi-Fi and Bluetooth [21]. With most existing research relying on fixed UWB transceivers in known locations [3], this thesis focuses on advancing towards mobile positioning systems or collaborative localization [19; 2; 10]. A recent trend has also been to apply deep learning in positioning estimation [115].

From the point of view of multi-robot coordination, maintaining connectivity between the different agents participating in a mission is critical. Connectivity maintenance in wireless sensor networks has been a topic of study for the past two decades [116]. In recent years, it has gained more attention in the fields of MRS with decentralized approaches [117]. Connectivity maintenance algorithms can be designed coupled with distributed control in multi-robot systems [118], or collision avoidance [119]. Xiao et al. have recently presented a cooperative multi-agent search algorithm with connectivity maintenance [120]. Similar works aiming at cooperative search, surveillance or tracking with multi-robot systems focus on optimizing the data paths [121] or fallible robots [122; 123]. Another recent work in area coverage with connectivity maintenance is available in [124]. A comparison of local and global methods for connectivity maintenance of multi-robot networks from Khateri et al. is available in [125].

In environments with limited connectivity, building and maintaining communication maps with information about the coverage and reliability of communication in different areas brings evident benefits. To this end, Amigomi et al. have presented a method for updating communication maps in an online manner under connectivity constraints [126]. A survey on multi-robot exploration of communication-restricted environments is available in [127].

It is worth noting that this section has been restricted to in-air communication and ground or aerial multi-robot systems. There is also a significant volume of research in the area of aquatic communication, which presents significant challenges. Aquatic multi-robot systems and their communication is, however, out of the scope of this thesis.

1.2.9 Localization in GNSS-denied environments

Localization is one of the main challenges in the deployment of mobile robots. Localization approaches can be divided among those providing global localization, and others focusing on relative localization (odometry) with respect to the initial position during deployment. The former case is most notably represented by GNSS sensors. However, robots are often deployed in GNSS-denied environments (e.g., underground, indoor fires) or environments where GNSS sensors do not provide enough accuracy (e.g., dense urban environments or forests). Global localization with other onboard sensors can be achieved, for example, with image matching [128], or lidar data matching [129].

Among the different approaches to onboard odometry, visual methods have gained significant traction due to their low price, passive nature and flexibility [130]. This is the case, for instance, of visual-inertial odometry with either monocular cameras [131], or multiple sensors [132]. However, these sensors present limitations in challenging environments with low-light or low-visibility conditions. In dense urban environments, lidar-based odometry is the only viable solution for long-term autonomy if high-accuracy localization is required [18].

Simultaneous Localization and Mapping (SLAM) approaches utilize odometry algorithms to build local maps [133; 134], while utilizing those maps later on for more stable and global localization, where now the global term refers to the scope of the mission since deployment, or since the process of building the map started. The different teams participating in the DARPA SubT challenge have employed various SLAM approaches with both lidar-based and vision-based approaches. Some of the specific algorithms have been ORB-SLAM in [52], or Hector SLAM [49].

1.3 Multi-robot systems for search and rescue

Many of the algorithms, methods and approaches introduced in this thesis do not target a specific application. However, the focus is often on operation in GNSS-denied environments and multi-robot coordination. Two projects that have shaped the research included in this thesis are AutoSOS[1] and RoboMesh[2]. The former one deals with the design and development of multi-drone systems for search and rescue, while the latter focuses on industrial applications and DLT solutions. It is therefore relevant within the context of this thesis to review applications of multi-robot systems within these application domains. The role that different robots can play in SAR operations is summarized in Figure 4.

Search and rescue (SAR) operations can take significant advantage from supporting autonomous or teleoperated robots and multi-robot systems. These can aid in mapping and situational assessment, monitoring and surveillance, establishing communication networks, or searching for victims. Here we briefly discuss multi-robot systems supporting SAR operations, with system-level considerations and focusing on the algorithmic perspectives for multi-robot coordination and perception.

Autonomous or teleoperated robots have been playing increasingly important roles in civil applications in recent years. Across the different civil domains where robots can support human operators, one of the areas where they can have more impact is in search and rescue (SAR) operations. In particular, multi-robot systems have the potential to significantly improve the efficiency of SAR personnel with faster response time [40; 41], support in hazardous environments [42; 43; 44], or providing real-time mapping and monitoring of the area where an incident has occurred [45; 46], among other possibilities.

Robotic SAR systems can differ in multiple ways: their intended operational environment (e.g., urban, maritime, or wilderness), the amount and type of robots involved (surface, aerial, ground or underwater - USVs, UAVs, UGVs, UUVs), their level of autonomy, and the ways in which humans control the robotic systems, among other factors.

[1] AutoSOS project: `https://tiers.utu.fi/project/autosos`

[2] RoboMesh project: `https://tiers.utu.fi/project/robomesh`

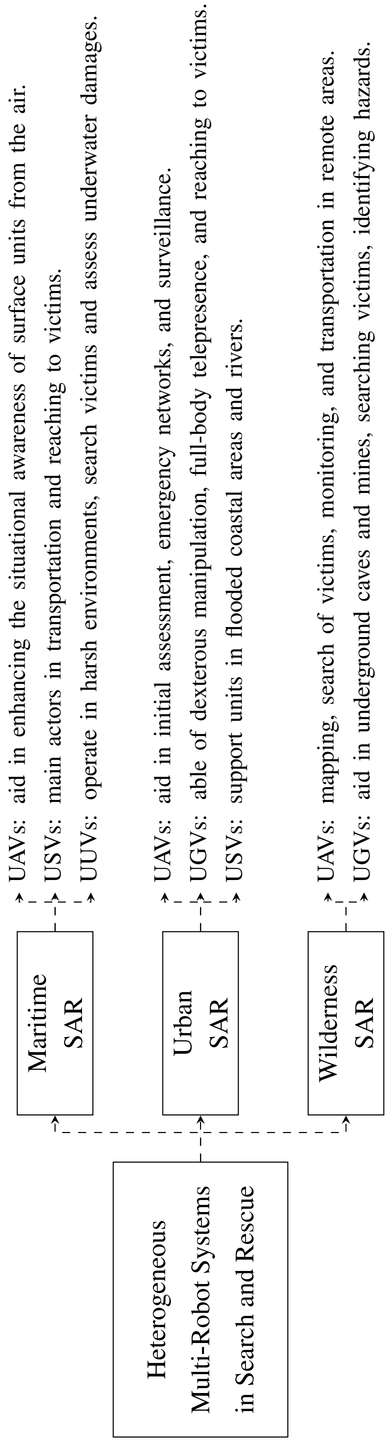

Heterogeneous
Multi-Robot Systems
in Search and Rescue
Maritime
SAR
UAVs: aid in enhancing the situational awareness of surface units from the air.
USVs: main actors in transportation and reaching to victims.
UUVs: operate in harsh environments, search victims and assess underwater damages.
Urban
SAR
UAVs: aid in initial assessment, emergency networks, and surveillance.
UGVs: able of dexterous manipulation, full-body telepresence, and reaching to victims.
USVs: support units in flooded coastal areas and rivers.
Wilderness
SAR
UAVs: mapping, search of victims, monitoring, and transportation in remote areas.
UGVs: aid in underground caves and mines, searching victims, identifying hazards.

Figure 4. Types of autonomous robots utilized in different SAR scenarios and their main advantages.

1.4 Terminology

Owing to the multi-disciplinary nature of this thesis, and the various domains covered, this section introduces a series of terms that are less common across fields. Additionally, we aim at clarifying the use in this thesis of some terms that do not have a unified meaning in the literature, such as the distinction between robot swarms and multi-robot systems.

Multi-robot systems and robot swarms

The terminology used in the robotics field to refer to systems composed of multiple robots is far from unified. Such systems are often called, in an interchangeable manner, multi-robot systems (MRS), multi-agent systems (MAS), or robot swarms [135]. In this thesis, the focus is largely on multi-robot systems. However, we refer to robot swarms when discussing more generic or abstract architectures, or when referring to large-scale MRS.

In general, we follow the definitions introduced in [135] when differentiating between MRS and swarms. Swarms of robots are scalable systems where interactions between robots are mostly local, driving emergent behaviors, and where members share equivalent or similar operational capabilities. In contrast, when we discuss about multi-robot systems, we often refer to a well-defined system composed of a, maybe predefined, series of robots that may or may not share operational capabilities, and where robot identities and global coordination often play an important role. Therefore, a robot swarm is a special case of a multi-robot system. However, in parts of this thesis we do use the term *swarm* referring to large-scale multi-robot systems, where individual robots no longer play a significant role.

Technologies in edge and cloud computing

The first few chapters of this thesis cover topics at the intersection of the robotics and IoT domains. Some of the terms introduced, such as network slicing or elasticity, are not necessarily common within the robotics field. We also discuss more general concepts such as application containerization. We refer the reader to our survey paper in this area for reference of these and other terms and technologies [136].

Distributed ledger technologies and blockchains

Also in the first few chapters, we discuss about the potential of blockchain technology and related cryptographic concepts. For an introduction to these terms, we refer the reader to one of the earliest works in the literature exploring the integration of blockchain and robotics [137]. A specific term worth noting here is, however, *byzantine robot*. In many related works, byzantine robots are malicious agents that affect the performance of a multi-robot system by any means, whether it is through alteration or fabrication of sensor or communication data, or by performing adversarial attacks on consensus algorithms. Throughout this thesis, we refer to the term byzantine robot more broadly within the context of a byzantine fault, which refers to any fault in a component in a distributed system that results in the system failing in its mission. Therefore, a byzantine robot is not necessarily malicious, but its byzantine actions might be also caused by malfunctioning sensors or actuators, or by external agents (e.g., within the context of data spoofing).

1.5 Contributions

The core objective of this thesis is to advance towards more robust multi-robot systems. The main design ideas pursued in this thesis are decentralization, collaboration and adaptiveness to different environments and operational situations. While pursuing those design principles, in turn, such a system must meet requirements in terms of security, trust and collaboration (to achieve efficient decentralization), and flexibility of the underlying collaboration mechanisms (to achieve solutions that do not depend, e.g., on external infrastructure such as GNSS). It is also worth mentioning that the thesis addresses, whenever possible, heterogeneity in the systems, from different types of robots to different operational capabilities. In the experiments, the focus is on using ground robots and aerial robots with different types of sensors.

This thesis therefore explores intercorrelated yet transversal topics including:

i. **Architectural design (Chapter 2):** we introduce a high-level design of reconfigurable robot swarms in the edge-cloud continuum [7], and blockchain-based architectures for edge-assisted solutions [8]. We have also reviewed the state-of-the-art in both key theoretical and practical aspects of multi-robot systems [9] and blockchain solutions for the general edge computing domain [12]. We explore the intersection of edge computing approaches and autonomy in mobile robotics [8; 11]. The key novelties in this chapter are in the study of edge computing architectures and potential implementation advantages, and a position on design architectures for multi-robot systems in the edge-cloud continuum.

ii. **Security and trust (Chapters 3-4)**: this thesis introduces blockchain-based approaches to the design and implementation of secure and trustable multi-robot systems, as well as for managing the collaboration between the robots. Importantly, we propose a novel approach to securing autonomous robot missions using encoded instruction graphs [1].

iii. **Localization (Chapter 5):** we explore approaches for GNSS-denied environments, from off-the-shelf and external UWB-based systems [3] to relative localization based on fusion of UWB ranging and VIO egomotion estimation [2], an approach also applicable outdoors in environments where GNSS signals degrade [10]. The key contributions are in the analysis of novel localization technologies and their potential for single and multi-robot systems.

iv. **Spatial coordination (Chapter 6):** we extend the state of the art with methods requiring zero to minimal communication (index-free and identity-free solutions [5], for robust operation in the lack or inability to communicate, but assuming robots can *see* each other) as well as signaling solutions with individual robot identities [4; 6]. In the latter approaches, UWB ranging can be used for simultaneous signaling and relative localization, while a blockchain framework can be leveraged for robot identities.

v. **Collaborative sensing (Chapter 7):** to pursue a higher-level result from an application perspective, we leverage relative localization solutions for multi-robot sensor fusion and collaborative scene reconstruction [2], in a work where we also introduce a novel path planning approach and graph rigidity monitoring to ensure a unique solution to the relative localization problem exists.

Parts of this thesis have already been published in different peer-reviewed papers. Other parts extend or further detail previous results, or have only been made available in online preprints. A significant amount of the theoretical results and approaches included in this thesis have been validated through different types of experiments with real robots [1; 2; 3; 5; 10], while other results have been validated in simulation [4; 6; 11; 13]. Finally, some of the works present peer-reviewed proposals to novel architectures or approaches in the design of multi-robot systems and the underlying technologies [7; 8; 12].

1.6 Thesis organization

The rest of the thesis is organized in 7 chapters that can be grouped in three parts, in addition to the conclusion chapter.

Part I. Architectures and design approaches. The first part of the thesis covers the description of different approaches to the design and development of heterogeneous multi-robot systems.

- **Chapter 2** focuses on architectures for heterogeneous multi-robot systems and the benefits of integrating edge and cloud computing into distributed robotic systems. We introduce and architecture leveraging the concept of *elasticity* from the cloud computing domain, and explore an example of computational offloading showing the potential benefits of edge computing architectures.

Part II. System security, trust and secure coordination. The second part of the thesis introduces the distributed ledger technologies and their potential for edge computing architectures and multi-robot collaboration. This chapter also presents a novel approach to securing single and multi-robot missions leveraging encrypted instruction graphs.

- **Chapter 3** delves into the potential role of distributed ledger technologies at the edge for multi-robot systems. We first describe the different consensus algorithms that are used in blockchain frameworks and more recent DLT solutions. We then propose a design approach to managing ad-hoc multi-robot collaboration and data sharing through a public blockchain. Finally, we discuss the different ways in which blockchain technologies can be deployed at the edge. We outline the main benefits and limitations of integrating distributed ledger technologies into distributed robotic systems.

- **Chapter 4** introduces a robotic mission encoding method that serves as an end-to-end validation framework for autonomous robots. In particular, we put our framework into practice with a proof of concept describing a novel map encoding method that allows robots to navigate an objective environment with almost-zero a priori knowledge of it, and to validate operational instructions. We also demonstrate the applicability of our framework through experiments with real robots for two different map encoding methods.

Part III. Localization, coordination and sensing. The third and last part of the thesis deals with both practical and theoretical aspects of multi-robot localization, coordination and collaborative sensing. We first introduce specific localization approaches (mainly UWB-based) for both individual robots and multi-robot systems. We then delve into spatial coordination or formation control algorithms requiring minimal to no communication (for which the UWB localization can be leveraged, but were formulated in earlier works). Finally, we show an example of collaborative sensing for scene reconstruction relying on UWB for relative localization.

- **Chapter 5** focuses on UWB-based localization for GNSS-denied environments. We first study the performance of state-of-the-art off-the-self systems for aerial robots. We then delve into the potential for UWB-based relative localization in heterogeneous ground and aerial multi-robot systems in areas where GNSS signals are degraded. We also introduce different sensor fusion approaches to UWB-VIO relative localization.

- **Chapter 6** introduces to new approaches to formation control. The first approach is index-free and communication-free, allowing for anonymous robots that are able to *see* other robots in their surroundings to converge towards a desired formation shape in a decentralized manner. However, this approach is not generalizable to any shape or configuration. The second approach presented requires one-way, local-only and minimal communication (signaling role self-allocation) and sensing of near robots. A progressive assignment of positions in the desired configuration is proposed to ensure solvability, while a leader-follower control approach is used to converge to the final configuration.

- **Chapter 7** combines approaches from the formation control domain (graph rigidity theory) into a collaborative localization system that monitors the rigidity of the localization graph to ensure uniqueneess of a relative positioning solution.

The thesis is then closed with a last chapter discussing the different methods and results, together with directions for current and future work.

- **Chapter 8** concludes the work with a discussion of benefits and limitations of the proposed approaches, as well as future research directions. We review the main contributions of the thesis to the state-of-the-art. We describe open issues and outline the advances in terms of designing and developing more robust multi-robot systems.

2 Multi-robot systems in the edge-cloud continuum

With robotic systems being increasingly connected, there is a growing intersection between the IoT and Robotics domains, in what is often called the Internet of Robotic Things (IoRT). This chapter covers the introduction of design approaches that leverage connectivity for decision-making and computational offloading. Through the chapter, we look at the intersection of robotic systems with concepts and elements from the IoT, IIoT, and multi-agent systems (MAS) domains. Throughout this chapter, we use the term *swarm* to refer to large-scale multi-robot systems.

Portions of text and a subset of figures in this chapter are reproduced from our previous works [7; 8; 11]

2.1 Reconfigurable swarm architecture

In this section, we present an architecture for reconfigurable multi-robot systems leveraging concepts and technologies from the edge and cloud computing domain. Parts and components of such an architecture are then presented across the rest of this thesis. However, the focus in this section is to provide a high-level vision of future robot swarms capable of self-reconfiguration (in terms of self-healing, role allocation or coordination, for example). At the same time, such swarms will potentially be able to exploit their heterogeneity in term of operational, computational and sensorial capabilities.

2.1.1 Swarm robotics and multi-agent systems in the IoT

A recent trend in cyber-physical systems and the Internet of Things (IoT) domain is to shift towards more distributed computation, a trend that has crystallized through the edge computing paradigm [138]. Similarly, recent advances in containerization, elastic computing, and dynamic resource management are materializing a decentralized cloud [139; 140]. Multiple researchers have explored the possibilities of integrating Multi-Agent Systems (MAS) theory within the IoT and cloud computing towards IoT MAS and Cloud MAS [141]. On the other side, the combination of MAS and robotics has brought swarms of robots with the potential for enhancing human responses in safety-critical applications such as firefighting [142], or post-disaster

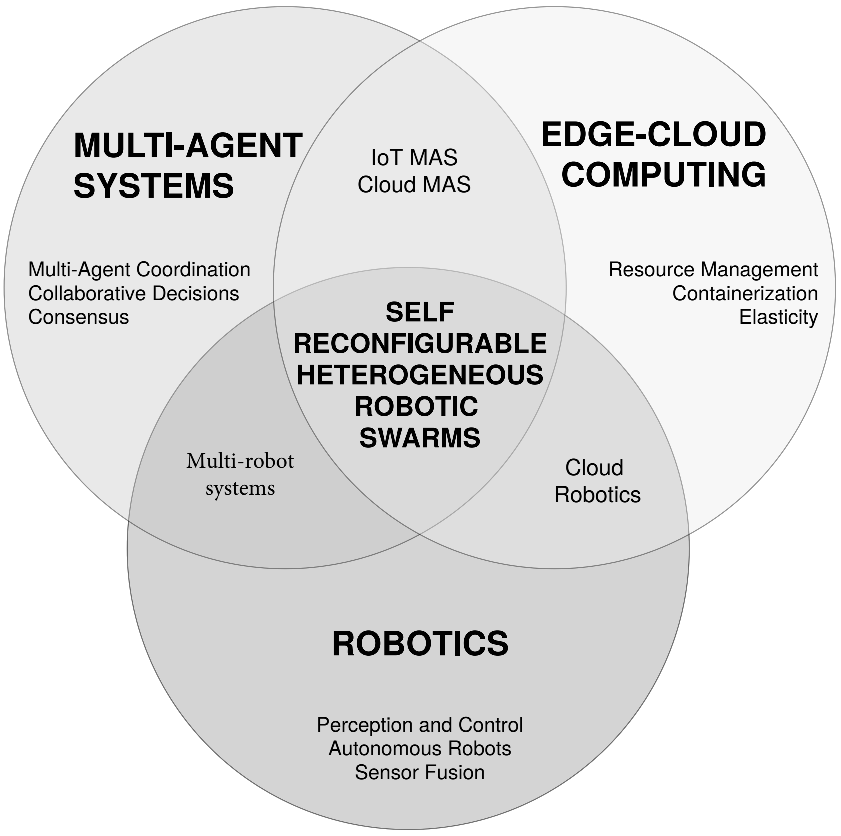


Figure 5. Intelligent and self-reconfigurable robot swarms can be designed at the intersection of the multi-agent systems (MAS), robotics and Edge-Cloud computing (also referred to as edge-cloud computing continuum) domains. This figure only represents a concept within the chapter context and does not necessarily generalize to other works.

scenarios [143], among others.

We extend these two approaches (MRS and IoT MAS) towards the intersection of edge computing and robotics. We then argue that with appropriate management and distribution of computing, sensing and communication resources within a robotic swarm, higher degrees of intelligence and operational flexibility and robustness can be achieved. This concept is illustrated in Figure 5. The capabilities of robotic swarms are currently limited by different factors, from the lack of methods and distributed collaborative sensing algorithms [144], to static and inflexible resource management with non-optimal utilization of hardware resources due to separate hardware and software design. This includes embedded hardware with relatively constrained computational resources due to payload constraints and uniform resources in swarms [145; 146; 147]. Some previous works have addressed these limitations through computational offloading and the definition of edge-cloud robotics architectures [148]. More recently, heterogeneous robotic swarms have been proposed to extend the flexibility and intelligence, opening the door to a wider array of more complex application possibilities [144]. Nonetheless, multiple challenges remain in terms of managing the collaboration within heterogeneous swarms [149].

Multiple research efforts within the fields of multi-robot systems and cloud computing have been directed towards distributed task allocation and distributed load management. For instance, autonomous mobile programs (AMPs) were an early introduction of a dynamic computational load management framework [150]. AMPs provided a distributed approach where autonomous agents were able to make decisions on a shared computational load, being aware of their own computational capabilities. AMPs share similarities with early load balancing techniques based and colony optimization for cloud computing [151]. More recently, multi-agent load balancing for resource allocation in a distributed computing environment has been proposed [152]. From the point of view of task allocation in multi-robot systems, K-means clustering and auction based mechanisms were introduced in [153]. In terms of spatial allocation, a workspace partitioning method was presented in [154] for indoor environments. In our work, we aim at combining these two approaches considering full reconfigurability through tight integration of methods from the edge computing domain and algorithms for cooperation in multi-robot systems. This is, to the best of our knowledge, the first study presenting such an approach.

In [155], our co-authors define the concept of resource ensembles from the perspective of the edge computing domain. This concept serves as the basis for abstracting edge resources and building dynamic management models on top of them. From the point of view of collaborative swarms of robots, we have presented a blockchain-based approach in [149]. In this chapter, we propose a blockchain as a medium for achieving consensus for bandwidth allocation and data quality ranking in a distributed multi-robot system. This, again, serves as the starting point towards distributed sensing and data processing in swarms of robots, where sensing, network and computational resources are abstracted and managed through collaborative decision making.

The techniques we propose have a clear impact, in particular, on swarms of drones. Current solutions for drone swarms require operators to either manually control drones or perform analysis of streamed data at a ground control center. This is a limitation for the deployment of drones in large areas where there have been natural disasters such as fires, or where people have gone missing, as the human resources necessary are too large. Even if the data is processed by a computer at the ground station, the need for a high-bandwidth channel between drones and the base station still limits significantly their operational capabilities. Therefore, there is an evident need for more intelligent drones that are able to perform data analysis independently and autonomously navigate large areas. Reconfigurable drone swarms empower complex edge data analysis at the swarm level through distributed edge computing. At the same time, this allows for long-term autonomous operation when energy constraints allow, as well as reconnaissance in remote areas with poor network connectivity.

The main objective of this section is to introduce a *new design approach* that enables efficient and dynamic resource management and autonomous reconfiguration of heterogeneous robotic swarms. We discuss the optimization of the various computing resources and sensing capabilities of robots in the swarm through a hardware (HW)/software (SW) co-design approach for the development of specialized robots. In particular, we incorporate elastic principles for coordinating and engineering collective capabilities of multiple heterogeneous robots. This elasticity takes into account the coordination at the level of multi-agent systems, but also the specific resources and algorithms utilized for robotic perception and navigation, among others.

Furthermore, we discuss how our proposed approach enables the reconfiguration of drone swarms and realize a distributed collective intelligence. This requires embedding intelligence and information processing in the drones themselves. With current technology, deploying multiple drones requires coordination among their operators, binding valuable resources from the actual mission. Even more, when multiple drones are being deployed in parallel [156; 157]. However, neither isolated intelligent drones nor simple task list orchestration is sufficient [158; 159]. Thus, it is essential to establish a collective intelligence that enables autonomous coordination and collaboration among the drones.

2.1.2 Models for reconfigurable swarms

We address the aforementioned challenges through an end-to-end design: from the robot hardware and local decision making on the computational and sensing resources to providing the swarm and its capabilities as a service for end-users. Two key aspects in our work are:

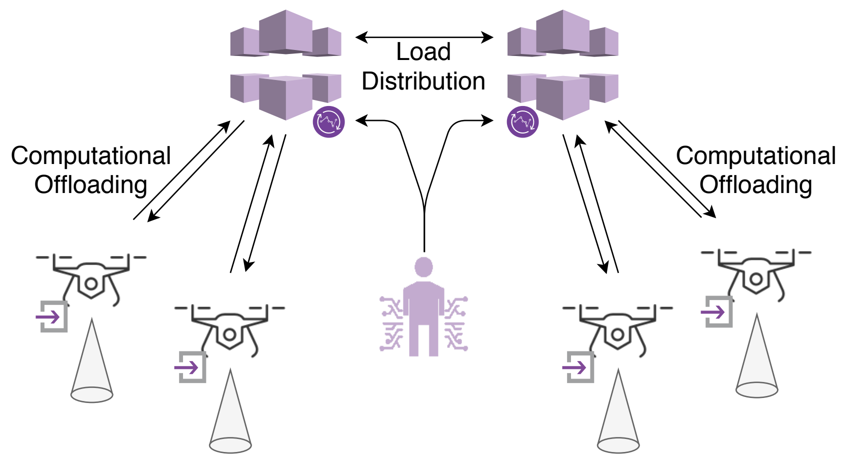
- Reconfigurable hardware resources for flexible and resource-rich computation platform at the swarm level
- Distributed data processing at the edge for collective swarm intelligence.

Reconfigurability and distributed intelligence provides more computational resources for multi-modal sensor fusion and distributed computation. Our model leverages methodologies and engineering techniques for distributed and dynamic management of elastic resources for swarms improving quality of results. The models are:

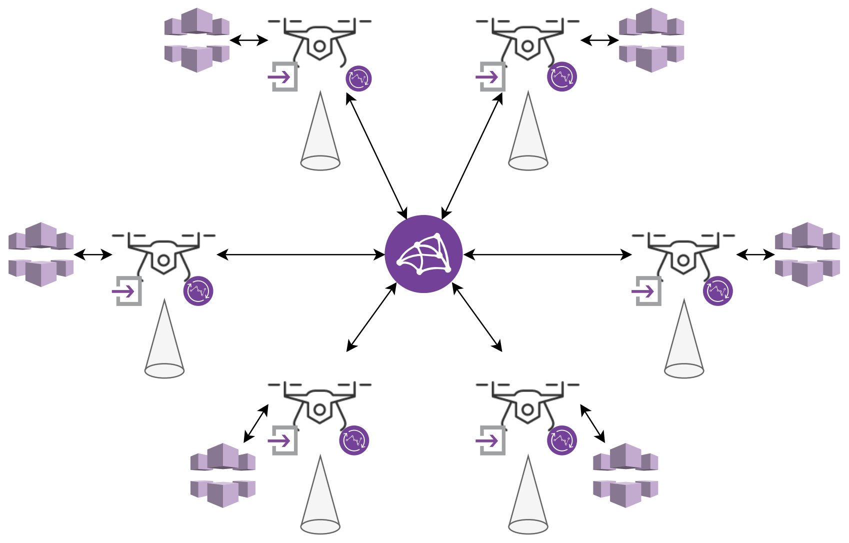
1) Swarm-as-a-Service Model: A swarm provides services to end-users with a control interface or API through what we call a Swarm as a Service (SwaaS). SwaaS offers an edge service model for swarm applications. In this view, each robot with its specialization is an edge services provider (or just an edge provider). Edge provider's sensing, computational and external communication resources are considered edge resources. Edge resources (hardware and software) are co-designed to make robots richer in terms of the resources that they can provide. With this approach, we bring various concepts of edge computing and services models to the field of multi-robot systems.

2) Application-Specific Resource Ensembles Model: Application Specific Resource Ensembles (ASREs) [155] define a specific organized set of edge resources (ASRE template or pattern) in swarms forming the edge infrastructure. An essential part of ASREs is the coordination and monitoring of resources together with swarm control and coordination. Resource management techniques enable service discovery, service end-to-end communication segmentation, and distributed task computation under unreliable and uncertain environments by incorporating uncertainty and elasticity [160]. For our knowledge, these have not been applied before to drone swarms. For example, [161; 162] are dedicated for containers and virtual machines. The proposed design for reconfigurable robotic swarms, therefore, includes the definition of templates and patterns for ASREs which can be used to tailor resource ensembles for application-specific needs.

In our end-to-end vision, the central point is to provide dynamic and flexible swarms as an *elastic heterogeneous multi-robot system*. In the system individual robots have different sensing and computational capabilities, a mesh network takes care of intra-swarm communication, and distributed algorithms enable the swarm to perform collective decision making as if it were a single unit. By *elastic*, we mean that the different resources of robots are abstracted and can be reconfigured depending on the application needs [163]. An illustration of this concept appears in Figure 7, compared to current cloud robotic and swarm robotic systems in Figure 6. In Figure 6a, each drone is independently connected to a cloud server, where it offloads part of its data processing. Any external control in this case goes through the cloud application, but direct control of drones could be enabled as well (for example, if the movement of the drone is controlled via a radio controller, and then mapping or other algorithms are run in the cloud). In Figure 6b, the drones form together a swarm. Each individual drone in the swarm has the same role initially, and both sensing and data processing occur individually at each drone. Algorithms describing the collaboration between drones could then run at the swarm level, but each drone would be still a separate entity. Finally, our approach is illustrated in Figure 7a, where all drones form a swarm as well. The key difference is that the swarm and its applications and resources are abstracted from individual drones and defined in a distributed manner at the swarm level. The communication with a controller or cloud services occurs from the swarm as a whole, and not from individual drones as separate entities. Moreover, sensing and computational resources, and the



(a) A cloud robotics system.



(b) A swarm robotics system.

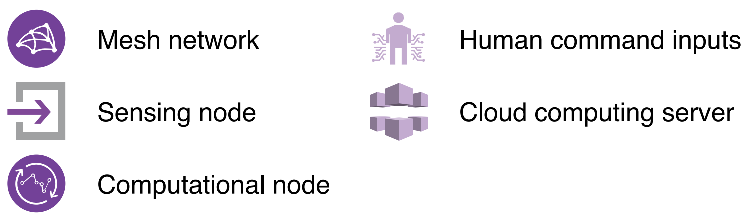
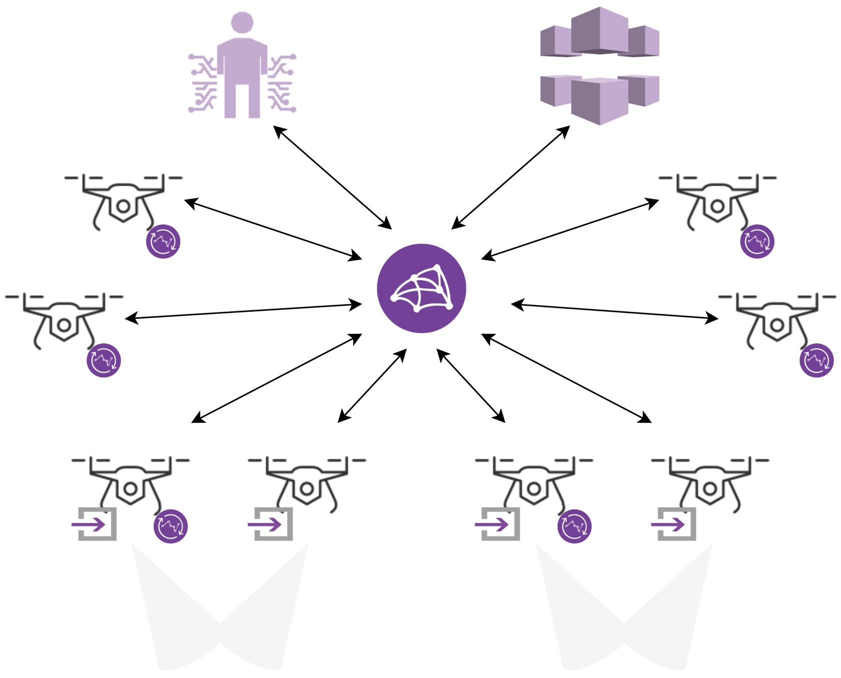


Figure 6. Sample typical architectures for cloud robotic and swarm robotic systems, illustrating sensing and computing roles together with interaction modalities.



(a) A reconfigurable robotic swarm.

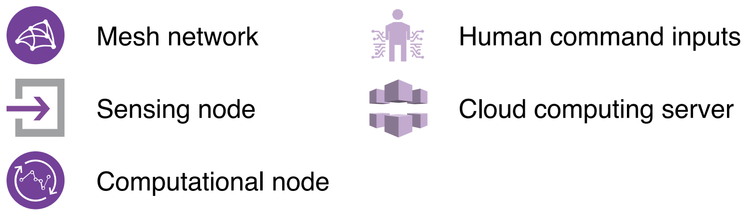


Figure 7. Illustration of proposed connectivity modalities for a swarm of drones, illustrating sensing and computing roles together with interaction modalities. In a reconfigurable swarm, roles may also be allocated beyond operational requirements considering also computational and sensorial capabilities. For example, part of the robots in the swarm might not be actively contributing to building a collaborative situational awareness model but instead process data generated by sensors onboard other robots.)

corresponding roles, are assigned dynamically among the swarm members. In the example illustrated in Figure 7a, half of the drones take a sensing role as their main role while data processing is offloaded to other drones assigned as computing nodes.

2.1.3 Architectural layers

We design an architecture with three layers and building blocks as illustrated in Figure 8. The layers are:

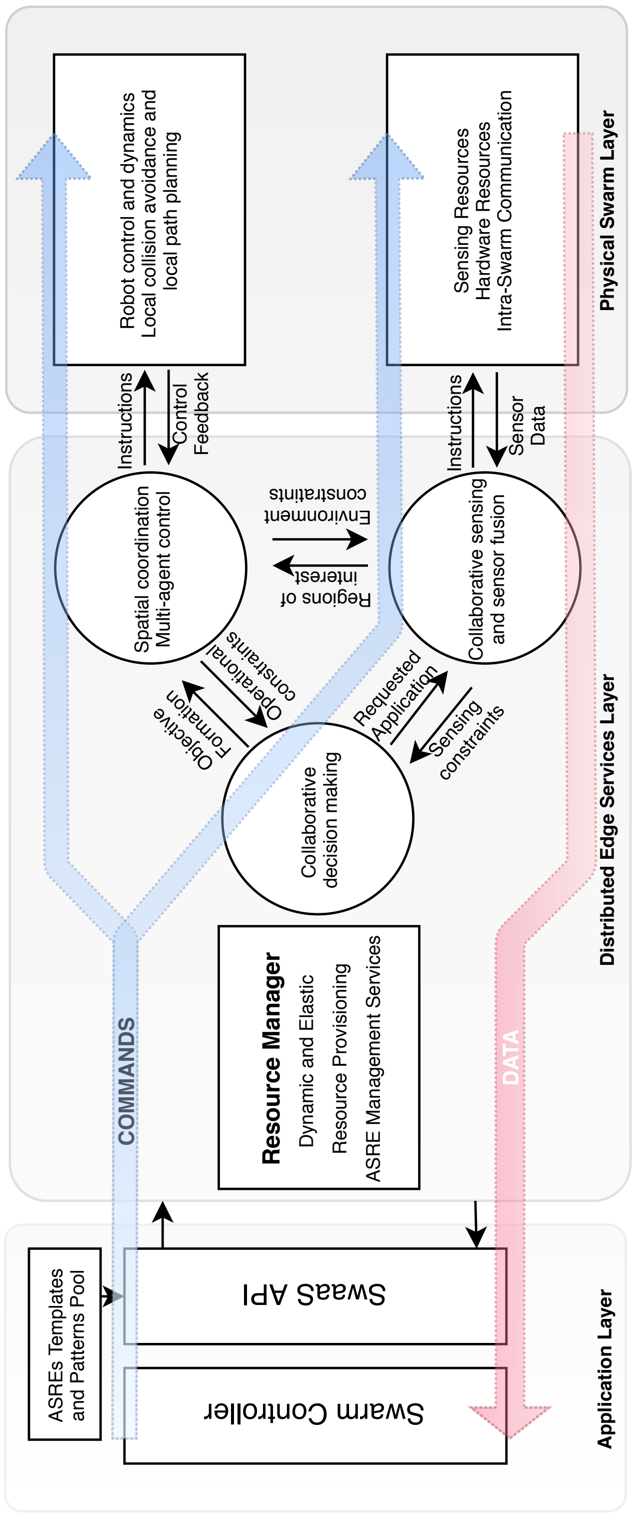

Application Layer
ASREs Templates and Patterns Pool
Swarm Controller
SwaaS API
Distributed Edge Services Layer
COMMANDS
DATA
Resource Manager
Dynamic and Elastic Resource Provisioning
ASRE Management Services
Collaborative decision making
Objective Formation
Operational constraints
Requested Application
Sensing constraints
Spatial coordination Multi-agent control
Collaborative sensing and sensor fusion
Regions of interest
Environment constraints
Instructions
Control Feedback
Instructions
Sensor Data
Physical Swarm Layer
Robot control and dynamics Local collision avoidance and local path planning
Sensing Resources Hardware Resources Intra-Swarm Communication

Figure 8. Proposed Architecture and Building Blocks for SwaaS and ASREs

Physical Swarm Layer. The actual control of robots is carried out at the physical layer, where the different hardware and mesh communication solutions are defined. The definition of a set of computing resources, sensors and actuators needs to be carried out as part of the swarm design in order to enable higher degrees of optimization when the sensing and computational resources are shared at the swarm level and applications run in a distributed manner.

Edge Services Layer. This is the main focus of the swarm design in terms of self-reconfigurability. This layer includes a Resource Manager (RM) for configuring and managing resources provided by the Physical Swarm Layer. Based on application requirements, the RM will assemble resources from edge providers into ASREs for SwaaS. The RM supports the automatic creation of ASREs by requesting, provisioning and orchestrating suitable resources.

This layer represents all the distributed services and processes running within the swarm and executed at each individual robot. These processes are classified in three main types: (1) spatial coordination (e.g., distributed formation control [5; 164]), (2) collaborative sensing (e.g. cooperative mapping [13]), and (3) collaborative decision making (e.g., role allocation [10]). These three apparently different parts of swarm control and decision making have a high synergy and their optimal operation depends on feedback from each other. These three topics have mostly been studied separately in the previous works [90; 165; 166]. Therefore, we have focused on the design and development of techniques for efficient communication between these processes. In summary, the key novelty is that ASREs and the RM implicitly manage collaboration within the swarm.

At runtime, ASREs will form an elastic and resilient edge mesh of services across robots in a swarm. The RM will dynamically provision new resources from different providers elastically. This kind of elasticity is carried out in an end-to-end and bi-directional manner: the resources are provisioned dynamically when new services are required or when the available resources change. The RM learns and optimizes the provisioning based on the reliability of resources, performance variations, bottlenecks, and failures. The RM provisioning is hidden from the application.

Application Layer. The application layer provides an interface for controlling and interacting with the swarm. We refer to this interface as the Swaas API. An external party or swarm controller can select from a pool of ASRE templates, which define the different patterns in which the swarm can be configured for different applications. By choosing an ASRE template through the SwaaS API, the swarm controller is implicitly selecting a set of resources and services. These resources are then provisioned and managed within the swarm itself through the RM. The services are provided based on the available distributed algorithms for sensing and coordination.

2.1.4 Reconfiguration processes in a drone swarm

We now discuss the specific technologies that enable the realization of the self-reconfigurable robotic swarm architecture.

Reconfigurability-enabling technologies

Currently, most small mobile robots and aerial drones rely mainly on CPUs in order to perform all the navigation and mission related computation, and microcontrollers for low-level control such as flight controllers [167]. We are leveraging existing technologies from other domains to enable reconfiguration within a drone swarm.

As a computing platform, we utilize FPGAs with embedded processors to extend the existing algorithms with custom hardware accelerators. FPGAs are reconfigurable hardware accelerators that can be exploited in computationally intensive and highly parallelizable tasks for autonomous robots, with higher performance/size and performance/power ratio as we have shown in previous works [168; 169]. Both the size and power consumption of hardware are essential aspects to take into account in drones. The use of FPGAs enables the RM to not only provision the existing resources but also dynamically provision new hardware accelerators on-demand. Some drones are equipped with FPGAs while others have embedded processors with GPU such as the NVIDIA Jetson TX2.

At the software level, we utilize containers to enable dynamic resource management and task execution. Container technologies are known but they have not been exploited for drones. Our goal is to use containers to enable dynamic resources management and task execution. All algorithms, from spatial coordination to collaborative sensing, are containerized and run in a distributed way. With efficient container orchestration, we are able to add flexibility and reconfigurability to the swarm. We bring specific techniques for computational load distribution, elasticity and resource management from the edge-cloud domain to the robotics domain.

In order to interface sensors, actuators, communication and the containerized algorithms, we utilize the Robot Operating System (ROS 2) which runs as a container application as well. ROS is the de-facto standard for robotic development in both academia and industry. ROS 2 focuses on distributed multi-robot systems and real-time computing, and will allow us to exploit container technologies for drones.

For network interfacing, there are no general solutions that integrate ROS 2 and mesh networking at the network level. Our experiments will utilize more traditional solutions at first, with all drones connected to a single Wi-Fi access point. Nonetheless, we will work towards the integration of ROS 2 and a Bluetooth 5 mesh network. Another recent technology that can provide significant advantages is ultra-wideband (UWB) [170]. UWB enables accurate localization in multi-robot systems, including drones [3], and has the potential for simultaneous communication and localization.

We will work on extending our current works on UWB-based mobile localization systems [19], studying the integration of UWB as a network interface between ROS 2 nodes.

Finally, we leverage blockchain-powered consensus algorithms suitable for multi-robot systems. Recent works [171; 149; 137] have proposed design concepts for integrating next-generation low-latency and scalable blockchains within heterogeneous multi-robot systems. Blockchains can be utilized as a distributed framework to achieve consensus in a multi-robot system, and also validate identities. This can be then utilized by ASRE management services, which could, in turn, be implemented as distributed Smart Contracts for resource coordination. The idea of utilizing a blockchain-based framework for managing edge resources has already been explored in our previous works [8].

Some of these planned implementations are described in the rest of the thesis, while others remain for future work.

Edge computing algorithms for sensing

We classify the algorithms in the edge layer in three main types: spatial coordination, collaborative sensing, and decision making. In previous works, these are typically defined with strong dependencies. For instance, depending on the sensing variable robots might be required to be in a specific spatial formation [4]. However, in our architecture these algorithms are designed independently and abstracted as edge services. This modular architecture brings multiple advantages. For instance, spatial coordination algorithms take feedback from the sensing algorithms regarding the location of regions of interest that should be analyzed more closely. This feedback is used to rearrange the drones in the proper shape and location. At the same time, the role of each drone within the spatial pattern is given by the collaborative decision making process. Analogously, the spatial coordination algorithms give feedback to other processes about the movement constrains of the swarm surroundings.

Resource management

The RM provisions and manages all resources, from hardware to edge services. Sensing resources are abstracted through ROS nodes (drivers) that produce data in standard formats. Each edge service is broken down into sub-services (for example, independent parts of an algorithm) and each of these is abstracted as a ROS node that consumes and produces different types of data (always in standard formats). All these ROS nodes are containerized and provisioned by the RM across the available the computing resources. The computing resources are modeled based on the amount and type of containers that they can run, and the performance for each containers. The provisioning is an optimization process that takes into account communication latency between data producers and consumers, and execution latency. The RM itself runs as a distributed and containerized application across the swarm, and manages the resources with elastic techniques.

Each application that utilizes the SwaaS API must define a Quality of Results (QoRs) requirement. We bring this concept from elastic computing models in which a QoR is defined in terms of performance, quality of data, type of output, and other measurable information [172], which the swarm provides to the controller through the SwaaS API. The QoR is essential for elastic resource management to dynamically provision the different resources taking the QoR requirement as an optimization constraint. We will extend the work by Mariani *et al.* on coordination-aware elasticity for developing primitives and algorithms to control the elasticity of swarms [173].

2.1.5 Architecture summary

We have proposed an architectural definition for reconfigurability in heterogeneous robotic swarms. This architecture is based on a combination of concepts and techniques from the robotics domain, multi-agent systems domain and edge-cloud computing domain. This is, to the best of our knowledge, the first work that proposes the abstraction and management of both hardware (sensors, actuators, computation and communication) and software (distributed sensing, coordination and decision making) as edge resources with elastic techniques. In particular, we explain how we are designing a reconfigurable drone swarm and what are the different hardware and software that make reconfigurability and elasticity possible.

2.2 Use case: offloading visual odometry to the edge

As we have discussed in the first chapter of this thesis, a key aspect in an autonomous robotic system is localization. In GNSS-denied environments, visual-inertial odometry (VIO) is an increasingly adopted choice of localization source owing to the ubiquity and availability of vision sensors. However, most existing VIO solutions require non-trivial computing resources to run in real-time. In building towards more distributed and cloud-enabled solutions, we investigate as an example how the accuracy of a reference VIO algorithm is affected by decreases in image quality. This is an important factor as streaming high-resolution images from multiple robots in a common network with low latency and at high framerates is not a trivial task.

In general, while visual-inertial odometry enables low-cost and accurate autonomous operation for small mobile robots, it still requires robots to have a minimum of computational resources available on their on-board computers. Most of the current research efforts are focused on algorithmic level optimization to achieve higher levels of accuracy and reliability in visual odometry on different hardware platforms. This has led to high-accuracy methods enabling long-term autonomy with efficient loop closure mechanisms [131]. However, small units such as micro-aerial robots (e.g., crazyflie drones) usually have constrained resources, including limited power and computational capabilities or reduced storage. In this situation, an aspect to consider is how to reduce the robots' computational burden while maintaining the VIO algorithm's high performance. If multiple cameras are utilized to reduce the blind angles for obstacle avoidance, path planning, and mapping, then the computational burden can increase considerably. This can have a significant impact on the performance and ability to autonomously navigate a complex environment in small mobile robots, including aerial drones. If additionally, multiple robots are operating in the same environment, accurate localization is essential to secure their operation and avoid collisions. In a multi-robot system where robots have equivalent sensing capabilities, the offloading part of the data processing can be a solution that not only increases the reliability of the system but also reduces the unit cost of each robot as the hardware can be simplified. In an industrial environment with large numbers of autonomous robots operating within a controlled area, reducing the cost of each robot can have a direct impact on the industrial ecosystem as a whole.

In recent years, some researchers have introduced the cloud robotics concept, in which the capabilities of small mobile robots can be enhanced by moving part or most of the computationally intensive data analysis tasks to a cloud environment [174; 175]. Nonetheless, streaming data to the cloud has the potential to significantly reduce the overall system reliability with uncontrolled latency or unstable network connection [168; 176]. We extend the recent trend in the IoT towards more decentralized network architectures with the fog and edge computing paradigms [177; 178; 179]. Edge computing crystallizes the idea of keeping the data processing as close as possible to where the data originates. With this approach, raw data is processed at the local network level instead of the cloud, decreasing the latency and optimizing the network load [180]. Furthermore, savings in hardware platforms and overall power consumption can be optimized with proper integration of edge computing [15]. In this work, we have moved the VIO computation towards a smart edge gateway to open the possibility for more intelligent, yet simple, large teams of autonomous robots that rely on edge services for offloading most of their computationally intensive operation.

The main motivation behind the work presented in this section is to study the optimal relationship between image quality and accuracy of a monocular visual odometry algorithm in a computational offloading scheme. Finding the proper trade-off between accuracy and image size has a direct impact on the computational resource consumption, algorithm runtime, network latency and, in consequence, the number of robots that can be supported simultaneously from a single smart edge gateway. Our goal is to provide a benchmark of the compression rate's influence on the VIO algorithm. To address these issues, we employ the state of art VIO algorithm VINS-Mono [131] and analyze its performance on an open dataset, the EuRoC MAV dataset [181], with varying image compression rate and picture quality. Our results show that the computational offloading scheme can be optimized in terms of bandwidth usage without compromising the accuracy of the visual odometry algorithm. Furthermore, decreasing the image quality reduces the processing time at the edge gateway. Therefore, finding the appropriate compression rate not only optimizes the network load but also enables a single gateway to handle the odometry for a larger number of connected robots.

The main contribution of [11] is on analyzing the performance of the state-of-the-art in monocular visual odometry with varying image quality and compression settings. We utilize the JPEG standard and examine the performance of a monocular visual odometry algorithm with the JPEG image compression setting varying from 1% to 100%. The implications of this study can be significant in a computational offloading scheme; an image size reduction of up to two orders of magnitude can be achieved without a significant compromise on odometry accuracy.

2.2.1 Cloud SLAM

The problem of SLAM has been traditionally considered either as an offline problem, where all accumulated data is utilized to rebuild the path, or an online problem for real-time image analysis with an on-board computer. However, if a large fleet of robots is considered, then a computational offloading scheme can considerably bring the cost down. To the best of our knowledge, computational offloading had been considered for mobile robot navigation a mapping only from the cloud computing point of view with cloud-centric architectures and data processing in powerful servers where the algorithms can be easily run in parallel at maximum efficiency. Yun *et al.* proposed a robotics platform to be deployed in cloud servers, RSE-PF, for distribution visual SLAM where data from different robots was aggregated and combined in the cloud [174]. An average network latency of approximately 150 ms was reported (round trip). Even with almost instantaneous data processing at the cloud servers, this either limits the image analysis rate to around 6 frames/second or induces a delay when parallel RX/TX channels are utilized. In the first case, an on-board computer such as a Raspberry Pi 4 or an NVIDIA TX2 could be able to provide a similar or better frame rate, while in the second case an accurate estimation of network latency must be available at the robot in order to interpret properly the processed information that the cloud servers return. The maximum number of robotic units that could be supported simultaneously was not reported; however, the authors utilized WebSockets in order to save bandwidth compared to HTTP. Dey *et al.* proposed a similar offloading scheme in which a multi-tier edge+cloud architecture was introduced [175]. Rather than concentrating on analyzing the performance, the authors shifted the research focus towards defining and solving an optimization problem in order to maximize the performance of the multi-tier architecture by offloading different processes to different layers. Their approach was to utilize integer linear programming for optimization of offloading design decisions utilizing the network bandwidth as a variable and adding latency constraints.

2.2.2 Monocular visual-inertial odometry

Visual-Inertial Odometry (VIO) is a common part of Visual SLAM (VSLAM), but is also used as a standalone state estimation method in robotic systems. VIO focuses on the local consistency of the robot movement trajectory, using real-time visual and inertial data to predict robot egomotion. The goal of SLAM is to achieve global consistency between the odometry and maps. VIO can therefore be considered as a building block for VSLAM, before tracking all the camera's historical data to detect loop closure and optimize the map.

Visual-inertial odometry algorithms combines camera and IMU data to implement SLAM or state estimation. The advantage of binocular VIO is that it can accurately estimate the motion trajectory and is able to recover the exact physical units. In Monocular VIO, it is only possible to obtain information regarding what the object has moved as a certain number of relative units in a given direction, while the binocular VIO is able to map these relative units to a metric system representing the real length or size. However, for objects that are far away, the binocular system degenerates into a monocular system. Monocular visual odometry has gained increasing attention in recent years because of the lower price and ease of automatic calibration. However, the data processing is more challenging.

VINS-Mono

VINS-Mono adopts a non-linear optimization-based sliding window estimator to predict a robot's position and orientation. This approach begins with the measurement preprocessing which will collect sensor data to detect feature and IMU preintegration. Through the initialization procedure, all values for bootstrapping the subsequent nonlinear optimization-based VIO will be calculated. The VIO with relocalization modules tightly fuse integrated IMU measurement processing, feature observation, and redetected features from a loop closure scheme. Finally, the pose graph module implements global optimization to reduce drift.

2.2.3 Experimental analysis

We have utilized an open-source dataset, the EuRoC dataset, in order to evaluate how the performance of the VINS-Mono algorithm varies when the image quality is reduced [181]. This is an initial approach and we have utilized the standard JPEG compression algorithm since it provides a high range of possible compression rates

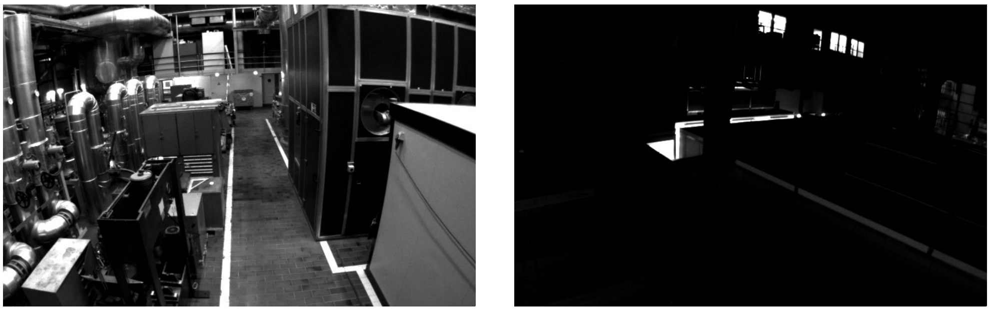

(a) Easier environment. (b) Harder environment.

Figure 9. EuRoC dataset samples with easier and harder environments for VIO algorithms.

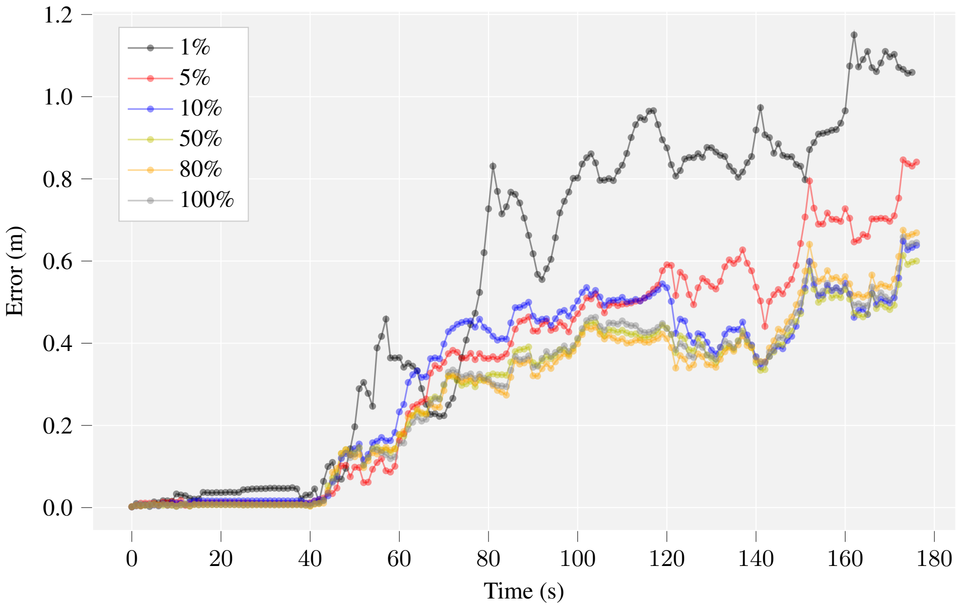


Figure 10. VINS-Mono error in the easier dataset.

through its image quality parameter. For instance, given a sample from the EuRoC dataset that has a size of 362 kB in PNG format, its size in JPEG ranges from 6.7 kB with 1% quality and 226 kB for 100% quality setting.

The EuRoC dataset is a binocular + IMU dataset for indoor micro aerial vehicles (MAV). It contains two scenes, one is a machine hall, and the other is a normal room. The dataset uses the flying robot AscTec Firefly as a data acquisition platform. It is equipped with binocular camera MT9V034 and an IMU ADIS16448. The camera frame rate frequency is 20 Hz, and the IMU frequency is 200 Hz. The authors utilize a Vicon motion capture system and Leica Nova MS50 as ground-truth for bench-

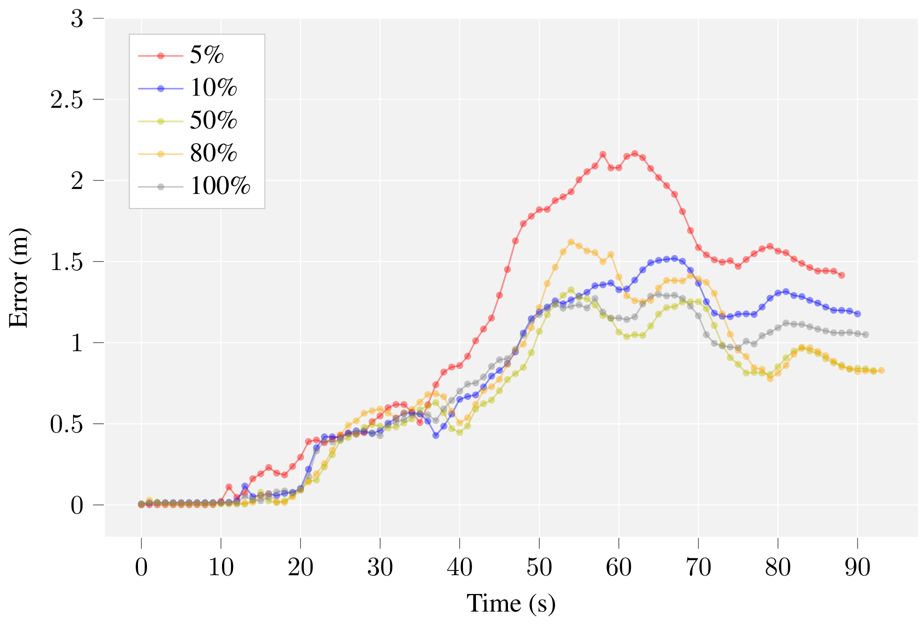


Figure 11. VINS-Mono error in the harder dataset. The error with 1% quality diverges within seconds of starting the sequence, and is therefore not included in the results.

Table 2. Average execution time of the different processes and network latency for a subset of image qualities.

	Image Quality				
	1%	**5%**	**10%**	**50%**	**100%**
Image size (kB)	5.7	7.9	11.2	28.3	202.2
Network latency (ms)	2.0	2.1	13.4	77.8	545.7
VIO computation (ms)	34.6	47.9	54.5	67.7	70.0

marking odometry algorithms. Due to the stable and reliable data provided, it has currently become a popular dataset [182; 183].

Our experiments have focused on the analysis of two parameters: the latency of the network and the accuracy of the odometry algorithm. We have also analyzed the processing time required for the feature extraction process and the pose estimation process for each of the image compression ratios. We have utilized two subsets of the EuRoC dataset which are considered easy and hard for visual inertial odometry algorithms, due to the extraction of less or more features. Samples from these two subsets are shown in Figure 9, where it can be seen that the image corresponding to the harder set is much darker and less features can be consequently detected. In fact, in this case, even if an image compression ratio of 5% has an impact of around 25%

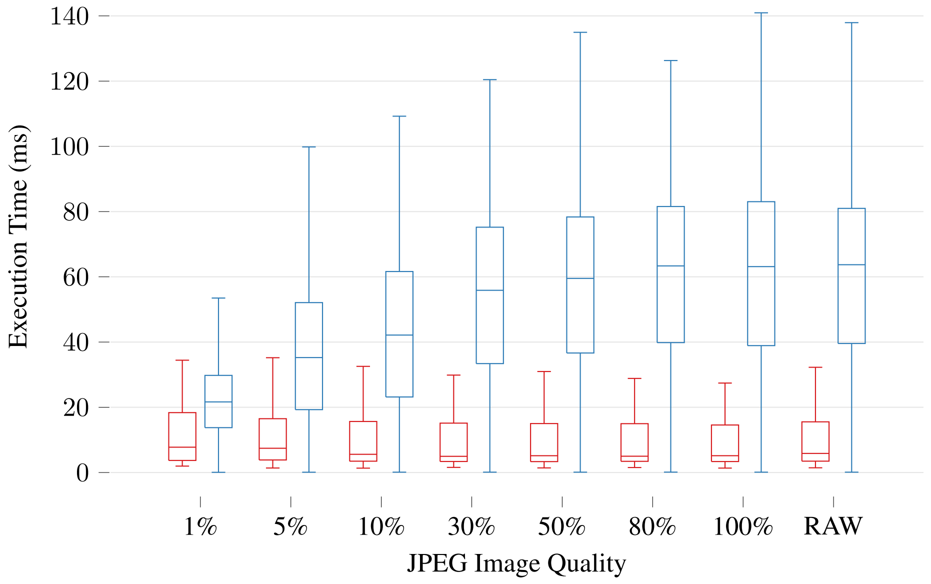


Figure 12. Execution times: feature extraction (red) and pose estimation (blue).

of the error at the end of the sample path (0.8 m error with 5% quality versus 0.65 m with 100% quality), and 1% quality renders a final error of around 1.1 m. In the harder dataset, however, only up to 5% image quality allows for a convergent path, as with 1% quality the algorithm is unable to calibrate the camera and IMU and the path diverges from the start. The errors accumulated with the VINS-Mono odometry algorithm over the easier and harder paths are shown in Figures 10 and 11, respectively. These indicate that the data quality can be reduced to as little as 10% without compromising the performance, while 50% quality gives the best performance in a harder environment. In the easier case, a 10% quality image matches the best performance with minimal odometry error while achieving two orders of magnitude of reduction in the network latency with respect to broadcasting a raw image.

The two main processes in which an odometry algorithm can be divided are feature extraction and pose estimation. The distribution of the execution times of these processes for a range of image qualities (1% to 100%) is shown in the boxplot in Figure 12, which have been obtained utilizing a 64-bit Intel Core i7-4710MQ CPU with 8 cores at 2.50 GHz. Each of the distributions has been calculated with 1000 images for which the different compression rates have been applied. While the feature extraction process has an execution time that remains constant with the increasing image quality, the pose estimation increases as more features are found in higher quality images. The network latency has an overhead effect that varies from under 1% (image qualities under 10%) to over 700% (100% image quality)

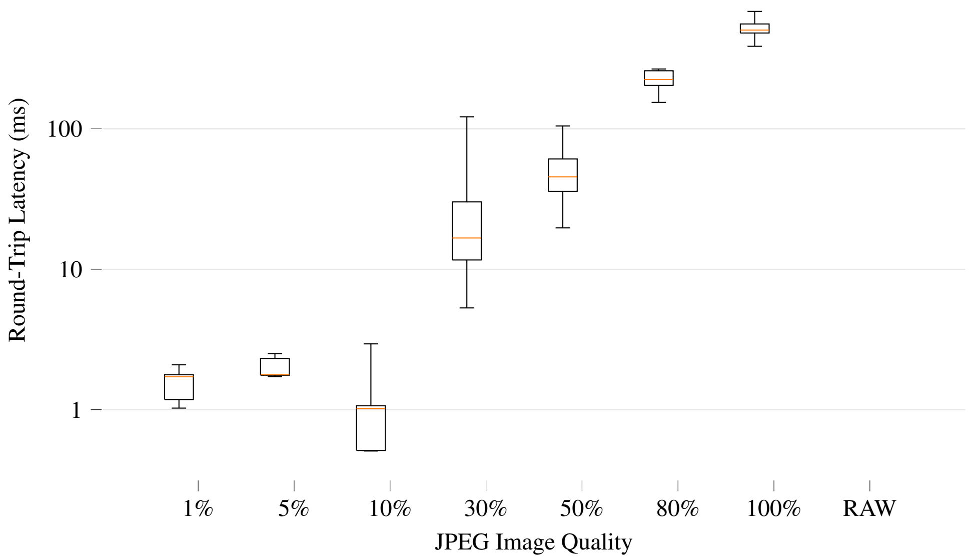


Figure 13. Average round trip latency with a UDP server.

when compared to the data processing time (feature extraction and pose estimation). The distribution of round-trip latency for a subset of image qualities is shown in the boxplot in Figure 13, where samples of 100 images have been utilized to calculate each of the distributions.

2.2.4 Remarks

We have evaluated the impact of image compression and quality in a visual inertial odometry algorithm. Our results show that image quality can be reduced up to a certain threshold, which depends on the ability of the algorithm to extract features from the environment, without a significant impact on odometry accuracy. This opens the door to the utilization of an efficient computational offloading scheme with edge computing. In turn, this enables the simplification of hardware onboard robots, a consequent reduction of power consumption and the ability to utilize a single edge gateway to offload the odometry computation from multiple robots. The latency of the network adds an overhead between 0.3% and 780% with respect to the processing time. In both datasets considered, a low accuracy loss could be achieved reducing the image quality to as much as 10%, where the network overhead is below 1%. In consequence, the offloading scheme does not induce significant delays to the odometry and has the potential to even improve the performance in terms of frame rate with more powerful edge gateways. The proposed edge computing offloading scheme can bring multiple benefits to a large multi-robot system, from cost reduction and energy

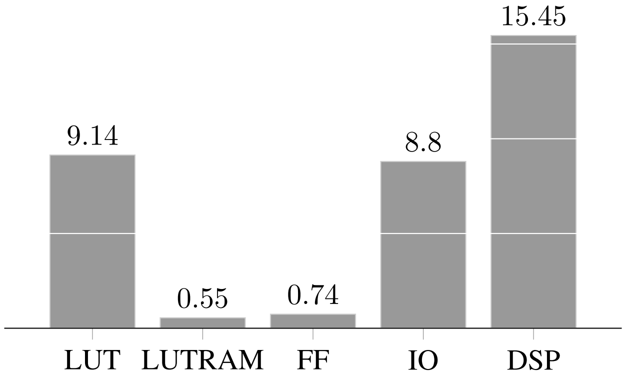


Figure 14. FPGA Resource Utilization Summary

efficiency to increased performance and reliability.

2.3 Use case: offloading lidar odometry with FPGAs

One of the key advantages of offloading computation is leveraging the same hardware for multiple robots, while simplifying the onboard hardware in small mobile robots. Lidar odometry algorithms use the lidar's data to compute the motion of a robot between two consecutive sweeps. In general, lidar odometry algorithms can be divided into three steps. The first step is to extract features from lidar data, the features can be the geometric distributions, or some stable points which can be observed in the two consecutive sweeps. The next step is to find the feature correspondence through the position difference between sweeps. The last step is estimating the lidar movement through the time between two sweeps [133]. FPGA's have the ability to process lidar data in real time with a limited resource utilization, and naturally parallelize the processing of data from multiple lidars [169].

Our goal in this use case study is to design a pure VHDL implementation for FPGAs. Instead of comparing complete lidar sweeps, as most recent implementations do, we aim at a implementation that analyzes lidar data as it is available and compares features in real-time. With this, we expect to be able to increase odometry accuracy and the positioning update frequency.

2.3.1 Initial implementation and analysis

In order to test the efficiency and usability of the feature-based odometry algorithm, we have first implemented in C++ within ROS. We use an RPLidar A1 for our experiment, a 360° two-dimensional lidar with 1° resolution at 10 Hz.

2.3.2 FPGA implementation

We have utilized a Zybo Z7-20 board for implementing our algorithm. This relatively small board is built around the Z-7010, the smallest chip in the Xilinx Zynq-7000 family. Even though the board integrates a dual-core ARM Cortex-A9 processor, we only use the FPGA logic in order to provide a generic design that can be easily ported to other platforms.

Implementing the proposed algorithm using VHDL hardware modelling presents some challenges. On one side, the need for calculation of trigonometric functions. In order to solve this, we utilize coordinate rotation digital computer (CORDIC) algorithms implemented in VHDL. In particular, we generate sine samples and from those calculate the values for cosine. On the other side, conversion types and integrating the CORDIC calculations into a state machine, requiring complimentary intermediate signals.

2.3.3 FPGA resource utilization

In order to study the potential of the FPGA-based implementation to be parallelized, we have synthesized an initial and unoptimized version of the design. The preliminary results show that the main resource utilization occurs with the IO banks (8.8%), Logic LUTs (9.14%) and DSP modules (15.45%). Nonetheless, the IO banks can be easily multiplexed, and additional lidars can be connected with a single input, so they do not represent a significant limitation. Moreover, a single nRF or Wi-Fi receiver can be utilized to receive information from multiple units. In terms of DSP utilization, the modules are used in the CORDIC implementations. A single CORDIC module can be shared among multiple parallel processes with a relatively simple state machine. Therefore, the proposed algorithm is not limited by the number of available DSPs. Similarly, we have estimated that around 90% of the LUTs can be shared. This is possible because the timing constraints that the frequency of the lidar scanner impose are very relaxed compared to the maximum performance that the VHDL implementation can deliver. Therefore, we expect that a single FPGA board will be able to run in parallel over 50 lidar odometry calculations. This can be combined with wireless communication solutions that provide enough available channels, such as Wi-Fi or nRF. In contrast, on an Intel Atom x5-Z8350 CPU @ 1.44 GHz × 4, the proposed algorithm can be executed approximately 5 to 15 times in parallel, depending on whether other processes are being run.

2.3.4 Remarks

We have presented preliminary work on an odometry offloading solution for multi-robot systems. We have designed and implemented a feature-based lidar odometry

Table 3. Performance Comparison Between FPGA and CPU Implementations

	Xilinx Zynq XC7Z010 (VHDL Impl.)	**Intel Atom x5-Z8350** (C++ Impl.)
Aprox. Resource	10% (fixed) + 1%	4% (fixed) + 7% CPU
Max. concurrency	>20-50*	< 5-15

*Expected range with optimized code.

Table 4. FPGA Resource Utilization Breakdown

Resource Type	**Used**	**Available**
Slice LUTs	4961	53200
LUT as Logic	4865	53200
LUT as Memory	96	17400
Slice Registers	791	106400
F7 Muxes	32	26600
F8 Muxes	0	13300
Bonded IOB	11	125
DSPs	34	220

algorithm that is flexible and can accommodate to a variety of indoors or outdoors environments, and implemented in an FPGA with pure VHDL modeling. The results presented in this section show potential for high parallelism and low-latency odometry at the edge with relatively small FPGAs.

2.4 Summary and conclusions

Throughout this chapter, we have introduced a high-level architectural proposal for designing large-scale distributed robotic systems able to leverage the edge-cloud continuum. We also looked at the potential of integrating blockchain technologies, which are discussed more in depth in the next chapter. At this point, these are just proposals and the two use cases presented later in the chapter only implement part of such architectures. The use cases have focused on analyzing the potential for leveraging shared computing resources at the edge in multi-robot systems. In the first use case, the focus is on how networking constraints might limit offloading computation from visual sensors, where lossy compression plays an important role. In the second use case, we have focused instead on studying the potential of specialized hardware, with FPGA-based accelerators for a lidar odometry algorithm. In summary, this chapter presents a position on the design of distributed robotic systems and an initial analysis of the potential implications for different state estimation algorithms.

3 DLTs for distributed robotic systems

Collaborative multi-robot systems are inherently distributed networked and mobile systems. With the vast majority of the coordination and planning algorithms in the literature requiring consensus among the different robots, DLTs are but a natural choice when designing decentralized, trustable and secure systems. During this chapter, we delve into the potential of DLTs for multi-robot systems. The focus is not on implementing or integrating specific algorithms, but instead on exploring the possible application scenarios and use cases from a conceptual and architectural point of view. This chapter discusses how different types of DLTs can serve as the underlying platform for collaborative decision making in multi-robot systems. Portions of text and a subset of figures in this chapter are reproduced from our previous works [8; 12]. In general, this chapter presents a position on the potential of blockchain and other DLTs for distributed and multi-robot systems.

3.1 Consensus mechanisms in blockchains

In this section, we introduce background concepts and theory. We focus on the concepts of consensus and, in particular, Proof of Work (PoW) and Proof of Stake (PoS), smart contracts, and scalability through sharding.

3.1.1 Consensus

Consensus mechanisms in a distributed system or decentralized network are those algorithms that allow agents to reach an agreement with respect to certain values, transactions or parameters whenever it is needed. Consensus mechanisms allow nodes in the network to trust others. The four most popular consensus mechanisms, according to [184], are Proof of Work (PoW), Proof of Stake (PoS), Practical Byzantine Fault Tolerance (PBFT) and Delegated Proof of Stake (DPoS), with other significant approaches including Proof of Authority (PoA), Proof of Elapsed Time (PoET) or Proof of Bandwidth (PoB). Both Bitcoin and Ethereum, the two most popular blockchains, utilize PoW, while Ethereum is shifting to a PoS-based consensus mechanism in Ethereum 2.0, also with implementations of PoA consensus available. Figure 15 shows typical consensus algorithms, DLT frameworks that use them, and a selection of their potential applications.

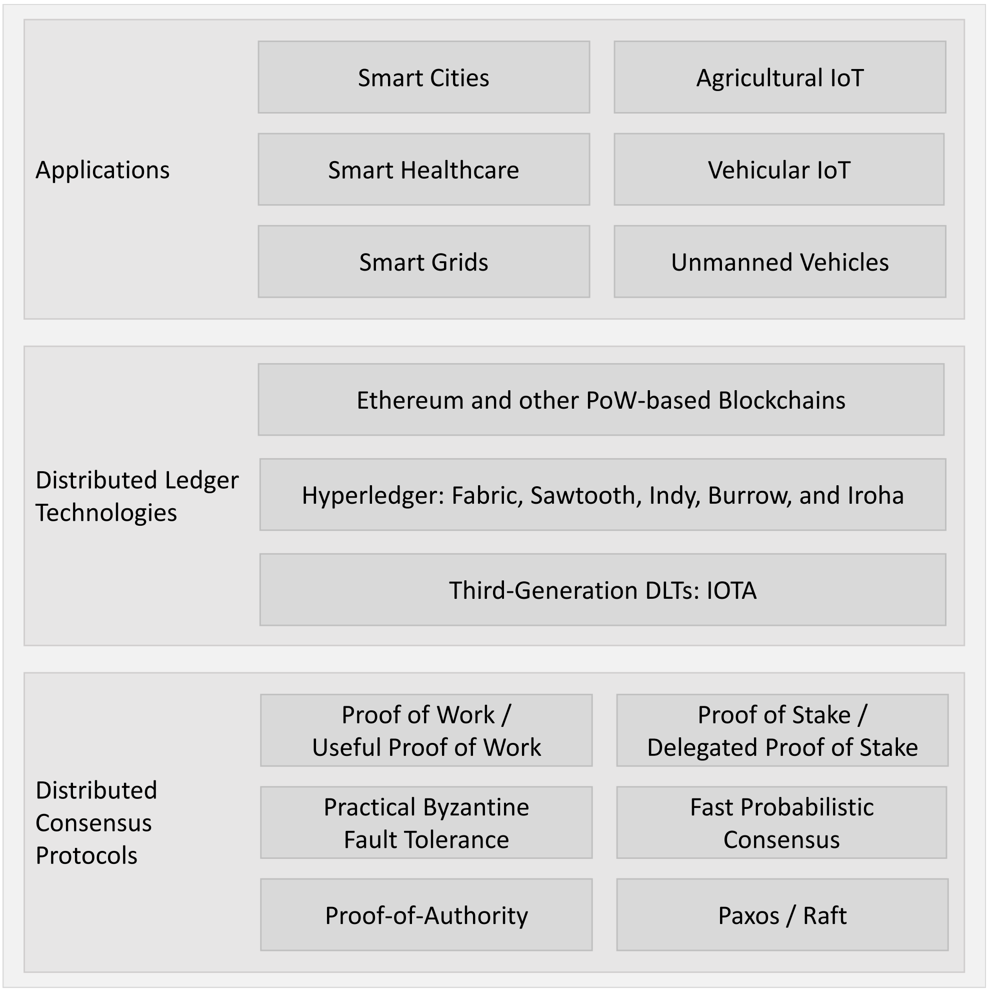

Applications
Smart Cities
Agricultural IoT
Smart Healthcare
Vehicular IoT
Smart Grids
Unmanned Vehicles
Distributed Ledger Technologies
Ethereum and other PoW-based Blockchains
Hyperledger: Fabric, Sawtooth, Indy, Burrow, and Iroha
Third-Generation DLTs: IOTA
Distributed Consensus Protocols
Proof of Work / Useful Proof of Work
Proof of Stake / Delegated Proof of Stake
Practical Byzantine Fault Tolerance
Fast Probabilistic Consensus
Proof-of-Authority
Paxos / Raft

Figure 15. Blockchain/DLT Consensus protocols, systems, and applications in integration with the Internet of Things.

Proof of work

The consensus mechanism introduced by [185] as part of the Bitcoin was Proof of Work (PoW), in what is the first known implementation of this mechanism as a way of achieving consensus in a distributed system. Firstly introduced by [186] as a filter to minimize the spamming capability of malicious email senders, the original idea of a PoW system has been invariant: to require a node in a network to solve a moderately computational intensive cryptographic problem in order to be able to make a transaction in the network, or validate its identity. In other words, a PoW is a cryptographic puzzle that is difficult to solve, while still being possible within a certain time and given certain hardware resources, and that it is very easy to validate.

Therefore, it takes a node a large amount of computing power to solve a PoW puzzle, but it takes other nodes in the network a small amount of computational resources to validate the solution.

In the Bitcoin and many successive blockchain systems, including Ethe-reum [187], a PoW-based consensus is introduced in order to validate new blocks in the blockchain. A block can be roughly described as a single entry in the distributed ledger containing information about a set of transactions. Each of the transactions in the block is confirmed when the block is validated by the network. In order to validate or *mine* the data in a block, or body, a header is generated by hashing the information in the block body. The PoW consists on adding an extra cell to the block body, a random number denominated nonce, such that the block header meets some predefined conditions. These conditions are set on a block header's target hash as a function of the hashes of previous blocks. Once a node finds a solution to the PoW puzzle, i.e., a nonce with which the target hash is obtained, then it broadcasts the block to the rest of the network. Other nodes can easily validate the proposed solution, and then start mining a new block. The miner that solves the PoW for a given block obtains a reward in the form of newly created, or mined, cryptocurrency. When a block is mined by solving the corresponding PoW puzzle, it is added to the blockchain and the transactions that it contains are considered validated.

A security concern arises when two nodes find a nonce at the same time or when a second node, which has not received the proof yet, starts broadcasting the solution as well. In Bitcoin, nodes accept the previous block for which they receive a proof. In the case of two near-simultaneous proofs, the blockchain separates in two branches, or forks. [185] introduced a rule in which the fork becoming longer or accumulating mining difficulty would be judged as the authentic one by the network. This is a practical solution as it is highly improbable that two consecutive blocks will be solved simultaneously by two pairs of nodes. In any case, even if two or more blocks are solved at the same time, at some points one of the forks will become longer. This defines the so-called 51% or double spending attack, as malicious nodes would need to to control at least 51% of the network's computing power in order to be able to introduce a faulty transaction in a block, validate it, and keep validating consecutive nodes in the corresponding fork so that it is accepted as the canonical fork by the network. When the size of the network and the number of miners increases, the probability of such attack is reduced, thus giving the blockchain its immutability and data integrity properties. At this point, the incentive introduced as a reward for solving a PoW puzzle is a key aspect of the blockchain in order to increase the number of miner nodes. The double spending attack is also one of the main vulnerabilities of blockchains based on a PoW consensus mechanism.

The benefit of having an expensive PoW solution in terms of hardware, energy consumption and time is that it is equally expensive for malicious nodes to attack the network. Part of the security of PoW thus comes from disincentivizing attackers because of the large a priori investment required in order to be able to attack and gain control of the network, which would not pay off even if the attack is successful [188]. However, this also means that miner nodes need to spend large amounts of resources, including electricity, in order to successfully solve the PoW problem and mine a block. This makes the blockchain growth unsustainable as all nodes in the blockchain actively try to solve the PoW puzzles while only one of them obtains the reward. In addition, it also creates inequality across the network and higher entry cost; i.e., a new node joining the network needs a large initial investment in hardware in order to be able to solve the PoW efficiently and compete for rewards, and nodes with the best hardware accumulate most of the rewards.

The estimation of computing resources relying on Proof of Work has been studied earlier by [189] and others [190; 191]. In his work, Eyal describes how not only full PoW solutions but also partial solutions can be utilized to ensure members in a mining pool do contribute to the collective mining effort. A mining pool is an association of nodes in a blockchain that utilizes PoW consensus in order to increase the probability of solving the PoW problem first, and therefore obtaining the corresponding reward. Nonetheless, if rewards are shared equally across the pool, then a malicious node might join the pool but never share its PoW solutions. The solution proposed by [189] is to ask all nodes in the pool to share partial proofs, which can be validated by the rest of the pool members, and utilized to quantify the effort or computational power that nodes are dedicating, in average, to the mining task. A key aspect to take into account, however, is the distribution of complexity of the partial proofs of work as described by [189]. This helps avoiding that an a malicious node always submits a partial proof of work with some minimum complexity but never tries to calculate the full proof. In a similar direction but with an opposite approach, [192] proposed a mechanism for the definition of non-outsourceable puzzles which would discourage miners from joining mining pools or creating mining coalitions due to the inability of the pool members to estimate the partial progress of each individual node.

Proof of useful work

Part of the research community has argued that taking into account the humongous amount of computational resources and electric energy put into mining to solve PoW puzzles, at least these could be defined in a way that the solutions found would help research in other fields. As an example, [193] proposed the definition of PoW puzzles that would find long chains of primes. Solving these PoW would be then dedicated to solve a mathematical problem which consists on finding the distribution of the Cunningham prime chain. In this case, the Fermat Primality Test would be used to validate the PoW solutions.

A different research approach is the definition of simpler PoW requiring less computational resources in order to reduce the entry barrier and provide a more uniform distribution of mined currency. [194] introduced the concept of Cuckoo hashing, in which the PoW difficulty would remain constant over time.

Proof of stake

While PoW introduces a secure consensus mechanism in distributed networks, the increasing amount of computational resources not only limits the options of new nodes in the network to obtain mined currency, but also limits the maximum amount of transactions that can be processed as it takes an average time of 10 minutes to validate a block and all the transactions it includes [195]. In order to reduce the transaction validation latency, a Proof of Stake (PoS) mechanism was first implemented within the Nxtcoin [196; 197]. A PoS mechanism chooses the block validator based on the stake of different nodes, assigning a probability of being validator that is directly proportional to the amount of coins that a miner owns. A similar approach was introduced by [198], where the pure stake a miner owns is directly related to the probability of the miner to mine a new block, while the exact chance is calculated also based on the state of the current block.

A direct consequence of a PoS-based consensus is that nodes accumulating large amounts of cryptocurrency have a higher change of mining new currency, additionally increasing their stake. Moreover, the network becomes more vulnerable to attacks from these nodes or coalitions of nodes with large stakes. In Ethereum 2.0, a PoS consensus mechanism will be introduced within the so-called Casper protocol [199]. Nonetheless, rather than considering the stake a miner owns, miners are required to put part of their coins at stake and locking them in a virtual safe during the validation process. Keeping the coins in a wallet without putting them at stake is not considered sufficient to be elected a validator in this case. In consequence, miners are incentivized to act in a honest manner as they risk losing all the coins they put at stake if faulty transactions are detected in their validated blocks. The mechanism through which a node loses the coins at stake because of faulty transactions are detected is called slashing.

A clear benefit of PoS over a PoW mechanism is the much lower computational complexity of the operations involved, thus having a much smaller footprint in terms of energy consumption and computational resources required. Multiple authors, such as [200] or [201], have studied the sustainability of Bitcoin's growth and its energy footprint, which researchers estimate to be the equivalent, on a yearly basis, to non-renewable energy resources consumed by entire nations of the size of Czech Republic or Jordan. Nevertheless, this also means that because miners do not need to dedicate large amounts of computational resources to mining, it is easier to perform Sybil attacks or spawn multiple identities within a single malicious node.

The 51% attack discussed in the PoW consensus mechanism is still a potential attack vector in a PoS system. However, while in the PoW case attackers need to obtain control over 51% of the network's computing power, which becomes increasingly easy as larger pools monopolizing the mining process are created, in a PoS system an attacker needs control over 51% of the cryptocurrency's total supply. This is, in theory, a more difficult problem than gathering enough computing power. Furthermore, the attack's incentive is additionally reduced due to the potential evolution of the cryptocurrency's market capitalization during and after the attack: while an attacker accumulates coins, the coin's value is likely to increase significantly, causing the attack to become increasingly expensive. In addition, if the coin's value drops after the attack due to a loss of trust in the network, the main adversely affected node would be the attacker itself as the value of the initial investment would significantly decrease.

In general terms, a PoS system relies on a validator or a set of validators which are eligible after depositing part of their stake. In other words, as described by [199], nodes earn the right to propose a block only after locking part of the coins they own coins on the blockchain. This is an extended definition over the pure PoS system firstly implemented by [202] as part of PPCoin, in which the total miner's stake is directly considered.

Practical byzantine fault tolerance

One of the most famous and successful blockchains to date is Hyperledger, a project initiated in 2016 within the Linux Foundation [203]. Hyperledger is a permissioned blockchain which has been successfully utilized in multiple industrial domains [204]. The objective of Hyperledger is the deployment of an open-source and cross-industry framework that can be utilized as a standard platform to run smart contracts within a decentralized ledger.

The consensus mechanism utilized in Hyperledger is a Practical Byzantine Fault Tolerance (PBFT) algorithm [205], first proposed by [206]. PBFT was the first algorithm with the ability to operate in large asynchronous networks such as the Internet, while providing over one order of magnitude in processing power improvement over previous methods, allowing for high-performance Byzantine state machine replication, and demonstrating thousands of requests per second. Byzantine fault tolerance can be described as the capacity of a system to maintain proper operation when multiple errors or unexpected behaviour occur within part of the system, but not its totality [207]. In a distributed network and considering the consensus problem, this is equivalent to the ability of the network to provide a robust consensus even in an scenario where a subset of nodes act maliciously, failing to forward valid data or sending invalid information.

In a PBFT system, nodes are distinguished between validating and not-validating peers [203]. The validating nodes run the consensus algorithm, in which they replicate a state machine and evaluate its result. A client makes a request that is transmitted over the peer-to-peer network through the non-validating nodes, which act as proxies between clients and validators. Non-validating nodes do not participate in the consensus mechanism, but are able to validate the results. The PBFT algorithm is able to provide consensus across the network when at most one third of the nodes behave arbitrarily or maliciously. Because the validator nodes need to arrive to the same results regarding the client request, the state machine that is replicated must be deterministic. Three types of transactions are valid operations: deploy, invoke and query. These involve accepting the code to be deployed (a smart contract), invoking it with a given series of accepted parameter and indicating the result, and querying a state from another node, respectively. The client awaits for a predefined minimum of replies with the same results, which depends on the maximum number of potentially faulty nodes.

In comparison with PoW and PoS systems, in PBFT individual transactions can be confirmed without the need to wait for a block including several transactions to be added to the blockchain. In terms of energy efficiency, PBFT requires less computational resources than a PoW consensus, but increases the probability of a Sybil attack, where a malicious nodes would create multiple instances pretending to be a large number of parties. In practice, PBFT is often combined with a PoW that must be solved in order to join the network and within certain time intervals to ensure that every node in the network is dedicating some minimum computational resources to the collective validation effort. An important benefit of PBFT over PoW and PoS is the low reward variance, as every node can be incentivized. This lowers the reward variance across miners. Nonetheless, the scalability of PBFT is an issue due to the large number of peer-to-peer communication exchanges required.

3.1.2 Smart contracts

The Ethereum blockchain [187] introduced one of the most notorious smart contract platforms based on blockchain, by providing a Turing complete language as part of its framework [208]. Ethereum introduced Solidity [209] as a language to implement smart contracts. In Solidity, a smart contract can be seen as a set of code instructions, or functions, and a set of initial, intermediate and final states (data). Both the data and the code resides at a specific address within the Ethereum blockchain. In general terms, a smart contract is simply a program that is run within the blockchain. In practice, it means that all nodes participating in the consensus algorithm run the same program and validate its output.

Smart contracts are part of the Ethereum Virtual Machine (EVM) [210]. The EVM is a completely isolated environment for executing smart contracts within Ethereum, with no access to other processes, network connectivity or files in the system. In addition, the way smart contracts can access data from other smart contracts is also limited. The EVM is based on the existence of contract accounts, which extend the functionality of external accounts, those controlled by a human or network node through a public-private key pair. Contract accounts operate in an automated way as a function of the code stored within the account. While external accounts are defined based on their key pair, with an address determined based on the public key being assigned to each node joining the network, contract accounts have addresses that are determined when the contract is created. In Ethereum, the address space is shared among both types of accounts.

Contract accounts are created through transactions that have a null or empty recipient. Those transactions must contain code that outputs the smart contract's code, which is then generated when the transaction's code is executed within the EVM. In general terms, transactions including a payload and Ether (Ethereum's cryptocurrency) between external accounts in Ethereum are extended so that when a transaction's target account is a contract account counting a set of code instructions, these are executed given the payload in the transaction. A key concept in Ethereum is gas. Upon creation, transactions are assigned a definite quantity of gas. The gas is a measure of the processing power that will be dedicated to that transaction. In other words, the gas is the transaction fee. The gas is initially charged into the transaction, and its reserve gradually decreases as a function of a set of predefined rules when the EVM executes the different transaction instructions. The gas that is left is refunded to the transaction creator. The gas price, which is paid upfront, is decided by the creator node. Miners, which obtain the gas price as a reward, decide which transactions to mine based on the amount of gas included. Therefore, the gas price is decided based on the market and the desired priority that miners will give to a specific transaction. In these terms, the Ether that a given node accumulates can be related to its ability to utilize the Ethereum network — the EVM — to perform computation based on instructions included in the transactions that the node initiates.

3.1.3 Sharding

One of the main disadvantages of the consensus algorithms presented above, specially PoW due the large computational resources required, is scalability [211]. While Bitcoin only requires one broadcast per block, PBFT is based on multicast messages and also suffers from scalability in terms of communication cost [212]. In all three cases, nonetheless, security does increase as the network becomes larger. [213] presented Elastico in order to overcome the scalability limitations of previous blockchains, with an approach in which the network is partitioned into a set of smaller subnetworks or committees, also called shards. The definition of such committees is referred to as sharding, and Elastico was the first implementation of a sharding protocol for permissionless blockchains that is able to tolerate a predefined fraction of byzantine nodes in the network. Another early proposal of sharding in blockchains was proposed by [214], in which Merklix trees, or Merkle Patricia trees [215; 216] are utilized to merge the state of different shards into a global state. These trees are, in general terms, an extension of Merkle trees for unordered sets that enable addition and removal of elements or branches in the tree with a logarithmic complexity and without the need of full tree recalculation.

Perhaps the biggest effort that is currently being put into the development of a truly decentralized, permissionless and scalable yet secure blockchain is the design and development of Ethereum 2.0 [217], where huge amounts of computing resources will be no longer required for mining [218]. In oder to do this, the Ethereum Foundation and other developers behind Ethereum 2.0 have embraced Proof of Stake as the main consensus mechanism, while still utilizing PoW to secure the network, and the concept of sharding towards scalability. The consensus is based on the Casper protocol [219], which incentives for mining have been described by [199].

While the impact that shards have on transaction scalability is relatively clear, with a much larger throughout being possible in terms of transactions validated per second, it is not straightforward to extend the implementation of smart contracts with sharding. As smart contracts have associated a series of data states corresponding to their code, each state change can be though of as a transaction. There are two basic approaches in order to deploy smart contracts in a network with shards: either contracts are executed within a single shard, or a cross-shard synchronization mechanism exists that allows for data to flow between shards. This also affects to transactions involving tokens that are assigned to different shards, as cross-shard communication is needed.

3.1.4 Scalability

The main goals of Ethereum 2.0 can be summarized in five items: decentralization (to allow single-shard or system-level validation with consumer off-the-shelf hardware), resilience (to maintain operational conditions through network partitions and even if a significant fraction of the network goes offline), security (to deploy advanced crytographic strategies that enable a large-scale participation and validation), simplicity (to keep the consensus layer and top-level definitions as simple as possible), and longevity (to utilize either quantum secure components and mechanisms, or design the system in a way that it can be easily updated when possible for quantum secure equivalents), according to the Ethereum 2.0 specification [218; 220].

The roadmap to Ethereum 2.0 [221] includes only a basic sharding approach in its initial phase, with no support for the EVM. The key innovations in the first phase of the Ethereum 2.0 deployment will be the utilization of the beacon chain with a PoS-based consensus mechanism [222; 223]. In the beacon chain, the full blockchain is not stored at all nodes. Instead, each shard has its own full record of transactions, i.e., its own blockchain, and the beacon chain can be understood as a central blockchain that coordinates the shards and stores a chain of global states. In this architecture [224], there are different types of nodes: (i) super-full nodes storing the beacon chain and every block from each shard's blockchain, (ii) top-level nodes storing and processing the blocks in the beacon chain and the headers in the shards' blocks, (iii) single-shard nodes, which are top-level nodes that process and verify at least one full shard, and (iv) light nodes which only process the headers of the main chain blocks. A full Ethereum 2.0 deployment with cross-shard transactions, support for light nodes and flexible smart contracts being able to be executed within a dynamic number of shards is still in early development stages. Nonetheless, most of the concepts have been laid down and development work is in progress. One of the main challenges is on how to support a dynamic reallocation of validators to different shards, i.e., the shuffling of shards that each single-shard node is validating. In this process, there is a trade-off between reshuffling overhead, as a single-shard node in charge of a new shard needs to download the corresponding data, and the time than an attacker needs in order to infiltrate a single shard once its validators are known to the rest of the network.

Some of the main ideas of Ethereum's solution in order to enable cross-shard communication, and the distribution of validators or shards for single-shard nodes, are the following [224]. First, the introduction of receipts, which are objects that are not stored in the shard's state but are defined in a way that Merkle proofs of their existence can be generated. This is useful for the simplest case where a larger number of applications have each a reduced number of users and do not need to share data often, so that each application can be contained within a shard and utilize receipts to communicate with other applications. Second, to offer transparent sharding, i.e., to dynamically create, merge or divide shards without the need for an application to be aware. Therefore, the sharding process is transparent to developers and they do not need to take sharding into account when defining smart contracts for different applications. Third, a solution for asynchronous cross-shard communication where receipts could be generated in order to revert transactions if necessary. If the system is biased such that reverts propagate faster than cross-shard requests, then this can effectively solve the problem of asynchronous cross-shard communication. Fourth, an strategy to avoid that an attacker sends multiple cross-shard requests from within a single shard. A proposed approach is to require an application that makes a cross-shard call to pre-purchase an amount of gas at the receiving shard (where the pre-purchase transaction occurs), which would be set as congealed. The amount of gas that can be congealed in a single shard is predefined, thus setting a limit to the amount of calls that can be made from other shards. Congealing gas avoids issues with volatile gas prices. In addition, a demurrage rate is included, such that the congealed gas is lost at a preset rate if it is not used within a receipt. Finally, congealed gas has the potential to be used for reliable intra-shard scheduling, even if only for the short term.

A security analysis of scalable blockchains through sharding has been carried out by [225] as part of the definition of Omniledger. While still in a proof-of-concept phase, the authors claim that Omniledger is the first decentralized ledger that is able to scale to Visa-level transaction throughput securely. Omniledger focuses on token transactions only, even though the authors lay down the directions for future work in order to support cross-shard communication for smart contracts. While the main concepts behind Omniledger will most probably have an impact in future blockchain implementations with sharding, the Ethereum approach is more robust in terms of strategy and definitions, as well as from the point of view of a thorough security analysis.

3.2 Blockchain-based services at the MEC layer

5G and beyond connectivity has the potential for bringing together the telecommunications, robotics [226], artificial intelligence (AI) [227], Internet of Things (IoT) [228] and blockchain domains [229], all of which share a recent trend in which computation is shifting towards more distributed architectures [230; 149; 231]. This comes together with the concept of network slicing and edge computing, key pillars behind the low-latency and network load optimization in 5G and beyond networks [232]. Through multi-tenant slicing, new business opportunities are being created at the edge of the network [233]. This section aims at connecting the potential that edge computing brings to multi-robot systems (presented in the previous chapter), to the opportunities brought by blockchain and other distributed ledger technologies from the perspective of decentralized robotic systems. We explore the potential for combining the backbone of today's autonomous robotic navigation and localization, the Robot Operating System [234], with the latest development in 5G and slicing strategies at the MEC layer. The MEC layer is an inherently distributed computing platform that enables high-performance computing (HPC) services with minimal latency [235]. The most direct application is to extend existing offloading schemes [236], and integrate them within the 5G stack [237]. This has clear potential in vehicular and robotic navigation, especially when combined with predictive schemes [238]. In addition, we envision that FPGA-based hardware accelerators at the base stations will provide new levels of reconfigurability, energy efficiency, and processing power within the offloading orchestrators. Moreover, we take into account integration between distributed robotic systems [239], and distributed computation platforms defined within a blockchain [187]. We argue that permissioned blockchains backed by a large public and trusted infrastructure will be a key element of MEC-based services. These blockchains will be able to provide a transparent and secure channel for connected vehicles to interact with third parties. Slicing at the MEC layer in 5G and beyond can reduce the computational load in connected robots and vehicles. This will allow units with more constrained resources, such as

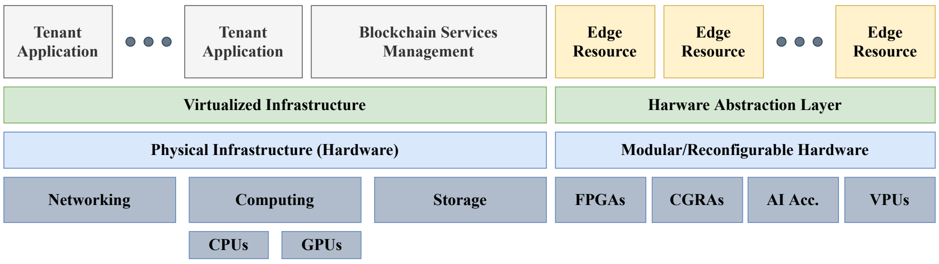

Tenant Application
Tenant Application
Blockchain Services Management
Edge Resource
Edge Resource
Edge Resource
Virtualized Infrastructure
Harware Abstraction Layer
Physical Infrastructure (Hardware)
Modular/Reconfigurable Hardware
Networking
Computing
Storage
FPGAs
CGRAs
AI Acc.
VPUs
CPUs
GPUs

Figure 16. 5G-MEC Computational Building Blocks

delivery drones, to enhance their situational awareness and increase their autonomy. In terms of safety and reliability in long-term autonomous operation in both self-driving vehicles and autonomous robots, challenges arise from the point of view of (1) localization accuracy [240], (2) situational awareness and level of understanding of the environment [241], and (3) limitations of computational capabilities in smaller robots or drones, with algorithms that might take longer to run depending on the complexity of the environment [242]. Slicing at the MEC layer has potential for providing services to support the operation of connected autonomous vehicles and robots by providing in respect to the above challenges (1) streaming services of high definition (HD) maps for accurate localization with online updates whenever the environment changes; (2) semantic information of the environment, as well as meta-data from other connected vehicles; and (3) an adaptive algorithm that autonomously provides in real-time map models and environment data according to the operational and computational capabilities of the vehicle requesting data.

We see the main opportunities as part of smart cities, where the blockchain can be supported by either RAN or 5G-connected public infrastructure, as well as industrial environments where there exists trust. The concept of Smart City has been mostly tied to the IoT since its inception [243]. Nonetheless, the IoT and the robotics domain have since been integrated as connected robots become the standard in industrial, civil and other application domains. The new edge and fog computing paradigms have only increased this synergy between the two domains [13]. Figure 16 illustrates a generic 5G-MEC architecture with a blockchain to manage services (tenant applications) and edge resources (reconfigurable and on-demand hardware). In a smart city, a public blockchain for data sharing could boost the deployment of autonomous robots from both private and public entities. Besides, the role of the infrastructure should be considered not only as a platform to manage the blockchain lifecycle but also as a static data source and validating platform, where traffic cameras and other sensors that already exist can be integrated.

3.2.1 Multi-Access edge computing and network slicing

The standardization of Multi-Access Edge Computing (MEC) has been promoted by the European Telecommunications Standards Institute (ETSI) [244], with the MEC Industry Specification Group (ISG) launched at the end of 2014. The ETSI MEC ISG aims at defining a multi-tenant distributed cloud platform to be located at the edge of the radio access network (RAN) [245]. Moving computation and data intensive tasks towards the edge of the network enables the low-latency and high-bandwidth requirements of 5G and beyond connectivity. Other fundamental technologies towards this end include containerization and virtualization, software defined networking (SDN), and network function virtualization [246].

One of the key pillars enabling multiple verticals within MEC, and opening the RAN edge to a wide variety of industries and users, is network slicing [247]. Network slicing consists of the co-existence of multiple logical software-defined networks (slices) on a common hardware infrastructure, i.e., a multi-tenant cloud infrastructure at the edge of the network, with each of the slices being optimized to meet the requirements of a particular application [248]. We are particularly interested in slicing for the automotive sector, where 5G will be the key in vehicle-to-everything communication [249].

3.2.2 Previous works

The literature shows that the integration of blockchain technology at the MEC layer has been proposed by multiple authors [250; 251; 252]. Nonetheless, these have been mostly focusing on the blockchain as a secure way of sharing or data or an immutable ledger to store transactions. However, one of the key applications of blockchains is their utilization as a robust decentralized computer that ensures the validity of execution of pieces of code called smart contracts [187]. We exploit these and the consensus protocols of blockchains to provide a framework for managing edge resources and services.

Xiong et al. proposed the utilization of edge services to offer resource-constrained devices opportunities to join a blockchain by mining at the edge. Then, the end-devices share the data with third-party applications through a pricing scheme, modeling the interactions within the IoT as market activities. While their focus is on utilizing blockchains as a cryptocurrency and auditable platform, we focus on the distributed computation that smart contracts enable instead.

Liu et al. presented a similar approach, where the MEC layer is used to offload mining operations [253]. Nonetheless, this was part of a wider offloading framework where the focus was on deciding which offloaded operation would be cached. A similar scheme can be integrated within the offloading slice proposed in this study.

Zhu et al.'s EdgeChain is the closest work to this work [254]. EdgeChain is a blockchain-based architecture that is utilized to place third-party applications across the MEC. We extend this idea for dynamic reconfiguration with smart contracts based on client-provider interactions, rather than considering the service providers only.

3.2.3 Managing MEC with permissioned blockchains

A consortium blockchain such as Hyperledger deployed across the MEC layer and connected public infrastructure, which are the nodes acting as validators, can be utilized to manage the interaction between connected clients and service providers, and at the same time orchestrating the hardware resources at the MEC layer. The proposed system architecture is illustrated in Fig. 17

We envision the existence of at least three separate network slices in order to support the aforementioned services. On one side, efficient offloading can be achieved with the on-demand reconfiguration of hardware accelerators, as well as AI accelerators. This slice focuses on low-latency in terms of fast data processing and optimization of computing resources to support as large number of connected devices as possible.

Some key benefits of this architecture are (i) identities are managed directly by the consortium blockchain and all transactions are signed and immutably recorded; (ii) the distribution of hardware resources or computing power is done through smart contracts and agreed across the MEC layer, with blockchain-enabled mobility; and (iii) a connected client, such as an autonomous robot or a self-driving car, requests a service from a third-party through the blockchain slice; if the smart contract approves the service, then the corresponding resources are configured and provisioned at the corresponding slice.

3.2.4 Distributed robotic systems

So far, the proposed architecture considers a distributed network architecture with distributed consensus algorithms. The last part of the piece is a distributed framework for deploying distributed robotic systems and algorithms to provide services to connected vehicles and robots. The Robot Operating System (ROS) has been the de-facto standard in production-ready robotic development for the past years [234]. However, wider adoption requires several challenges to be solved, including automated node discovery, real-time systems, non-ideal networks and distributed multi-robot deployments. These and other use cases are being developed within ROS 2 [239]. We believe that ROS 2 will be an essential piece in connected vehicles and robots by providing a common framework and standardization to the MEC-based services. ROS 2 can solve key challenges in flexible service definitions at the MEC layer. It will provide standardization of data formats, channels and deployment of algorithms, with a common underlying logic for all service providers as will be discussed in the

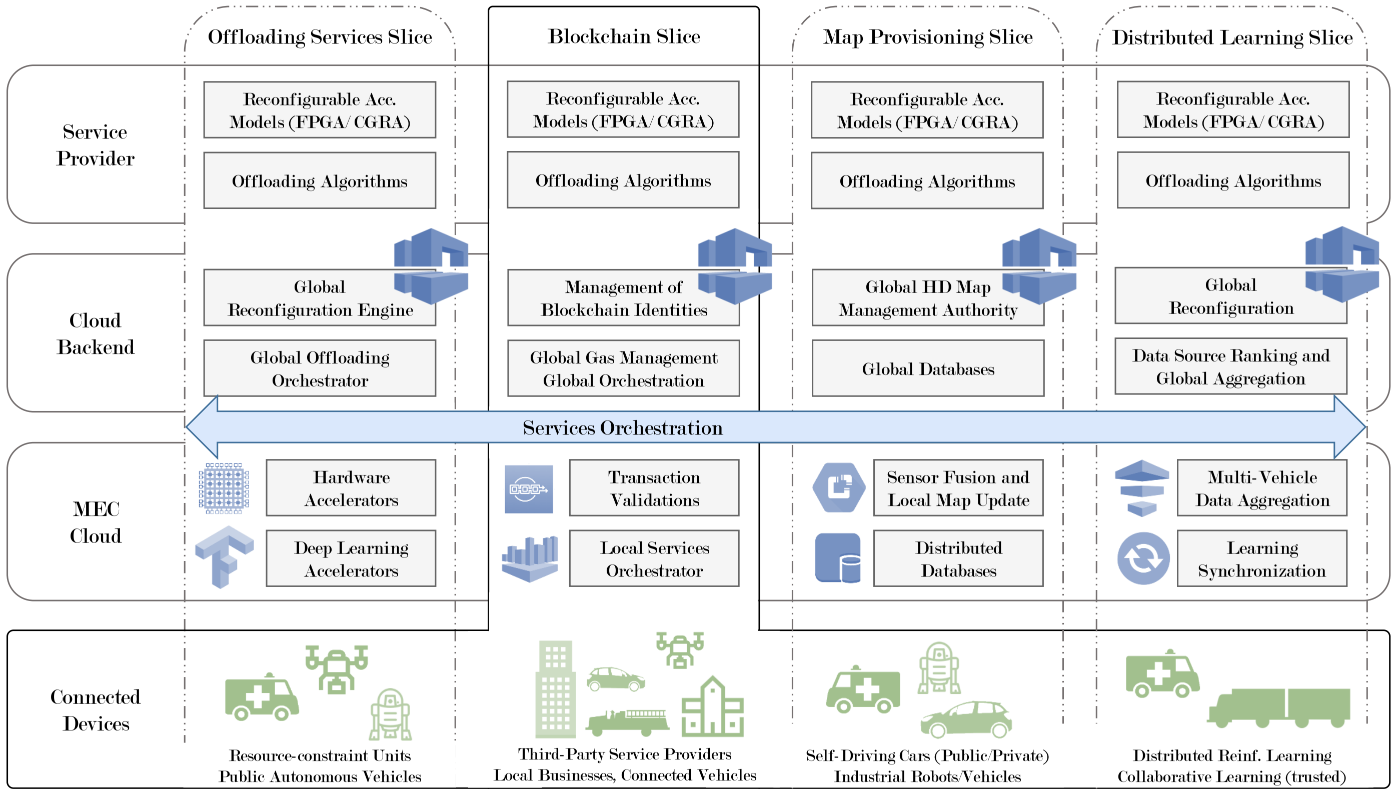

Offloading Services Slice
Blockchain Slice
Map Provisioning Slice
Distributed Learning Slice
Service Provider
Cloud Backend
MEC Cloud
Connected Devices
Reconfigurable Acc. Models (FPGA/ CGRA)
Offloading Algorithms
Global Reconfiguration Engine
Global Offloading Orchestrator
Management of Blockchain Identities
Global Gas Management Global Orchestration
Global HD Map Management Authority
Global Databases
Global Reconfiguration
Data Source Ranking and Global Aggregation
Services Orchestration
Hardware Accelerators
Deep Learning Accelerators
Transaction Validations
Local Services Orchestrator
Sensor Fusion and Local Map Update
Distributed Databases
Multi-Vehicle Data Aggregation
Learning Synchronization
Resource-constraint Units Public Autonomous Vehicles
Third-Party Service Providers Local Businesses, Connected Vehicles
Self-Driving Cars (Public/Private) Industrial Robots/Vehicles
Distributed Reinf. Learning Collaborative Learning (trusted)

Figure 17. Architecture for Blockchain-Based MEC Autonomy Services

next section.

3.3 MEC for autonomous robots in smart cities

The proposed architecture can be utilized as a general framework to provide services through MEC slicing. Nonetheless, we focus on how this architecture can be integrated to support the autonomous operation of autonomous robots and vehicles, opening the door to new applications and business opportunities in cities or areas which support such seamless integration of third-party services within the telecommunications network stack. We also outline the role of ROS2 and the blockchain in the different application scenarios. The key areas of the system are illustrated in Fig. 18. In the following, we propose different areas where this technologies have potential, with a purely hypothetical discussion.

3.3.1 Provision of HD maps in real time

Autonomous navigation in dense urban areas requires self-driving cars and more diverse autonomous robots to have the ability to localize themselves with very high-accuracy. This is enabled by high definition (HD) maps of the environment [255]. However, HD maps are expensive to generate, update and maintain [256]. In terms of generation and real-time updates, the proposed architecture can smoothly integrate within the streaming slice a data fusion module that gathers information from different sources to obtain these maps, as described below. Regarding the maintenance, the main disadvantage of HD maps is that they require large storage on-board the vehicles, and it is impractical to keep maps of large areas within vehicles themselves. An evident solution is to provide streaming services at the MEC layer. However, this requires tight mobility and latency control. We believe that this can be achieved with the combination of ROS2 as a standardization framework, 5G and beyond networks for low-latency and predictive mechanisms to provide in advance data about the areas that robots or vehicles will travel through. ROS messages serve as a standard for data formats, which can be then processed by multiple third parties without requiring extra communication to instruct on the data structure. Moreover, the nature of ROS topics enables consistent integration within streaming services to provide HD maps, with subtopics defining various parameters, e.g. location or point cloud density.

3.3.2 Online update of local HD maps

In Smart Cities, administrators can provide a framework to support the online update of HD maps and utilize existing infrastructure as a source of data. Traffic cameras and other sensors utilized for monitoring can be repurposed and their data forwarded to a data fusion scheme within a dedicated MEC slice. Moreover, connected infras-

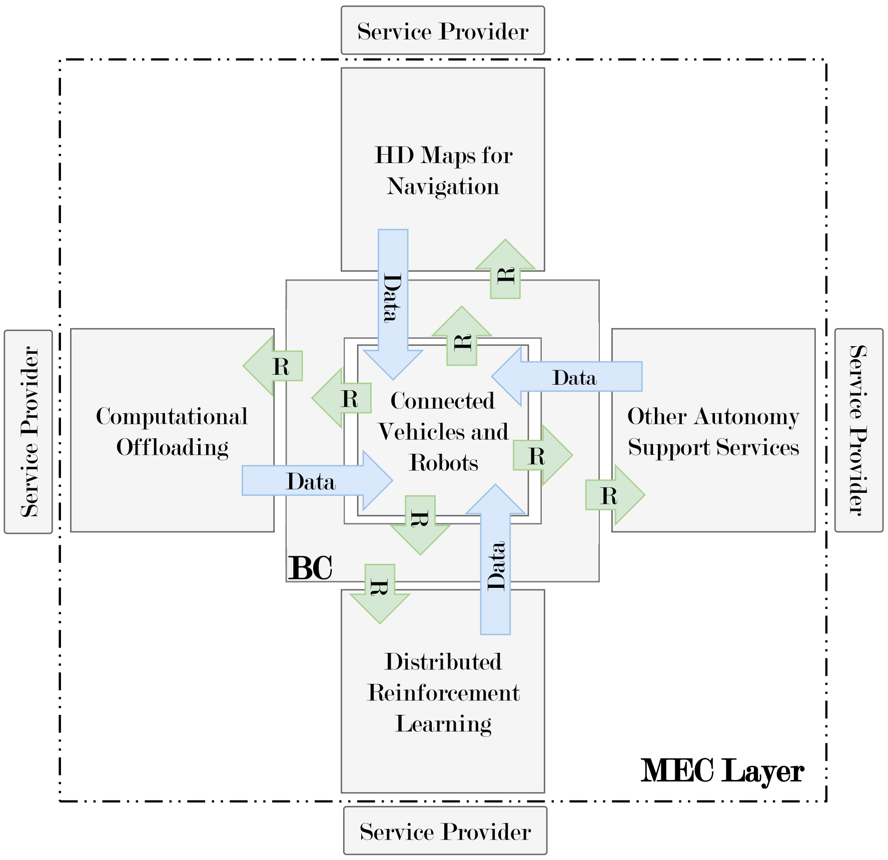


Figure 18. Autonomy Support Services at the Multi-Access Edge Computing (MEC) layer. Requests (small yellow arrows) go through the blockchain slice (BC), which is in charge of the local service orchestration. Data is streamed directly to end-users to reduce latency and increase throughput.

tructure with enough processing power can serve to increase the range or capacity of the streaming network. The role of ROS open source algorithms is essential to deploy the state-of-the-art in multi-source and multi-sensor data fusion in public infrastructure. Moreover, the smart contracts within the blockchain can be utilized to rank the available sources of data.

By providing an open framework, city administrations open the door to new local applications such as drone delivery or various types of autonomous robots surveying, monitoring and performing other tasks across the city. This has the potential to boost both the city's economy and technology innovation.

3.3.3 Distributed reinforcement learning

As the robotic field has evolved over the past two decades, deep learning has become an essential aspect in complex robotics systems [28]. In particular, reinforcement learning has allowed for traditional dynamics models to be replaced for neural networks that have been able to outperform any previous approaches.

With the first semi-autonomous cars roaming the roads of large cities around the world, humongous amounts of data are being collected to improve the performance of deep learning algorithms. This is a process that requires offline training of neural networks. However, various distributed reinforcement learning algorithms enable robots and autonomous vehicles to have online improvements of their models not only from the real-time data and experiences but also from those of cooperating vehicles.

Offering a distributed reinforcement learning service at the MEC layer would enable connected vehicles to take advantage of the data and experiences of other vehicles to learn faster and better, with more and different experiences being analyzed in shorter periods of time. Nonetheless, such as a service would require a tight control on identities and a mechanism to ensure that model updates are valid and do provide an improvement. A permissioned blockchain provides a transparent and secure identity management framework, while the robustness and vulnerabilities in distributed multi-agent reinforcement learning is still an open problem [257]. If raw data is provided to the learning service, then the model updates can be validated. If data is protected due to privacy concerns and only the model updates are shared, then it becomes considerably more challenging to determine the validity of a given update. A blockchain can provide part of the solution to this problem through its consensus mechanisms. They have been shown to outperform traditional consensus mechanisms in the presence of erroneous or malicious data in other scenarios within the robotics domain [171].

3.3.4 Offloading services

Reliable connectivity and existence of MEC services in a large area opens the doors to robots and vehicles relying on network-enhanced intelligence for their operation. Instead of developing and building complex robotic systems able of long-term self-supported autonomy, local organizations and businesses can build simpler products with similar capabilities, relying on computational offloading to achieve certain functionalities. Not only does this reduce the development and production cost, it also potentially decreases time-to-market, further boosting innovation.

We propose a separate slice for the offloading orchestrator and services because the focus is on optimizing computing power and reducing execution time when possible, compared to the storage and mobility requirements of the streaming slice. Even if the underlying hardware is the same, a different degree of reconfiguration is expected in order to optimize the offloading scheme.

GPU-based and FPGA-based accelerators have been widely used in sensor development and deep learning acceleration over the past few decades, being a perfect match for the requirements of edge computing [258]. More recently, autonomous navigation, localization and mapping algorithms have started to use FPGA-based implementations for real-time matching of HD maps or odometry [168]. Some of these operations are inherently parallelizable, and therefore FPGA-based accelerators have the potential for decreasing the latency by several orders of magnitude. In the proposed architecture, we envision that dynamic reconfiguration of FPGA-based hardware accelerators will play an important role in optimizing edge resources for computational offloading, increasing the number of nodes that can be supported and reducing the execution time of different processes.

ROS services can be directly utilized in offloading schemes, where third party services simply offer these to the network. In addition, there has been a recent interest in developing ROS-compliant accelerators to match the rising computational needs [259]. Having reconfigurable hardware available on-demand at the edge can help third-party service providers integrate these solutions. The reconfiguration and provisioning of resources can then be made through smart contracts executed in the MEC-hosted blockchain. Hardware accelerator models can be naturally abstracted in terms of the number of processing units or resources required, and multiple models can co-exist within a single chip. Finally, data partitioning schemes at the blockchain level and its modular architecture with the aforementioned concepts put together an efficient, open and flexible framework for offering offloading services.

3.3.5 Security concerns

The blockchain is a key piece in the proposed architecture as a source of trust. We consider that the main security concerns in a MEC service framework is not the exposure of data but instead its validity and reliability. This is exemplified by threats identified by ENISA such as the manipulation of the network resources orchestrator (unreliable orchestration or invalid data regarding the resource orchestration) [260]. As we are discussing services that support the operation of autonomous vehicles, we need to take into account that this is a safety-critical application scenario where sending wrong data to a connected vehicle or robot might put in danger pedestrians and drivers. While the blockchain is not able to provide a robust way to validate data by itself, a ranking of the different identities offering services can be created and updater in real-time. Moreover, by implementing the resource orchestration and management of edge resources with smart contracts, the reliability of services providing mission-critical data can be kept under tighter control. Finally, the immutability of the transaction record can be utilized as a posteriori to assign liability and ensure accountability.

Figure 19 shows a classification of edge computing services that can benefit from, integrate or leverage blockchain technologies. Distributed robotic applications can be found within all these areas, as well as other, more generic, IoT use cases.

3.4 Consensus in swarms with blockchain technology

We now move towards the opportunities within collaborative robot swarms. The last two sections covered the potential of blockchain technology and multi-access edge computing for distributed robotic systems. In this section, we define an strategy for managing collaboration and establishing consensus in a collaborative robotic swarm utilizing a blockchain. Furthermore, we discuss how different aspects of the blockchain could be adapted to the specifics of robotic cooperation, where the most valuable token that can be exchanged between robots is data. Therefore, part of the security focus is shifted from the transaction validation point of view to the data quality aspect. In that regard, a blockchain can be utilized to establish a secure way of evaluating and ranking the quality of data provided by the different robots.

The approach presented in this section can be generalized towards achieving consensus in a large multi-robot system, or swarm. Nonetheless, we focus on a specific problem that is particularly significant: cooperative mapping and collaborative sensing or perception within a heterogeneous team of autonomous robots operating in the same environment. With perception, localization and mapping being three of the cornerstones behind fully autonomous operation, the problem of collaborative sensing for enhancing the situational awareness of each of the individual robots sets the basics towards more complex collaboration.

We focus on heterogeneous robotic systems because of their dynamism and the wider variety of applications that they enable. Heterogeneous multi-robot systems or larger swarms have been studied for over two decades [261; 262; 263; 264; 265]. In addition, we consider ad hoc swarms where the number of robots can change over time, and their properties are not within a predefined set. These changes present multiple challenges from the perspective of a heterogeneous resources management system. In a homogeneous ad hoc robotic system, different parameters such as computation power, bandwidth distribution or type and amount of data to be shared can be either predefined or calculated based on a preconfigured strategy. This means that, in most cases, the way that robots interact with each other will not suffer from sudden changes. However, the same does not apply to heterogeneous robotic systems, where a new robot joining the collaboration effort might have a sensor suite or computational resources very different from the rest of collaborating robots. In that case, all the robots need to adapt their collaboration schemes, with potentially significant changes in the way information flows within the network and in the selection of robots that have priority over others to either share or receive information.

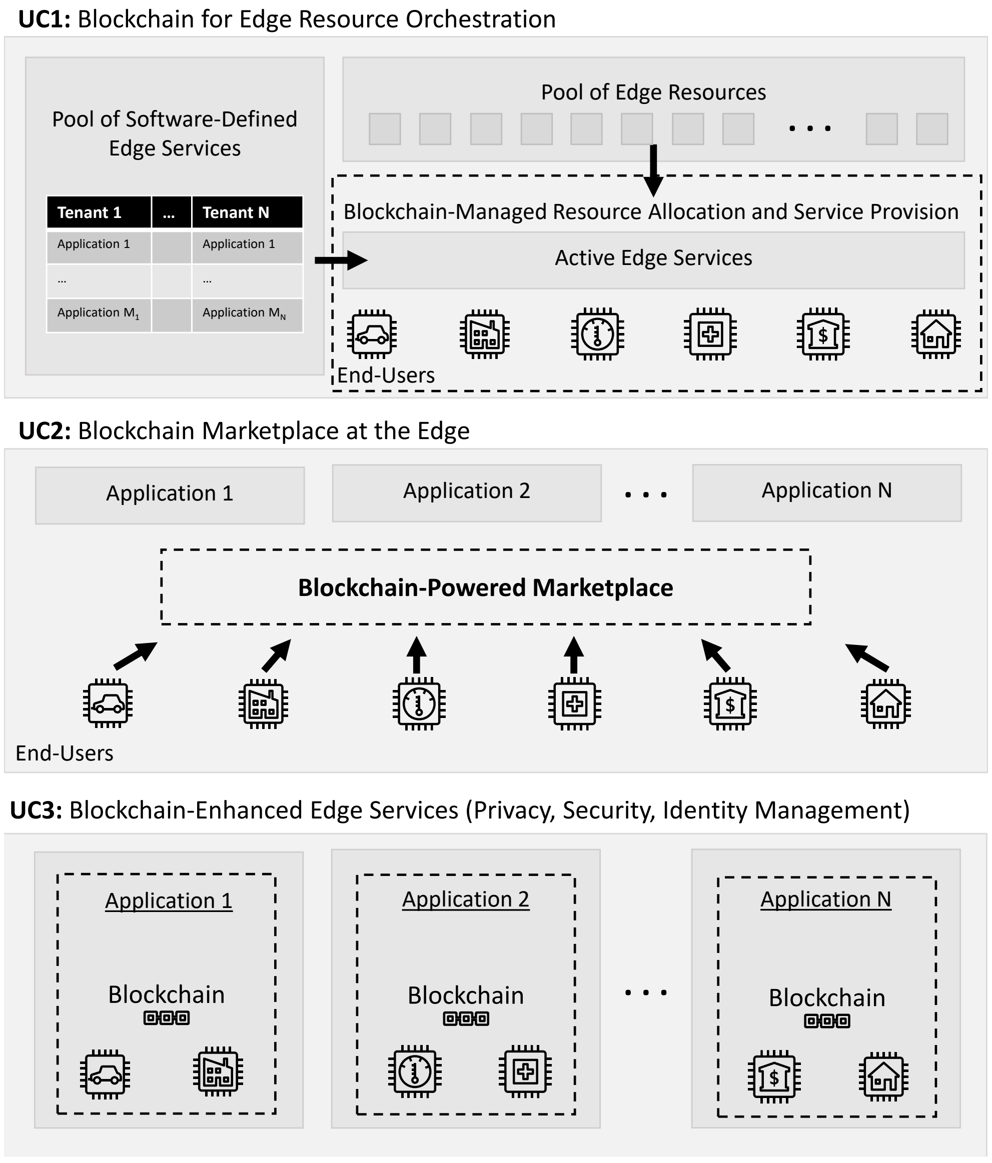

UC1: Blockchain for Edge Resource Orchestration
Pool of Software-Defined Edge Services
Tenant 1
...
Tenant N
Application 1
Application 1
...
...
Application M1
Application MN
Pool of Edge Resources
Blockchain-Managed Resource Allocation and Service Provision
Active Edge Services
End-Users
UC2: Blockchain Marketplace at the Edge
Application 1
Application 2
Application N
Blockchain-Powered Marketplace
End-Users
UC3: Blockchain-Enhanced Edge Services (Privacy, Security, Identity Management)
Application 1
Blockchain
Application 2
Blockchain
Application N
Blockchain

Figure 19. Main use cases for blockchain within edge computing systems. (UC1) Blockchain-powered resource allocation and service provision; (UC2) Blockchain-powered marketplace for interfacing users and services; (UC3) Blockchain-enhanced individual edge services relying on blockchain technology for security, privacy, data management and audits or identity management, among others.

An additional assumption that we make is that robots can be anonymous and that all robots have the same role within the blockchain. However, some conditions must be set in this regard. As we are utilizing a blockchain in order to manage the collaboration and consensus between the different individual robots, a minimum number of nodes must remain in the network in order to keep the blockchain alive. Alternatively, infrastructure in the operation environment can be utilized in order to provide the backbone of the blockchain, making sure that it stays alive and with a minimum level of security so that previously stored data can still be trusted or utilized by new robots joining the collaborative network.

Keeping the blockchain alive and all previous records has the disadvantage of a higher overhead when a robot joins the network. Nonetheless, in this section we present an approach to evaluate the quality of data provided by the different robots and utilize a ranking based on this metric in order to manage the collaboration. In the case of having robots operating only at sparse time intervals, with long idle periods in which the robots are off or offline, these might lose their status and need to regain it every time they join the network. This has a negative impact not only in the robot itself, which would need to regain the trust of its peers in terms of data quality, but also for the rest of robots, which might be receiving less accurate data until the previous status of the new collaborating peer is regained. Therefore, the level of optimality of the collaboration might be reduced. In general, there is a trade-off between the benefits that the data history stored in the blockchain brings to the collaboration with the drawbacks in terms of synchronization and overhead when robots join or leave the network. It is left as a design decision up whether only the blocks generated in a certain recent time frame are downloaded by new robots or the whole blockchain is. Not having the complete blockchain is not an issue in terms of transaction validation because we introduce a demurrage effect inspired by the design ideas of Ethereum 2.0 to the network's cryptocurrency. Therefore, the tokens that individual robots had in their stake earlier than a certain time threshold are no longer usable.

Having a connected infrastructure that supports the blockchain is an interesting approach that has the potential to enable multiple applications in the era of 5G and beyond connectivity. With network slicing, softwarization and virtualization, and the ability to support a variety of different verticals, blockchains and other systems could be deployed within the base stations. In those cases, permissioned blockchains might be considered a more suitable solution, where the role of the infrastructure could be more related to validating and increasing the level of security. In any case, in this section we also consider fully distributed and anonymous permissionless blockchains. A more in depth analysis on this topic is given within the discussion of the design principles.

We focus on providing a strategy with potential to solve two existing challenges in heterogeneous multi-robot collaboration: (1) the management of bandwidth in a peer-to-peer network between different robots, potentially having a myriad of sensing capabilities and computational resources, choosing which robots have priority to transmit their data and which robots will receive specific data batches; and (2) the decision making in terms of what specific data do robots share in order to optimize the benefits that results from the collaboration process. Both concepts are closely related; if a specific robot is able to obtain higher quality data, then a more optimal collaboration can be achieved if its data is shared among the interested peers. In that case, this particular robot should have priority in terms of bandwidth usage. While this argument utilizes only the sensing capabilities of robots, a similar approach can be taken in terms of the computational resources of each robot. In a smart city environment, an autonomous delivery drone with limited sensing abilities, for example running visual inertial odometry with a single onboard camera, could benefit from the data extracted by a self-driving car equipped with multiple cameras, radars and a high-quality multi-channel 3D lidar. However, if the car simply streams its data, the drone would be barely able to process most of it due to constrained computational resources in its onboard computer. Therefore, it might either discard most of the data when receiving it, or accumulate it and induce a delay in the processing with the consequent latency increase in localization or mapping tasks.

In summary, the main difference of our approach with respect to previous works is that robots do not need to share information about their sensing and computational capabilities, yet they can be able to optimize the way they are collaborating. We understand collaboration optimality as the enhancement of each of the robot's situational awareness resulting from the analysis of data provided by their collaborating peers. By utilizing a blockchain framework, and adapting the latest developments in the blockchain field, we are able to provide robots with means for more efficient ways of collaboration without having to share, trust and interpret specific data about their sensors or data processing capabilities.

3.4.1 Blockchain-powered collaboration

The utilization of a blockchain for managing the collaboration between robots, based on the assumptions and situations described above, can have multiple benefits. However, multiple design aspects that can significantly affect the way the blockchain is utilized have not been specified so far. It is not the aim of this section to provide a specific set of methods for heterogeneous robotic collaboration, but instead to provide a set of basic strategies and design concepts that can be utilized towards the definition of advanced collaboration schemes in the future. In this section, we overview the different approaches that can be taken in different blockchain aspects.

Blockchain genesis

The first design decision is whether a longevous blockchain is preferred, or ad hoc blockchains are created when needed. Both options present significant challenges from the implementation point of view, and both can be utilized in different environments or specific application scenarios. We overview the main benefits and drawbacks of each of them. In future work, we will put into practice these two options and evaluate different parameters in order to provide more insight.

I. Long-term support

In the case of a single longevous blockchain (i.e., persistent in time even with nodes entering and leaving the nerwork) with ad hoc collaboration where new robots might join the network, and other robots might go offline for long periods of time, the main challenge is to ensure that a minimum number of nodes is able to securely maintain the blockchain state. We believe that this type of blockchain can only be effectively implemented when connected infrastructure is added into the blockchain. Whether it is a private or semi-private blockchain in an industrial environment, or an open permissionless blockchain that robots can join to improve their operational performance in a smart city, there must be a minimum set of nodes that are fixed in the environment and are able to support the blockchain even in the case that no robots are active. Furthermore, connected infrastructure, such as mobile network access points or smart gateways near public Wi-Fi hotspots around a city, can be utilized to define a standard for communication within the blockchain and to publish its existence so that anyone can decide to join it.

An immediate question that raises when considering connected infrastructure, which naturally has a different role from robots joining the network, is whether a permissionless blockchain is the best option, or a permissioned consortium blockchain where the infrastructure nodes have a predefined role of validators is more secure. We favour the permissionless and open blockchain option because its benefits in terms of flexibility and because the key aspect of a blockchain for robotic collaboration does not rely as much on the transaction security as it rather does in the integrity and quality of shared data, which can not be validated with a traditional blockchain approach.

II. Ad-hoc blockchains

An alternative to a single blockchain, potentially supported by connected infrastructure, is to create and destroy collaborative networks on the fly. The main disadvantage of this approach is on how to define the conditions under which the blockchain is started, and which entities are allowed to initiate the process. This option is not suitable for applications where individual and anonymous robots are collaborating, with potentially independent developers. Instead, this might be a more suitable approach for situations where a single controller or developer is deploying a large robotic swarm, which may already include a collaboration scheme or not. In that case, the blockchain genesis can be established by the developer, and its existence made public, opening the door to other robots or swarms to join and share data. In this case, the newly joined members should put trust on the blockchain initiators.

A similar situation where an ad hoc blockchain could be applied is in the case of multiple end-devices, vehicles or robots being produced by a single manufacturer but utilized by different individuals. For instance, this could be the case of a company selling self-driving cars. These could be preconfigured to automatically detect other cars from the same manufacturer in the vicinity. In the event of multiple cars converging in a near area, a blockchain could be started without the car owners being aware of it, and these could start benefiting from the data collected at other vehicles, with enhanced autonomous operation. It would be then a decision of the manufacturer whether to broadcast the existence of this network for other vehicles to connect or not. In comparison with a direct cooperative data sharing approach, the utilization of the blockchain and the data quality validation strategies presented in this section would ensure that faulty sensors or tampered sensor data can be detected by the network if certain conditions are met under which vehicles are able to validate each other's data.

Consensus mechanism

While PoW shows an additional potential other than providing a means for consensus, its many drawbacks described in Section 2 make it unsuitable for long-term scalability. In consequence, we propose the utilization of a Proof-of-Stake system for validating transactions, while maintaining a periodic PoW for computational resource estimation. In any case, the energy and time spent by miners towards the PoW is not futile, as it will be utilized by the network to estimate the available computational resources at different nodes. In general, we propose the utilization of a protocol similar to the Casper PoS protocol in Ethereum 2.0, while a PoW is required to be executed periodically for nodes to be still considered part of the peer-to-peer network [219; 218].

In a traditional blockchain architecture, a transaction is validated when the corresponding block where it has been included is mined. In our case, transactions in the proposed network architecture are data exchanges between one robot and a subset of its peers. The data itself is shared within the peer-to-peer network but outside of the blockchain. A certain sample is shared within the blockchain as part of the transaction body so that the whole network can run the data quality evaluation and ranking procedure. Robots do not wait for transactions to be validated within a block before sharing the data. Rather than coining new cryptocurrency though the consensus mechanism, we propose the utilization of of the periodic PoW mechanism to provide robots with a preconfigured and fixed amount of cryptocurrency. While in the case of the Ethereum blockchain the Ether can be utilized both to perform trade between peers, and to purchase the network's execution time, in this section we only consider the latter scenario. Therefore, the only use of the cryptocurrency is to be able to perform data transactions and be sure that data will be forwarded through the peer-to-peer network.

Rather than providing rewards for block mining as an incentive, the PoW puzzles are compulsory for nodes to be considered part of the network. The time between PoW requests can be preset within the network configuration, as well as the procedure by which the next PoW is automatically calculated based on the blockchain's state or other parameters. In order to ensure that nodes do not collect a large amount of coins, we propose the implementation of a demurrage mechanism as in Ethereum 2.0, where all the cryptocurrency is effectively congealed and disappears within a certain time interval. In addition, penalties must be included to further control the utilization of cryptocurrencies. In doing so, we can limit the amount of data that dishonest nodes are able to send over the peer-to-peer network.

Security concerns

In a robotic collaboration system, data exchanges are the most valuable tokens. However, including all data in the blockchain would significantly reduce its usability because of the impact on scalability and the latency that such amount of data processing and validation in all nodes would induce. Therefore, in order to provide basic means for the collaborating robots to decide the level of trust that they put into a certain robot, data samples are submitted to the blockchain and evaluated within the network through smart contracts. These samples are then ranked and utilized in order to estimate whether a robot is honest or not, but also which robot is able to provide more accurate or useful data given a particular request.

It is not within the scope of the collaborative decision making presented in this section the definition of how the data quality ranking is taken into account at each individual robots. This is for application developers and must be implemented separately at each robot controller.

Scalability

In order to ensure efficient scalability of the proposed blockchain architecture, we propose the utilization of spatial shards for local decision making in terms of data quality evaluation. Therefore, both a local ranking and a global ranking are kept in record at the shard chains and global chain. This is a useful approach in order to reduce the network load and induced latency. The consensus mechanisms would run in parallel shards which would be defined based on the Ethereum 2.0 standards. A single validator thus belongs to two kinds of shards, spatial shards utilized for running local smart contracts regarding local data quality ranking, and randomized shards running the consensus mechanism and maintaining a global ranking with separate smart contracts. More insight into the definition of the local and global rankings is given in the data evaluation section.

3.4.2 PoW for online estimation of computational resources

In this section, we propose a methodology for estimating the computational resources of each of the collaborating robots by exploiting the PoW puzzles utilized in the blockchain in order to validate blocks. The time required to solve a PoW puzzle can be utilized as an indicator of the available computational resources at a given robot, and partial proofs can be used in case robots are not able to solve a PoW puzzle within a certain predefined time interval. We utilize the term available, rather than total computational resources, because we assume that robots are able to operate autonomously on their own, and utilize the collaboration in order to improve the accuracy of the different methods that they already run. Therefore, robots must decide which amount of resources do they want to reserve for the collaborative effort; the more resources they put into solving PoW puzzles, the more data they are able to obtain, as the amount of data is calculated based on the available processing power in order not to overload the receiving robot with more data that it is able to process.

In a typical PoW utilization for block mining in a blockchain, once a miner finds a solution to the PoW puzzle and broadcasts it, all other miners automatically discard their solutions and start working on mining a new block. However, this can only give an idea about the processing power of the node that was able to mine the block. In order to be able to obtain useful information regarding all nodes in the network, partial proofs of work can provide more insight into the effort that different nodes put towards the PoW puzzles.

The utilization of partial proof of works has been previously been proposed in different mechanisms that secure and raise the level of fairness in mining pools [189; 266; 190; 267; 191]. Mining pools utilize various payout systems in order to distribute the mined coins between their miners even if individual nodes have not been able to provide a full PoW solution. For the estimation of computation in a robotic system, a simplistic approach is enough. One naive solution is, for instance, that each robot is assigned a different PoW puzzle with equivalent complexity. This helps to avoid two robots submitting the same partial or full PoW solution while only one is actually calculating it. In this case, we do not need to consider the presence of malicious nodes that put less computational resources towards solving PoW that they can. This is because the conclusion from the network would be that the processing power is more limited at those nodes, and other robots would therefore send less data. In this approach, solving PoW puzzles is not the means towards a monetary reward, but instead towards a data reward. The faster a PoW problem is solved, the more data a robot is likely to receive from its peers. Therefore, individual nodes would gain nothing and only incur in their own detriment by lying to the network with less complex partial PoW solutions. Because the result of the PoW has to be shared with the rest of robots and can be easily validated by each of them, robots cannot provide fraudulent data regarding their computing capabilities. The reason robots might perform such malicious actions could be because either they want to destabilize the network or they want to increment the amount of data they are receiving in a network with limited bandwidth where most robots might have extra processing power that is not being put into use.

The mining difficulty should be set to a fixed value, in contrast with Ethereum's adaptable PoW complexity, so that robots with lower computational resources can also be part of the collaborative sensing scheme. Nonetheless, the complexity should be enough to ensure that only robots with a minimum level of computational capabilities are able to participate. In a similar way, the network connectivity of the robots must be put to test before joining the network in order to avoid bottlenecks and dub-optimal collaboration. At the same time, if the PoW is too easy to solve, then the communication overhead might play a more significant role. In general, if the network-wide communication latency is at least one order of magnitude smaller than the minimum time required to solve a PoW puzzle, then a simple averaging could suffice for more accurate, long-term estimation. Nonetheless, the estimation should be able to adapt to changes in the available computational resources. This can happen because robots might be running other computation intensive processing algorithms that are only executed at certain intervals, or only when a series of events occur. In order to do this, a naive approach would be, for instance, to select the last N PoW proofs or partial proofs such that for all (or most) $M < N$, the estimated computational capabilities C_M and C_N have low variance, i.e., $\mu_{C_M} - \sigma_{C_M} \leq \mu_{C_N} \leq \mu_{C_M} + \sigma_{C_M}$. However, potential outliers should be taken into account and a minimum number of partial PoW solutions $N > \varepsilon$ utilized in the collective estimation procedure. If the nature and capabilities of robots collaborating through the network changes, with significant increase or reduction of the average computing power, an alternative approach to adapting the PoW complexity is to set a maximum time that robots dedicate to the mining effort, even if none of them is able to produce a full PoW solution. In this case, however, the maximum latency in the peer-to-peer network should be taken into account in order to calculate the timeout interval, and its value should be negligible in comparison.

3.4.3 Data evaluation - proof of quality

We are basing the collaboration management in two parameters: the estimation of the available computational resources and the evaluation and ranking of the quality of data provided by different robots. For the first aspect, we have proposed the utilization of a PoW scheme in order to maintain an online estimation during operation. For the second aspect, the main idea presented in this section is to share within the blockchain a data stamp, or sample, whenever a data exchange transaction between two robots, has been made. Thus, not all the data is stored in the blockchain, but can be transmitted through a direct connection, an external network or the peer-to-peer network. We assume that the only connection between robots is the peer-to-peer network. This matches with the assumption that robots are anonymous, and therefore they do not necessarily have any means of contacting their peers. The main argument behind this assumption is that collective decision making in terms of what data is shared between certain pairs of robots is strongly affected by the constraints of bandwidth or latency inherent to the peer-to-peer network. In external channels exist, it is not straightforward to consider them within this distributed process.

The cryptocurrency in the proposed collaborative network has no real value. Instead, the most valuable asset is data. Therefore, we put the focus around the data and how to evaluate and rank its quality. The following parameters are utilized to evaluate the data stamp: (i) the type of data that has been provided; (ii) the density or resolution of the data sample; and (iii) the comparison of this data sample with historical samples from other robots that are or have been operating in a close location. Regarding the type of data that is being shared, it can be classified, for instance, into visual data (camera images), point cloud data (from lidars, depth cameras), or radar data. Then, the resolution of these images, or the density of a lidar point cloud is also taken into account. Based on this evaluation, penalties can be defined for nodes that fail to provide certifiable data stamps by providing a reduced share of newly coined cryptocurrency.

A ranking of the quality of data is not kept within the blockchain. This is because the type of data can only be evaluated based on the global needs of the system, which can considerable change over time. What is stored in the blockchain is the results of comparison of data stamps from the same environment. This result can be (i) a confirmation that the data matches, with no ability to provide further information; (ii) a confirmation that the data matches and that the current sample is either less dense and included in the sample, or that the new sample has more resolution; or (iii) a mismatch event where the robot has been unable to confirm that both the historical sample and its new sample represent the same object or environment.

A trustability concern arises when considering the event of a subset of nodes submitting bogus data stamps to the blockchain, which are in turn validated by other robots in an attack coalition. However, in order to do this, the attacker coalition must hold a majority in a given spatial localization. We propose the utilization of a validation graph, where two nodes are connected if any one has validated the other's data stamp, and its analysis in order to detect fully connected or almost fully connected subgraphs, or whether it is a disconnected graph with multiple separated components. While this can give an idea of trust, the decision-making process from this information is not straightforward. Consider for instance the case in which a certain group of honest robots are sharing and validating each other's real data, and an equivalent number malicious nodes is sharing and validating counterfeit data. If these are the only collaborating robots in the vicinity, then for a robot in another location it is impossible to discern between them. However, this might not necessarily be a problem if no other robot is utilizing data from that particular location. In the long-term, given a majority of honest collaborating robots, and assuming that most locations are visited by a large enough number of the honest robots, then the counterfeit data stamps will be eventually detected and the set of malicious nodes will be labelled as dishonest.

3.4.4 Peer-to-peer data sharing scheme

So far we have provided an approach that assumes that robots do not share explicit information about their sensors, or the on-board hardware resources. This means that the maximum level of optimality that can result from such collaboration is limited by how well can the different proposed approaches abstract and model the robots' resources and capabilities. In other words, if the estimation of computational resources that is obtained via PoW cannot be performed in an accurate manner, this inherently limits the maximum level of performance of the proposed system. A similar situation occurs with the estimation of data quality and robot trustability level.

If robots decide to share data without filling a transaction within the blockchain, it must occur either through a direct link or outside of the peer-to-peer network. If a third node receives a request to forward data between a given pair of robots, it only does so if the corresponding hash of the data sample has been already included in the blockchain.

In the same sense than in Ethereum nodes buy gas in order to execute smart contracts, each transaction consumes a given amount of cryptocurrency. In order to avoid a situation in which a malicious node would start multiple transactions, effectively double-spending its stake, sending large amounts of data in order to collapse the network, we propose that a strategy similar to the gas congealing scheme in Ethereum 2.0 can be utilized. The key difference is that all the cryptocurrency that a robot has in its stake is congealed and subject to the demurrage effect.

The decision making in terms of the data to be transmitted can be established as an optimization problem where the bandwidth of the peer-to-peer network and the available computational resources at the nodes receiving the data are considered constraints. The function to be maximized is a weighted sum of the data that robots receive based on their requests, and a measure of the trust put into the quality of that data. In this section, we provide a high-level approach and do not delve into the details of specific calculations, which will be considered in future work.

Consider that at a certain time instant the blockchain is formed by a coalition of N nodes, with their indexes represented by the set $[N] = 0, \dots, N$. A new PoW problem is considered by all robots, where L_H represents the size of the PoW hash in bits, and $PoW_i \; \forall i < N$ is the full or partial solution that robots provide and is verifiable by their peers, $PoW_i \in \mathbb{Z}_2^{L_H}$. The available processing power at each robot is then estimated relatively to that of other robots, and mapped to the interval $\mathbb{R}^{[0,1]} = \{x \in \mathbb{R} | x \in [0,1]\}$. We denote the estimator with $\hat{\mathcal{C}} : \mathbb{Z}_2^{L_H N} \implies \mathbb{R}^{N[0,1]}$ which takes as input the set $\{PoW_i\}_{i \in [N]}$ and outputs the set $\{\hat{\mathcal{C}}_i\}_{i \in [N]}$. During the estimation calculation, an additional parameter $\mathbb{D}_{max}$ is calculated, which defines the maximum amount of data that the robot with the most computing power is able to process per second, in bytes. Upon submitting a PoW full or partial solution, each robot i also submits a data request, denoted DR_i, which contains a requested amount, in bytes, type of data (image, point cloud, radar or others) and a minimum and maximum resolution or density $DR_{i,j} = struct\{type, max\_size, min\_res, max\_res\}$, where j varies for each different data type. Additionally, robots submit information about the data they are able to provide, or available data, with the same type of information $AD_{i,j} = struct\{type, max\_size, min\_res, max\_res\}$, where again j iterates over the available data types. We suppose that an error function $\mathcal{E}_{DR_{i,j}, DA_{k,j}} > 0$ is given that increases as the mismatch between the desired data size and resolution in a data request from robot i and the available data at robot k. Finally, robots share their position in a global reference frame and an estimation of its error. This is utilized in order to divide the robots in spatially-defined shards, and at individual robots to decide how they utilize the received data.

We assume that the data quality evaluation provides a value $\mathcal{Q}_i \neq 0 \in \mathbb{R}$ that can be negative and represents the trust that the network puts on robot i. We do not provide a specific formula to calculate this value, but instead refer to the guidelines described in the previous section. Given the maximum bandwidth of the link between robots i and j in the peer to peer network, $BW_{i,j}$, we can now formulate the optimization problem that is solved as part of a smart contract deployed in the network in order to make decisions with respect to the data that is shared between robots:

Table 5. Hashing power of different boards typically utilized as onboard computers in robotics (in hashes per second). The hashing algorithm was SHA256 and the tests involved solving PoW puzzles with 22, 23 and 24 bits of difficulty. The hashing puzzles were solved running within a single thread. The standard deviation shows *NA* when it is below 1000 hashes/second.

	Intel Up	Intel Up Gateway	NVIDIA Jetson TX2	Intel i5-6200
μ_h **(h/s)**	89000	79000	184000	561000
σ_h **(h/s)**	NA	NA	1000	16000

$$\begin{aligned}
\arg\max_{X} \qquad & f(X) = \sum_{i=1}^{N} \left(\sum_{j \mid x_{ij} \in X} \alpha \mathcal{Q}_j + \beta \frac{1}{\mathcal{E}(x_{ij})} \right) \\
\text{subject to:} \qquad & x_{ij,\,size} \le \mathbb{D}_{max} \cdot \hat{\mathbb{C}}_i \qquad \forall x_{ij} \in X \\
& x_{ij,\,size} \le BW_{i,j} \qquad \forall x_{ij} \in X
\end{aligned}$$

where $X = \{x_{ij}\}$ represents a data exchange between the pair (i, j) with a given size $x_{ij,\,size}$, and $\mathcal{E}(x_{ij}) = \mathcal{E}_{DR_{i,j},\,DA_{k,j}}$. The parameters α and β define a weighted sum between data usability and data trustability that must be set depending on the range than the data quality and data error match functions can give. We have considered that the bandwidth $BW_{i,j}$ of the link between robots i, j is independent of data that might travel between the same link but does have a different recipient. This is a unrealistic assumption in a peer-to-peer network that potentially relies on a mesh network for communication. However, that is a problem more related to the graph theory domain and we do not consider within the scope of this section.

3.5 Initial assessment of a blockchain solution

In this section, we partly evaluate the potential of the proposed methods. In particular, we focus on studying the correlation between the computational resources required to solve a PoW problem with the execution of different algorithms widely utilized in robotics. In addition, we show different data samples that could be utilized within the data evaluation scheme.

3.5.1 PoW metrics

We have utilized four different computing platforms to evaluate the consistency of the relationship between the hashing power and different types of algorithms that autonomous robots might run during their operation. The PoW solver has been implemented a single-thread process so that it can then be run in parallel in order to

Table 6. Classification latency in tensorflow for a CNN classifying the CIFAR-10 dataset [268], and visual odometry (VO) latency for the Kitti dataset [269]. The standard deviation shows *NA* when it is below $10\mu s$.

	Intel Up	Intel Up GTW	Jetson TX2	Intel i5-6200
$\mu_{class\_latency}$ (μs)	4400	5000	700	770
$\sigma_{class\_latency}$ (μs)	500	NA	40	60
$\mu_{VO\_latency}$ (ms)	200	210	108	59
$\sigma_{VO\_latency}$ (ms)	112	119	50	29

take into account also the number of available threads or cores in the robot's onboard computer. The four computing platforms are (1) an Intel Up board, (2) an Intel Up Gateway, (3) an NVIDIA Jetson TX2, and (4) an Intel i5-6200U CPU. The NVIDIA Jetson TX2 has been specifically selected because it has an embedded Pascal GP. As the PoW relies only on a processor, it is not able to model accurately the processing power for applications that can be inherently accelerated with a GPU.

The hashing power of the different devices is shown in Table 5. The number of hashes varies almost an order of magnitude between the least and most capable ones. The standard deviation of the hashing is below 3% for all the devices under test. Table 6 then shows the latency of two different types of data processing processes: a convolutional neural network (CNN) classification for the CIFAR-10 dataset [268], and visual odometry for the Kitti dataset [269]. The standard deviation in the case of the CNN classification latency is always lower than 12%, while the visual odometry latency is more variable and has a standard deviation of over 50% in some cases.

Figures 20 and 21 show the relationship between the hashing power and the performance of the classification and odometry algorithms, respectively, for the different devices. In the case of the CNN classification, the ratio is mostly constant, except for the NVIDIA Jetson TX2. This shows that the PoW puzzle can model the processor capabilities with high fidelity, but fails when other types of resources (GPU in this case) play a significant role. The CNN classification runs at lower latency on the NVIDIA Jetson TX2 than on the Intel i5 processor.

In a full system implementation, multiple PoW schemes should be designed to model the different types of computing resources that could be available within the collaborating robots. In future works, we will analyze different types of PoW puzzles and hashing algorithms, and to avoid specialized accelerators such as ASICs.

3.5.2 Data sharing scheme

For the data sharing scheme, we show the amount of data that would need to be shared within the blockchain to compare it between different vehicles. We also show how the data characterization can depend on the point of view or distance to the fea-

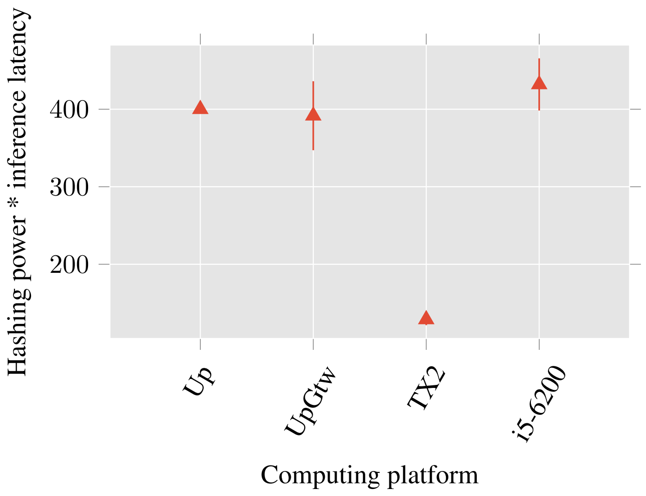


Figure 20. Relation between hashing power and classification latency for the different computing platforms. The ratio is mostly constant except for the NVIDIA Jetson TX2, which offers higher performance owing to its integrated Pascal GPU.

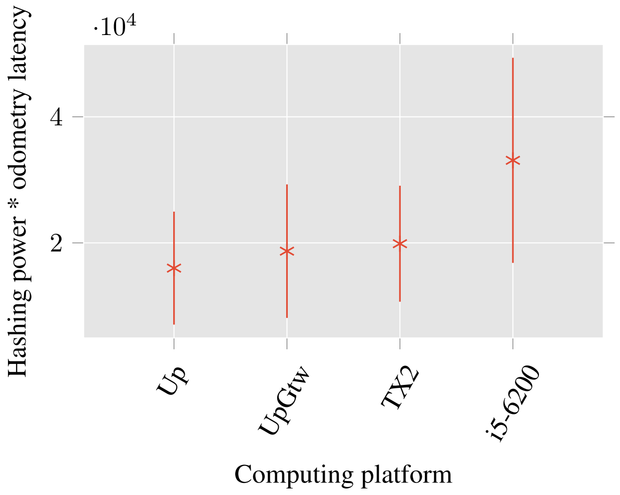


Figure 21. Relation between hashing power and visual odometry latency for the different computing platforms. The ratio is maintained mostly constant if the high variance is taken into account, due to the task being run completely on the CPU.

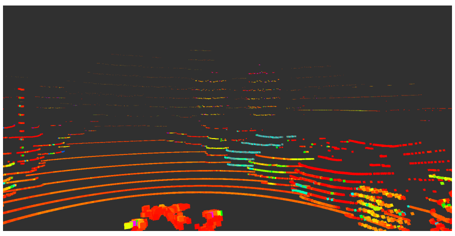
ture that is being utilized for the data sample. This is important to take into account when comparing the same feature from data samples submitted to the blockchain by different vehicles or robots at different times. Because the point of view of the object in the data sample and the distance to it might affect the properties of the data in the sample, a relation between these must be defined beforehand. Then, different data samples can be compared appropriately.

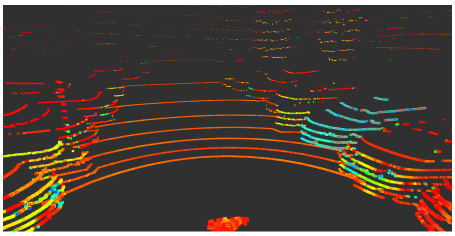
We have utilized 3D lidar data from a 32-channel Velodyne laser scanner, which has been previously utilized for localization in [18], where the platform utilized for gathering the data is described in detail. We have focused on analyzing how the data can be characterized based on the distance to the object that is being utilized for the data sample. Figure 22 shows the extracted point cloud from a car in a parking lot next to the road. In this case, both the distance to the object and the visible are important factors because there are other objects nearby. Therefore, different parts of the car are visible from different points of view, and the relationship between the segmented point cloud size and the distance to the object is no longer linear. In this case, moreover, the number of channels that have a projection over the car varies with the distance. We have extracted only eight of the channels from the lidar data to show that ratio between a 16-channel and an 8-channel point cloud size is not constant.

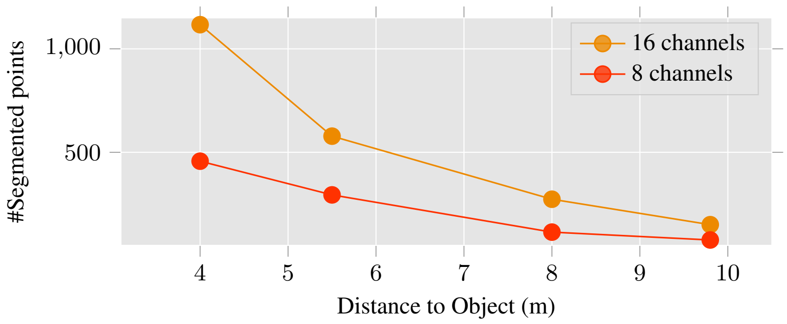
Taking into account the above data, in order to design the data characterization function different types of objects or features and their characteristics must be considered. From these, we can then define a function that defines the data density based on the distance to an object (as in the tree), the area of the feature being encoded in the data sample (as in the case of the building corner), or both the distance and the visible area (as with the car). These are just some examples and by no means do they provide an exhaustive classification. However, in this section, we have focused on showing different possibilities to define the data characterization scheme and justified them with different real-world cases.

3.5.3 Initial implementation

An initial implementation with Ethereum has been made, where the data samples are submitted as data payloads in individual transactions between devices. In this setting, smart contracts are not yet utilized to implement the data ranking. Instead, the data is submitted as a payload in standard Ethereum transactions together with 1 Ether. To do this, a private Ethereum network has been deployed with PoW difficulty set to 0 bits. Therefore, the mined blocks and the Ethereum generated is only limited by the time between blocks specified in the genesis file. Table 7 shows the gas necessary to submit a transaction for inclusion in a blockchain block depending on the size of the data payload. This relationship must be taken into account when deciding the amount of ether or gas given to each robot based on the amount of data that they are


(a) Car at 8 m (in blue)


(b) Car at 5.5 m (in blue)



(c) Size of the segmented tree point cloud seen at different distances

Figure 22. Size of 3D lidar data samples for a car in a roadside parking (point cloud data). The number of points does not follow a linear ratio with the distance because the area of the car visible to the sensor increases with the distance, together with the density of the point cloud. We show the number of points in the data samples for a 16-channel and a (simulated) 8-channel lidar, where the number of points depends on the number of channels that are projected onto the car surface. Data described in [18].

Table 7. Gas needed in order to attach a given amount of data to a Ethereum transaction. All transactions involved 1 ether and a data payload.

	Transaction				
	#1	#2	#3	#4	#5
Payload (bytes)	20	1080	2160	4320	8640
Gas	21680	57720	94440	167880	314760

going to share with their collaborating peers. Then, the amount of data in each data sample can be predefined based on the total amount of data shared, and checked by the nodes receiving the data. This helps ensuring fair use of the bandwidth in the peer-to-peer network.

3.6 Discussion

In general terms, the approach proposed in this section requires more maturity of some blockchain technologies that are not widely in use yet. Some of these can be seen within the roadmap towards Ethereum 2.0. The collaboration process described in the previous section can be summarized with the following steps:

1. Genesis of the blockchain. A decision is taken regarding whether it is supported by fixed assets, such as infrastructure, or automatically destroyed when the number of collaborating robots reaches a minimum threshold.

2. A new robot is able to join the network by providing a PoW solution, in order to avoid Sybil attacks and ensure that all robots have a minimum of available computational resources. Upon joining, the bandwidth of the connection between the robots and the peers that it is connected to is put to test. This can be done periodically in order to have an estimation of the peer-to-peer network bandwidth, if other means of calculating its capacity are not available.

3. Periodic submission of partial or full PoW solutions. This is utilized to have an online estimation of the available computational resources at the different robots. Together with the PoW solution, a series of data stamp is submitted. Each data stamp must represent the type and density of data that the robot is able to share with the rest of the network. An estimation of the maximum data throughput that it can stream for each of the types must be included as well. If the robot is located in a location where previous data stamps exist, and it is able to capture comparable data, then additional data stamps are submitted as well. These additional stamps must be accompanied by a comparison result and corresponding details, which must be verifiable by robots with enough computing power.

4. Together with the PoW and data stamps, robots inform their peers about the type of data and a range of resolutions that would benefit their autonomous operation.

5. If data stamps which can be comparable to previous entries in the blockchain are submitted, then all robots with enough processing capabilities must perform the comparison. An PoS approach is utilized in order to validate a comparison, where the stake is calculated based on the number of positively confirmed data stamps submitted by each individual robot.

6. Upon receiving all partial PoW and the corresponding data stamps, a smart contract is executed to perform the online estimation of available processing power.

7. Utilizing the processing capabilities at the receiving robots and the peer-to-peer network bandwidth as constraints, an optimization problem is solved in order to decide the usage of the network.

8. Robots receiving data must confirm that its properties are equivalent to those of the submitted data stamp. In the event of a mismatch or the inability of the receiving robot to verify that the data stamp is part of the received data, a negative receipt is issued which affects the evaluation of data quality and trustability together with the stamp-to-stamp matching process.

An illustration of some of the steps summarized above is provided in Fig. 23, where different robots have various computational capabilities and share data stamps, partial or full PoW solutions, data requests and information about available data in order to collaborative decide on the best usage of the peer-to-peer network and the data that will be shared. After this, robots start collaborating, utilizing the newly coined cryptocurrency to make data exchange transactions until the next PoW puzzle has to be solved. Therefore, the behaviour of the collaborating robots in terms of communication and data sharing is static between two consecutive PoW solving rounds.

3.6.1 Challenges

In order to arrive to a successful implementation of the strategies presented in this section, numerous challenges need to be overcome. Nonetheless, the main objective of this section is to define a basic set of design strategies and architectures that have potential to enable secure and efficient ad hoc robotic collaboration in the near future.

Some of the main challenges of the proposed, which will be studied further in future work, are the following: (i) the management of blockchain lifecycles, or the management of trust where a predefined set of assets is in charge of maintaining the blockchain; (ii) the prevention of broad-scale attacks where a large group of attackers submit counterfeit data to the network and continuously provide new stamps and confirmations of such data; and (iii) how to ensure that robots do not join the network only to obtain data but do not show the real quality of their data, for instance by downsampling point clouds or inducing blur in images. This last point presents particular complexity because it is virtually impossible for robots to evaluate whether their peers do have more sensing capabilities or not, and limiting the amount of data that robots receive based on the quality of data that they provide would have a significant negative impact in resource-constrained robots with limited sensor suites such as drones or small delivery robots.

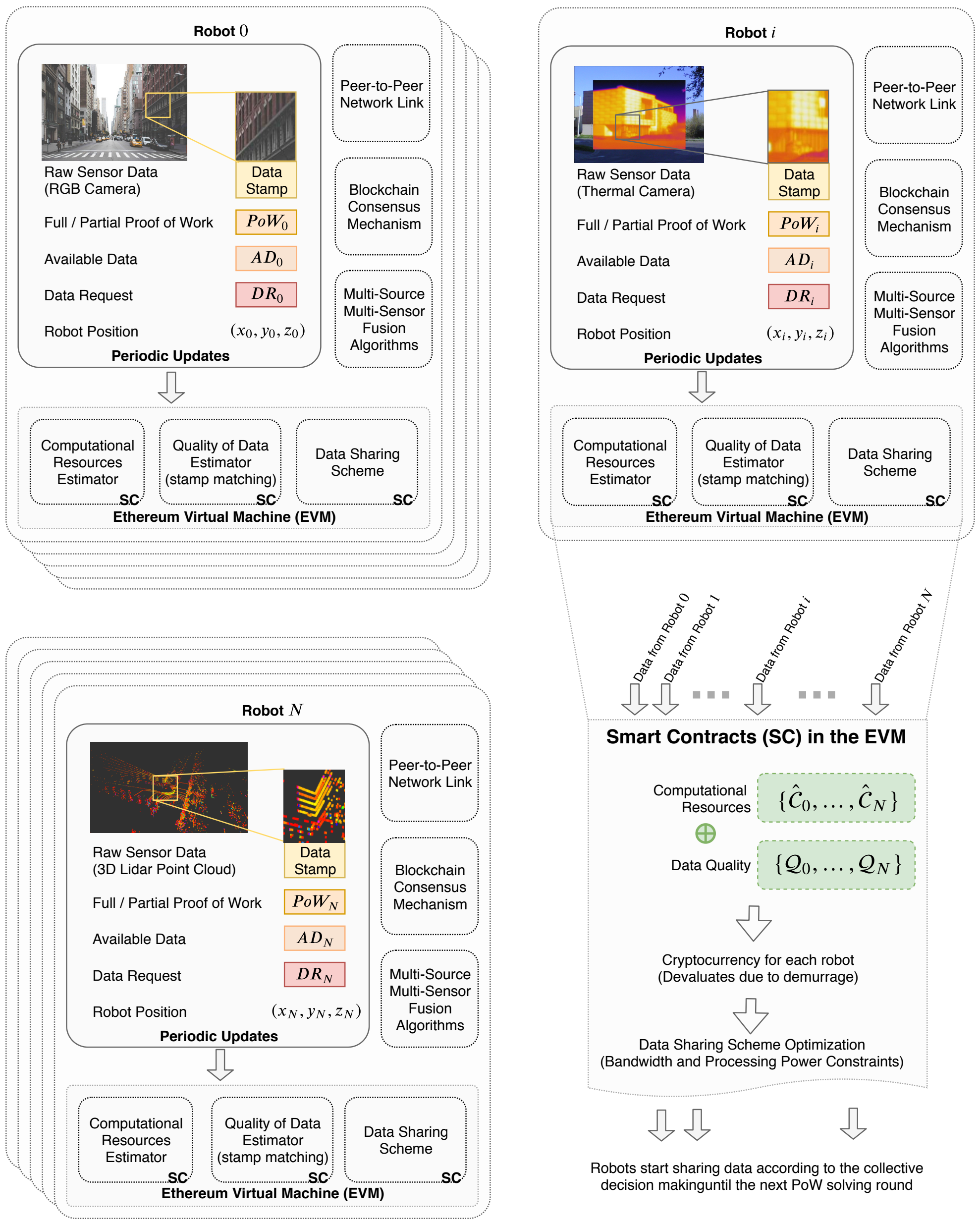

Robot 0
Raw Sensor Data (RGB Camera)
Data Stamp
Full / Partial Proof of Work
PoW_0
Available Data
AD_0
Data Request
DR_0
Robot Position
(x_0, y_0, z_0)
Periodic Updates
Peer-to-Peer Network Link
Blockchain Consensus Mechanism
Multi-Source Multi-Sensor Fusion Algorithms
Computational Resources Estimator
Quality of Data Estimator (stamp matching)
Data Sharing Scheme
SC
Ethereum Virtual Machine (EVM)
Robot i
Raw Sensor Data (Thermal Camera)
PoW_i
AD_i
DR_i
(x_i, y_i, z_i)
Robot N
Raw Sensor Data (3D Lidar Point Cloud)
PoW_N
AD_N
DR_N
(x_N, y_N, z_N)
Data from Robot 0
Data from Robot 1
Data from Robot i
Data from Robot N
Smart Contracts (SC) in the EVM
Computational Resources
$\{\hat{C}_0, \ldots, \hat{C}_N\}$
Data Quality
$\{Q_0, \ldots, Q_N\}$
Cryptocurrency for each robot (Devaluates due to demurrage)
Data Sharing Scheme Optimization (Bandwidth and Processing Power Constraints)
Robots start sharing data according to the collective decision makinguntil the next PoW solving round

Figure 23. Illustration of the inputs utilized in order to estimate computational resources and data quality. The data stamps are taken from specific features from the environment that can be easily recognized by other robots, such as the structure of windows or the corners in a building.

3.6.2 Opportunities

We see the main opportunities of the proposed system as part of smart cities, where the blockchain can be supported by deployed infrastructure, as well as industrial environments where there exists trust between parties but different actors deploy robots in the same environment. In a smart city, a public blockchain for data sharing could boost the deployment of autonomous robots from both private and public entities. In addition, the role of the infrastructure should be considered not only as a platform to manage the blockchain lifecycle but also as a static data source and validating platform, where traffic cameras and other sensors that already exist can be integrated. With this sensor data, connected infrastructure can validate data stamps from the robots, but also provide data to interested parties.

3.7 Summary and conclusions

This chapter covers a position on the use of blockchain and other distributed ledger technologies within multi-robot systems. Throughout the chapter, we discuss potential uses at the edge from the perspective of distributed systems, but also in terms of driving collaboration within multi-robot systems. Within the scope of this thesis, this chapter presents a more generalist view of the opportunities, while in the next chapter we delve into a particular implementation using only a small subset of technologies. This chapter also extends the previous chapter with a mode in-depth discussion of the blockchain potential. Out of the scope of this thesis, the ideas presented in this chapter have matured with our ongoing work in the use of both permissionless and permissioned blockchains. For instance, in [270] we explore the integration of the Hyperledger Fabric blockchain framework with ROS 2 and show its potential for managing robot fleets and a proof of concept in multi-robot inventory management. We also present a methodology for integrating the IOTA DLT platform with ROS 2 in [271].

4 Securing single- and multi-robot missions

Throughout the thesis, we have presented different design architectures that are based in distributed ledger technologies. In this chapter, we delve into more specific methods for securing robot behaviour. We extend previous works from our co-authors [272], where Merkle trees (data structures typically employed in blockchain frameworks for validating data) are used to create secret mission instructions. In this chapter, we only utilize cryptographic hashing methods for encoding instructions and defining new secure ways in which an autonomous robot can interact with its environment.

Portions of text and a subset of figures in this chapter are reproduced from our previous work [1].

4.1 Premise and motivation

With robots and autonomous robots having an increasing penetration across multiple aspects of our society, more attention is being paid to the safety and security aspects in robotic operation [273]. The differentiation between safety and security often becomes fuzzy, with the safety term being utilized to refer to human-robot interaction [274], or to the safety of the robot itself [275]. In either case, safe operation of an autonomous robot requires tight control over the security of the data being used, from data defining mission instructions to sensor data. Figure 24 shows a layered classification of stages in which information is either collected or processed by an autonomous robotic system. This figure extends the cyberattacks categorization in [276], and also takes into consideration that the internal processes can be modeled as a software-defined network from a more abstract point of view [277]. Many robotic frameworks, such as the Robot Operating System (ROS) fall into this consideration [278]. From the cybersecurity domain point of view, the acquired sensor data needs to be secured as well. This represents an additional challenge. Therefore, an essential aspect in the operation of autonomous robots is to be able to validate both data being shared among subsystems and external systems (a controller or other robots), but also data defining or characterizing the way the robot, seen as a cyber-physical system, interacts with its environment.

A relevant precedent in securing multi-robot cooperation was introduced by Castelló Ferrer *et al.* in [272], were the authors leveraged Merkle trees to cope with byzan-

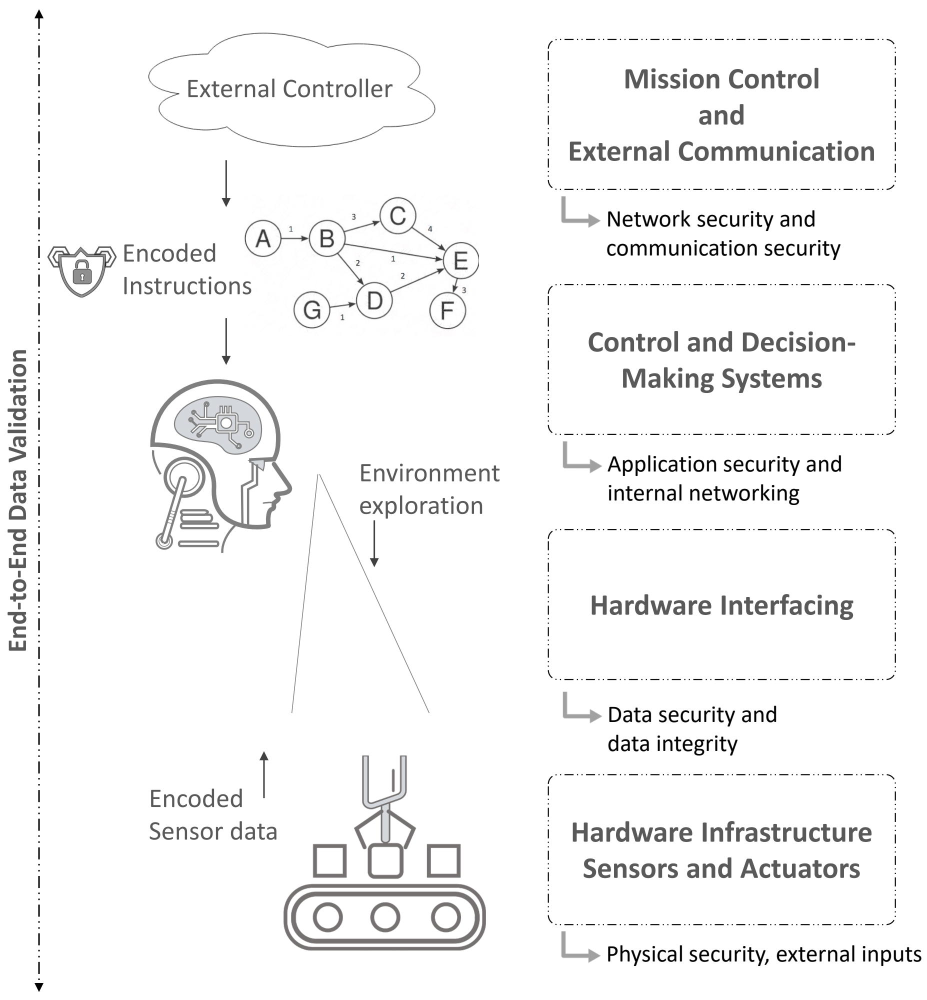

External Controller
Encoded Instructions
A
B
C
D
E
F
G
Environment exploration
Encoded Sensor data
End-to-End Data Validation
Mission Control and External Communication
Network security and communication security
Control and Decision-Making Systems
Application security and internal networking
Hardware Interfacing
Data security and data integrity
Hardware Infrastructure Sensors and Actuators
Physical security, external inputs

Figure 24. Classification of data acquisition and analysis processes in autonomous robots and matching security layers.

tine robots in cooperative missions within swarms of robots. The key objective is to improve the security and secrecy aspects of swarm robotics missions. The main novelty of their work is the introduction of a framework for validating data in robots without relying on the data itself, by encoding mission instructions in Merkle trees. Merkle trees are cryptographic structures that enable validation of data through cryptographic proofs that do not involve the data itself. In their approach, robots randomly navigate the environment because they have no knowledge about it. In the study presented in this chapter, we go beyond separating data verification from the data itself, towards defining new implicit ways of guiding robots without any direct instructions. Only by interacting with the environment can the appropriate instructions be decoded by the robot, in a process that serves as a simultaneous end-to-end validation of both sensor inputs and mission instructions. The methods introduced in this chapter can provide full mission instructions that ensure a robot is able to complete a task while keeping the almost-zero a priori knowledge about the mission objective and its steps. What is more important, the robot is not only able to validate the integrity of both the mission instruction and its sensor data, but also that it is actuating as expected as chained instructions are decoded sequentially.

A common aspect across different robotic frameworks is that robots are given raw information about their operational environment before the mission starts. For example, this is a common step in autonomous navigation processes. In the case of robots running SLAM, they accumulate raw data by themselves. Self-driving cars in dense urban areas, for instance, often rely on high-definition maps for localization by matching their sensor data with a part of the map [255]. This arises security concerns from two points of view regarding data integrity. First, if a robot sensor malfunctions or the data is tampered with, then wrong landmarks could be matched or the data could match a wrong part of the map. Second, if the map or landmarks given to robots are either corrupted or modified intentionally, then a robot might be tricked into following wrong directions. In a worst case scenario, all sensor inputs (either modified or not) could match wrong mission instructions and a robot could be navigating towards a different position without being aware of it. The same reasoning can be applied to other types of missions where a robot has to interact with its environment, such as assembly robots or healthcare robots.

4.1.1 Novelty

We aim at extending previous works into a more general framework focusing on encoding not only mission instructions but also the relationship between the possible ways in which a mission can be completed. In [272], one of the main research questions is whether it is possible to provide the "blueprint" of a robotic mission without describing the mission itself. In this chapter, we delve into the possibilities and limitations of encoding instructions, exploring different application scenarios and developing more specific ways in which this idea can be integrated into real robotic missions. In this work, we go beyond separating data verification from the data itself, as introduced in [272], towards exploring new implicit ways for defining complex robotic workflows. One of the key differentiating points with respect to the previous work is that the complete mission is encoded, whereas in previous works the robots would perform a known mission (e.g., exploration with random walks) and only specific actions are defined whenever the robots are able to reproduce predefined encoded sensor data. In addition, in this work we utilize a graph structure, which can be directed, to provide a specific mission flow and explicit topology on the connection between the possible mission instructions or states. Through the chap-

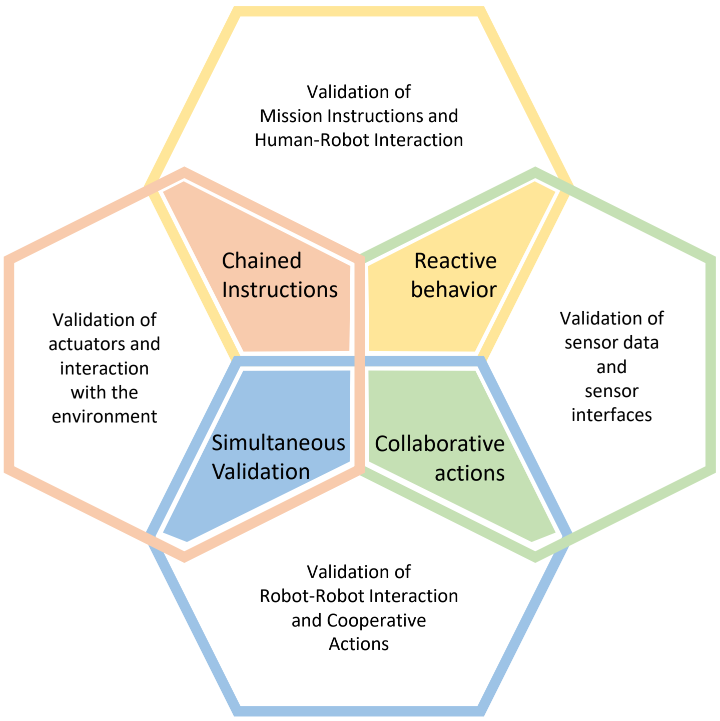


Figure 25. Our end-to-end validation framework can be utilized for single autonomous robots but also in human-robot and robot-to-robot cooperation.

ter, we first describe the framework and the different possibilities, from human-robot interaction to collaborative robots, and then provide a proof of concept with navigation instructions. This proof of concept demonstrates that traditional robot behavior can be maintained with minimal impact on robustness and applicability even when providing full mission codification, opening the door to more secure and safe deployment of autonomous robots.

The basis of the framework described in this chapter is that only by interacting with the environment can the appropriate instructions be decoded by the robot. This decodification process serves as a simultaneous end-to-end validation of both sensor inputs and mission instructions. The methods introduced in this chapter can provide full mission instructions that ensure a robot is able to complete a task while keeping an almost-zero a priori knowledge about the mission objective and its steps. What is more important, the robot is not only able to validate the integrity of both the mission instruction and its sensor data, but also that it is actuating as expected as chained instructions are decoded sequentially. We also show how this can be extended to interaction with humans or other robots. The potential validation processes are illustrated in Fig. 25.

We propose the utilization of one-way cryptographic hashes to encode a set of features or elements in the objective environment, which are then utilized as landmarks for exploration. This requires robots to know in advance the procedure for calculating the hashes, which is equivalent to robots being able to understand high-level mission instructions in a traditional deployment. By utilizing cryptographic hashes, we encode the landmarks in a way that no explicit information about the environment is exposed even if the data is accessed by a non-trusted third party. These hashes can be defined by extracting different features of the environment: from raw sensor data (e.g., objects with a certain color) to processed data (e.g., the output of a deep learning classifier), or extracting geometric properties of the environment (e.g., position and size of doorways or rooms). Additionally, the content of traditional landmarks can be encoded, such as the position of wireless localization beacons or the content of QR codes. Nonetheless, an important aspect to consider is that small changes in the hashed information can produce significant changes in the resulting hash. Therefore, the definition of the features must be done in a way such that hashes can be reproduced by robots with a certain noise tolerance. In this chapter, we discuss the different options for generating hashes and defining landmarks. Then, we utilize a certain set of geometric features that can be extracted from the environment utilizing range sensors (e.g., lidars) and provide experimental simulation results to validate the encoding scheme.

With the proposed approach, an external controller can provide new mission instructions to one or more robots even over an untrusted network, assuming that robots are aware of the procedure to calculate the hashes. This can be utilized, for instance, in remote areas or in order to enhance the security of the information if new instructions are provided for groups of robots in industrial warehouses at various locations from outside their local network. An additional requirement is that at least one of the positions in the navigation graph must be known to robots. In a cooperative exploration task, different robots can share the positions that they have visited by simply sharing the hashes. If, additionally, these hashes are generated by combining two or more other hashes (for instance, different properties of a single feature), then these can be shared instead so that robots can prove to their peers that they have visited the location and still not reveal explicit information.

4.1.2 Research questions

Summarizing the previous considerations, we have found an unexplored research gap in robust data validation schemes for autonomous or semi-autonomous robots. The main research questions that we find so far unanswered extend the work in [272] towards broader data validation in autonomous robots:

- Is it possible for a robot to safely and securely interact with its environment, operators or other robots in a way that it has zero a priori knowledge of the mission itself;

- can encoded information revealing no explicit mission instructions maintain the level of efficiency and applicability of current robotic workflows; and, finally,
- can these encoded mission instructions be utilized to simultaneously validate their integrity but also the progress of the mission, together with local information such as sensor inputs and operation of actuators.

The first of the above questions applies to a wide variety of situations. For instance, whether a robot in a factory can be given assembly instructions that it cannot understand until the assembly process starts, or whether a robot can be given a map that it can only understand when it starts to navigate its environment. An even more interesting situation occurs when a robot can interact with a human or another robot in a predefined way only when a series of conditions are met. The latter two questions extend the same concept towards more concrete aspects: whether these encoded instructions can be adapted to current robotic algorithms and workflows, extending previous works relying on random movement, and, finally, whether instructions understood by the robots can, *simultaneously*, validate that all processes are working properly. Thus, we are asking whether a single framework can define both end-to-end data validation and define encoded instructions that enable a robot to securely and safely interact with its environment.

These considerations cover the validation of a robot's operation from end to end: from validating sensor data and the correct operation of actuators to validating mission instructions and information received from an external controller. In our experiments, we consider a navigation or exploration mission where a priori information (features) about the environment is available to the mission controller or robot operator (operator hereinafter). In this scheme, an operator generates a set of encoded instructions by hashing the description of a set of landmarks (waypoints) in the environment. The set of encoded landmarks is then given for autonomous robots which utilize them to navigate the objective environment with zero a priori knowledge of it. The encoded landmarks are given to the robots in the form of a navigation graph which also includes information about how to navigate between consecutive landmarks. Because all information is encoded, we minimize the amount of raw data a priori exposed at the robots.

4.1.3 Contributions

The main contributions of this chapter are

1. the definition of an end-to-end validation framework for autonomous robots based on encoded instruction graphs;
2. the introduction of a novel approach that encodes a navigation graph utilizing cryptographic hashes to encode environment features, and
3. the definition of a set of methods that allow robots to follow encoded mission instructions while validating their sensor data without external feedback.

4.1.4 Chapter organization

In the next section, we focus on describing encoded instruction graphs for securing missions and enabling end-to-end data validation. We also delve into the different possibilities and application scenarios, describing how the encoded graphs can be defined for different situations or interaction modalities. We also explore the different opportunities in terms of data validation, and on how interaction with the environment or other robots creates different forms of validating instructions, sensor data and the robot's own actuators.

The second part of the chapter is then related to applying the aforementioned framework to a navigation problem in autonomous robots. We study two different scenarios in which (1) geometric and topological information is extracted from the environment to define the landmarks, and (2) markers are placed in the environment and utilized as waypoints for navigation. In the former case, the operator only needs minimal information about the geometry and topology of the environment. In the latter, the operator needs to place the markers in the environment before the mission starts. Through simulations and experiments with real robots, we show that these methods do not add a significant computational overhead compared to existing navigation methods while maintaining similar levels of accuracy and performance.

4.2 Background and related works

There has been a growing interest in securing robotic systems, partly owing to the increased connectivity with which robots are equipped. Indeed, there is an increasing amount of data exchange modalities with new attack vectors in remote control commands [279], telemetry [280], offloading computation [281], robot-to-robot communication [272], and human-robot interaction [282].

4.2.1 Security in robotics

Multiple research efforts have been devoted to studying cybersecurity issues in robotics. In [276], Clark et al. review and discuss the main security threads to robotic systems, from spoofing sensor data to denial of service attacks, and including other typical vector attacks such as malicious code injection or signal interference. However, this analysis only takes into account the security aspect in robotics from the cybernetic point of view, without considering the physical dimensions in which robots operate. The interaction of a robot with its environment presents key issues that cannot be addressed with traditional risk mitigation techniques from the cybersecurity domain. An early work in this direction was presented by Py et al. in 2004 [283], where the authors introduced an execution control framework for autonomous robots that would analyze the data obtained from the behavior of the robot through a *state checker*. A more recent work taking into account the nature of robotic operations was presented by Tang et al. [284], where sensor data was estimated through a denial of service attack. Similarly, in [285], Tiku et al. introduced a methodology for overcoming security vulnerabilities in a deep learning localization method by introducing adversarial training samples. All these approaches, however, take the point of view of data security in information systems and do not explicitly involve the cyber-physical nature of autonomous robots and their interaction with the environment, which is the objective of this chapter. In terms of distributed and multi-robot systems, most efforts have been directed towards the analysis and mitigation of security issues from a networking perspective [286].

From the point of view of data validation, Legashev et al. described an approach for monitoring, certification, and validation of the operation of autonomous robots [287]. The authors' aim was to define a generic framework from a legislative approach, relying on periodic telemetry data obtained from autonomous robots and, in particular, autonomous vehicles. The framework focuses on validating the robot's operation but not the data itself. The only method available to validate the data itself in this work is through statistical analysis and detection of statistical abnormalities. Data integrity was also the objective of Yousef's et al. study on cyber-physical threats on robotic platforms [288].

4.2.2 Indoor mobile navigation

In this chapter, we focus on demonstrating the applicability of the proposed approach to different indoor navigation methods. As such, we here review

Early works on indoor mobile robot navigation utilized landmarks in order to enable localization of the robot through longer missions. Lazanas *et al.* presented a method that allows for bounded navigation errors and localization uncertainties given a properly distributed set of landmarks along the path [289]. However, landmark-based localization with limited sensor capabilities introduces high sensitivity to noise. To cope with this problem, Madsen *et al.* provided an online localization algorithm that optimizes the selection of landmarks utilized for localization [290], assuming that there is a variable and large enough number of detected landmarks.

More recently, high-accuracy landmark-based localization has been possible with the utilization of wireless sensor networks [291], wireless or light-based beacons (IR, UWB, Bluetooth), and QR codes [292] or other identifiable images[293; 294; 295]. Nazemzadeh *et al.*'s approach to utilizing inertial and odometry data to increase the accuracy can be directly leveraged to improve the robustness of our proposed methods in terms of hash reproducibility [292]. It increases the probability of robots being able to match encoded features with positions in the navigation graph. One benefit of utilizing landmarks that are already encoding certain information, such as QR codes or other text representations, is that additional information can be embedded into the landmarks. In an industrial scenario, this can be utilized to provide further instructions for robots [296]. Wireless sensor networks or beacons can, on the other side, only provide location information in most cases. However, it is not strictly necessary to install certain infrastructure beforehand in order to enable robust navigation. For instance, Gadd *et al*'s work on infrastructure free navigation can be modified to provide an encoded topological map rather than storing raw features [297].

Recent advances in high-accuracy odometry methods show that robust autonomous operation is possible, specially in short distances, without the need for real-time mapping or a pre-built map [298; 131]. An estimation of a robot's motion can be obtained with the integration of inertial data [299]. Alternatively, different odometry methods can be applied based on non-inertial sensors. Visual odometry algorithms utilize feature extraction and tracking from cameras [300; 301], while lidar-based odometry uses mostly geometric information [133]. These can be leveraged in our approach for robust navigation between landmarks and ensure that the inter-landmark localization error is within the noise tolerance of the hashing scheme. In known outdoors environments, high-accuracy positioning can be achieved through differential GNSS measurements [302].

In the case of image-based localization, scale-invariant feature transform (SIFT) descriptors can be encoded with a certain resolution. Zamir *et al.* introduced a tree-based indexing scheme for SIFT descriptors to find the GPS location of images based on a dataset built from Google Maps Street View [293]. Additionally, the authors proposed the utilization of group-based localization with multiple images match where there was less confidence for individual images. If an encoding approach is integrated with this, the group-matching approach might be affected as the confidence measure would be instead whether hashes can be reproduced or not. Thus, there is a clear trade-off between flexibility and security. Rather than extracting features from individual images, Sattler *et al.* introduced an efficient and effective method for reconstructing scenes with an structure-from-motion approach [294]. In their work, a 3D-to-2D search was utilized to cope with matches lost due to quantization. If a method of this nature is utilized within an encoded navigation approach as the one proposed in this chapter, this search needs to be modified to be done after the one-way hashes have been calculated. More recently, Thoma *et al.* have shown the resurgence of image-based landmark definition for navigation and localization with the recent advances in feature representations for image retrieval [295]. The authors focused on generating a set of landmarks from a larger subset of images. This work can be directly integrated with the encoding approach presented in this chapter.

In terms of cooperative exploration, an early approach to multi-robot mapping was proposed in 2002 by Williams *et al* [303]. An adaptive methodology for collaborative localization and mapping was proposed with novel map fusion techniques with estimation of transformations between relative frames of reference. Michael *et al.* used multi-robots to realize collaborative 3D mapping after the Tohoku earthquake in 2011 [304]. Over the last two decades, more advanced algorithms have been proposed for map matching [305] and map merging in collaborative SLAM [306]. The benefits of collaborative mapping are clear and have been presented by multiple authors [307]. Of special interest to this work is Dedeoglu *et al.*'s work on a landmark-based matching for multi-robot cooperative mapping [308]. The author's approach to matching topological maps could be extended to matching a subset of hashes, in a situation in which robots would explore an unknown environment and map different parts separately.

4.2.3 Research gap and novelty

In general, we see a research gap in terms of addressing the physical dimension in robotic operation from a security and safety point of view. This becomes even more evident when analyzing the most widely used robotic frameworks. Among them, the Robot Operating System (ROS) has become a standard across both industry and academia. Multiple researchers have studied the security flaws of ROS [309], and proposed different approaches to address these issues [310]. Moreover, many of these are being mitigated in the newest version, ROS 2 [311]. However, these efforts are again mostly directed at securing ROS as a distributed and networked system, and not from the point of view of a robotic framework meant for robots to interact with their environment. While it is highly important to provide security from the data flow point of view, we direct our efforts in this chapter towards the gap in securing and validating the way robots are controlled and interact with their environment.

In this chapter, we focus on providing a framework for validating data integrity. Other types of cyberattacks such as denial of service attacks, in which the communication channels are congested, are not considered. Nonetheless, it is worth noting that our proposed approach can provide some benefits even in such situations. While the communication channel utilized to transmit the encoded commands to the robot might be known to an attacker, the sensor data or inputs triggering the different actions are unknown even to the robot itself, if multiple possibilities exist. Therefore, our framework also provides partial mitigation for other types of cyberattacks where the channel utilized to trigger a robot's actions might be disguised within the encoded instruction graph sent to the robot before the mission starts.

4.3 Encoded instruction graphs framework

This section covers the main framework components and how it can adapted to different application scenarios from a design perspective, before delving into a specific use case in the remainder of this chapter.

4.3.1 Encoding robotic instructions

We follow up on the instruction encryption ideas from [272], where missions instructions are given to a robot by encoding combinations of sensor inputs and a set of robot actions. We do not consider explicitly multi-robot cooperation at this point but instead focus on describing robust options for encrypting and decrypting mission instructions. Moreover, we also evaluate the performance degradation inherent to encrypting data when compared to standard robotic operation, avoiding random behavior. In order to do that, the first step is to not only encode a set of actions and features from sensor data, but also at least one more variable that enables a *hash search*, a trial-and-error process in which a robot does not need to be able to reproduce a specific hash but instead can try multiple hashes until finding a match. An example of this is the addition of a spatial or temporal component. The second step is then to define not only a series of encoded actions but also encoded states, which can be defined based on a combination of variables (e.g., position, time, sensor data or other external inputs). By encoding both states and actions, we can then wrap the set of encoded information into a graph structure, such that the encoded information in an edge of the graph gives the robot information about how to arrive to a different state or process. We call this an encoded instructions graph. A sample encoded instructions graph is shown in Fig. 26. In this graph, the initial instruction is encoded into a hash H_1, which the robot is able to decode by combining a predefined action (e.g., movement in one specific direction) with equally predefined sensor data (e.g., visual or geometric features extracted from the environment). Most of the nodes in the graph represent states, with the majority of the nodes in this example being defined by a position and a specific feature defined a priori on the basis of specific sensor data (e.g., predefined landmarks). However, the information that nodes encode is not restrictive, as it can be both actions or states. The edges, however, should encode information that enable the robot to transit between nodes, and therefore should include one of the robot's possible actions or an external input, such as a message from a controller or another robot. An edge can also be empty, i.e., if the robot can gather enough information to decode a different node without any intermediate step. In the example in Fig. 26, the initial node triggers an action by the robot whenever it is able to acquire sensor data that reproduces a hash in the graph. It can then proceed to two different states in which it must to be able to reproduce both its position and sense a different variable. The encoded information in the second node, H_2, can then be decoded after a certain time, which can for example be used as a failsafe to go back to the initial state if the sensor data defining H_3 cannot be acquired in time. In practice, the robot is not aware of the type of information encoded in each hash, and must therefore perform a continuous trial and error process to try to reproduce the hashes by all the different means it has been preprogrammed for. In our experiments, we show that this process of trial-and-error has a mostly negligible impact compared to the computational cost of extracting features or processing sensor data in standard robotics algorithms. In any case, the process of deciding how to define the encoded instructions is not trivial, as they must be reproducible, while concise enough to avoid data mismatches.

An encoded instruction graph as described above is a directed graph. For a mission that has only one possible solution and where each step is followed by one and only one other step, then the graph is reduced to a path or linear graph. In many cases, the graph will be acyclic. This happens when there are multiple options to complete a mission, but once a set of steps is taken then the same ones cannot be taken again. For instance, in a manufacturing process, the order in which a set of parts are moved to a working bench might not matter, yet every part must be moved exactly once. In a general case, the graph can be arbitrarily complex and contain any number of cycles. For instance, a reconnaissance mission in which a robot has to navigate an objective area without any specific order could have multiple cycles. Higher mission control embedded at the robot should then be able to understand these more complex graphs and provide a planning strategy that is not directly sent by the mission controller. An example of this will be given in our experiments.

4.3.2 Validation modalities

The proposed approach can be extended to multiple scenarios, as the encoded information cannot only be a set of predefined actions and features extracted from sensor data, but also other external inputs, variables defining the state of the robot, or even timing constraints. The possibility of utilizing external inputs is particularly interesting as it enables secure and secret multi-robot cooperation but also new ways of defining under which conditions human-robot interaction can happen. Since the inputs can already be encoded, the information exchanged between robots or utilized as external signals triggering robot actions can be defined in a way that they are totally meaningless and only usable when combined with other data. Therefore, if the data is spoofed or a third party gains access to this communication medium, no real data is actually compromised. Different possibilities for encoding and decoding instructions are shown in Fig. 27. In the figure, we show the different approaches to encoding a robot's interaction with its environment. From left to right, the options include (i) simply encoding a predefined action, (ii) combining actions with a timestamp when they should be carried out, (iii) actions triggered by specific sensor data

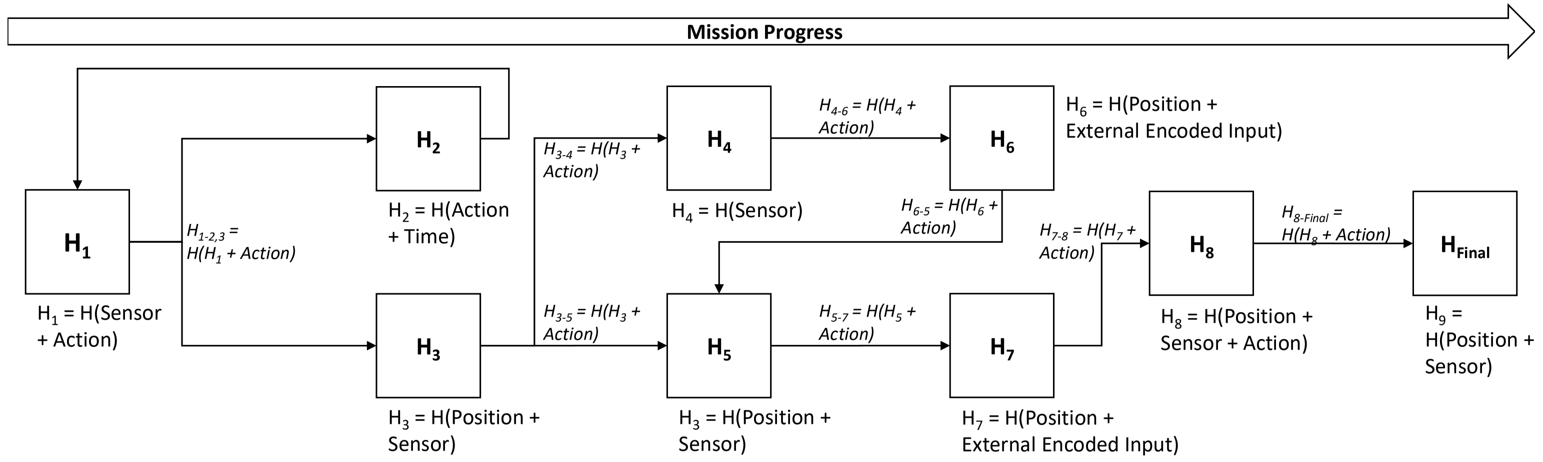


Figure 26. Sample encoded instruction graph, with hashes being defined from different environmental information, or combinations of sensor data, localization data, and one or more of the available actions. The mission can be completed successfully only if the hashes are decoded sequentially. Note that there is not necessarily a single way of progressing in the mission.

obtained from the environment, and (iv) the combination of all the previous. The actions can then be chained including the previous hash or action within the next codified node in the graph. These approaches are described in more detail and put into context in the following subsections:

Independent validation

The simplest approach is to encode mission instructions individually and independently. In this case, an encoded instruction set can be sent to the robot, similar to the approach followed in [272] but where only the leaves of the Merkle tree would be sent. This set of instructions does not define a graph structure and does not represent the main interest of this chapter. However, these instructions can be utilized as the root of an encoded instruction graph, or as a trigger for starting a parallel process at the robot.

An additional layer of security can be added by introducing time or spatial constraints. Time constraints (e.g., introducing a timestamp in the hash) provide an extra layer of security against attacks that could spoof the encoded data and reproduce it later, even if the data itself cannot be decoded. Similarly, spatial constraints can be added by including the robot's location in the hash. This, however, only prevents the replication of the robot's behavior in other locations.

Iterative validation

An iterative validation happens when a robot is able to validate its own actions. This particular modality will be the case studied in the next sections with the introduction of an encoded navigation graph.

Encoded instruction graphs defining an iterative validation process can contain different types of encoded data in their nodes and edges. For instance, sensor data can be encoded in the graph nodes, which serve the purpose of validating the process. Additionally, this sensor data can encode other information, such as positional information or time information that can be utilized as a part of the control loops at the robot. Then, the actual instruction for the robot to move towards the next step is encoded in the edge of the graph, which encodes both the data in the current node being validated and the action or actions that will enable the robot to decode the next node. An illustration of this process is shown in Fig. 28. In the figure, we show how, at each step, instructions can be validated simply by being able to reproduce the corresponding hash. Moreover, as actions accumulate leading to new reproducible states (defined through chained hashes in the encoded instruction graph), we are able to iteratively validate the actions confirming their output with an expected outcome.

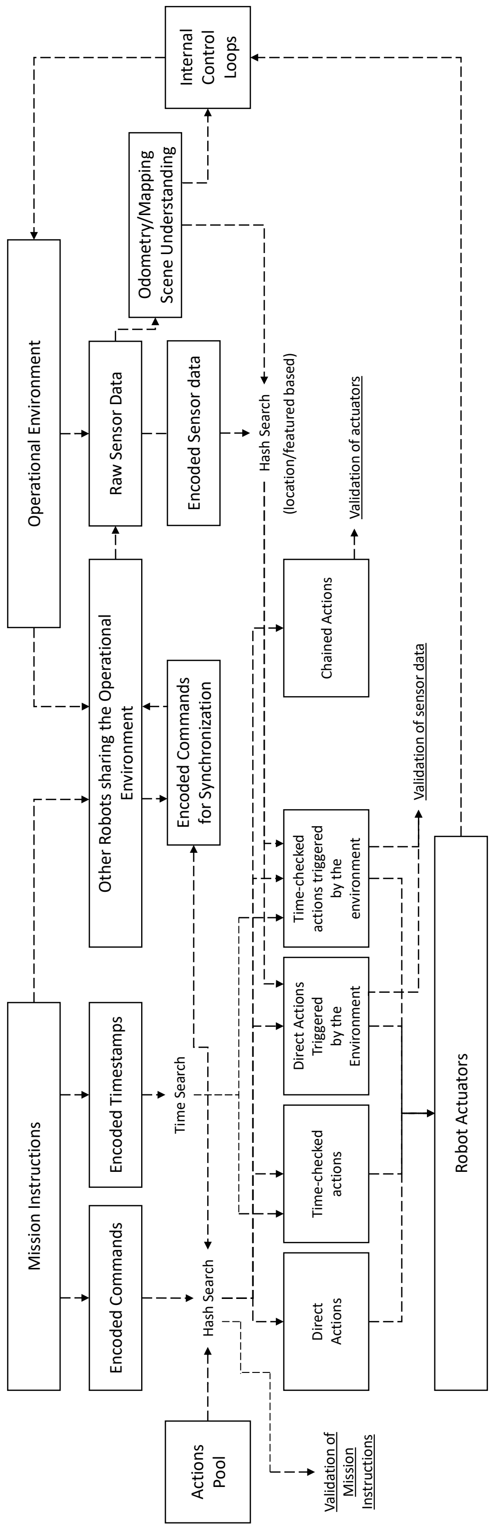

Actions Pool
Validation of Mission Instructions
Mission Instructions
Encoded Commands
Encoded Timestamps
Hash Search
Time Search
Direct Actions
Time-checked actions
Direct Actions Triggered by the Environment
Time-checked actions triggered by the environment
Robot Actuators
Validation of sensor data
Chained Actions
Validation of actuators
Other Robots sharing the Operational Environment
Encoded Commands for Synchronization
Operational Environment
Raw Sensor Data
Encoded Sensor data
Hash Search
(location/featured based)
Odometry/Mapping Scene Understanding
Internal Control Loops

Figure 27. Different validation modalities and data flows. The same approach can be utilized for individual mission instructions, event-based commands, chained instructions, or multi-robot communication.

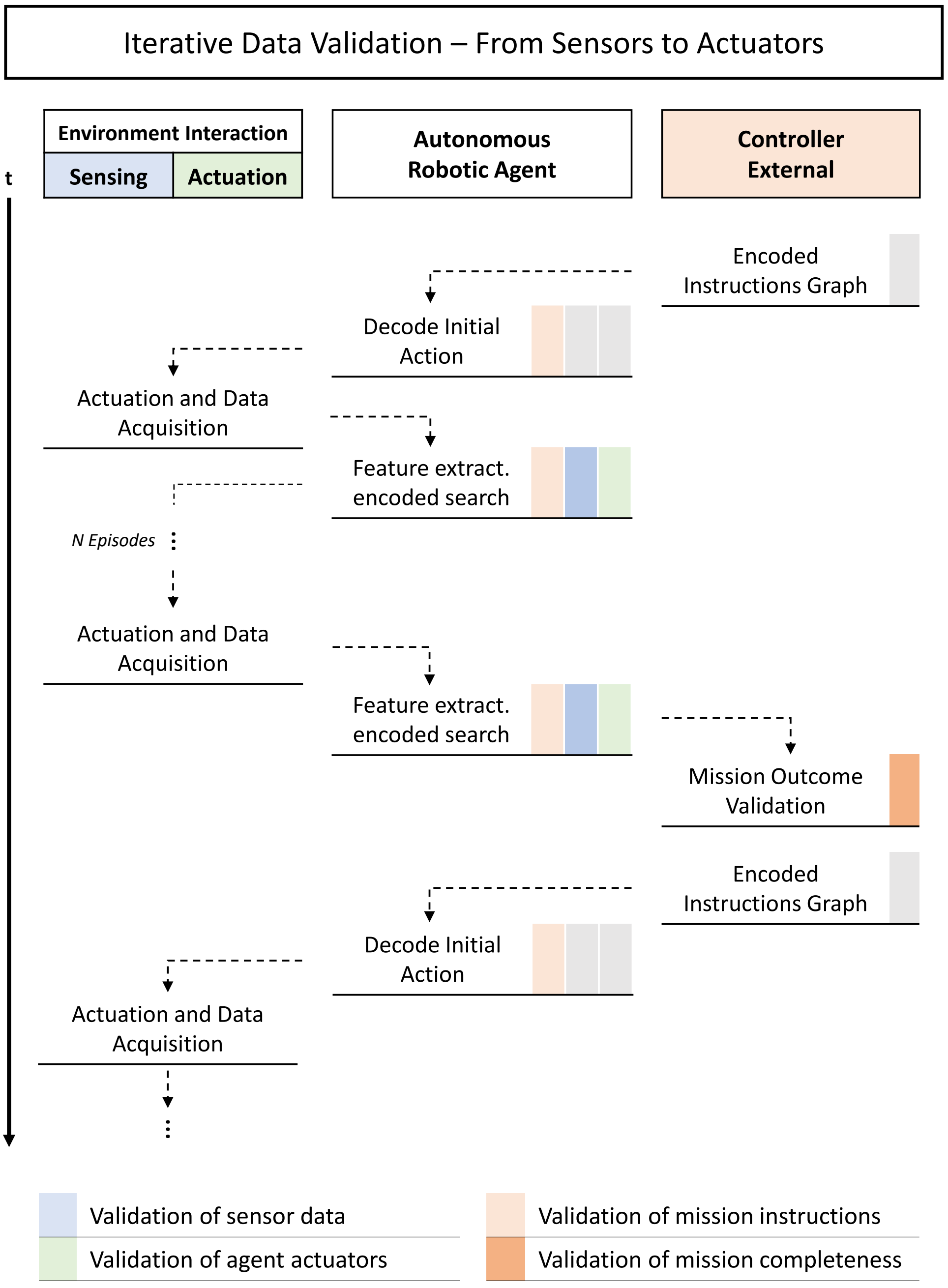

Iterative Data Validation – From Sensors to Actuators
Environment Interaction
Sensing
Actuation
Autonomous Robotic Agent
Controller External
t
Encoded Instructions Graph
Decode Initial Action
Actuation and Data Acquisition
Feature extract. encoded search
N Episodes
Actuation and Data Acquisition
Feature extract. encoded search
Mission Outcome Validation
Encoded Instructions Graph
Decode Initial Action
Actuation and Data Acquisition
Validation of sensor data
Validation of agent actuators
Validation of mission instructions
Validation of mission completeness

Figure 28. Illustration of an iterative validation process.

Multi-robot simultaneous and mutual validation

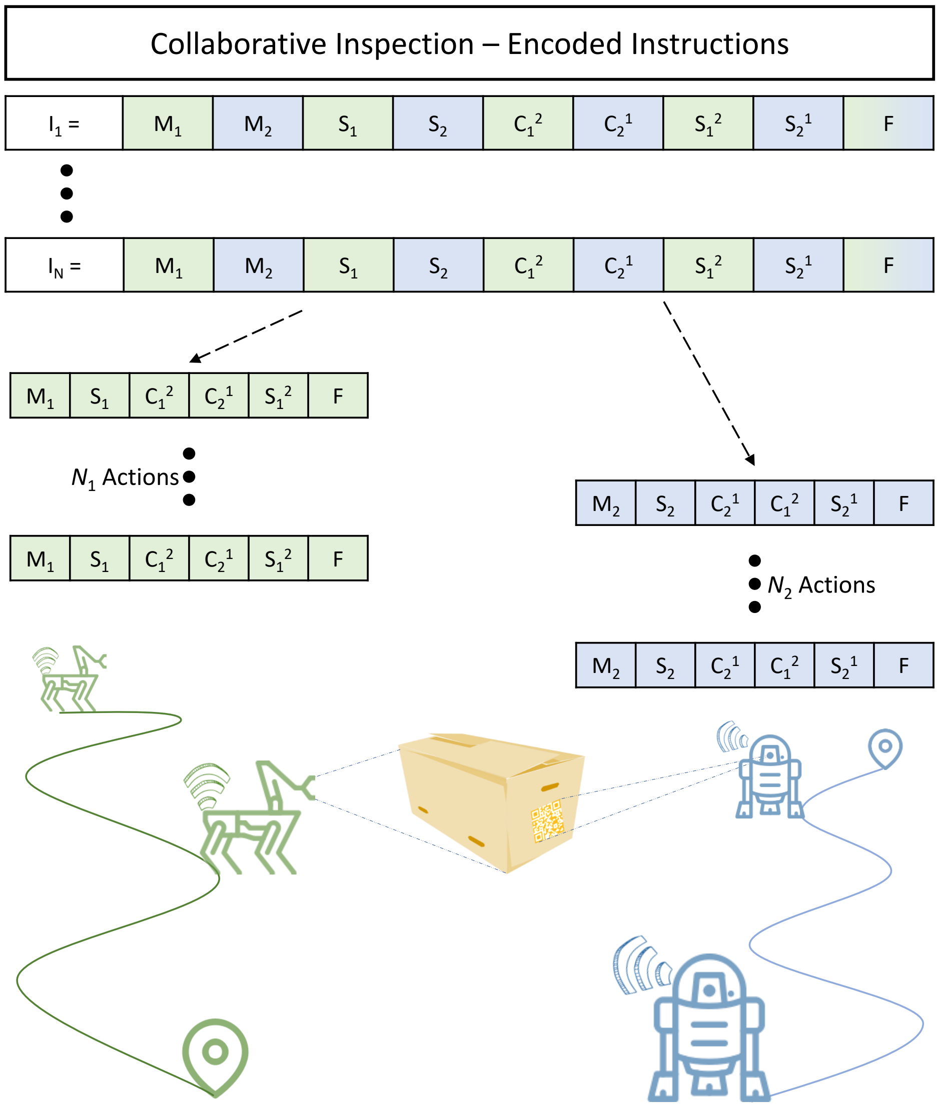
As we have mentioned earlier, one significant scenario where this validation framework can be applied is in multi-robot cooperative behavior. Complementing the ideas proposed in [272], we are now also able to break down a mission in two disjoint parts that can be given to two different robots. An example of this is shown in Fig. 29, which illustrates a collaborative inspection process. In the example in Fig. 29, an encoded instruction is defined by combining a set of different signals or parameters. First, data triggering an action is set from features extracted by the robot from its own sensor data. In addition, once multiple robots share the same operational environment, we can now also differentiate between hashes obtained from sensing the environment and those obtained from sensing the behavior of other robots. Second, the two or more robots can also exchange messages in order to trigger each other's actions. These messages do not hold any valuable data to the sender but only to the receiver as part of a hash decoding process. The messages can be predefined and based on the robot state, or generated as a function of the features sensed in the environment.

4.4 Use case: encoded navigation graph

One of the most fundamental ways in which a robot interacts with its environment is by navigating it. Maps have long been utilized for autonomous navigation and exploration in mobile robots to increase the robustness of long-term autonomous operation [312; 313]. Maps or landmarks provide robots means for localization in a known reference frame, while enabling the calibration and adjustment of on-board odometry and localization algorithms.

Maps can be either given to robots, or built by themselves through a simultaneous mapping and localization (SLAM) approach [133; 300; 314], and allow robots to continuously localize themselves within a global reference system through different map matching techniques, such as iterative closest point (ICP) [315], perfect match (PM) [316], or normal distribution transforms (NDT) [317] algorithms.. Landmarks, on the other side, enable robots to utilize odometry methods for short-term localization and correct their position when a landmark is identified [290; 318].

Landmark-based navigation has been successfully implemented in various mobile robots with quick response (QR) codes [319; 320; 292] or other identifiable images[293; 294; 295], wireless sensor networks [291], or ultra-wideband (UWB) markers [19; 170; 321], among others including IMU fusion [292], or topological maps in for infrastructure-free navigation [297]. When utilizing landmarks that are already encoding certain information, such as QR codes or other text representations, additional information can be embedded into the landmarks. In an industrial scenario, this can be utilized to provide further instructions for robots [296].


Collaborative Inspection – Encoded Instructions
I_1 = M_1 M_2 S_1 S_2 $C_1{}^2$ $C_2{}^1$ $S_1{}^2$ $S_2{}^1$ F
I_N = M_1 M_2 S_1 S_2 $C_1{}^2$ $C_2{}^1$ $S_1{}^2$ $S_2{}^1$ F
M_1 S_1 $C_1{}^2$ $C_2{}^1$ $S_1{}^2$ F
N_1 Actions
M_1 S_1 $C_1{}^2$ $C_2{}^1$ $S_1{}^2$ F
M_2 S_2 $C_2{}^1$ $C_1{}^2$ $S_2{}^1$ F
N_2 Actions
M_2 S_2 $C_2{}^1$ $C_1{}^2$ $S_2{}^1$ F

M_i	Encoded instruction to robot *i*: encoded action + triggering data (S_i / S_i^j / C_j^i/ F).
S_i	Sensor data triggering instruction M_i, describing the operational environment.
S_i^j	Sensor data at robot *i*, describing robot *j*'s actions.
C_i^j	Encoded message that robot *i* sends to *j* to trigger action M_j.
F	Indicates whether the action is triggered directly at either robot (first action).

Figure 29. Illustration of a collaborative inspection process where robots only have partial instructions.

In order to analyze the viability of this idea and discuss the potential directions for solving the research questions defined in Section I of this chapter, we consider the most essential aspect of a robot's interaction with its environment: *the navigation*. Therefore, we capsulize the research questions to more concrete considerations regarding the navigation of autonomous robots:

1. Is it possible to provide a description of the environment (e.g., a map or a set of landmarks and how to travel between them) to an autonomous robot, in a way that the robot is unable to understand the map until it starts navigating, and such that it can only decode the information in that map if a series of conditions on how it sees its environment are met?

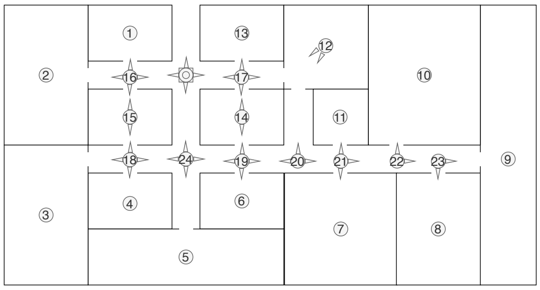
2. Is there a way of defining navigation instructions for an autonomous robot such that any modification of those instructions automatically renders them unusable ensuring that if wrong sensor data is fed to the robot's controller, the instructions cannot be followed?

4.4.1 Encoded graph definition

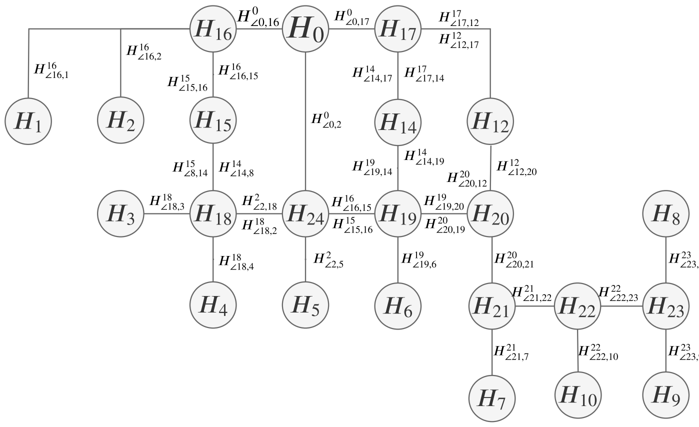
Rather than modeling a map of the objective exploration area and utilize it for navigation, we utilize a landmark-based navigation graph that encodes the position of the different landmarks and the navigable directions between landmarks. In this graph, each vertex represents one encoded position in the map, and each edge represents a straight or unique path between two positions. By unique we mean a path that might not be straight but such that the robot can realistically follow. A sample map and the corresponding encoded navigation graph are illustrated in Fig. 30. In this and latter sections, we utilize the following notation. A graph is an ordered pair $\mathcal{G} = (V, E)$, where V represents a set of vertices, and E represents a set of edges associated with two distinct vertices, i.e., a set of tuples $\{(V_i, V_j) \mid V_i, V_j \in V\}$. We consider a directed graph, were the order of these tuples matters.

The most straightforward approach to landmark encoding is to define the hash of a position given its coordinates $\vec{r} \in \mathbb{R}^3$. Thus we would define $H_i = H(\vec{r}_i)$. In order to ensure that hashes will be reproducible, the coordinates need to be given in a coarse grid with a resolution that is dependent on the accuracy of the robots' onboard odometry.

If the environment is accessible a priori, elements can be installed that facilitate the localization of robots when they are nearby, such as QR codes, or Bluetooth/UWB beacons. The QR codes contain hashed data and can encode additional information, for example, instructions for a robot to operate in a given room or area. An alternative approach is to utilize the environment geometry and topology. The coordinates of the features can still be utilized to define their hash without using a



(a) Sample floorplan.



(b) Corresponding navigation graph to floorplan in (a).

ID	H_N (256 bits)
0	H_0 = 5341dfa945ca9e52334 … 8446048
1	H_0 = a6340c2ed22ff55a475 … c3e92b4
$\vdots$	
24	H_{24} = 2522cfaa21faaa45a … d9de50a

(c) Landmark hashes given to robots. The hashes are calculated based on the position (x, y, z) and the landmark type (LT): $H_i = H(LT_i + x_i + y_i + z_i)$, where H is the hashing function and $+$ means concatenation.

$$\begin{pmatrix}
0 & 0 & 0 & \dots & 0 & H^0_{\angle 0,16} & H^0_{\angle 0,17} & \dots & H^0_{\angle 0,24} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
H^{16}_{\angle 16,0} & H^{16}_{\angle 16,1} & H^{16}_{\angle 16,2} & \dots & H^{15}_{\angle 16,15} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \dots & \dots & 0 & 0 & \dots & H^{24}_{\angle 24,0}
\end{pmatrix}$$

(d) Adjacency matrix with the edge hashes to aid navigation between landmarks. The edge hashes are $H^i_{\angle i,j} = H(LT_i + x_i + y_i + z_i + \angle i, j)$, where $\angle i, j$ represents the navigation direction from i to j.

Figure 30. Encoded navigation graph construction process.

predefined grid. Rather than having a robot utilizing its own or near position to calculate the hash, it can calculate it based on the coordinates of a position that depends on the robot's current local environment.

4.4.2 Deployment and navigation

We assume that the position where a robot is deployed is either known in an absolute reference frame, or utilized as a common reference in the robots' local coordinate system. If only local references are utilized, these must have a common orientation. The initial position is encoded with a hash but also known to robots.

In the encoded navigation graph, each edge in the graph is given two hashes, as the robots might reach these from different directions. Therefore, the adjacency matrix containing the edge hashes shown in Fig. 30 is not symmetric. Only minimal information about the local environment required for navigation purposes needs to be stored at the robots. Odometry-only (map-free) navigation, when possible, would be preferred to minimize the amount of raw information that robots store.

The directions between features, or the initial position and near features, is encoded in a way that can be matched by robots on a basis of trying multiple possible directions until finding one that produces the corresponding hash. The edge hashes are calculated on a trial-and-error basis, and thus they can be defined with an arbitrary division of the $[0, 2\pi]$ interval. However, this decision must take into account the trade-off with the inherent computational overhead. Furthermore, not all the navigable directions are necessarily selected, and therefore the real topology of the objective environment can be, to some extent, hidden. In addition, multiple features can be selected within a single room or small area, but even if all are detectable at the same time, a fully connected subgraph does not need to be generated within the navigation graph. In general terms, there is a trade-off between the number of actual connections between features that are encoded in the navigation graph, and the robustness of the navigation in the event of robots not being able to reproduce a certain subset of hashes.

4.4.3 Landmark-based localization

The accuracy of the feature's position directly affects the error tolerance for the odometry method utilized for navigation when no landmarks are detected. In order to cope with the odometry error, if it can be estimated then it can be taken into account to calculate the hashes from the position of landmarks, following a trial-and-error approach within a certain spatial area around the landmark. The number of trials that a robot needs to perform depends on the accuracy of the odometry method utilized, and the granularity of the grid utilized to define the position of the landmarks and calculate the hashes. Additionally, the possibility of the robot identifying a wrong landmark that is nearby must be taken into account. Therefore, there is a trade-off between accuracy and robustness with multiple factors to take into consideration.

4.5 Navigation graphs: methodology

In order to test the feasibility of the encoded navigation approach presented in this chapter, we have run a series of simulations and experiments. In these, we analyze the overhead and performance impact from calculating hashes and utilizing the encoded navigation graph. In all cases, we make the assumption that the environment is known to the mission controller.

The nature of the information to be hashed can vary significantly depending on the operational environment of the robots, and how accessible it is to the mission controller. The main bottleneck is in defining hashes from a selection of features that robots can identify with high probability. Because hashes have to be reproducible, features have to be constant over time, and be identifiable in a variety of conditions affecting either sensors or actuators. These can range from environmental conditions affecting sensors to taking into account the different possibilities for traversing a given room through various paths. Based on these considerations, we have devised two types of application scenarios in which we test the proposed framework.

First, we consider an environment where only robots operate. In this case, we have simulated the interior of a building with empty rooms. For this scenario, we utilize a simulation environment, in which we encode geometric features in the navigation graph: doorways, corners and rooms. For the simulations, we consider a fully automated environment with no dynamic obstacles and known geometry. This can be applied, for instance, to logistic warehouses where only autonomous robots operate. It can also be applied to autonomous cleaning machines operating at night, or, in general, any scenario where the environment does not change significantly over time. In our simulations, the robot relies on a two-dimensional laser scanner for feature detection.

Second, we consider a real office setting with a dynamically changing environment, people moving in it and a wide variety of objects populating the different rooms. The experiments are carried out relying on visual markers that can be placed in multiple fixed locations. For this application scenario, real experiments are carried out in an office environment with people and a variety of furniture across different rooms. Because of the large amount of desks, chairs and other equipment, detecting geometric features from the environment would render irreproducible hashes and multiple situations in which features can not be detected due to either objects or people blocking the field of view of sensors. To tackle this issue, we have utilized QR codes as markers to encode the landmark positional information.

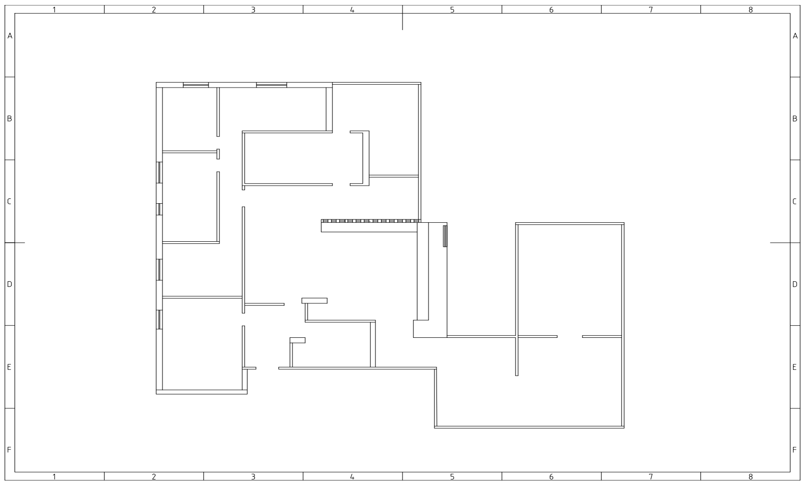


Figure 31. Floorplan of the simulation environment. All doors, rooms and corners are utilized as encoded landmarks.

4.5.1 Simulation environment

The proposed encoding approach has been implemented within the Robot Operating System (ROS) in Python. ROS is the current de-facto standard for production-ready robot development [234; 322]. The simulations were carried out within the ROS/Stage environment. A TurtleBot 3 is simulated with a 2D lidar and wheel odometry. The robot is set to explore an indoor environment with a floorplan illustrated in Fig. 31. The environment is $40 \times 40\,m^2$, and the robot has a circular shape with a diameter of $0.35\,m$. The simulated environment contains 9 rooms with a single entrance and 6 more spaces and corridors in between. The starting exploration position of the robots is near the main door, in the bottom-left. The 2D lidar has a field of view of 270° and produces 1080 samples (0.25° resolution) in each scan, with a scan rate of 10 Hz. This work presents a proof of concept, and therefore we do not study the effect that different odometry methods have in the exploratory mission. Instead, we utilize wheel odometry and vary its error to study the impact that the corresponding computational overhead has due to a larger number of hashes being calculated.

Feature extraction

In the simulation experiments, we utilize three types of features to localize the robot and navigate the environment: doorways, concave corners and rooms. These are defined from the same set of F feature points which we denote as Features of Interest $FoI = \{fp_1, \ldots, fp_F\}$, where $fp_i \in \mathbb{R}^3$. The feature extraction process is

Algorithm 1: Feature Extraction and Hash Calculation

```
Callback:
    Calculate:
        $\mathbf{F} = getF(data)$;                                  // Orientation-ordered $F$ set
        $\mathbf{F}_{cv} = getCv(\mathbf{F}) \subseteq \mathbf{F}$;         // Set of concave features
        $\mathbf{F}_{cc} = getCx(\mathbf{F}) \subseteq \mathbf{F}$;         // Set of convex features
    Define:
        $\mathbf{H} = []$;                                          // List of hashes
    foreach $fp_i,\ fp_j\ \in\ \boldsymbol{F}_{cv}$ do
        if $\|fp_i - fp_j\| < \delta_{dw}$ then
            $H.append(doorwayHash(fp_i, fp_j))$;
    foreach $fp_i\ \in\ \boldsymbol{F}$ do
        if $fp_{i+j} \in \boldsymbol{F}_{cx}\ \forall j \in \{-1, 0, 1\}$ then
            $H.append(roomHash(fp_{i\text{-}1}, fp_i, fp_{i\text{+}1}))$;
    foreach $fp_i\ \in\ \boldsymbol{F}_{cv}$ do
        if $isCorner(fp_i)$ && $notDoor(fp_i)$ then
            $H.append(cornerHash(fp_i))$;
    // Utilize any matching hashes to update the robot's
    // position with respect to the global reference frame
    if $\exists\, h \in\ H \mid h \in\ NavGraph$ then
        $updateAbsolutePosition(H))$;                           // Use matching hashes
```

outlined in Algorithm 1. The $NavGraph$ variable stores a list of hashed positions as well as an adjacency matrix with the edge hashes. A sample of this is shown in Fig. 30, subfigures (c) and (d). The function $search()$ calculates a certain number of hashes over a predefined area around the identified feature until it either finds a matching hash from $NavGraph$ or ends the search unsuccessfully. This function ensures that the hashes are reproducible even if odometry error accumulates over the inter-landmark navigation. The search area is defined based on the expected odometry error as well as the granularity of the grid utilized to define the hashes. Finally, the function $updateAbsolutePosition()$ takes the matching hashes as arguments, calculates the relative position of the robot with respect to the landmarks that have been identified and utilizes the known position of the landmarks (which is encoded in the hashes) to recalculate its own position and restart the odometry estimation.

Doorways

We define doorways as any set of two concave feature points that are within two predefined distances $(\delta_{dw,min}, \delta_{dw,max})$ from each other. In our simulations and experiments, we set these distances to $\delta_{dw,min} = 1.2\,m, \delta_{dw,max} = 2.5\,m$. Note that these feature points might not be consecutive if we consider the ordered set of feature points by orientation. We define the corresponding waypoint to be encoded according to (1):

$$H_{dw}(fp_i, fp_j) = H\left(\text{"doorway"}, \frac{fp_i + fp_j}{2}, \angle fp_i fp_j\right) \tag{1}$$

Corners

For each concave corner not in a doorway, we define its corresponding hash with (2):

$$H_{cv}(fp_i) = H\left(\text{"corner"}, fp_i, \angle fp_i\right) \tag{2}$$

where $\angle$ now represents the orientation of the normal vector to the wall surface at the position of the corner.

Rooms

A room waypoint is defined as the centroid of any three consecutive convex points, calculated as the arithmetic mean of their positions. To reduce the probability of having a mismatch in rectangular rooms where two consecutive subsets of three convex corners are visible by a robot, we add the area Δ of the triangle that the points define:

$$H_{dw}(fp_i, fp_{i+1}, fp_{i+2}) = H\left(\text{"room"}, \frac{fp_i + fp_{i+1} + fp_{i+2}}{3}, \Delta_{i,i+1,i+2}\right) \tag{3}$$

4.5.2 Real-robot experimental settings

The experimental environment has a size of 30 m by 25 m. For the experiments, an EAIBOT DashGo D1 has been utilized. We have installed on the mobile robot a 16-Channel Leishen 3D Lidar, an SC-AHRS-100D2 IMU, and a Logitech c270 USB camera. The DashGo platform also provides wheel odometry from its differential drive system. The 3D lidar is utilized to accurately localize the landmarks and provide ground truth with map-based localization algorithms for three-dimensional point clouds introduced in [255], with the ground truth trajectory shown in Figure 32.

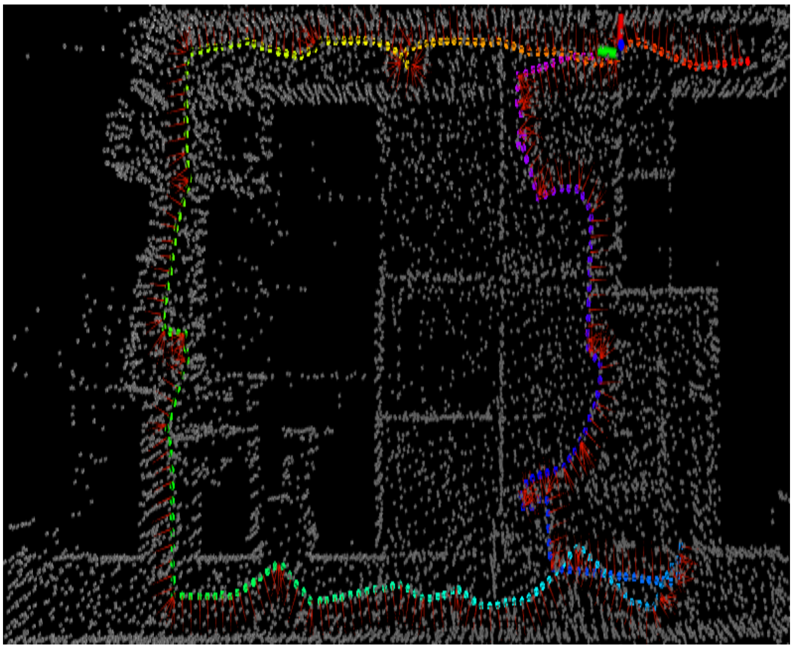


Figure 32. 3D point cloud (bird's eye view) of the experiment environment utilized as ground truth in our experiments. The path of the robot is shown with colored points, where red represents the start of the mission and purple the end.

The camera is utilized to detect the QR codes and extract the encoded information in them. Figure 33 show the implementation diagram with different ROS nodes. The 3D lidar odometry and mapping are adapted from the LeGo-LOAM-BOR package [323]. The QR code decoding node has been written in Python using OpenCV and the Zbar library. The hash based localization node utilizes the QR codes for localization when available and the wheel and inertial odometry as an estimation between landmarks. The QR codes utilized during the experiment are of known size (12 cm by 12 cm), and the localization node has been calibrated to map the size in pixels of a detected QR code in the camera to the distance to it. The localization also takes into account the relative orientation of the QR code.

4.5.3 Feature hashing

We utilize SHA3-256 for hashing [324], which generates 32 byte hashes. It takes an average of under 500 ns on an Intel(R) Core(TM) i5-6200U CPU with the pysha3 implementation in Python. If additional security is required against offline attacks on exposed hashes, other hashing algorithms such as Bcrypt [325] can be utilized. Bcrypt needs around 300 ms to generate a hash with the same CPU. However, there is a trade-off between security and real-time operation as robots need to calculate

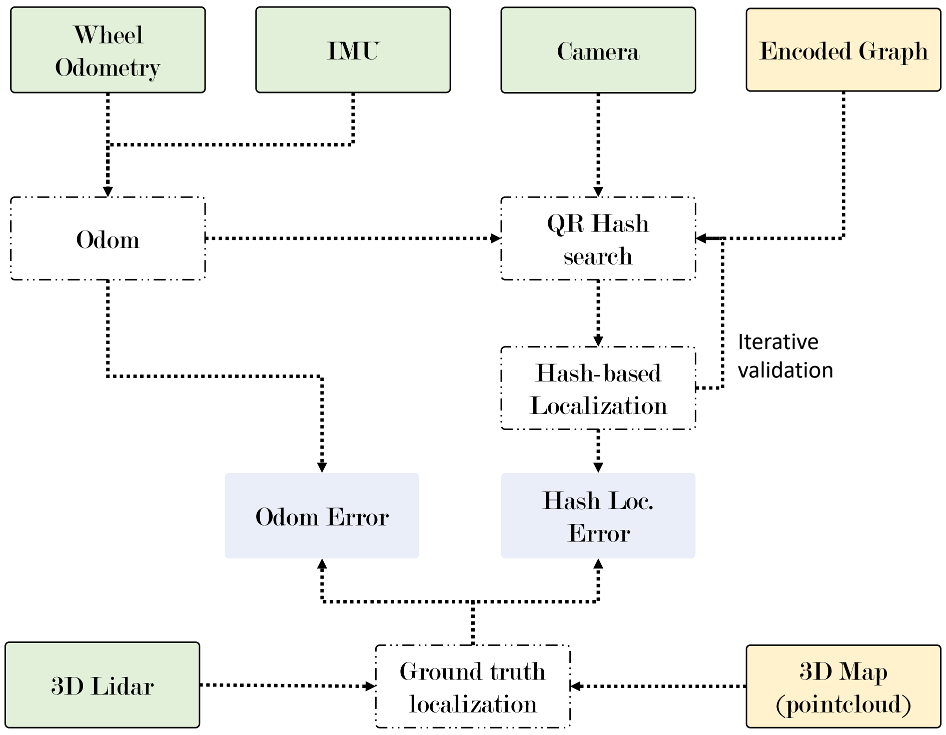


Figure 33. Data flow in the experiments. Each box represents a ROS Node which has been implemented either in C++ or Python. The outputs are the ground truth, odometry (odom) error and has-based localization error (loc. error).

multiple hashes per lidar scan. We believe that, in most applications, SHA3-256 is enough and can be utilized even in resource-constrained devices.

4.6 Simulation and experimental results

We have carried out a series of simulations with one and multiple robots to evaluate mainly the cost of utilizing hash matching for localization and navigation, but also the impact on accuracy of the encoded landmarks.

4.6.1 Metrics

In order to evaluate the simulation results, we measure the absolute localization error of the robot with odometry only and hash matching. Furthermore, we analyze the distribution of the computational load among the different tasks that the robots are carrying out: feature extraction, hash calculation and hash matching. In the simulations, we also measure the effect of the odometry-based localization noise and the choice of spatial granularity for landmark positions.

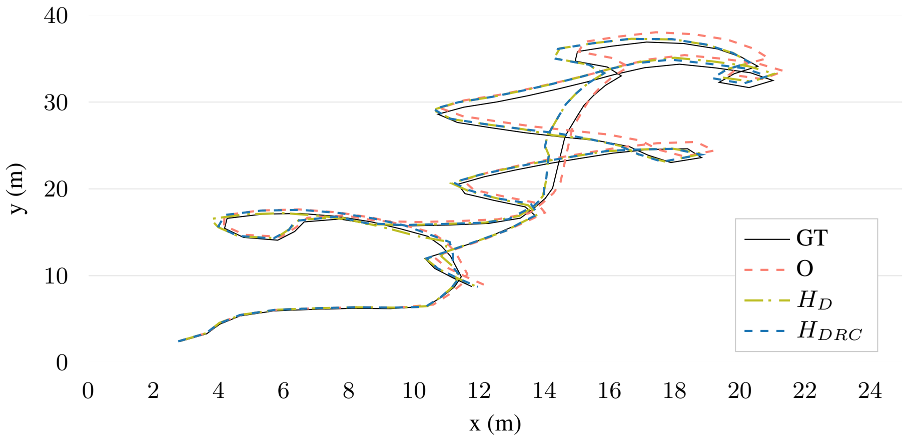
4.6.2 Single-robot simulation results

The aim of the simulations is to prove whether our encoded landmark localization and navigation scheme is viable and adds a significant computational overhead or not.

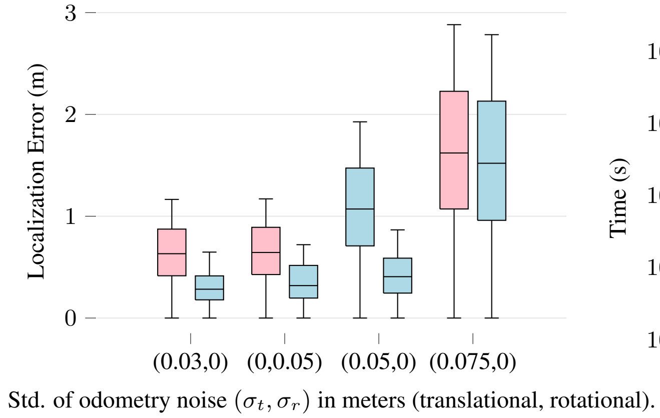
Figure 34 (a) shows the path recovered from odometry measurements and hash-based localization with doorway hashes only, and all three types of hashes, together with the ground truth (GT). The data is recorded over 150 s; the translation odometry noise is set with $\sigma_t = 0.03$, and rotation noise with $\sigma_r = 0.05$. The error distribution for the two methods is given in Fig. 34 (c). Because the doorway-type landmarks are predominant in the chosen simulation environment, the localization error does not decrease significantly when also considering rooms and concave corners.

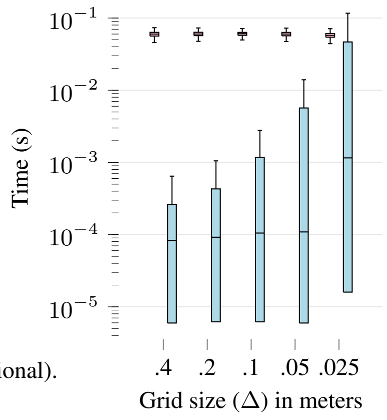
In the simulated environment, we have predefined the position of landmarks with an accuracy of 0.1 m. Therefore, when analyzing the errors in Fig. 34 (b), any values below 0.15 m represent virtually zero error. Figure 34 (b) shows that the localization method is robust even when the odometry error increases significantly ($\sigma_t = 0.05$). However, there is a limit, around $\sigma_t = 0.06$, for which the size of the environment is big enough so that the robot is unable to match landmark hashes due to accumulated odometry drift. In order to calculate these hashes, we assume an error tolerance with respect to its estimated position of ± 0.5 m, independently of the size of the grid utilized to locate the landmarks and generate the hashes. This limit defines the computational time required for the hash search together with the grid size.

Regarding the computational overhead necessary to calculate the hashes, estimate the robot's position, and perform path planning accordingly, Fig. 34 (c) shows the distribution of computational time utilized to extract the set of features, or points of interest, from the raw lidar data and the distribution of computational time utilized in calculating and matching hashes. For an error tolerance of ± 0.5 m, the graphic shows situations in which the robot tests up to 9, 25, 121, 441 and 1681 grid positions, respectively. The search for a hash match is gradually done in a spiral manner around the estimated position and within the aforementioned error tolerance. These results show that even with fine-grained grid search, in average the time required to localize the robot based on hashes is two orders of magnitude smaller than the time required to extract features from lidar data. In the worst-case scenario, the time required can be comparable, with an equivalent order of magnitude for both hash matching and feature extraction.



(a) Ground truth and estimated path during simulation.





(b) Odometry and hash-based localization error for different odometry noise levels.

(c) Execution time for a varying grid size.

Figure 34. Simulation results. Subfigure (a) shows the reconstructed path with ground truth (GT) wheel and inertial odometry (O), only doorway hashes (D), and all features: doors(D), rooms (R) and concave corners (CC). Subfigure (b) shows the odometry and hash-based localization errors for different odometry noise levels. Finally, (c) shows the execution time distribution for the feature extraction (red) and hash matching (blue) processes, where the grid size represents the search space when trying to find a hash match.

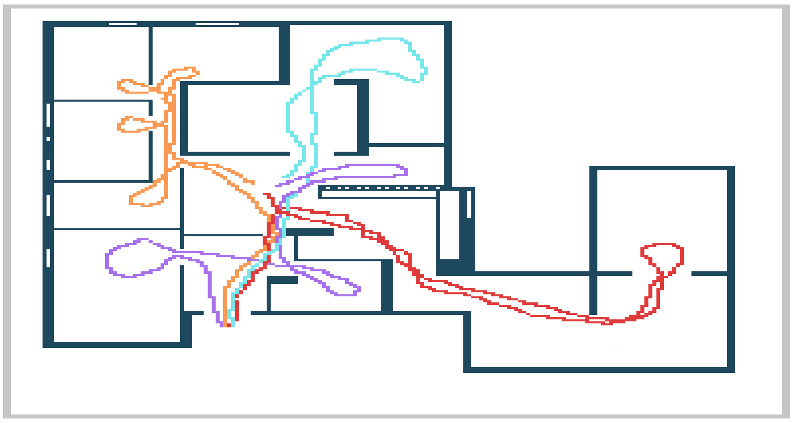


Figure 35. Illustration of the paths followed by four robots during the multi-robot collaborative exploration simulation.

Table 8. Environment knowledge distribution during the collaborative exploration simulation.

	Hashes found	**Area**
Robot$_1$ (purple)	12/32	23%
Robot$_2$ (orange)	16/32	29%
Robot$_3$ (red)	15/32	41%
Robot$_4$ (cian)	11/32	27%

4.6.3 Multi-robot exploration simulation results

In terms of cooperative exploration, we provide a qualitative analysis of a four-robot cooperation. Figure 35 shows the paths of four robots exploring different areas of the same simulation environment. By utilizing encoded landmarks, these robots can share their progress upon meeting in the center of the maze without revealing the raw data they have acquired. If the mission's nature is to perform surveillance or detect a series of items, robots do not need to store raw map data at all. Nonetheless, even if they did, the knowledge of the objective environment remains divided, as shown in Table 8. In this case, robots acquire in average raw data form only 30% of the objective exploration environment, and 41% at most.

4.6.4 Experimental results

Figure 36 (a) shows the path recovered from odometry measurements and hash-based localization (QR codes). The error in the odometry is significantly higher than in the simulation experiments due to a drift in the yaw measurements. However, the translational odometry error is much smaller. The hash-based localization is able to correct this orientation whenever a QR code is within the field of view, and therefore it does not suffer from the yaw drift. The maximum hash-based localization error that we observed was of 41.3 cm (between observations of landmarks and owing to the accumulated odometry drift). This allowed the utilization of a fine grid of 2 cm for calculating the landmark hashes. We set, experimentally, a $1\,m^2$ hash search area around the estimated location.

A total of 23 QR codes were installed in the office environment, and the tests were done with a small number of persons in their offices. Out of those, 17 QR codes were utilized by the robot during its navigation.The mission times at which at least one code was in sight, the error from each observation, and the global error distribution are shown in Fig. 36 (b). The execution time of the hash matching algorithm was on average over one order of magnitude smaller than the time required to extract the QR codes from camera images. Thus, the overhead was mostly negligible. Only in a reduced number of occasions was the latency of these two processes comparable, as Fig. 37 shows.

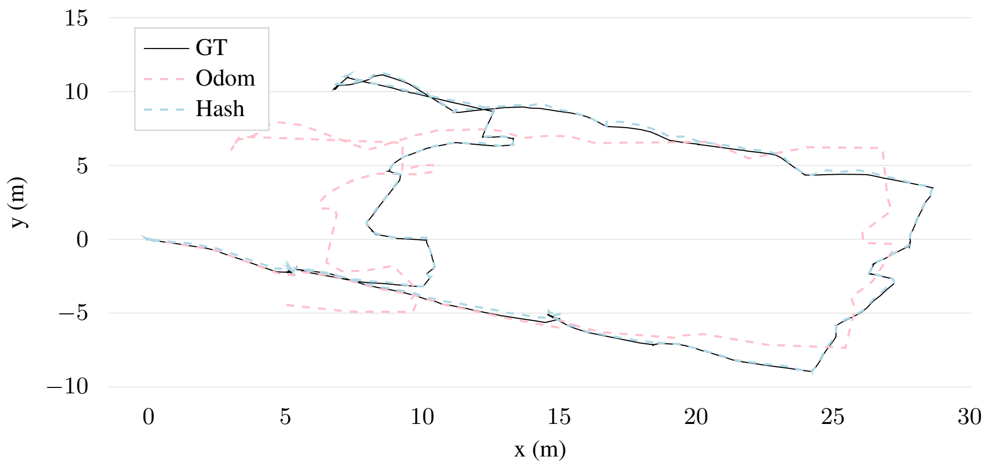
4.7 Summary and conclusions

The work discussed in this chapter was motivated by the problem of securing robot actions in the real world. While much research has been recently devoted to increase the security of robotic systems from the cyber-security point of view, we see a gap in the literature in terms of securing a robot's interactions with its environment. This chapter suggests a new approach with a framework to secure robotic workflows by encoding mission instructions. The process of decoding the instructions serves, simultaneously, to validate the correct operation of sensors and actuators. This framework can be better applied in controlled environments or when there is a priori knowledge of the operational environment of the robot. We show that the integration of this framework within robotic navigation algorithms has little to no impact in the robot's performance or the computational costs.

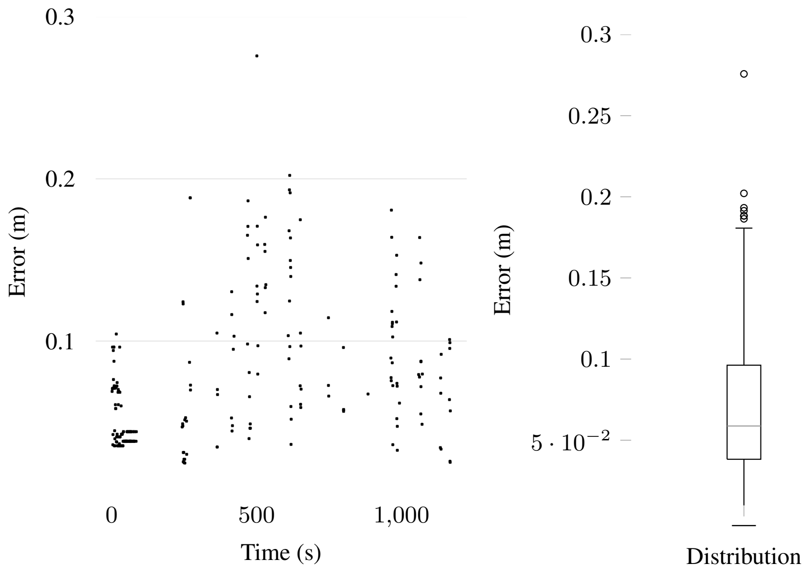
4.7.1 Chapter summary

Security and safety in robotics are crucial aspects to take into account with the current surge of autonomous robots penetrating multiple aspects of our society and the increasing interaction between robots, and between robots and humans. In this direction, further research needs to focus on the validation of data at the different layers of robotic systems, and in particular the validation of the interaction of a robot with its environment. This interaction often starts with navigation, which has been the main aspect studied in this chapter.

Navigation and localization in autonomous robots require large amounts of raw



(a) Estimated path and ground truth during the experiment.



(b) Localization error when a QR code is in sight over the complete experiment (left) and distribution of the error (right).

Figure 36. Experiment results. Subfigure (a) shows the reconstructed path with ground truth (GT), wheel and inertial odometry (Odom) and hash-based localization (Hash). In (b), we show a scatter plot with the localization error every time that a QR code is within the field of view of the camera, and a boxplot with the global error distribution. Finally, (c) shows the execution time for the different processes involved in the localization, with the hash search and matching being one to two orders of magnitude below the image processing processes that would still be in use in a traditional approach.

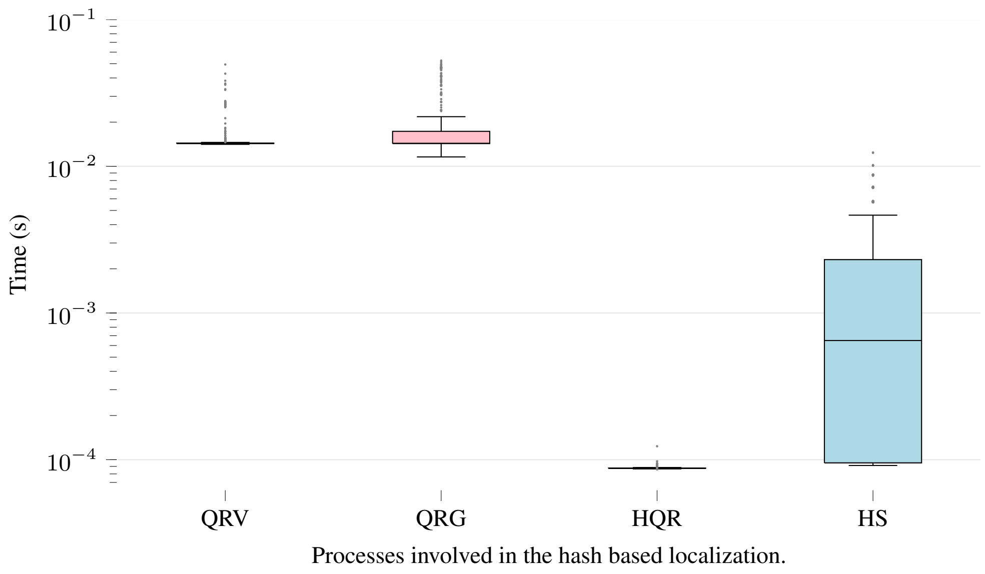


Figure 37. Execution time for the different processes involved in the localization: QR code extraction when in view (QRV), global QR search (QRG), hash search for the detected code (HQR) and hash matching (HS). We can observe that the hash search and matching methods require one to two orders of magnitude less computation time than the image processing processes. The latter would still be in use in a traditional approach.

data for long-term operation, either given a priori by a mission controller or acquired by robots while performing their missions. In addition, validating the integrity of both mission instructions and sensor data without any external feedback is an open problem. We have presented a framework that enables robots to validate both the correct operation of their onboard hardware and sensors, and the integrity of information received from an external controller.

In particular, to the best of our knowledge, this chapter introduces and evaluates the first end-to-end validation framework that focuses on navigation and localization with encoded landmarks, which allows robots to effectively perform their missions while performing end-to-end validation of information. We have shown that utilizing an encoded navigation graph adds only a negligible computational overhead even when high-accuracy positioning is required.

The end-to-end validation scheme demonstrated in this work for navigation tasks can be naturally extended to cover virtually all domains of robotic operation. In future work, we will focus our research efforts towards experimentation in more relevant environments, and in particular industrial settings. We will aim at extending this approach to other interaction forms between a robot and its environment, from multi-robot collaborative assembly to human-robot interaction and control.

4.7.2 Viability and usability

We have seen that the computational overhead added when encoding landmarks is mostly negligible. Thus, our approach could be incorporated on top of many existing navigation and localization schemes, whether they are landmark-based or not, to increase the level of security if the error tolerance allows.

This approach has additional uses when more than one robot is taken into consideration. In multi-robot cooperation, different robots can share their plans, progress or position (based on the navigation graph only) with others by utilizing the same hashes or parts of them. This would reduce the possibility of raw data being exposed but also virtually eliminate the options for attackers or byzantine agents to affect the mission, as has been shown in [272]. Moreover, as we have shown with the multi-robot simulations, the framework allows for multi-robot exploration or other collaborative mission, reducing the fraction of raw data or environment information that has to be made available to each individual unit.

4.7.3 Trade-offs and security considerations

We have shown through simulation and real-world experiments that the proposed framework has negligible impact in terms of usage of computational resources. In the simulations, performing a hash search in a grid map with a resolution of 2.5 cm requires in average over an order of magnitude less computational resources than the lidar feature extraction that is inherent to any localization process. Nonetheless, for each specific application in which an encoded instruction graph is used, there will be a certain size of the hash search space where the instruction decoding is no longer negligible.

Another trade-off occurs between real-time operation and security. The larger the hash search space, the more computational resources are needed to perform a brute-force attack on the encoded information. Nonetheless, performing such attack requires information on the encoding process and algorithm, in turn requiring physical access to the robot or access to the autonomy algorithms and data processing stack. The specific threshold will be defined, among other factors, by the hashing algorithm, and the ways in which the hashes are generated (e.g., involving time or unknown external inputs from the controller can, in many situations, render the brute-force attacks unfeasible).

In terms of the resilience of the proposed framework against adversarial attacks, the main security vulnerability that we have detected is the ability of an attacker to reproduce the encoded commands even without decoding them, potentially triggering the robot into repeating actions. If data is spoofed when transmitted to the robot, the robot's behavior could be studied under different encoded commands, and then an attacker could trigger a known mission. While this cannot be completely mitigated within our proposed approach, we have introduced time-checked actions and event-triggered actions. If a one-time action is required and either the start of the mission or its timing is known, then it is feasible to include the time component into the encoded instructions to avoid repeated actions even if the encoded data is spoofed. Moreover, other generic strategies designed against data spoofing could be introduced on top of our framework.

Another key issue is the potential deadlock state into which a robot could run into if it is unable to decode an essential subset of the encoded instructions. This can happen, for instance, if the sensor data drifts too much from the expected value and beyond the maximum error tolerance. In that case, a robot might be unable to reproduce the hash of one or more key features from its environment. While we have shown that reasonable error tolerance can be taken into account in the encoded navigation graph by doing a spatial search until matching the hash, this is an issue that requires further study. Nonetheless, it is part of our objective to actually force a deadlock state when a robot either malfunctions or data is tampered with. Therefore, this behavior is expected based on the definitions introduced in this chapter, and we will focus our future research not towards making hashes more reproducible but instead towards designing strategies for getting out of the deadlock state. This will involve estimating the origin of the error and defining the corresponding control strategies. In the current framework, it might be unfeasible to conclude whether an instruction cannot be decoded due to a sensor malfunction or a faulty actuator.

4.7.4 Validation of sensor data

With the navigation experiments, we are also able to leverage the encoded landmarks for validating the sensor data leading to odometry estimations. Indeed, the robot is able to estimate the drift or error in the odometry as a function of the computational time required to decode the landmark position (i.e., the number of hashes that need to be tested against the encoded map information). If the odometry error is too large, or a sensor that is used for odometry data malfunctions, then the hash cannot be decoded unless the hash search radius and computational timeout are extended consequently. In the experiments, we utilize only IMU and wheel odometry data (e.g., versus visual-inertial or lidar odometry) to evaluate the performance of the proposed methods in more adversarial conditions where the odometry drift between landmarks may increase significantly.

4.7.5 Secure and trustable multi-robot systems

The research in Chapters 3 and 4 of this thesis has opened the door to further uses of blockchain technology and its components for building security and trust in multi-robot systems. Below we outline some of our ongoing research in this area.

One of the key practical problems of utilizing blockchain technology in real-world multi-robot systems is the strict requirements form the point of view of connectivity and network topology. Most if not all existing approaches require either global connectivity at all times or reliable networking solutions. To tackle this issue, we have been working on a partition-tolerant and byzantine-tolerant multi-robot collaboration framework in [271]. This is made possible by IOTA, a next-generation blockchain solution that uses a directed acyclic graph (DAG) as its underlying data structure instead of the traditional *linked list* in a blockchain. Our new framework allows for robots to disconnect and reconnect from the network, while still trlying in smart contracts to build byzantine tolerance.

We have also worked on extending to more realistic scenarios and use cases the security considerations. For example, in [271], the partition-tolerant framework is tested with cooperative mapping simulations and real-world experiments. We show that with our approach, we are able to deliver a system that identifies and eliminates fabricated lidar data that would otherwise affect the global map generated in cooperation by multiple robots. In [326], we extend the framework to detecting a byzantine robot based on visual data, leveraging our work in vision-based change and anomaly detection [327]. While these experiments are proofs of concept, they also show the potential of the approaches introduced in this thesis in real-world applications.

5 Localization

Autonomous robots that interact with their environment require means for localizing themselves. Global or absolute localization, while essential for many applications (e.g., drone logistics, self-driving cars, or many service robots) are not always necessary. Simpler missions might find it sufficient to estimate relative localization (e.g., following a person or a robot leader) or might not need localization at all but only obstacle and collision avoidance approaches (e.g., moving in a given direction).

In this chapter, we focus in localization in GNSS-denied environments and relative localization within multi-robot systems. In other works, we have studied localization techniques based on GNSS and other sensors, mainly lidars, in both urban environments [18] and in unstructured environments in the wild [328; 329]. Over this and the next chapter, we rely on lidar odometry and visual-inertial odometry for sensor fusion and as part of the localization methods. However, owing to the need for robust and scalable relative localization in multi-robot systems, we focus our work around UWB-based localization. For more details on lidar and visual odometry, we refer the reader to related works that are not included in this thesis [330].

Portions of text and a subset of figures in this chapter are reproduced from our previous works [3; 10; 19; 21; 331].

5.1 UWB-based localization

We first introduce the basic concepts and theory behind UWB-based localization. These methods are common to other means of radio-based localization, where time-of-flight (ToF), time-of-arrival (ToA) or angle-of-arrival (AoA) of signals is measured. Time-of-flight measurements directly lead to estimation on the distance between a transmitter and a receiver. Absolute ToA measurements can be used to estimate the distance between synchronized pairs of nodes, while the distance differences between a transmitter and a series of synchronized receivers can be estimated with time-difference-of-arrival (TDoA). Additionally, passive TDoA can be employed to estimate differences of distance from a passive receiver to an active transmitter-receiver pair. These different approaches are described in more detail in the following parts of the section.

Some commercial UAVs already utilize UWB for indoor localization. For instance, Bitcraze's Crazyflie [332], with its Loco positioning deck, utilizes a UWB tag for indoor positioning. The Loco add-on relies on Decawave's DWM1000. In this chapter, we work with the latest generation of UWB transceivers, the DWM1001, that provide improved localization accuracy. Another company that utilizes a similar, but undisclosed, technology, is Verity Studios [333]. Verity develops multi-UAV systems for light shows indoors. In terms of UWB localization systems, higher-end solutions integrated within ready-to-use systems are available from vendors other than Decawave, such as Pozyx [334], Sewio and OpenRTLS [335].

UWB wireless localization technologies have gained increasing attention in mobile robot applications in the past few years. UWB is a mature technology that has been studied for over two decades [336], with the IEEE 802.15.4a standard including specifications for UWB over a decade ago. UWB systems can be utilized for communication and localization [337], or as short-range radar systems [338]. UWB systems enable localization of a mobile tag from distance-only measurements between the tag and fixed anchor nodes with known position. The distance can be estimated via either time-of-flight (ToF) or time difference of arrival (TDoA). In the former case, the tag can calculate the distance to each of the anchors separately, while the latter estimation requires all anchors to be either connected in a local network or have a very accurate clock synchronization [339].

More recently, with increased accuracy and more commercially available radios, UWB has been applied for indoor positioning and navigation in the field of mobile robotics [340; 321]. UWB-based positioning has been applied in mobile robots to aid navigation as part of multi-modal simultaneous localization and mapping (SLAM) algorithms [341], or for aiding odometry [342]. In the field of aerial robotics, it can aid vision sensors during the approximation for docking in a moving platform [343], or for navigation in warehouses [344].

The distance between a UWB emitter and a receiver node can be estimated, given a known speed of transmission of the electromagnetic waves in air, from the time of flight of the signal. In most cases, the clocks of the two nodes will not be synchronized, or standard network-based synchronization is several orders of magnitude larger than the actual time of flight of the UWB signal. To avoid the need of synchronization, a two-way ranging approach can be utilized, following Eq. 4 to obtain the propagation time:

$$T_{prop} = 0.5 * (T_{round} - T_{reply}) \tag{4}$$

The method above is also called single-sided two-way ranging (SS-TWR). To remove the need for calibrating intrinsic delays, a double-sided two-way ranging (DS-TWR) approach is a more typical solution. Both are illustrated in Fig. 38 The propagation time in DS-TWR can be calculated as shown in Eq. 5:

$$T_{prop} = \frac{T_{round1} * T_{round2} - T_{reply1} * T_{reply2}}{T_{round1} + T_{round2} + T_{reply1} + T_{reply2}} \tag{5}$$

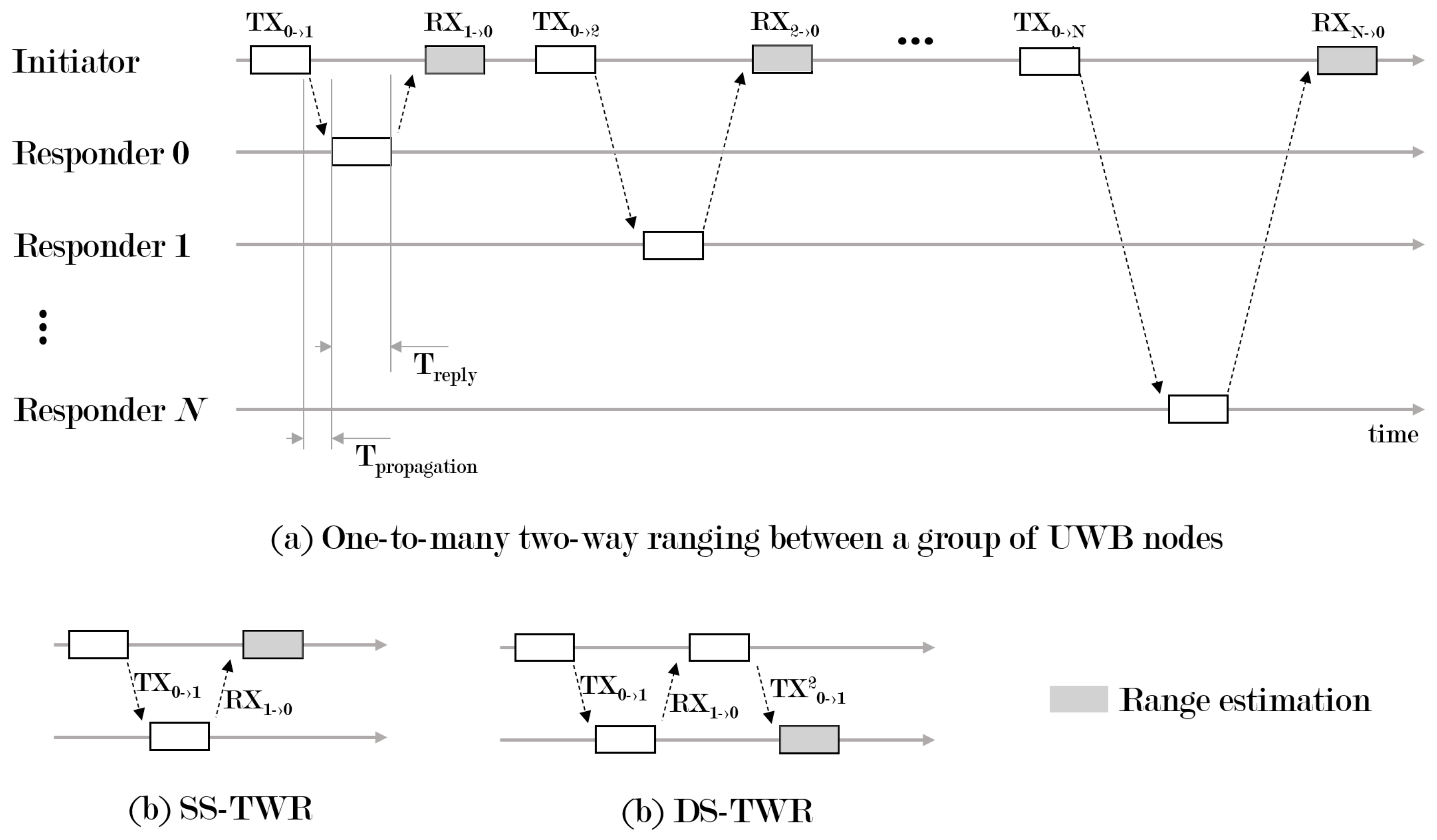


Figure 38. Transmissions required for ranging between UWB nodes in different modalities.

Alternatively, and given a synchronized set of UWB nodes in known locations, a signal by an emitter can be localized utilizing the time difference of arrival to the different nodes. The position in this case is found at the intersection of a set of hyperbolas. If the set of nodes in known positions is actively communicating, a passive listener can also localize itself based on a TDoA scheme, given that the messages include enough information about timing relative to each transmission. More details of such an approach can be found in [345]. Some of the above possibilities for localization are illustrated in Fig. 39.

5.2 Previous works

UWB ranging has been used in multiple mobile robots to air localization or navigation, often when fused with data from other sensors. For instance, UWB has been utilized to bypass the complexity of visual loop closure detection in [340]. In lidar-based SLAM, UWB ranging has been utilized to avoid laser range limitations of inexpensive 2D scanners in tunnel-like environments [321], where the UWB measurements are not utilized for positioning but as part of range-only SLAM. This simplifies the installation, as the position of the UWB anchors can be unknown.

UWB in mobile and aerial robots

Multiple works have utilized UWB-based localization systems to enable indoor UAV flights. In one of the earliest implementations of such system, Tiemann *et al.* study the robustness of a predefined UAV flight that relies on UWB ranging [31]. This has potential applications in the logistics sector. For instance, in [31] the authors rely on UWB-based UAV navigation for fast and flexible automated stocktaking in a warehouse by scanning the good's QR-codes. A similar study for UWB-based autonomous flight in warehouses was carried out in [346]. When mapping with UAVs, accurate UWB-based positioning allows to shift the focus from odometry and position estimation to sensing, enabling high-fidelity three-dimensional reconstruction with RGB-D cameras [341].

A more complex experiment was carried out in [347], where the authors show how UWB-based localization can be combined with vision position estimation for docking UAVs on a ground vehicle. The work is extended in [343] with a focus on GNSS-denied environments. In their setting, four UWB anchor nodes are placed in the ground robot while one UWB tad is placed on the drone. However, the ground robot is relatively large, 2 m long and 1.5 m wide. In our dataset, we experiment with different anchor configurations, including a setting that can be installed on a small ground robot of about 0.6 m by 0.6 m, and a case where all four anchors are near to the ground.

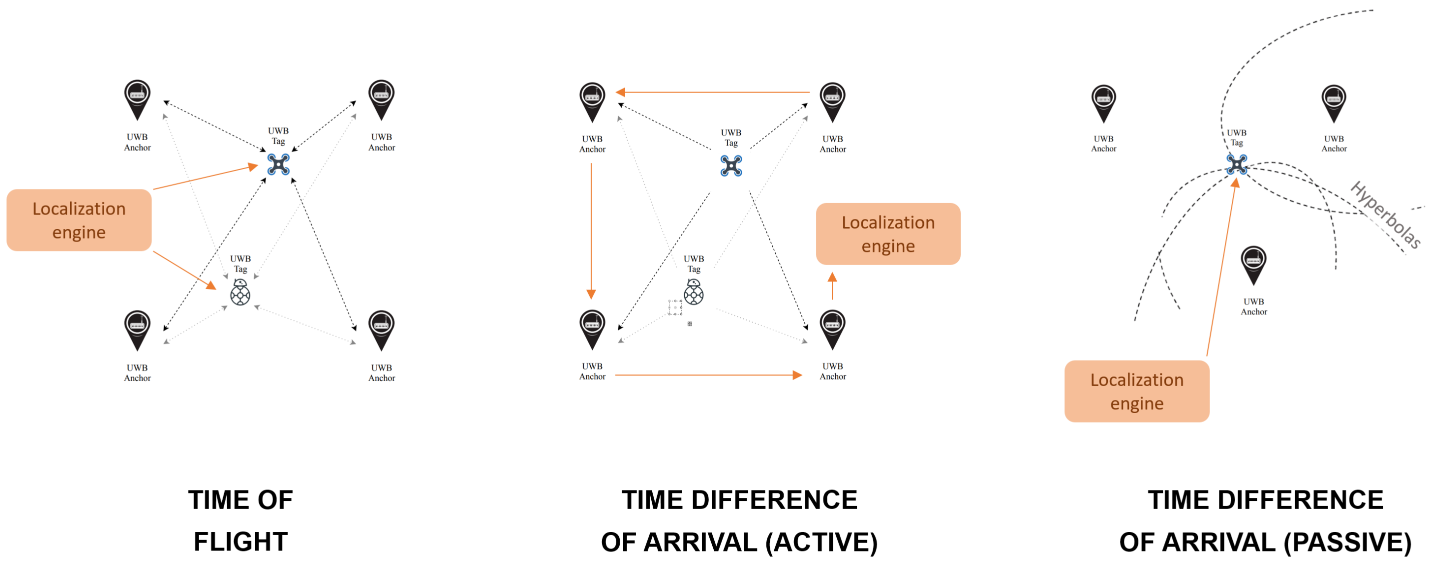

UWB Anchor
UWB Tag
Localization engine
Hyperbolas
TIME OF FLIGHT
TIME DIFFERENCE OF ARRIVAL (ACTIVE)
TIME DIFFERENCE OF ARRIVAL (PASSIVE)

Figure 39. Illustration of different possibilities for achieving localization using UWB ranging. The illustrated list of possibilities is not exhaustive but represents the most common deployments.

5.2.1 Formation control

One of the first applications of UWB in multi-UAV systems is formation control algorithms. Formation control refers to the design of multi-robot control systems for spatial coordination [4]. In many cases, these rely only on distance measurements [348], but more often relative positioning between the robots is required [164]. Some algorithms do not need accurate localization but only rough estimations of neighbor positions, and collision avoidance is often a part of the control scheme [5].

UWB has already been utilized in formation control algorithms for multi-UAV systems. K. Guo *et al.* [349] and S.Q. Cao *et al.* [350] both proposed a distributed formation control scheme. In their work, a UWB ranging and communication network was used for relative localization (RL) estimation in two dimensions. From such a network, the distances and relative speed between the UAVs can be estimated.In [349], the authors achieved formation flights with three UAVs. Finally, in [350] the authors realized a leader-follower formation flight with two UAVs. In this latter work, the authors also performed an outdoor positioning comparison between UWB and GPS, concluding that UWB can fully meet the requirements of relative positioning for formation control algorithms at a similar level than GNSS sensors.

A more recent application of UAVs has been their utilization in light shows both indoors and outdoors. At the 14th Moscow International Aerospace Expo (MAKS), a Chinese drone company: DAMODA, performed a ”human-machine dance” with UWB-aided UAVs [351]. The entire indoor UAV formation is based on UWB localization technology to achieve precise positioning and precise control. Another notable company utilizing undisclosed UWB technology for indoor shows is Verity Studios, a Swiss company that designs multi-UAV systems for indoor spectacles [333].

5.2.2 Multi-robot systems

In multi-robot systems, having multiple UWB nodes in each robot enables robust relative localization, both in position and orientation [352]. In the case of multi-robot coordination with an anchor-based positioning system, UWB distance measurements can be utilized in formation control algorithms [353], also fusing them with odometry estimation [354]. Finally, the positioning system can also be decentralized with individual node-to-node distance measurements in a cooperative multi-agent system [355].

Multi-modal sensor fusion in heterogeneous multi-robot systems can significantly increase the situational awareness of individual robots [149]. However, the data fusion mechanisms often need accurate relative localization between the robots. The high-accuracy short-range distance estimation that UWB enables can then be exploited in this direction. Deploying multiple UWB transceivers in each robot enables not only relative localization between each pair of robots in terms of position but also orientation. Nguyen *et al.* demonstrated the viability of this idea by designing an extended Kalman filter for robust target-relative localization in a heterogeneous multi-robot system, where UWB was utilized for both ranging and communication [352]. In utilizing UWB for relative positioning, one of the most relevant publications to date in this area is Nguyen *et al.*'s work on the first autonomous docking of UAVs in a mobile UGV platform that relies on UWB localization for approaching the mobile docking station [343]. The final docking maneuvers, however, are based on onboard vision and known markers on the docking platform. While the size of the mobile platform was relatively large, 2 m long and 1.5 m wide, recent datasets with multiple anchor configurations show the viability of this idea maintaining relatively high accuracy [3]. Other works have focused on utilizing UWB for specific maneuvers. In [356], a multi-robot collision avoidance scheme was developed and tested with UWB transceivers for global localization of the agents via deep reinforcement learning. Qiang *et al.* developed a multi-robot localization platform built on top of the Robot Operating System (ROS) [357]. Their platform, which can accommodate various types of robots, was then extended and applied to a formation control problem in [358].

5.2.3 Distributed estate estimation

More recently, a series of works have emerged in which UWB ranging, fused with other onboard sensor data, is used for distributed estate estimation in robot swarms. This has been shown to be especially effective in swarms of aerial robots [359; 360; 35; 361]. Xu et al. have presented a robust multi-modal sensor fusion algorithm exploiting UWB ranging and VIO that provides a decentralized and collaborative localization framework for multi-UAV systems [359]. In a subsequent work, the authors further developed the system for robust localization both indoors and outdoors. In [35], Zhou et al. present a novel aerial swarm capable of operating in dynamic unknown environments. The methods presented by the authors are also robust to partial observations, with the swarm being capable of complex behaviour such as following a person that is only visible by part of the swarm through a forest. As indicated in [361], the aerial swarm presented by Zhou et al. is probably the first one capable of operating in unstructured environments, with the presence of dynamic obstacles, and in a fully decentralized manner with only onboard computation and sensing.

5.2.4 Contributions

The main contributions of this chapter are the following:

1. The introduction of a novel dataset that relies on ToF measurements of UWB signals for positioning of UAVs, meant for fast and mobile deployments with ground robots acting as anchors. This includes an accuracy and latency analysis of the auto-positioning of the anchors. To gather this dataset, we also develop new firmware for UWB nodes to be able to perform all required localization and calibration calculations.

2. The characterization of the UAV positioning accuracy as a function of the spatial distribution of the anchors, the distance of the UAV to the center of mass of the anchors, and its speed and height. Moreover, initial experiments show the feasibility of stable autonomous flight based on the proposed localization system. We also provide open-source code for the automatic calibration of anchor positions, as well as the ROS nodes used for interfacing with the UWB devices in different modes.

3. The introduction of a relative localization method based on a least-squares estimator for ground and aerial multi-robot systems. We show that this approach is more accurate than GNSS in areas where GNSS signals degrage (e.g., near tall buildings).

5.3 Characterization of UWB localization

We investigate the properties of a mobile and inexpensive ultra-wideband (UWB) wireless positioning system that can be quickly deployed in GNSS-denied emergency scenarios, or in general for indoor environments. Compared with other indoor localization systems such as motion captures [362], a drastic decrease in both system complexity and price only has a relatively small impact on positioning accuracy. More importantly, even if the accuracy is reduced, the localization estimation is stable and does not threaten the smooth flight of a UAV. These type of system can complement existing motion capture (MOCAP) systems providing more flexible and mobile deployments.

Several datasets and analysis reports exist for indoor localization of mobile robots based on UWB [363; 364], including UAVs [365]. However, we have found that all previous studies involving the localization of UAVs are based on TDoA measurements which are more accurate but limit significantly the mobility of the system as a whole. Instead, we rely on ToF distance estimation only and analyze the self-calibration of anchor positions for a mobile setting.

Finally, we study the localization accuracy as a function of the UAVs speed, height and position with respect to the anchors, both within and outside the convex envelope defined by the anchor positions. This type of characterization based on the operational details of the UAV does not exist in previous works. In this chapter, we utilize Decawave DWM1001 UWB modules, the latest generation with an advertised

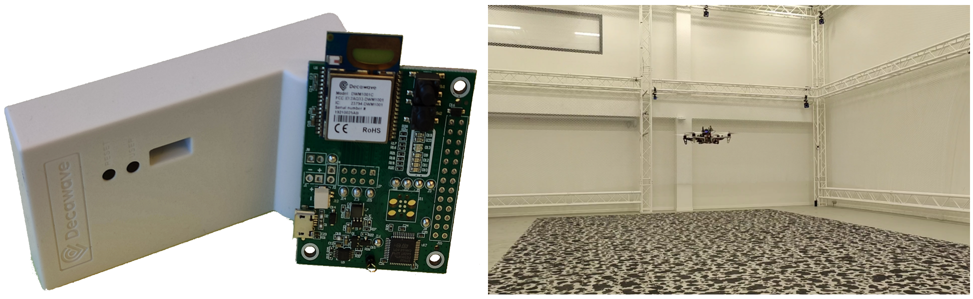

(a) UWB Node. (b) Quadrotor in the MOCAP system.

Figure 40. (a) The DWM1001 DEV board with and without case. (b) The F450 quadrotor used in the experiments in the Optitrack MOCAP arena.

accuracy of up to 5 cm. The data is acquired using the Robot Operating System (ROS). Custom ROS packages have been written for interfacing with the DWM1001 modules depending on their configuration (anchor, active tag, passive tag) and made publicly available, while the position information is given by Decawave's UART API, which is closed source.

5.3.1 UWB characterization and existing datasets

There are some previous works which have already characterized UWB localization systems [366], including in the field of aerial robots [344]. However, to the best of our knowledge, previous analysis of the localization accuracy were done based on a fixed and well-calibrated anchor system. Moreover, existing datasets are small and very specific. Therefore, we believe there is a need for a more comprehensive understanding of the advantages and limitations of UWB-based flight for autonomous UAVs, in particular with fast ad-hoc deployments and movable anchor systems where the relative position of the anchors can also change over time. Through the rest of this section, we explore recent works in UWB-based or UWB-aided localization for mobile robots and, in particular, UAVs. This covers use cases in both industry and academia. Then, we analyze existing datasets for UWB-based localization of mobile robots and compare then with ours.

Raza *et al.* introduced a dataset for indoor localization with narrow-band and ultra-wideband systems [363]. The dataset includes data from both a walking subject and a remotely operated radio control car. This dataset only one specific anchor position, and the UWB tag in the remotely operated car is at a constant height. In consequence, the analysis of the data can only be partly extrapolated to other use cases, such as aerial robots.

Barral *et al.* presented a dataset acquired using ROS and Pozyx UWB devices [364]. This dataset only contains range information between two tags. It

Table 9. Latency and Accuracy of the Autopositioning method from Decawave's DRTLS localization system compared to a custom self-calibration method for anchors.

	Latency	**Distance**	**Max. Error**
RTLS Autopositioning	$40\,s \pm 5\,s$	10 m	1.2 m
		4 m	0.75 m
Custom Calibration (x50)	$2.5\,s \pm 0.1\,s$	10 m	0.4 m
		4 m	0.25 m
Custom Calibration (x5)	$0.9\,s \pm 0.05\,s$	10 m	0.5 m
		4 m	0.3 m

enables the characterization of inter-device distance estimation in both line of sight (LOS) and non-line of sight (NLOS) conditions. A similar study was carried out by Bregar *et al.* in [367] and [368] with Decawave's DWM1000. In both cases, the distance between a single anchor and a tag was estimated in multiple locations, with both LOS and NLOS ranging.

Regarding the utilization of UWB localization for UAVs, Li *et al.* published a dataset recorded over an indoor flight of a UAV with UWB-aided navigation [365]. In their paper, the authors also introduce an Extended Kalman Filter (EKF) that enables very accurate 3D localization by fusing UWB and IMU data. Their dataset, however, contains data from a single flight with a single anchor setting. In contrast, our objective is to analyze how different anchor configurations affect the accuracy of the localization. In particular, most of our subsets of data have been recorded with anchors situated in a two-dimensional plane, so that it can mimic a more realistic and quick deployment in, for example, post-disaster scenarios, where the anchors might be mounted on ground robots. As most drones have some type of accurate onboard altitude estimation (sonar, lidar, or infrared), it is enough if the UWB system provides position information in two dimensions only. Moreover, we provide subsets of data were the quadrotor is equipped with one, two or four UWB tags, therefore enabling orientation estimation as well. Finally, commercially available UWB-based localization systems have significantly improved over the past two years since the previous dataset was published [365]. In this section, we report even smaller localization errors *out of the box*, without the EKF to fuse with IMU data. The device we have utilized, the DWM1001 from Decawave, is illustrated in Fig. 40 (a).

Another key difference of our dataset is that we rely on ToF measurements only for the UWB position estimation. While this can reduce the accuracy when compared to TDoA, it does not require the anchors to be connected and synchronized. We believe this is an essential enabler of ad-hoc mobile deployments. A more detailed comparison of our dataset with existing ones is shown in Table 10.

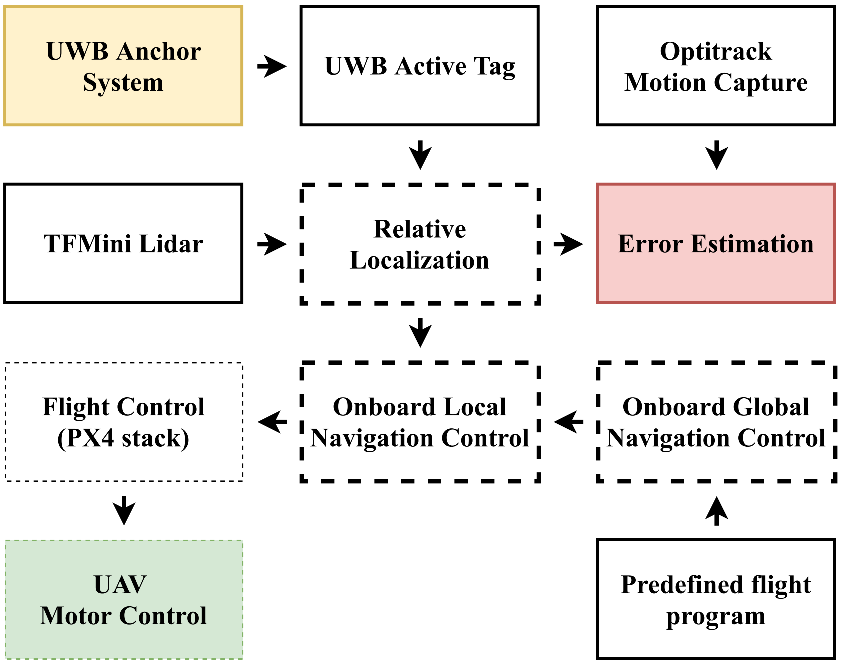


Figure 41. Controller blocks in the UWB-aided autonomous flight. The boxes with continuous border represent data acquisition ROS nodes, while dotted lines represent the custom ROS nodes where the actual control happens. The PX4 stack has not been modified.

5.3.2 TIERS UWB dataset

This subsection describes the data that is included in the dataset as well as the steps followed when recording the different subsets. All the data, ROS nodes and firmware for the UWB devices is made publicly available in Github[1].

The dataset has been recorded using two different methods, with recordings on the UAV or at a ground station. The first case includes data which has been acquired using an onboard computer on a quadrotor equipped with an active UWB tag and flying autonomously. The second case refers to data from a passive tag connected to a ground station, while the quadrotor is being flown manually. In the second case, a delay exists between the ground truth data given by the motion capture system and the UWB data due to the passive nature of the tag being used for recording the positions.

The autonomous flight tests have been done with an F450 quadrotor equipped with a Pixhawk 2.4 running PX4, an Intel Up Board as a companion computer running Ubuntu 16.04 with ROS Kinetic and MAVROS, and a TF Mini Lidar for height estimation. The quadrotor flying in the motion capture arena (Optitrack system) is shown in Figure 40 (b).

Data subsets

The dataset presented in this chapter contains 7 subsets listed in Table 11. In all cases, the number of anchors in use was either 3, 4 or 6. Four anchors is the minimum required for robust localization, as the position of a tag can still be calculated if the connection with one of the anchors is intermittent. However, at some points we disconnected one of the anchors to emulate the situations in which not all four anchors are reachable and study the impact on accuracy. Besides, while a larger amount of anchors can lead to an increase in accuracy, having ad-hoc and movable anchor networks with very large numbers might be impractical. In general, based on our experiments we believe that four anchors give enough accuracy to enable autonomous flight of an UAV, and therefore we find it the most suitable solution. The only scenario with 6 anchors was set in order to analyze if the vertical accuracy would change significantly, also when some of the anchors were positioned at different heights.

The first four subsets were recorded in a more traditional setting with all anchors located in the corners of the motion capture arena. In the case of six anchors, the two extra ones were located on the walls at different heights from the original four. Then, the subset #6 was recorded with four anchors in a single corner and near the ground, emulating a setting that could be installed onboard a single ground robot. Finally, subset #7 was recorded with 4 anchors at a height of 10 cm and a separation of about

[1] `https://github.com/TIERS/UWB_DRONE_DATASET`

Table 10. Comparison of existing UWB-based localization and positioning datasets and ours.

	Distance Est.	Height Est.	Mobile tags	Mobile anchors	Anchor settings	Subsets	Convex Envelope	Dims.	UWB Node
Bregar [367] (2016)	ToF	-	-	-	-	-	-	1D	DWM1000
Bregar [368] (2018)	ToF	-	-	-	-	-	-	1D	DWM1000
Raza [363] (2019)	TDoA	-	✓	-	1	1	IN	2D	DWM1001
Barral [364] (2019)	ToF	-	-	-	-	-	-	1D	Pozyx
Li [365] (2018)	TDoA	UWB	✓	-	1	1	IN	3D	TimeDomain
Ours	ToF	UWB+Lidar	✓	✓	4	12	IN/OUT	3D	DWM1001

1.8 m. This configuration can be achieved with 4 small ground robots porting a UWB anchor each. While a bigger separation can result in better accuracy, our aim in this case was to test the localization estimation robustness when flying both inside and outside the convex envelope defined by the anchor positions.

Whenever more than one tag has been utilized, they were always part of a single rigid body. In subsets #3 and #4, two tags separated 30 cm are mounted on top of the UAV, one in the front and one in the back. In the experiments with 4 tags, these were forming a rectangle of 20 cm by 30 cm. The position of the anchors is calculated through triangulation, and the measurements are taken at the maximum frequency allowed by the UWB module. This results in a higher frequency and larger number of measurements when compared to Decawave's solution, which is built to support a larger number of UWB nodes simultaneously.

Dynamic anchor reconfiguration

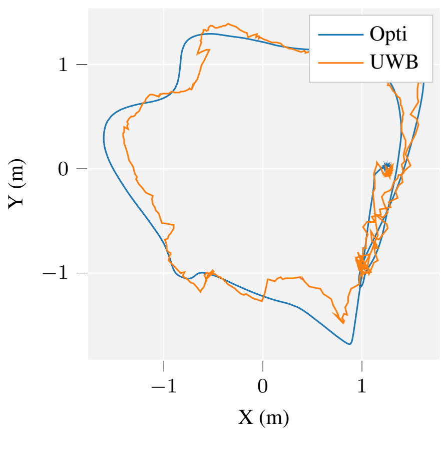
The public dataset that we have made available contains only data that has been acquired using Decawave's RTLS system with their proprietary firmware flashed onto the DWM1001 nodes. The product's UART API has been utilized as an interface to read distance and position information of the different nodes. However, we have found the RTLS calibration of anchor positions to be slow, taking around 40 s, and inaccurate, with errors exceeding 1 m in some cases. Therefore, we also make available an initial implementation of a custom anchor re-calibration system. Our system does between 5 and 50 measurements for each of the distances, with a total latency varying from just under 1 s to 2.5 s.

Energy efficiency

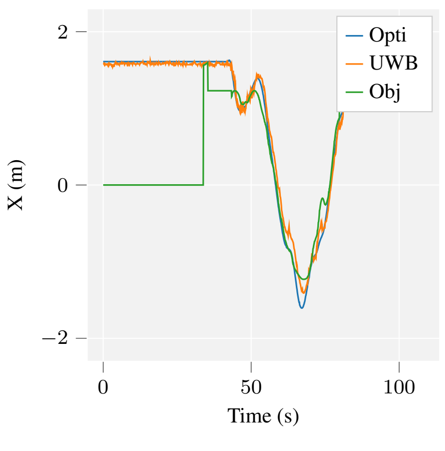
We have also analyzed the power consumption of the UWB nodes in different modes. The power consumption has been monitored with Monsoon's High Voltage (HV) power monitor. While for UAVs the impact on total energy expenditure might be insignificant during flight, this can be a key aspect to take into consideration in mobile settings. For instance, if ground robots move only from time to time.

5.3.3 Dataset analysis, experimentation and results

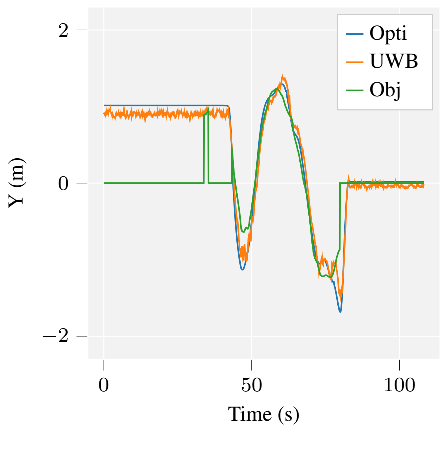
To test the feasibility of an autonomous flight based only on UWB ToF measurements and a 1D lidar for height estimation, we show a simple circular trajectory.



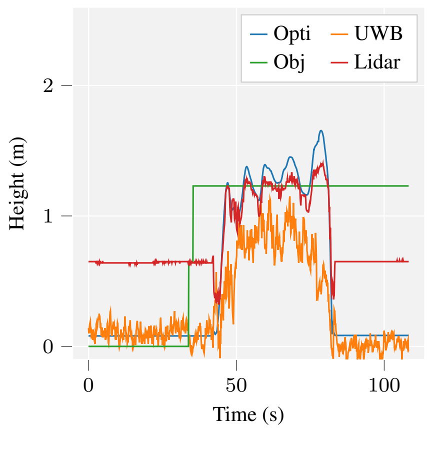
(a) UAV path in the XY plane



(b) UAV position along X-axis

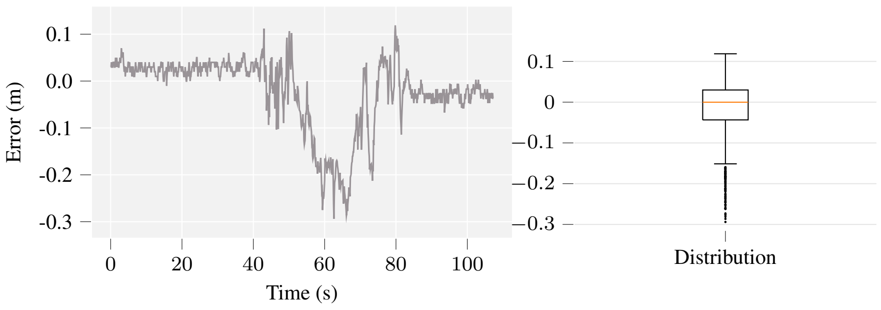


(c) UAV position along Y-axis

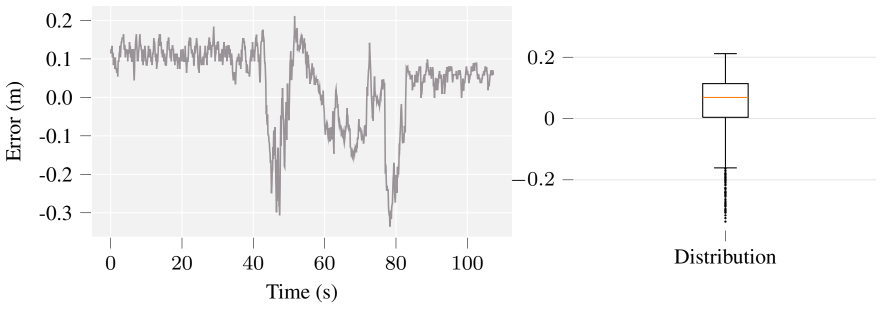


(d) UAV height over time

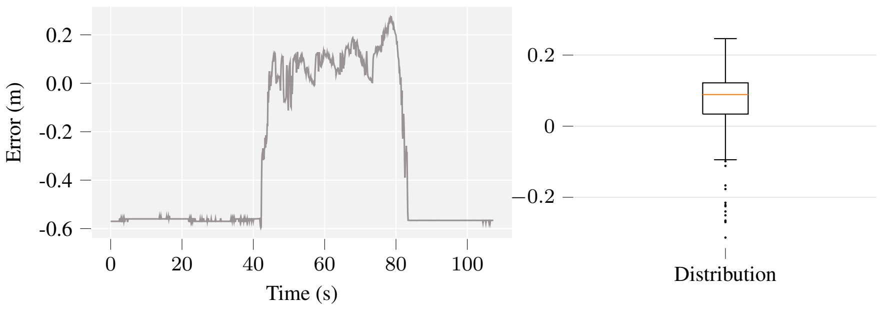
Figure 42. Autonomous flight based on UWB for localization in the XY plane and a 1D lidar for height estimation. For the height error, only the lidar data was taken into account, and only while on flight for the boxplot. The Optitrack (Opti) system gives the ground truth reference, while the position is estimated based on the UWB system and 1D lidar. The position control input (Obj) utilizes the position estimation as well.



(a) X-coordinate error



(b) Y-coordinate error



(c) Height error

Figure 43. Distribution of errors during the autonomous UAV flight shown in Fig. 42.

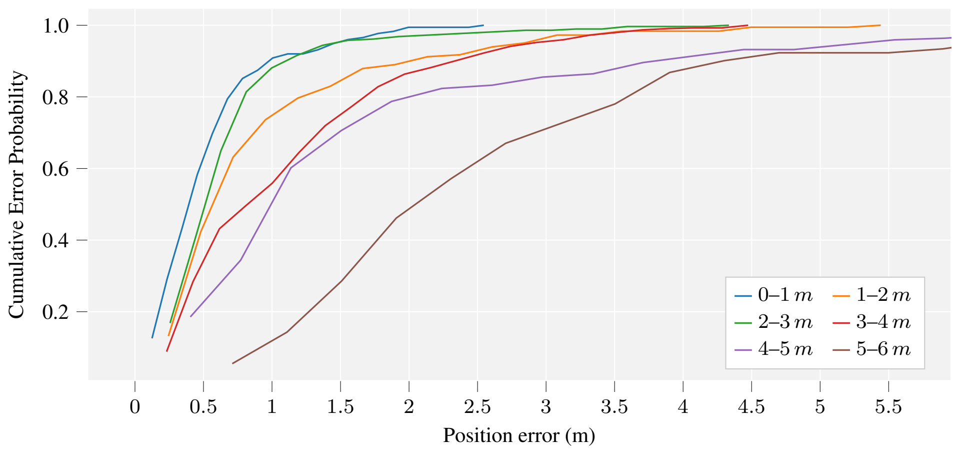


Figure 44. Cumulative probability distribution of the positioning accuracy for different distances to the center of mass of the anchor system. The anchors are positioned in the room corners forming a square of $16m^2$.

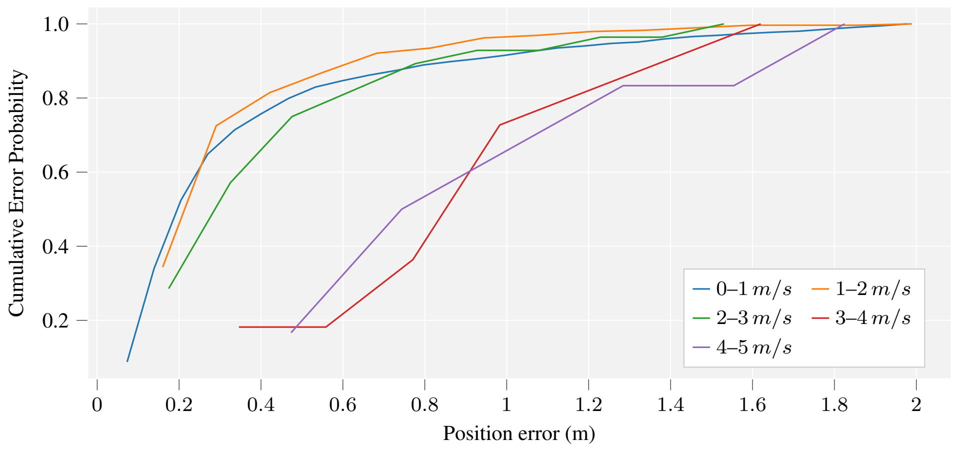


Figure 45. Cumulative probability distribution of the positioning accuracy for different speeds of the UAV. The anchors are positioned in the room corners forming a square of $16m^2$.

Table 11. Description of the data subsets. For each subset, we include the description of the anchor locations, the height at which the anchors are installed, the number of anchors and number of tags, the number of flights recorded separately, the total number of individual distance measurements or position estimations at 10 Hz (#meas). In addition, we describe the platform where the data was recorded, whether at the companion computer connected to active tags, or at the base station via a passive tag, and whether the UAV flight happened inside or outside the convex envelope defined by the anchor positions.

	Anchor positions	Anchor Height	#Anchors	#Tags	#Meas	#Flights	Recording (UAV/GS)	UAV Position (w.r.t. Convex Env.)
#1	Room corners	1.8 m	4	1	1000	1	UAV (Autonomous)	Always in
#2	Room corners	1.8 m	4	2	1000	1	UAV/GS	Always in
#3	Room corners	1.8 m	3–4	1	9000	3	GS	Always in
#4	Room corners	1.8 m	3–4	2	7000	9	GS	Always in
#5	Room corners	0–2.4 m	6	2	1000	1	UAV	Always in
#6	Single corner	0 – 0.5 m	4	1	1700	1	GS	Always out
#7	Room center	0.1 m	4	1–4	2500	3	GS	In & Out

Flight with UWB-based localization

Figure 42 shows the path and localization error recorded over an autonomous flight using UWB for localization in the horizontal plane and a 1D lidar for height estimation. The control input, in green in the figure, given to the drone in the form of waypoints is described by (6):

$$\begin{aligned}\mathbf{p}_{obj} &= (r_{obj}(t), \theta_{obj}(t), z_{obj}(t)) \\ &= \begin{cases} (r_0, \theta_0, z_0) & if \;\; |\|p(t)\| - \|p_0\|| > \varepsilon \\ (r_0, \theta(t) + \Delta\theta, z_0) & otherwise \end{cases}\end{aligned} \tag{6}$$

where $\mathbf{p}(t) = (r(t), \theta(t), z(t))$ represents the current position of the UAV, $\mathbf{p}_{obj}$ is the waypoint given to the UAV as its objective position, $p_0 = (r_0, \theta_0, z_0)$ is the entry position to the circular path, ε is a threshold to consider that the UAV is within the predefined path, and $\Delta\theta$ defines the angular speed when it is considered together with the waypoint update rate and the maximum linear speed of the UAV. In the experiment shown in Figure 42, we have defined $p_0 = (1.23, 0, 1.23)$, $\varepsilon = 0.3\,m$ and $\Delta\theta = 0.05\,rad$. All the positions are represented in cylindrical coordinates, and the radial distance and height are given in meters.

Spatial distribution of anchors

One of the novel analysis in this chapter is the study of different spatial anchor distributions. The accuracy recorded during the autonomous drone flight in Fig. 42 (d)-(f) relates to a typical setting where the anchors are positioned in the corners of a room. One thing to note in this case, nonetheless, is the enhanced accuracy of the latest DWM1001 transceiver. Through our experiments, we have also noted that over 50 % of the samples acquired during the flight had an error under 10 cm. In particular, the lower and upper quartiles in the case of the x-coordinate error in Fig. 42 (d) reflect an error smaller than 5 cm.

An extreme case for the location of anchors would be when these are located in a single robot or movable unit. This has been done, for instance, by Nguyen *et al.* in [343] for an UAV to dock on a moving platform. However, in that article, the authors report using a movable platform of $3\,m^2$, which is impractical in most situations, in particular in post-disaster scenarios where access to the objective operation area might be limited. In order to assess the viability of a more practical usage, we have located four anchors in four corners of a cube, with one of them representing the origin of coordinates and each of the other three the axes. The separation between the anchor in the origin and each of the others was just 60 cm. Fig. 49 shows the position estimation error in this case. We can see that the position estimation is highly unstable. However, the lower and upper quartile in the boxplot relate to relatively low errors. With proper filtering and sensor fusion it might be possible to utilize such anchor settings when the error margin allows.

The last anchor distribution included in the dataset represents, to the best of our knowledge, the most usable for a moving platform, with four anchors near the ground and separated only around 2 m. One of the flights recorded and the corresponding errors are illustrated in Fig. 46. The localization accuracy in this case doubles but still allows the possibility of autonomous flight.

5.3.4 Autopositioning of anchors

The localization estimation provided by Decawave's UART API has given us better results than the utilization of raw individual distance measurements applied to multiple open-source multilateration algorithms. The code to interface the API with ROS for both passive and active tags is made available, together with the data, in the Github repository. However, Decawave's function to autoposition the anchors has not given good results in our experience. Moreover, the calculation takes around 40 s, which is unassumable in some mobile settings. In order to tackle this issue, we have written our own firmware for the anchors in order to recalculate their position if they move. In our experiments, we utilize separate UWB devices for the autopositioning. Each anchor location is equipped with one device flashed with our code for autopositioning only, and another one as an anchor for the localization of the UWB tag, flashed with Decawave's proprietary firmware. We utilize the UART API to set the anchor positions after the autopositioning. In a real scenario, a single device could be used as both but it would need to be reprogrammed on the fly.

Figure 50 shows the average error of the distance estimation between two anchors during the self-calibration process. The measurements are taken at over 35 different distances up to 22 m, with 50 measurements for each distance. The standard deviation for each particular distance ranged from 0 to 4 cm, while the standard deviation of the error altogether was under 3 cm in over 50 % of the cases, as shown in the boxplot in Fig. 50. The comparison between Decawave's autopositioning and ours is shown in Table 9.

In addition, we have measured the power consumption of the UWB devices in different modes, as shown in Table 12 and Fig. 51. This includes anchors, active tags and passive tags, as well as the device running our autopositioning firmware. In the latter case, we provide an initial implementation with no power usage optimization, and during the autopositioning the nodes are transmitting at high frequency. Therefore the power consumption is very high. We differentiate between *responder* and *initiator* types. Each of the four anchors takes the role of *initiator* one time, sending a message one by one to each of the other three (in *responder* mode), and calculating the distance via two-way ranging. The distance between two nodes is thus calculated twice during the autopositioning.

During the autopositioning process, the first anchor to become an initiator, which is activated via a *start* command through the UART interface, is considered the origin of coordinates. Then, we assume that some information about the position of the other anchors exists. The minimum information required is to know the order of the anchors over the boundary of their convex envelope in a counter-clockwise direction. We also predefine the x axis to follow the direction of the vector that is defined from the first to the second anchor following the aforementioned order.

5.3.5 Characterization of UWB localization accuracy

We have classified the accuracy of the UWB localization based on the distance to the center of mass of the anchor system, the height of the UAV and its speed.

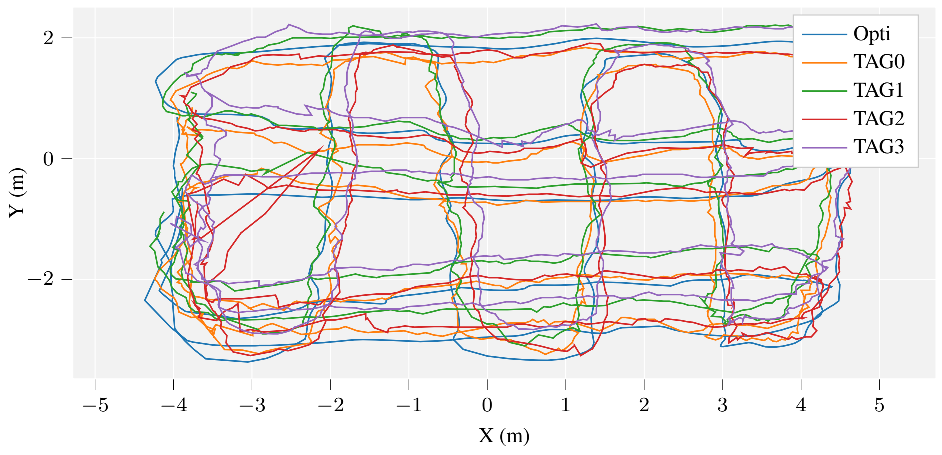
The dataset introduced in this chapter includes data of an UAV flying both inside and outside the convex envelope defined by the anchor positions. In general, the closer an UWB tag is to the center of mass of the anchor system, the higher the position estimation accuracy is. This is illustrated in Fig. 48 (a) for the case where the anchors are in the corners of the room, and in Fig. 47 (a) for the case in which the anchors are in the center and close to the floor.

Based on the rest of the measurements in the same two figures, we can also see that the position estimation error is smaller with lower speed, as illustrated in Fig. 48 (c). Regarding the height, when the anchors were all near the floor, we did not obtain significant differences, as sown in Fig. 47 (b). However, when the anchors were at the height of 1.8 m in the corners of the room, higher flight altitudes resulted in smaller errors, as Fig. 48 (b) shows. While LOS was always ensured during the experiments, the error was smallest near the constant z plane defined by the anchor positions.

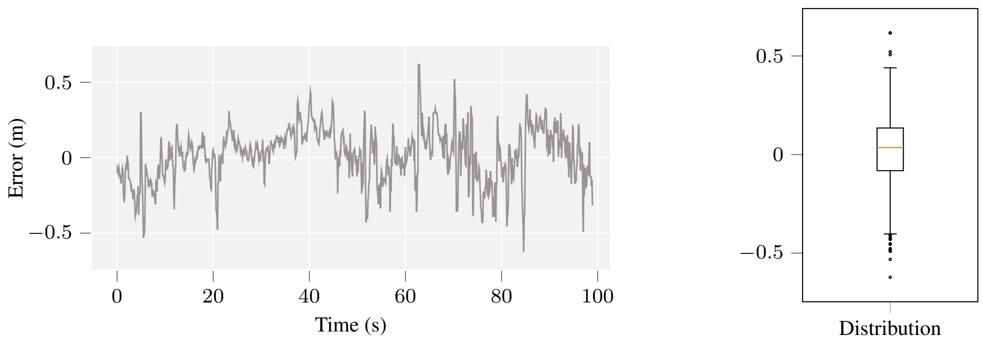
The conclusions from the above characterization of accuracy based on speed, height and distance to the center of mass of the anchor system allow a more efficient control of autonomous UAVs in a real deployment, where strategies can be defined to adjust the error estimation based on these parameters.

5.3.6 Remarks on the UWB dataset

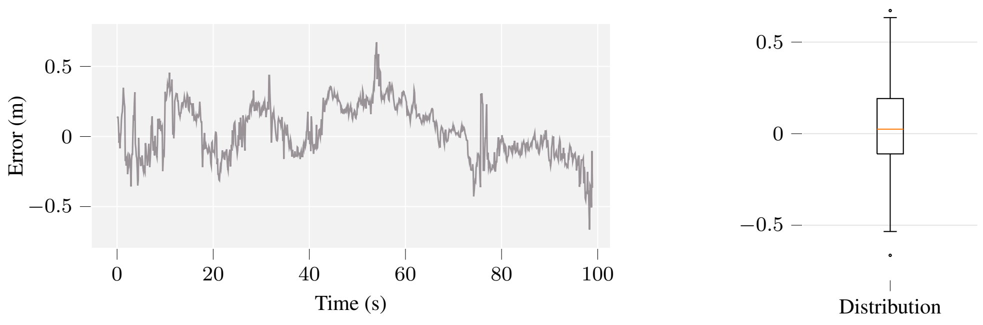
We have presented a novel dataset for UWB-based localization of aerial robots. We have focused on studying the localization accuracy for ad-hoc deployments with fast self-calibration of anchor positions. Up to the authors' knowledge, the dataset presented in this section was the largest and most complete at the date of publication. The dataset includes multiple anchor configurations, as well as data from UAVs equipped with a variable number of UWB tags. The dataset includes data from an autonomous flight with an UAV. The ground truth in all cases has been recorded using an Optitrack motion capture system. It is also the first comprehensive analysis of



(a) XY Pahts

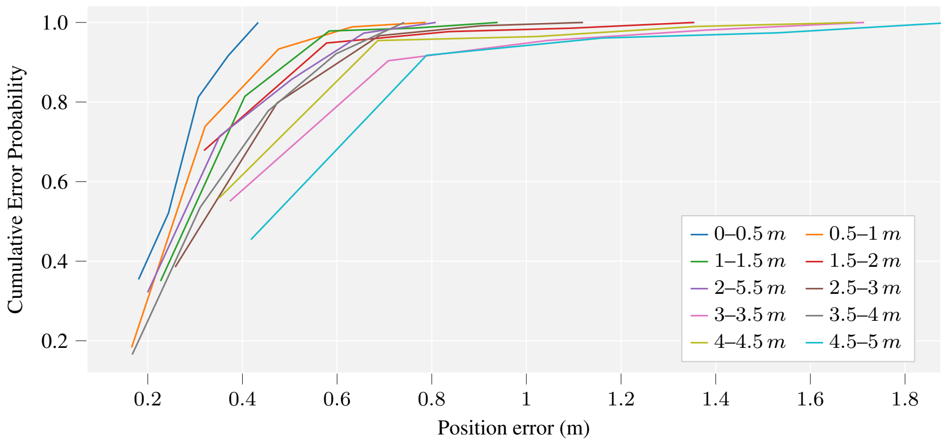


(b) X-coordinate error

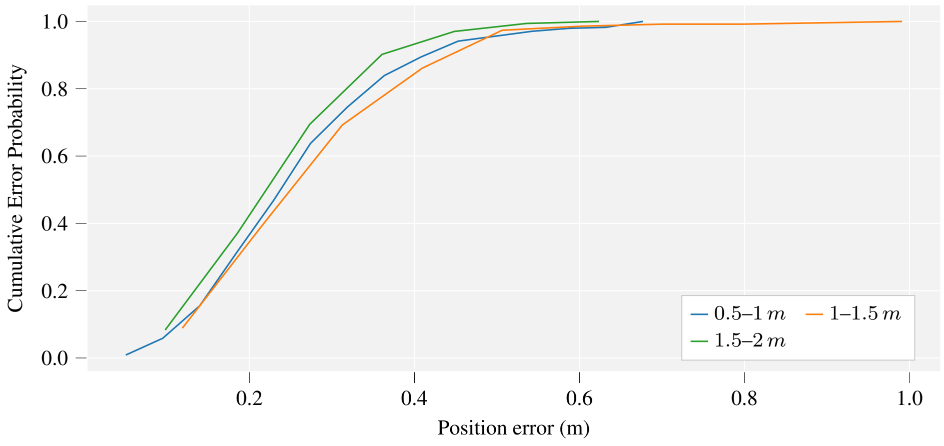


(c) Y-coordinate error

Figure 46. Data recorded with 4 tags (single rigid body) and four anchors positioned in the center of the motion capture arena, separated 1.8 m only.



(a) Cumulative error probability based on the distance to the center of mass of the anchor system

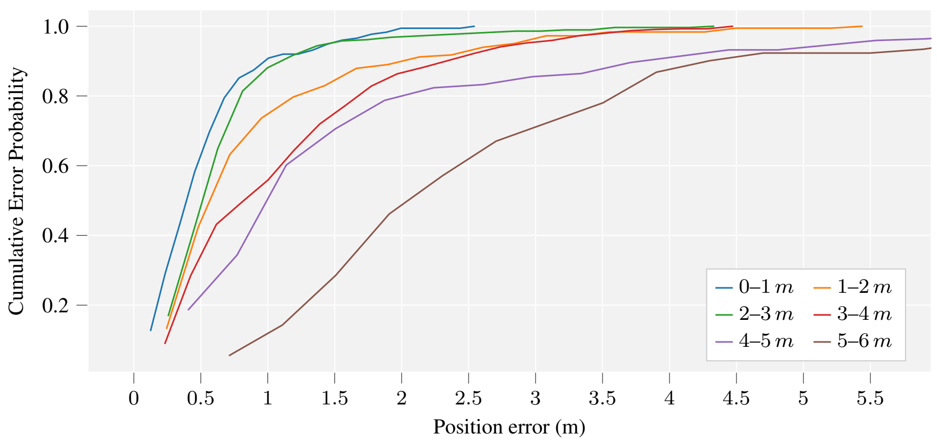


(b) Cumulative error probability for different heights.

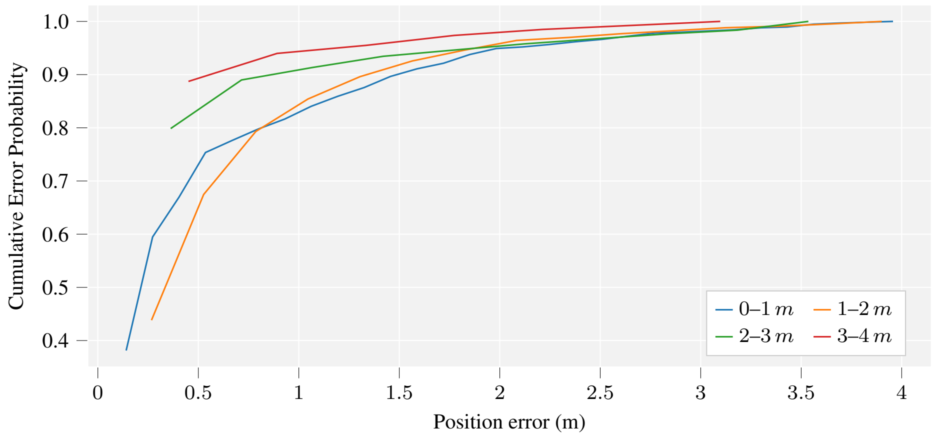
Figure 47. Positioning error probabilities for anchor positioned at $(1.79, 0.58, 0.1)$, $(0.01, 0.58, 0.1)$, $(1.8, -1.2, 0.1)$, and $(0, -1.21, 0.1)$.

the UWB localization accuracy based on the UAVs speed, height, and distance to the center of mass of the anchor system.

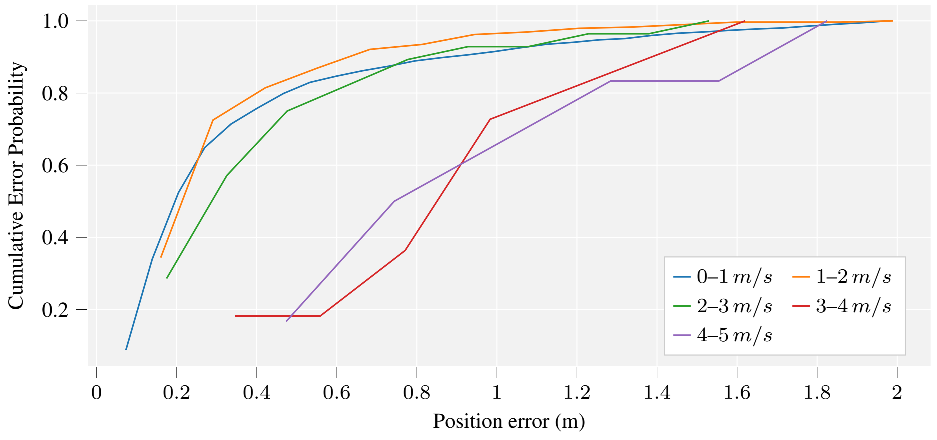
We believe that the dataset presented in this section will enable the research community to further explore the possibilities of robust and accurate autonomous flight in GNSS-denied environments with ad-hoc localization networks via a combination of UAVs with reference ground robots.



(a) Error based on the distance from the UAV to the center of mass of the anchor system.



(b) Error based on the height of the UAV.



(c) Error based on the speed of the UAV.

Figure 48. Cumulative probability distribution of the positioning accuracy of the UAV when the anchors are positioned in the room corners forming a square of $64\,m^2$.

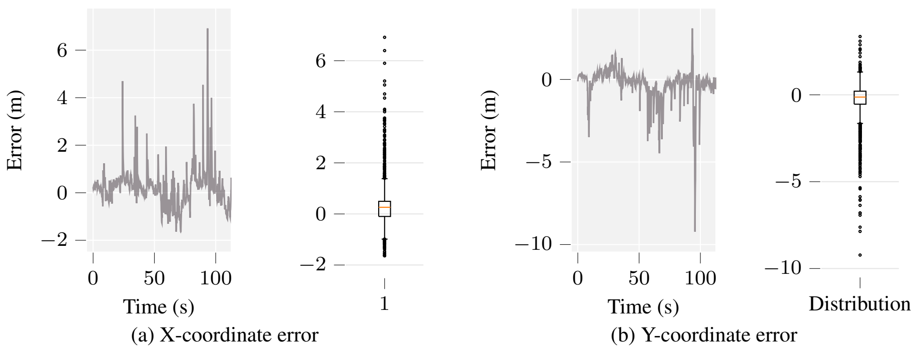


Figure 49. Localization error when all the four anchors are nearby, defining three faces of a cube. While the average error is small, and 50 % of the measurements are relatively accurate, the localization estimation is highly unstable. Further filtering is needed to enable accurate flight in this case.

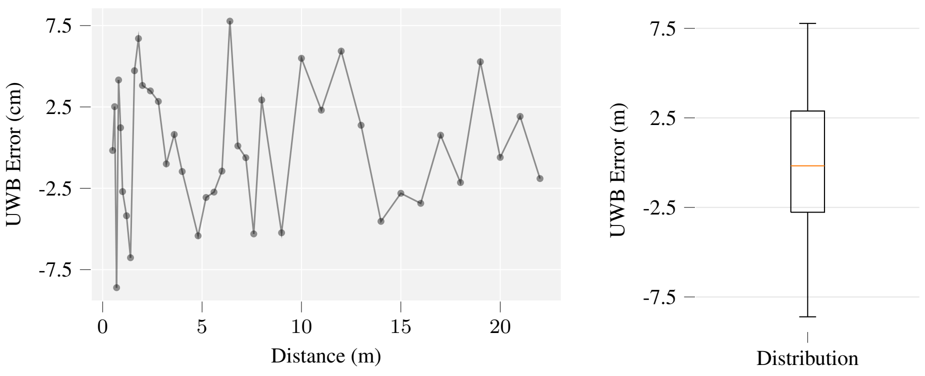
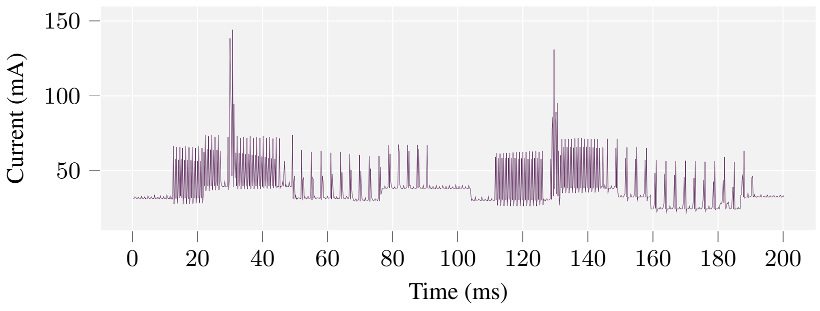


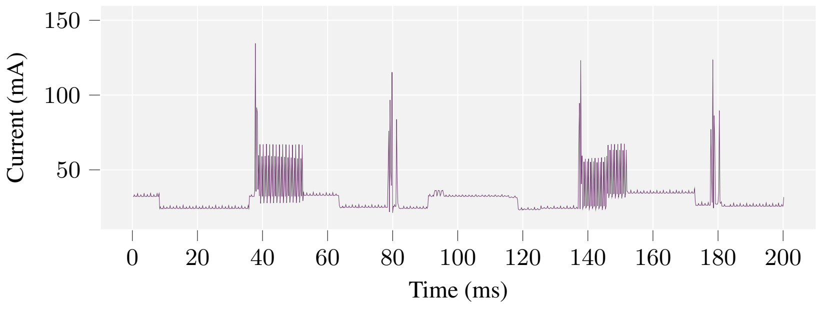
Figure 50. Average distance estimation error between two DWM1001 nodes in line of sight. The nodes have been calibrated to take into account the antenna delay. The mean error is smaller than 1 mm while the standard deviation is 3.9 cm. The maximum error is 8.6 cm.

Table 12. Power conssumption of UWB tags and anchors.

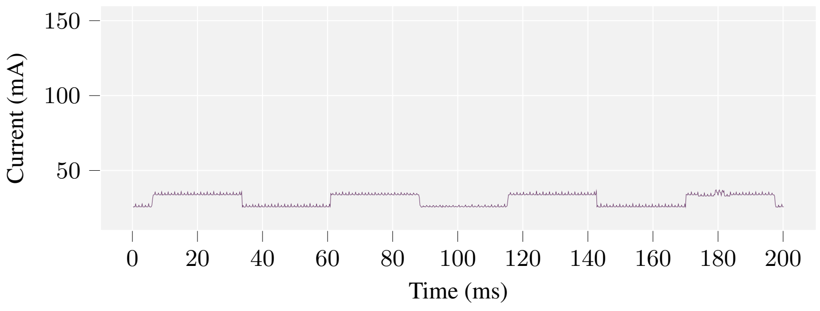
	Power (5 V supply)		Power (3.7 V supply)	
	Avg. (mW)	Max. (mW)	Avg. (mW)	Max. (mW)
Anchor	171	699	129	545
Active Tag	161	687	115	543
Passive Tag	155	189	114	503
Custom Init.	440	726	341	554
Custom Resp.	523	731	358	557



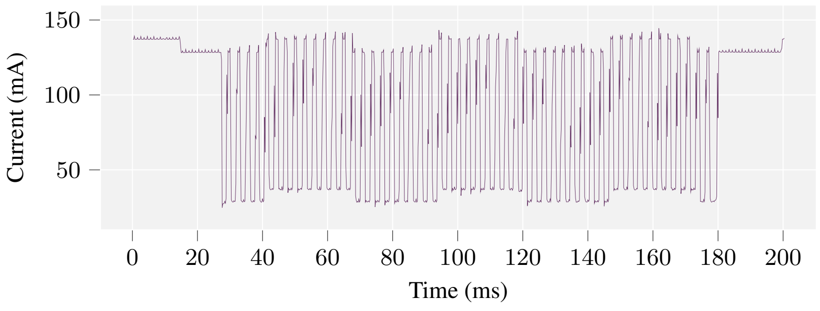
(a) DRTLS Anchor



(b) DRTLS Active Tag



(c) DRTLS Passive Tag



(d) Custom Calib. Anchor

Figure 51. Power consumption of the UWB anchors and tags in different modes. In the self-reconfigurable anchor, the communication is constant and therefore the power consumption is greater than in Decawave's DRTL setup.

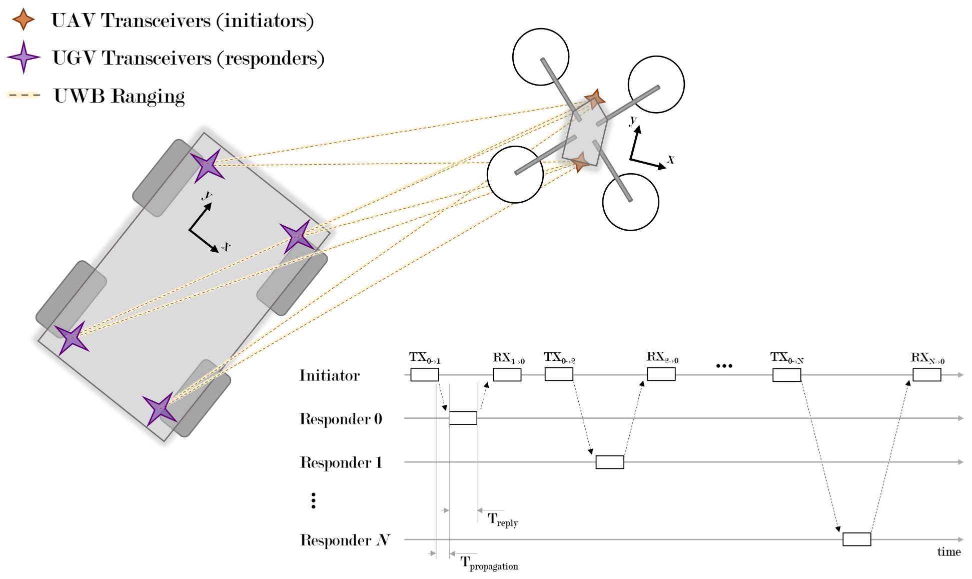


Figure 52. Cooperative localization approach based on UWB ranging measurements from multiple transceivers in different robots

5.4 Cooperative localization

In this section, we now explore the potential of relative UWB localization in environments where GNSS signals degrage. Outdoors, GNSS-RTK is the de-facto standard for gathering aerial data with UAVs [369]. For example, high-accuracy photogrammetry [370], civil infrastructure monitoring [371], or in urban environments where GNSS signals suffer more degradation [369]. As UAVs become ubiquitous across different domains and application areas [9], having access to more flexible and lower-cost solutions to precise UAV navigation can aid in accelerating adoption and widespread use. In this section, we consider the problem of UAV navigation through relative localization to a companion unmanned ground vehicle (UGV). We consider a ground robot as a more flexible platform from the point of view of deployment, but in simulations, we also consider localization based on fixed beacons in the environment, closer to how GNSS-RTK systems are deployed.

The system we analyze in this section consists of a UGV equipped with four UWB transceivers and a UAV equipped with two transceivers. The UAV transceivers act as initiators, taking turns in sending signals to each of the UGV transceivers. When these respond, the time of flight of the signal is calculated and the distance between each pair of transceivers is calculated. This process is illustrated in Fig. 52. The main contribution of this section is thus on evaluating how UWB-based relative localization can improve the positioning of UAVs when supported by ground robots. We simulate different trajectories to evaluate the performance of the system and compare the accuracy of the GNSS, UWB, and VIO approach to localization with field tests in an urban environment. In the simulations, we consider different configurations of transceivers in the ground to compare the localization and navigation performance.

5.4.1 Cooperative UWB-based localization

We consider the problem of relative localization between a UAV and a UGV based on UWB ranging between transceivers installed onboard both robots. The objective is to leverage this relative localization to improve the accuracy of the UAV navigation outdoors. We are especially interested in improving the navigation performance in urban areas where the accuracy of GNSS sensors is degraded due to the signal being reflected at or occluded by nearby buildings.

Let us denote by I the set of N transceivers onboard the UAV. These will act as initiators, i.e., will actively transmit messages to initiate ranging measurements between them and the responder transceivers on the ground. We denote the latter ones by the set R of size M. An initial approach, which we implement, is to iteratively range between each initiator and the set of responders. If the number of nodes increases significantly, more scalable approaches can be used where, for example, a single initiator message is answered by several or all responders with different delays [372].

We model the UWB ranges between an initiator i and a responder j with

$$\mathbf{z}_{(i,j)}^{UWB} = \|\mathbf{p}_i(t) - \mathbf{q}_j(t)\| + \mathcal{N}(0,\, \sigma_{UWB}) \tag{7}$$

where $\mathbf{p}_i$ and $\mathbf{q}_j$ represent the positions of the initiator and responder transceivers, respectively, and $\mathcal{N}$ is Gaussian noise. Based on the ranges, different approaches to localization include, e.g., multilateration or a least squares estimator (LE). We implement the latter, and hence the position of each tag can be calculated based on the known anchor positions by

$$\mathbf{p}_i = \underset{\mathbf{p}\in\mathbb{R}^3}{\operatorname{argmin}} \sum_{j=0}^{M} \left(\mathbf{z}_{(i,j)}^{UWB} - \|\mathbf{p} - \mathbf{q}_j\| \right)^2 \tag{8}$$

Alternatively, assuming that the position of initiators in the UAV ($\{\mathbf{p}_i\}$) is given based on the UAV's position and orientation ($\mathbf{p}$ and θ, respectively) by a set of rigid body transformations f_i, i.e., $\mathbf{p}_i = f_i(\mathbf{p}, \theta)$, then the estimator can be used to obtain the full pose of the UAV directly with

$$\mathbf{p},\, \theta = \underset{\substack{\mathbf{p}\in\mathbb{R}^3 \\ \theta\in(-\pi,\pi]}}{\operatorname{argmin}} \sum_{i=0}^{N} \sum_{j=0}^{M} \left(\mathbf{z}_{(i,j)}^{UWB} - \|f_i(\mathbf{p}, \theta) - \mathbf{q}_j\| \right)^2 \tag{9}$$

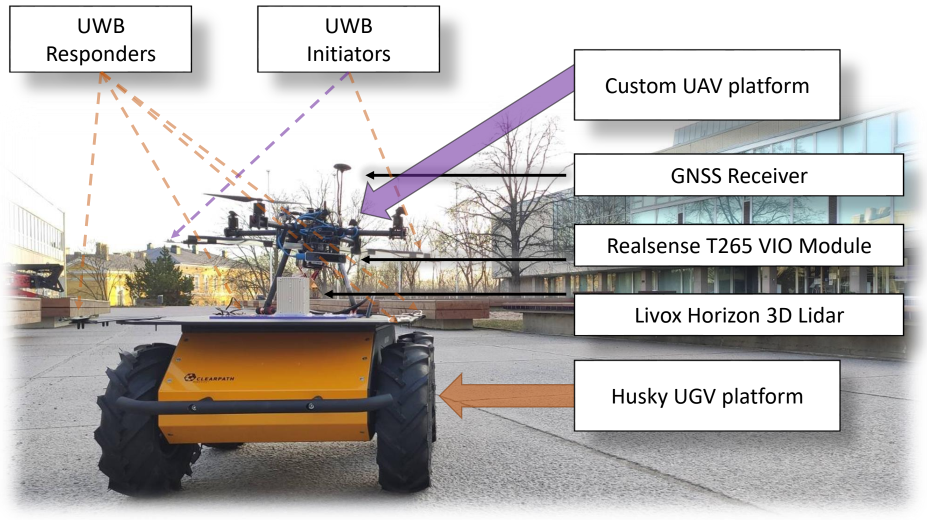


Figure 53. UAV and companion ground robot utilized in the experiments.

```
Algorithm 2: Ground truth extraction
  Input:
    3D lidar point cloud:       $\mathcal{P}$
    Last known MAV state:       $\mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1}$
  Output:
    MAV state:   $\{\mathbf{p}_{MAV}^{k}, \dot{\mathbf{p}}_{MAV}^{k}\}$
1 while new $\mathcal{P}_k$ do
      Generate KD Tree:        $kdtree \leftarrow \mathcal{P}$;
2     MAV pos estimation:      $\hat{\mathbf{p}}_{MAV}^{k} \leftarrow \mathbf{p}_{MAV}^{k-1} + \frac{\dot{\mathbf{p}}_{MAV}^{k-1}}{I}$;
      MAV points:              $\mathcal{P}_{MAV}^{k} = KNN(kdtree,\ \hat{\mathbf{p}}_{MAV}^{k})$;
      MAV state estimation:    $\mathbf{p}_{MAV}^{k} = \frac{1}{|\mathcal{P}_{MAV}^{k}|}\sum_{p\in\mathcal{P}_{MAV}^{k}} p$;
```

5.4.2 Multi-robot system

The multi-robot system employed consists of a single ground robot and a UAV. The ground robot is a ClearPath Husky outdoor platform equipped with four UWB responder transceivers for cooperative positioning and a Livox Avia lidar utilized to obtain ground truth. Owing to the lack of a reference system such as a GNSS-RTK receiver, we extract the UAV position from the lidar's point cloud and utilize this as a reference. The point cloud is automatically processed following the steps described in Algorithm 2, and manually validated. We refer the reader to [373] for further details on this method. Based on indoor testing with a reference anchor-based UWB

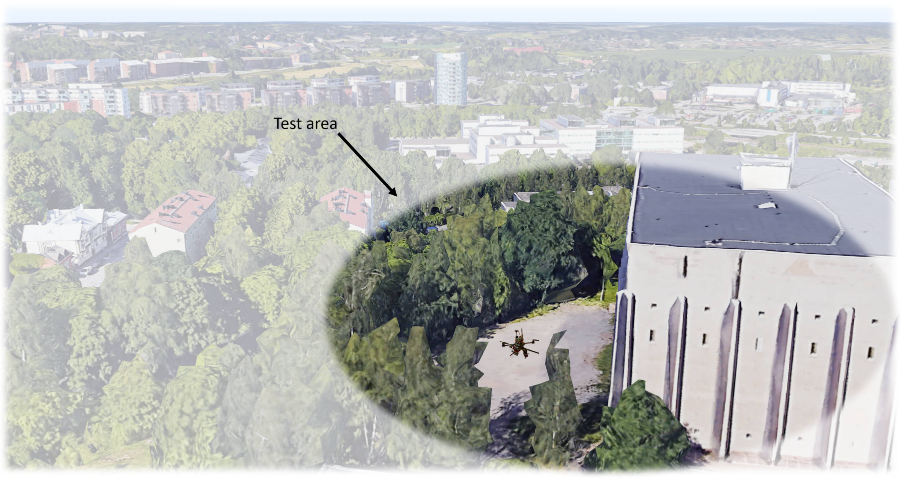


Figure 54. Test area for the outdoors experiments.

system, we have evaluated the ground truth accuracy to be in the order of 10 cm. The UGV and the custom UAV are shown in Fig. 53. The UAV is equipped with two UWB transceivers and a RealSense tracking camera T265 for VIO.

5.4.3 Experimental settings

The field experiments are carried out in Turku, Finland (the precise location can be found with GNSS coordinates 60.4557389° N, 22.2843384° E, illustrated in Fig. 54. The test site lies between a short line of trees and a large building that presumably blocks and reflects GNSS signals. The UAV runs the PX4 autopilot firmware, which is unable to obtain a stable GNSS lock in the test location. This location is chosen as an example of an urban location where GNSS receivers operate in suboptimal mode.

5.4.4 Experimental results

Results from outdoors experiments with real robots are reported in Fig. 56 and Fig. 57. The former shows a partial extract from the trajectory in 3D, where we can observe that the UWB error is significantly smaller even when the altitude reaches 30 m. In the latter plot we can see that the overall error more than 5 min flight time. The cooperative UWB approach particularly outperforms both VIO and GNSS estimations in terms of vertical accuracy. In terms of planar xy error, VIO is more accurate but only during the first few seconds of flight, before it rapidly loses accuracy and diverges when the UAV altitude increases. In any case, the cooperative UWB-based localization provides consistent accuracy throughout the flight and therefore has potential for better collection of aerial data through autonomous flights.

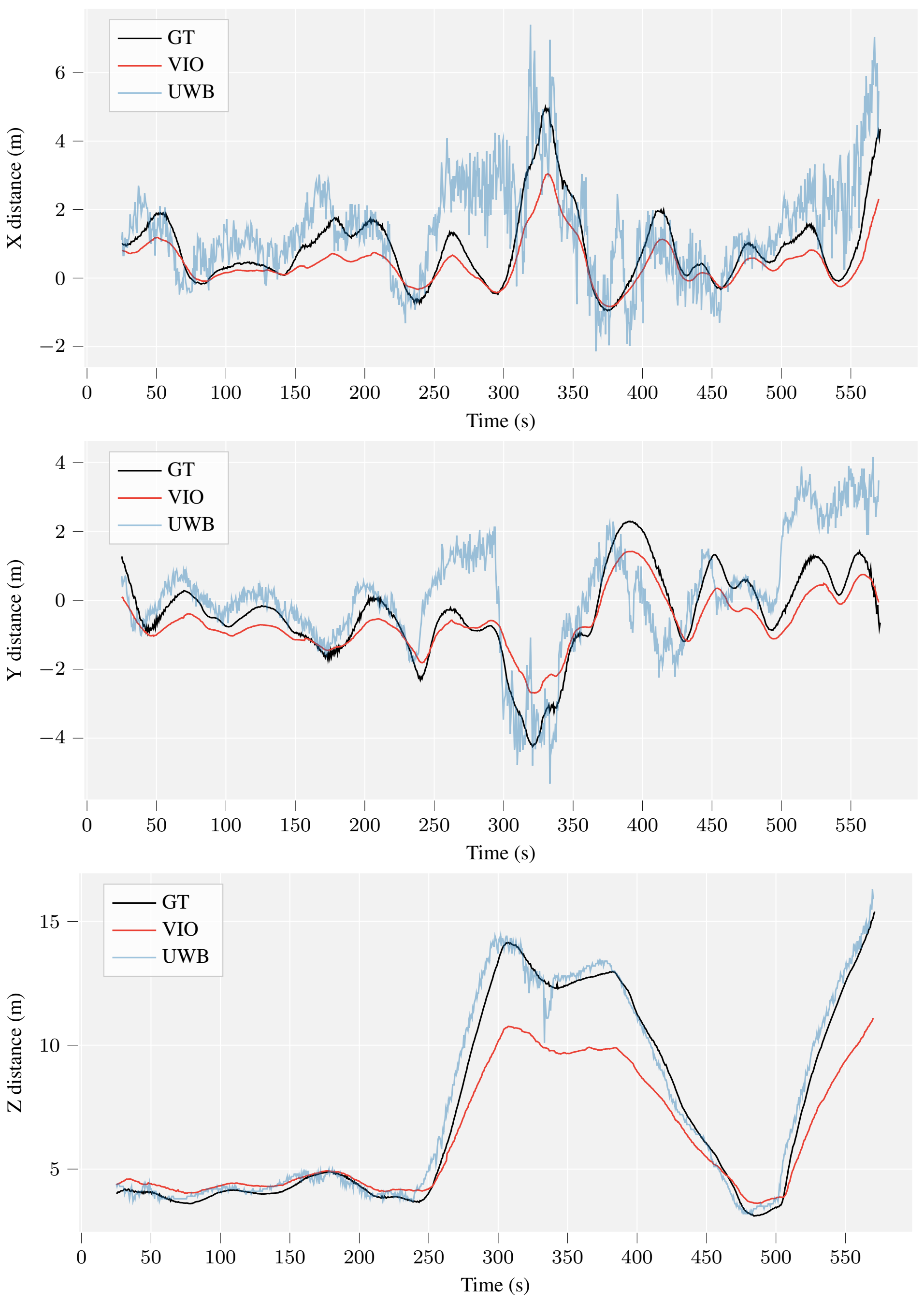


Figure 55. Extracted trajectory from experiments before the VIO estimations diverged.

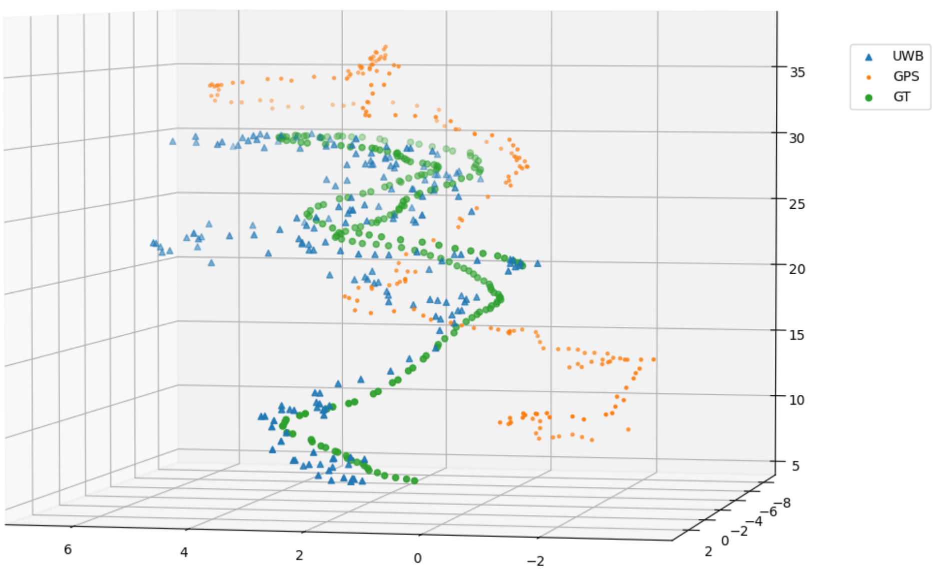


Figure 56. Partial trajectory of the UAV during the outdoors experiment. VIO is not included because it becomes unusable once the UAV reaches 8 m of altitude.

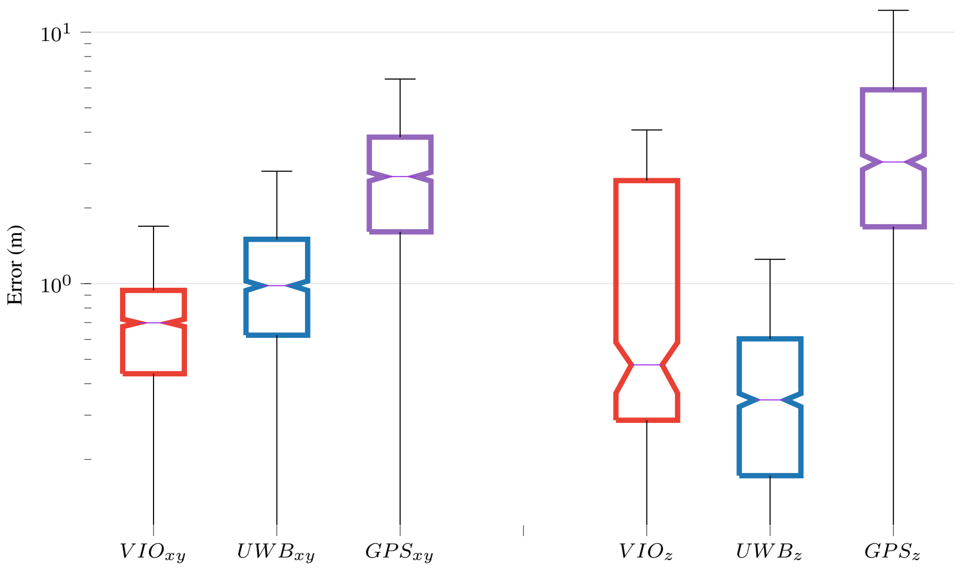


Figure 57. Planar and vertical errors of the different methods during the outdoors flight. The VIO has low error but was only valid for the first few seconds of flight. The UWB error is particularly low in the vertical dimension owing to the horizontal planar distribution of nodes mounted on the UGV.

5.5 Summary and conclusions

Throughout this chapter, we have been looking at UWB-based localization approaches for aerial robots and multi-robot systems. In the first part of the chapter, we introduced a novel dataset for anchor-based UWB localization in autonomous aerial robots, studying both the potential and the main limitations and constraints (e.g., anchor distributions and velocity constraints). In the latter part of the chapter, we have focused instead on relative UWB-based localization. In particular, we show that UWB-based state estimation has potential not only in GNSS-denied environments, but also in locations where GNSS signals degrade.

This chapter does not go deep into localization algorithms or the different approaches to using UWB ranging in multi-robot systems. Out of the scope of this thesis, in [345] we introduce a novel approach to more scalable UWB-based localization in multi-robot systems with a dynamic ToF and TDoA approach through evolving active and passive roles. This is latter integrated in [374] with blockchain smart contract for managing the role allocation process in a more secure and trustable manner.

There is indeed still significant potential and open research areas in the use of wireless ranging technologies for multi-robot systems. In other ongoing works, we are currently exploring more advanced sensor fusion algorithms for (i) suboptimal spatial distributions of nodes and (ii) single-range relative localization. The latter focuses on fusing odometry and ranging measurements, while the former solves the problem of a robot operating far outside the convex envelope of the anchor nodes.

6 Spatial Coordination in Multi-Robot Systems

The problem of formation control in multi-agent systems has drawn increasing interest of researchers in the past two decades [375; 376]. Formation control, or pattern configuration, is the basis for cooperation among a multi-agent system [377]. With an increasing number of applications for multi-agents systems, the significance in having efficient, flexible and robust formation control algorithms is evident [378].

Pattern configuration and formation control algorithms for swarms of robots have been extensively studied for the past decades [379; 380; 381; 382; 383; 384; 385; 386; 387]. We can classify pattern formation algorithms among those that require agent indexing or those in which agents are anonymous. Most of the work to date in formation control algorithms for multi-agent systems lies in the former category [388; 386; 389]. In a swarm of robots where all units are indistinguishable, a more natural approach is arguably to assume a homogeneous set of agents without identity. In practical deployments of multi-robot systems, however, different robots have identities. These identities can also play a key role in securing the behaviour of the multi-robot system and the interaction within it, as we describe in other chapters of this thesis.

In this chapter, we introduce two novel approaches to index-free and indexed formation control. The index-free approach requires only mutual sensing (e.g., relative localization estimation methods introduced in the previous chapter), without explicit peer-to-peer communication. The indexed approach requires mutual sensing in addition to mutual identification (e.g., through situated communication). Portions of text and a subset of figures in this chapter are reproduced or adapted from our previous works [5; 4; 6].

In summary, in this chapter we introduce two approaches to formation control in multi-robot systems:

- **Communication-free** formation control, where we assume robots are able to *see* each other (or at least those in their vicinity) but do not *talk* to each other.
- **Progressive** formation control with **minimal, one-way communication** where robots are able to *signal* neighbors in their line-of-sight about their *intentions* (their self-allocated role in the objective formation).

6.1 Index-free, communication-free formation control

Many efforts have been devoted to the study and replication of collective motion in nature, such as birds in a wedge formation or flocks of fishes. In these cases, agents are anonymous and homogeneous, and therefore interchangeable. Algorithms that model this behavior may use indexing for formulation but are not affected by an indexing permutation as all agents are considered equal. To the extent of our knowledge, algorithms that rely on anonymous definitions and do not require communication can only be applied in formations of trivial patterns such as flocks or regular polygons, where all positions are equivalent. In the case of flocks, the algorithms are usually based on distance-only measurements [387], while bearing-only measurements can be used to achieve regular polygon configurations [380; 381; 382]. A limitation of these algorithms is that the set of possible formations is restricted to those configurations in which all agent positions are equal. More complex configurations are possible with index-free control if communication is used throughout the system to ensure consensus [379]. Index-free refers to the fact that agents are anonymous and that a permutation of the indexes does not affect the calculation of the control inputs. However, wireless communications between robots used in these algorithms can cause a large overhead of latency and energy consumption. When the number of agents increases, this becomes a serious issue. Some research has been focused on generalizing index-free control by designing algorithms that are invariable to indexing permutations [390; 391; 392]. However, these often use an equivalence relation, or are significantly affected by the non-scalable size of the permutation space with respect to the number of agents in the configuration [393].

Distance-only measurements or bearing-only measurements also appear in the literature within algorithms that are able to achieve more complex formation configurations where the positions are not equivalent. For example, bearing-only measurements are used in [394] to allow non-trivial pattern formations. Alternatively, distance-only measurements are equally employed by [385; 389].

6.1.1 Formation definition

Formation control algorithms often define formation positions in a way that are directly related to the variables sensed by agents. Some widely used formation control algorithms are position-based, distance-based and bearing-based algorithms [386]. Position-based formation control algorithms define the positions in the formation with a set of points in space, which are assigned to individual agents. Distance-based formation algorithms (e.g. algorithms for achieving flocking formation) define a position in the formation by distances which are calculated by the agent and its neighbors. Similarly, bearing-based formation control algorithms define a position by the set of bearings, given a common orientation for the measurement. In this section, the proposed formation control algorithm is a distributed and index-free pattern formation algorithm considering the position of neighbor agents simultaneously.

Comparing to index-based formation control algorithms, an index-free algorithm has many advantages. For instance, it is challenging or even impossible for index-based robots to form an accurate formation when one of the robots cannot get into the predefined position due to a hardware failure or other errors. This issue of an inaccurate formation can be avoided with an appropriate index-free formation control algorithm as agents are interchangeable. Furthermore, the proposed algorithm is communication-free, which helps to avoid the overhead of large energy consumption and latency caused by wireless communicating between agents.

Spherical distribution

Our objective is to design an algorithm that requires, at most, position measurements, and provide a definition for a formation configuration that is independent of the agent indexing while allowing almost arbitrary patterns. Therefore, we introduce a new definition for each of the positions in the formation.

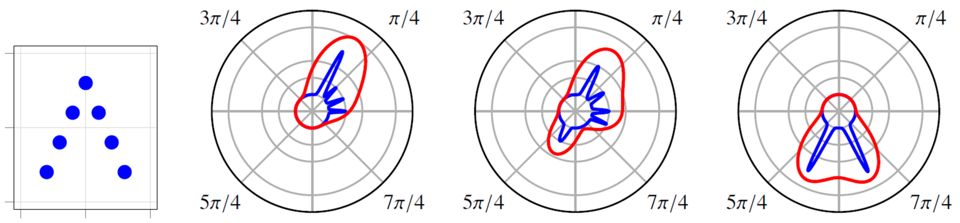
***Definition* 6.1.1** (Spherical Indicator Distribution). *Given a set of N agents with positions $(x_i, y_i, z_i) \in \mathbb{R}^3 \ \forall\, i = 1, \ldots, N$, let N_i be the set of agent i's neighbors. Their positions are measured within agent i's local reference frame and represented by $(x^i_j, y^i_j, z^i_j) \equiv (r^i_j, \theta^j_i, \varphi^j_i)\ \forall j \in N_i$ in Cartesian coordinates or spherical coordinates, respectively. Then we define agent i's spherical indicator distribution by*

$$\psi_i(\theta, \varphi) = \alpha \sum_{j \in N_i} w(r^i_j)(\theta, \varphi) * \delta(\theta - \theta^i_j, \varphi - \varphi^i_j) \qquad (10)$$

where $\delta(\theta, \varphi)$ is the Dirac delta function defined in in the subset $[0, \pi) \times [0, 2\pi) \subset \mathbb{R}^2$, $\alpha > 0$ is a constant and $w(r)(\theta, \varphi)$ is a function of the type $w(r)(\theta, \varphi) = \exp\left(-(\theta, \varphi)^T Q(r)(\theta, \varphi)\right)$ for a positive definite matrix $Q(r) \succ 0$.

The definition of the spherical distribution in (10) is inspired by an agent's view of its environment. The Dirac delta marks the two-dimensional bearing of each of the neighbors while $Q(r)$ shapes the space around those points as a function of the distance. We have defined an agent's spherical distribution over (θ, ϕ) only and not over r as well because the bearing coordinates are defined over a compact subset of $\mathbb{R}^2$. Thus, they can be represented by two-dimensional matrices if we consider a discretization of $\mathbb{R}^2$. From a computational point of view, this means a finite and fixed memory allocation. In particular, for our algorithm formulation, we define $Q(r)$ in Definition 6.1.1 as

$$Q(r) = \beta \begin{pmatrix} r^2 & 0 \\ 0 & r^2 \end{pmatrix} + \begin{pmatrix} \tau & 0 \\ 0 & \tau \end{pmatrix} \qquad (11)$$



(a) Pattern (b) —$\psi_{0|\tau=0}(\theta)$, —$\psi_{0|\tau>0}(\theta)$ (c) —$\psi_{1|\tau=0}(\theta)$, —$\psi_{1|\tau>0}(\theta)$ (d) —$\psi_{3|\tau=0}(\theta)$, —$\psi_{3|\tau>0}(\theta)$

Figure 58. Illustration of the spherical indicator distribution. Figure (a) shows a wedge configuration of 7 agents. Figures (b), (c) and (d) show the spherical indicator distribution of the first, second and fourth (top) agents, if indexed from left to right, for $\tau = 0$ and $\tau > 0$.

for constants $\beta, \tau \in \mathbb{R}$, $\beta, \tau > 0$. The value of β is the same for all agents and governs the variable width of $w(r)$ along (θ, φ), while τ adds a constant minimum width. We can think of $1/(\beta r^2)$ and $1/\tau$ as variances of two convoluted Gaussian distributions. We extend on the significance of τ later in this document. Then we can expand an agent's spherical distribution (10) into (12). illustrated in Fig. 58,

$$\psi_i(\theta, \varphi) = \alpha \sum_{j \in N_i} \exp\left(- \left(\beta {r_j^i}^2 + \tau \right) \left|\left| \left(\theta - \theta_j^i, \varphi - \varphi_j^i \right) \right|\right|^2 \right) \tag{12}$$

Autonomous Position Assignment

The first step in a formation control problem is the position assignment. In displacement-based control this is predefined beforehand, whereas in anonymous distance or bearing-based control this step is skipped due to all positions being equivalent, for instance. In our algorithm, agents perform autonomous self-position assignment as described by Definition 6.1.2.

***Definition* 6.1.2** (Position objective). *Given a desired position ψ_j^* and an agent in a position ψ_i, we define the cost of achieving the position j for agent i as*

$$D_{\psi_i}(\psi_j^*) = \int_0^{2\pi} \int_0^{\pi} \left|\left|\psi_i(\theta, \varphi) - \psi_j^*(\theta, \varphi)\right|\right|^2 d\theta d\varphi \tag{13}$$

and, therefore, an agent i decides its objective position by minimizing

$$\psi_i^{obj} = \psi_j^* : j = \underset{j=1,\ldots,N}{\operatorname{argmin}} D_{\psi_i}(\psi_j^*), \;\; D_{\psi_i}^{obj} := D_{\psi_i}(\psi_i^{obj}) \tag{14}$$

from where we define the cost of achieving the objective position

In order to achieve the objective position, we give the following problem formulation. Given a formation shape defined by a set of N positions $\{x_i\}_{i=1,\dots,N}$, and their corresponding spherical indicator distributions $\{\psi_i(\theta,\varphi)\}$, then

***Definition* 6.1.3** (Formation objective). *Given a pattern configuration defined by a set of N positions, from which we can calculate their individual spherical distributions, the desired formation shape of a group of agents is represented by the set $\mathbb{E}_\psi = \{\psi_1^*, \dots, \psi_N^*\}$ of desired spherical indicator distributions. Therefore, the formation objective is to have a permutation $\sigma \in S_N$ such that $D_{\psi_i}(\psi^*_{\sigma(i)}) < \delta$ for a constant $\delta \in \mathbb{R}$, $\delta > 0$. The value of δ is the minimum error allowed to assume a successful convergence to the desired position. The set $\{\psi_i(\theta,\varphi)\}$ represents the agents' positions and $\{\psi_j^*(\theta,\varphi)\}$ the desired positions.*

The definition for the position objective in (6.1.2) does not ensure, in its current form, that there exists a permutation σ that maps the set of agents into the set of formation positions. This section presents a preliminary exploration of the potential of the position definitions introduced in this section. The results included in this document involving simulations and tests are obtained under the assumption that the initial distribution of agents in space ensures the existence of σ.

After the autonomous self-position assignment is made by each of the agents, the next step is to minimize some cost function in order to arrive to the convergence conditions given in Definition 6.1.3. When minimizing the cost of achieving the objective position introduced in (14), however, a problem arises from the nature of the definition we are using for $\psi_i(\theta,\varphi)$ in (12). If the value of τ is too small, as happens in the example illustrated in Fig. 58, then the minimization problem can be non-convex. This might happen because $\psi_i(\theta,\varphi)$ is a function formed up by individual peaks with almost-zero intervals in between. Local minima exist when some of the peaks in an agent's spherical distribution and its objective position's spherical distribution are overlapped but others are not. Therefore, depending on the desired pattern configuration, the value of τ must be adjusted to prevent this from happening. We have developed a graphical interface for simulation purposes where we can adjust the parameters of the algorithm in real-time and during a simulation to see how this affects the position definitions as well as the algorithm performance.

Collision avoidance

So far, our algorithm does not take into account collision avoidance; for example, in the limits $r_j \to 0$ and $r_j \to \infty$, the contribution of an agent j to ψ_i is reduced to a constant over θ or a Dirac delta function in θ_j, respectively. In the limit $r_j \to 0$, this might lead to collisions between agents in the case of distributions in which the angular positions of neighbors are densely distributed across all θ. In that case, ψ_i would be an almost-constant function, and the error incurred when inter-agent is

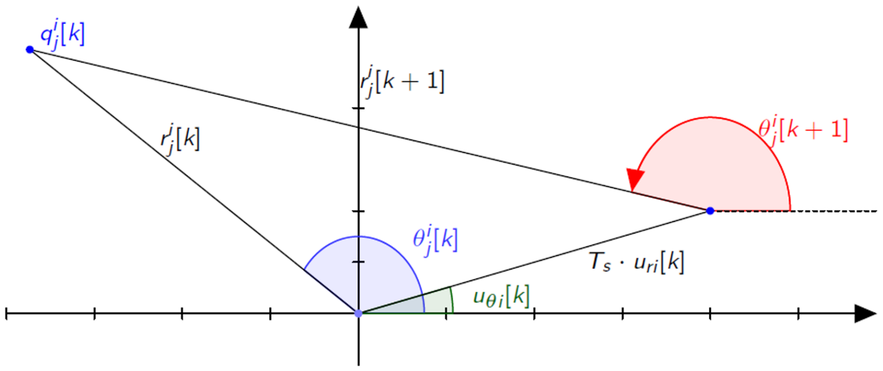


Figure 59. Illustration of the calculation of $r_j^i[k+1$ and $\theta_j^i[k+1]$.

reduced would be negligible.

In order to introduce collision avoidance to our algorithm, we consider the following modification of the spherical indicator distribution for a position. Suppose an agent is considered to move safely when its distance with respect to all other agents is bigger than a threshold r_s. An agent is considered to be in collision danger if its distance with respect to a neighbor is below a threshold r_d. Then we can modify (10) into

$$\begin{aligned} \psi_i(\theta,\varphi) &= \sum_{j\,\in N_i} \alpha_j \exp\left(-\left(\beta {r_j^i}^2 + \tau\right) \|(\theta-\theta_j^i, \varphi-\varphi_j^i)\|^2\right), \\ \alpha_j &= \begin{cases} \alpha & if\;\; r_s < r_j \\ r_j/(r_j - r_d) & if\;\; r_d < r_j \leq r_s \\ \infty & if\;\; r_d \leq r_j \end{cases} \end{aligned} \tag{15}$$

where we have introduced the constant α inside the sum and now varies for each of the agents, which always ensures that $r_j > r_d$ and therefore agents are safe from collisions.

6.1.2 Control inputs

In this section, we introduce the control laws that define our algorithm for both a single and double integrator dynamics model for the agents. For simplicity, we reduce the problem from here on to the two dimensional case. Therefore, an agent's spherical distribution is given now in polar form $\psi_i(\theta)$.

First, we suppose that the dynamics of each of the agents in the swarm follow a single-integrator model given by $p_i[k+1] = p_i[k] + T_s u_i[k]$ where $p_i[k], u_i[k] \in \mathbb{R}^2$ denote the position and velocity, respectively, of agent i, at the time step k, and T_s is the sampling period. We use radial and angular control inputs $u_i[k] \equiv (u_{i_r}[k], u_{i_\theta}[k])$ that represent the speed and direction of movement, respectively. The polar input is well defined because all agents share a global orientation. Now we can calculate $r^i_j[k+1$ and $\theta^i_j[k+1]$ based on the control input for agent i, using basic trigonometry (following Eq. 16 and Eq. 17, as illustrated in Fig. 59, and define a cost function.

$$r^i_j[k+1]^2 = r^i_j[k]^2 + T_s^2 u_{i_r}[k]^2 - 2T_s r^i_j[k] u_{i_r}[k] \cos\left(\theta^i_j[k] - u_{i_\theta}\right) \tag{16}$$

$$\theta^i_j[k+1] = \arccos\left(\frac{r^i_j[k]\cos\left(\theta^i_j[k]\right) - T_s u_{i_r}[k]\cos\left(u_{i_\theta}[k]\right)}{r^i_j[k+1]}\right) \tag{17}$$

***Definition* 6.1.4** (Cost function)*. Given an agent with dynamics described by a single integrator, its position represented by $\psi_i(\theta)$, and its position objective by $\psi_i^{obj}(\theta)$, then we define its cost function at a time step k by*

$$J_i[k] = \int_0^{2\pi}\left(\psi_i - \psi_i^{obj}\right)^2 d\theta + \gamma||u_i||^2 = D^{obj}_{\psi_i}[k] + \gamma u_{i_r}[k] \tag{18}$$

where $\gamma > 0$ controls the weight of the closed loop control.

From Definition 6.1.4 we propose a control law that minimizes (18). First, we use numerical methods to calculate the angular control input, obtaining $\psi_i\left[k+1 \mid u_{i_\theta}[k] = \widetilde{\theta}\right]$ from (16) and (17). Regarding the control input u_{i_r}, we propose a control law based on the instantaneous position error.

$$u_{i_\theta}[k] = \widetilde{\theta} : \widetilde{\theta} = \underset{\widetilde{\theta}\in[0,2\pi)}{\operatorname{argmin}} \int_0^{2\pi}\left(\psi_i\left[\text{k+1} \mid u_{i_\theta}[k] = \widetilde{\theta}\right] - \psi_i^{obj}[k]\right)^2 d\theta \tag{19}$$

$$u_{i_r}[k] = \nu \frac{D^{obj}_{\psi_i}}{10\delta + D^{obj}_{\psi_i}} \tag{20}$$

where δ is the maximum error allowed for each position to assume that the system has converged to its formation objective, as defined in the problem formulation in Definition 6.1.3. The factor 10 introduced in the equation is motivated by a preferred faster convergence of $u^2_{i_r}$ to 0, with respect to the position error term of the cost function (18).

If we suppose that the dynamics of each of the agents follow a discrete double-integrator model, then the position of a agent at the next time step is given by $p_i[k+1] = p_i[k] + T_s q_i[k]$, $q_i[k+1] = q_i[k] + T_s u_i[k]$, where $p_i[k]$, $q_i[k]$ and $u_i[k]$ are agent i's position, velocity and acceleration (control input), respectively. Then, we adapt the cost function (18) to take into account the agent's velocity as $J_i[k] = D^{obj}_{\psi_i}[k] + \gamma_r u_{i_r}[k] + \gamma_\theta \left|u_{i_\theta}[k]\right|$. Now u_{i_r} and u_{i_θ} represent the radial and angular acceleration, respectively. We take into account the radial acceleration to avoid rotations. In this case, instead of defining the control inputs at each time step by a closed formula, we adapt the desired angular speed from (19) into

$$\begin{aligned} q_{i_\theta}[k] &= \widetilde{\theta} : \widetilde{\theta} \\ &= \underset{\widetilde{\theta} \in \left[q_{i_\theta} - u_{i_\theta^{max}}, q_{i_\theta} + u_{i_\theta^{max}}\right)}{\operatorname{argmin}} \int_0^{2\pi} \left(\psi_i\left[k+1 \mid u_{i_\theta}[k] = \widetilde{\theta}\right] - \psi_i^{obj}[k]\right)^2 d\theta \end{aligned} \tag{21}$$

where we have only changed the minimization interval by adding the maximum allowed radial acceleration and taking into account that the operations $u_{i_\theta} - u_{i_\theta^{max}}$ and $u_{i_\theta} + u_{i_\theta^{max}}$ are modulus 2π. Equivalently, we calculate the ideal radial speed using (20) and then limit the radial acceleration. Then the control inputs are simply $u_{i_\theta} = q_{i_\theta}[k] - q_{i_\theta}[k-1]$ and $u_{i_r} = q_{i_r}[k] - q_{i_r}[k-1]$.

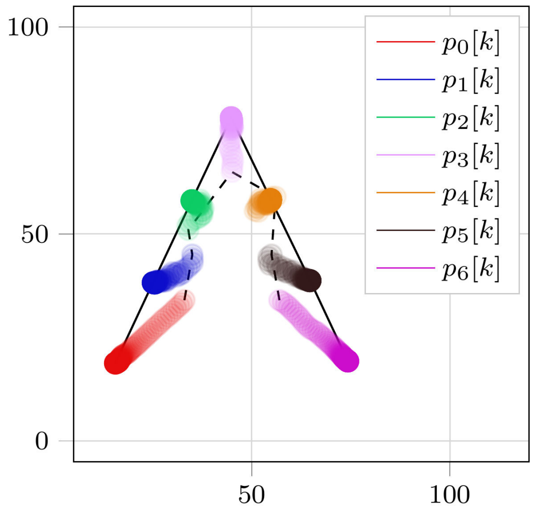
6.1.3 Numerical analysis

In this section, we show the computational simulations results that we have obtained after implementing our algorithm for non-trivial T-shaped and wedge formation configurations. The algorithm has been implemented using Python. The implementation models the robots as single points in space.

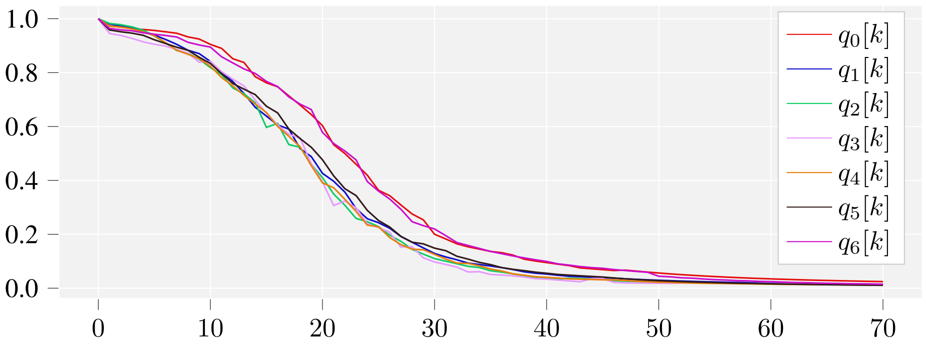
Figs. 60 and 61 show the result of two simulations. The wedge formation is achieved using a single integrator model for the dynamics of the agents, while the T-shaped formation is achieved with a double integrator dynamics. We have tested both single and double integrator models for those two and other formation configurations, including a square, a square with a fifth agent in the center, and a diamond. We have found little difference in the performance of the algorithm under the single and double integrator models. The main difference can be seen in Fig. 60 (b) and Fig. 61 (b), where we can see that there is an acceleration limit.

We set the maximum speed to $\nu = 1$, and the maximum position error allowed to assume convergence is set to $\delta = 1$. This means that when the position error of an agent is $D^{obj}_{\psi_i} = 1$, then its radial speed is $q_{i_r} = \nu \cdot D^{obj}_{\psi_i}/(10 + D^{obj}_{\psi_i}) \approx 0.11$. We can see that this effectively happens in Fig. 61, (b) and (c), around iteration 50, in which the total error of the system is approximately $\sum_i D^{obj}_{\psi_i} \approx 5$ and the speeds are $q_{i_r} \approx 0.1$.

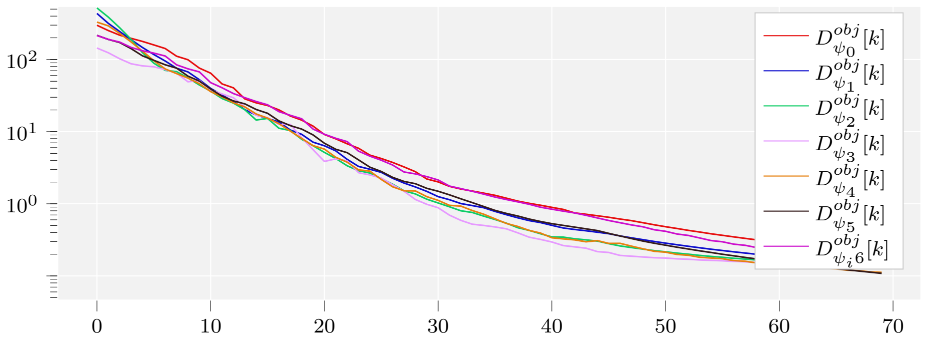
We also partially test the role of the neighborhood set, N_i, in the convergence of the algorithm. In the wedge configuration simulation, we assume that agents are able to sense the position of all other agents. In the case of the T-shaped configuration, we introduce a sensing radius, and define each position in the formation based only on the *visible* positions. Throughout the simulation, we impose that Agents 0, 3 and



(a) Single integrator paths.

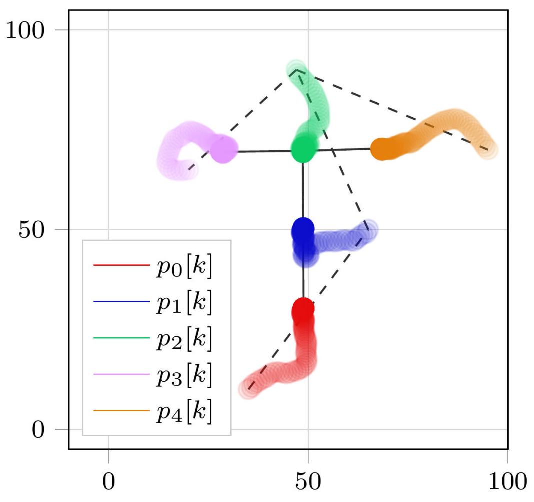


(b) Single integrator speeds.

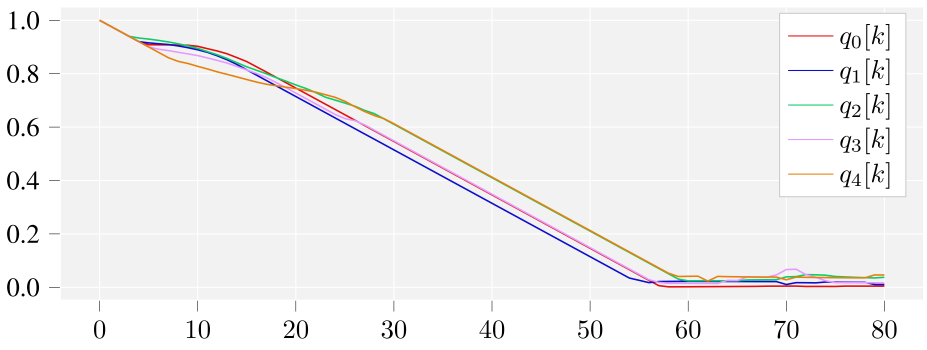


(c) Single integrator individual errors

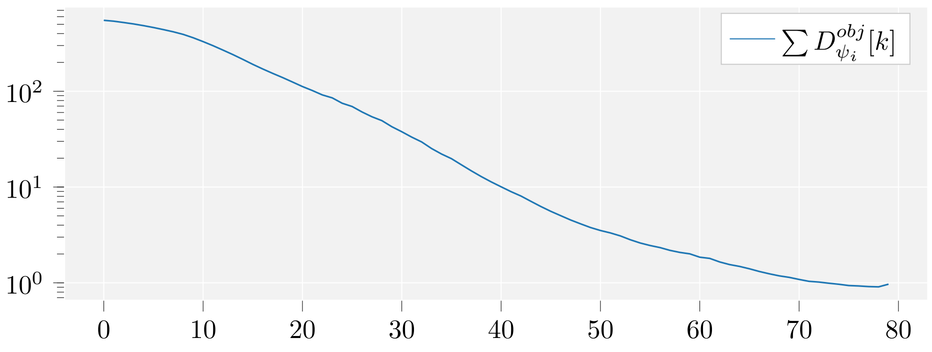
Figure 60. A wedge configuration of 7 agents is achieved, where the movement of the agents is displayed with increasing opacity over time.



(a) Double integrator paths.



(b) Double integrator speeds.



(c) Double integrator system error.

Figure 61. A T-shaped configuration of 5 agents is achieved using double integrator models.

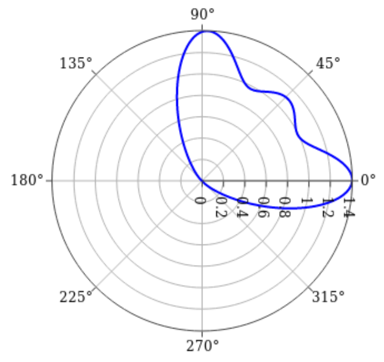
4 are not able to *see* each other, and the same for Agents 0 and 2. Agent 1 is the only one able to sense the position of all other agents. The definitions that we have given so far do not take into account the possibility that the number of neighbor positions that is used to define a given position might differ from the number of agents that an agent trying to reach that same position is able to sense. This is an important aspect of the algorithm under study and more results will be reported in future works.

6.1.4 Testing in a lab environment

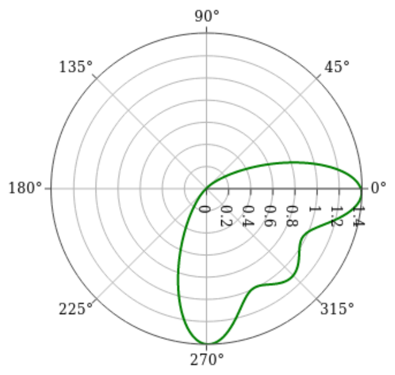
We have made an initial test of the proposed algorithm with real robots. In order to analyze the effectiveness of our algorithm in a real scenario, we use a small RC car, a radio controlled car where we have replaced the radio receiver by a Raspberry Pi. The motion of the car is controlled via two servo motors, one for the turning direction and one for speed control. The turning direction is limited to the interval $(-0.35rad, 0.35rad)$, of about 40 degrees, 20 in each direction. This is the value of the maximum radial acceleration $u_{i_\theta^{max}}$. While each iteration of the algorithm needs a relatively low execution time, reading and processing the lidar data takes more computing resources. Moreover, we run a web server on the Raspberry Pi in order to have real time monitoring and visualization of the lidar data and agent view, as well as calculated control inputs. As a result, instead of keeping the car moving in a continuous way, we move the car in steps. The car speed given by the control input is translated into the distance that is moved in each step. This can be improved with a more powerful computing platform.

To obtain the position of other cars we use a RPLiDAR A1M8 lidar with a range of 12m that offers a 360-degree view of the car environment. In order to calculate the car orientation, we use a column in the testing environment as a reference. We calculate the instantaneous car orientation supposing that the two walls closer to the column represent north and east directions. In a real scenario, potentially unknown, agents would be equipped with digital magnetometers or other sensors in order to ensure a shared global orientation. However, the settings we use are enough for the purpose of the experiment.

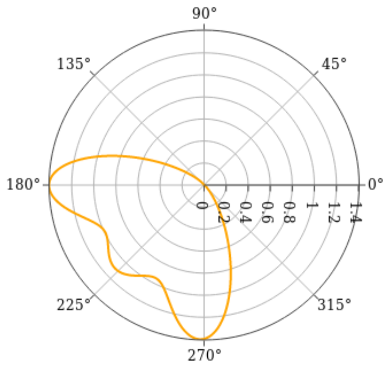
The results of our test are shown in Fig. 63. We use 4 cars and choose a square pattern configuration as the formation objective. The first four graphs, (a) to (d) show the definition of the four positions in the square. The lab environment is illustrated in (e), with the white column used as an orientation reference. The column is not used for localization but only for aligning the orientation of local reference frames. We follow the movement of a car placed near the center of the room. The initial position of the car and the other three cars can be inferred from the initial lidar view shown in (f), where the car being monitored is in the center of the graph. The four cars are initially placed in a nearly-triangular configuration where one of the cars is near the center of the other two.



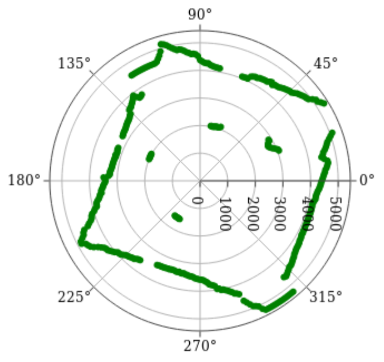
(a) Position 0



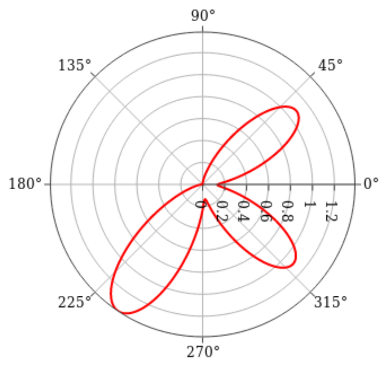
(b) Position 2



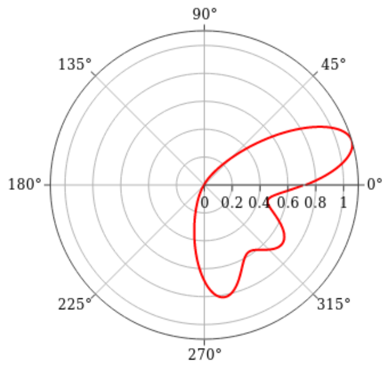
(c) Position 3



(d) Lab environment

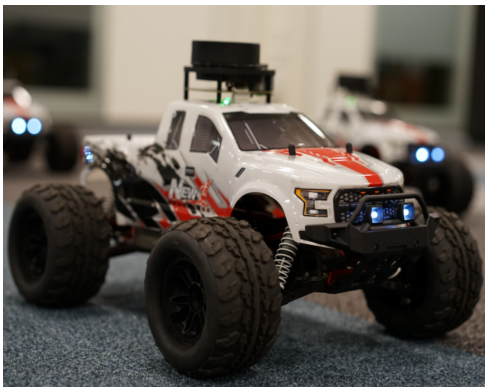


(e) Lidar output



(f) Initial Car Vision

Figure 62. Experiment with the lidar-equipped RC cars. Figures (a) to (c) show the definition of 3 positions in a square configuration (they are all equivalent with respect to a rotation). Figure (d) shows the lidar output, with one point for each degree, for a car placed near the center of the room detecting three other cars and the reference column. Figures (e) and (f) show the view of the car, $\psi(\theta)$, at iterations $it = 0$ and $it = 5$, respectively.


(**a**) Car unit.

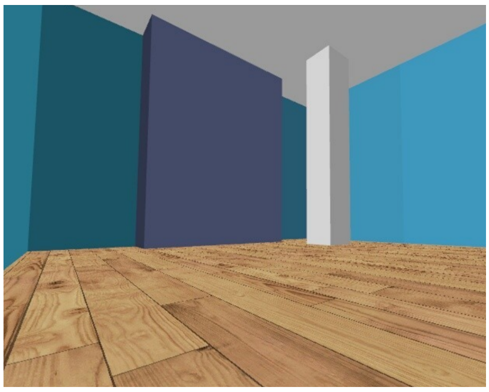

(**b**) Test environment.

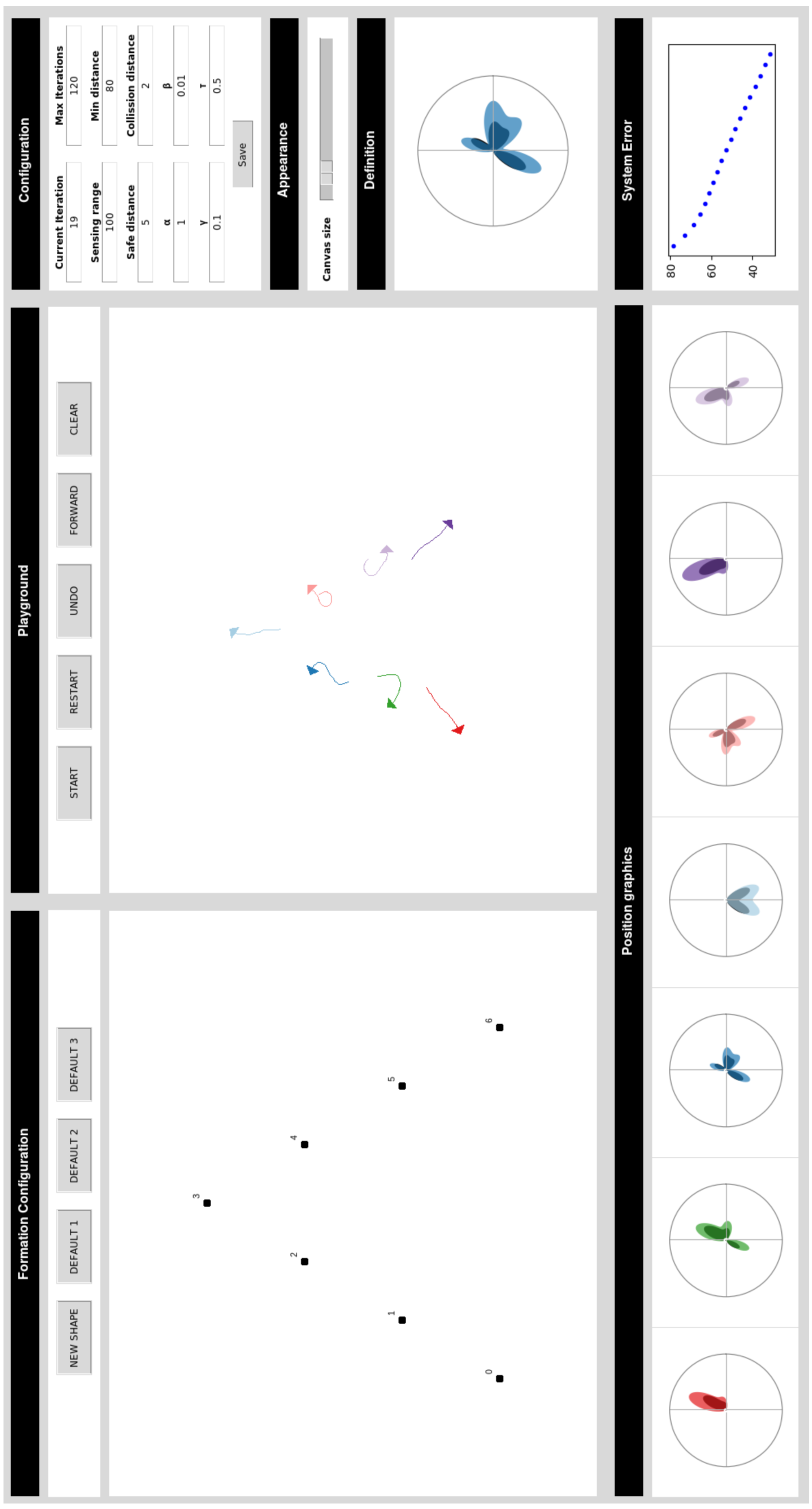
Figure 63. Experiment setup: one of the four cars used in the experiments equipped with a 2D lidar and 3D model of the test environment.

Due to the non-negligible size of the cars compared to the room space, and the fact that they are not small robots that can turn around while keeping their position, the formation can only be achieved to some extent, and the maximum required error is achieved in just 5 iterations. Graphs (g) and (h) show the initial and final agent view after those 5 iterations, where the car is trying to achieve the position defined in (b). This experiment proves that convergence to the desired configuration is possible in a real scenario without communication or predefined assignments. The angular distribution of the four position definitions makes the role assignment simpler in this case, but more complex configurations have been simulated. Future work will include extensive testing in larger environments and more complex formation configurations. We will also take into account more realistic models for the agent dynamics.

6.1.5 Discussion on index-free formation control

In this section, we have proposed a formation control algorithm that enables almost arbitrary shapes to be formed up without the need of communication between the agents and without identifying each of the agents with a unique label. This algorithm is based on an index-free definition for a position within a formation that requires distance and bearing measurements to express the position of a neighbor agent in spherical coordinates. Moreover, the definitions can be adapted to avoid collision between agents.

The algorithm introduced in this section consists of a set of preliminary ideas and is in an early stage of development. Therefore, further development is required for a robust formulation and implementation. We have assumed the existence of a



Configuration

Current Iteration	19
Max Iterations	120
Sensing range	100
Min distance	80
Safe distance	5
Collission distance	2
α	1
β	0.01
γ	0.1
τ	0.5

Figure 64. Screenshot of the simulation program developed in order to ease the adjustment of model parameters for different types of formation configurations.

permutation that ensures a surjective mapping to the set of desired positions. We base this assumption in the use of an initial distribution that naturally produces such surjection. Moreover, the role of the neighborhood set in the convergence has not been studied in depth.

To the extent of our knowledge, other distributed algorithms that are able to achieve wedge configurations, or a T-shaped formation, require either communication among the agents to achieve consensus or agent labeling and a predefined assignment of the positions in the formation to specific agents. The algorithm presented in this section is inspired by the anonymous nature of a homogeneous system. It only requires measurement of the position of other agents, and a shared global orientation. The former can be obtained through multiple solutions such as lidars or cameras together with computer vision, while the latter can be easily achieved by using a digital compass or similar sensors.

6.2 Progressive formation control

We propose a distributed formation control algorithm that requires no a priori position assignments and minimal one-way communication between agents. Positions are assigned autonomously by the agents in a progressively through a directed acyclic graph.

Different distributed approaches exist depending on the communication within the agents, and the a priori information given such as predefined position assignments. If no assignments are made, when no communication is allowed, only formation configurations where all positions have an equivalent definition in terms of their neighbors is possible. This is the case of flocks, where distance between neighbors is constant, or regular polygons, where the angle between the positions of the two nearest neighbors is also constant [387; 383]. When agents can achieve consensus through communication, then arbitrary formation shapes are possible. This can be done either by using system-wide communication to achieve consensus or via local interactions only. In the former case, auction mechanisms have been used to perform task allocation, which is equivalent to assign positions in a formation [395; 384]. Regarding the latter scenario, a very interesting approach has been proposed by Pinciroli *et al.*, in which positions are assigned progressively via local interactions between neighboring agents [396; 397]. The authors propose a solution that is easily scalable, robust, and specifically suited for the natural scenario where agents are deployed progressively and not at once. Moreover, only minimal communication is required at the time of joining the formation, and a given agent only needs to exchange information with two agents that are already part of the objective configuration. Compared to their approach, we propose a method that reduces further the amount of necessary communication. The proposed algorithm only requires one-way communication by broadcasting status information. However, this simplification limits the number of possible configurations. We prove that if the set of initial positions of deployed agents is a *convexly layered set* with a constraint below the agents' sensing range, then agents converge to the objective formation. This mainly requires that agents can be assigned to the vertices of a series of convex polygonal layers, such that they follow an inclusion relation, and all pairs of agents forming a polygon edge are within sensing range.

The progressive position assignment inherently introduces latency in the system when compared to algorithms that do not require communication between agents as it has been presented previously in this chapter [5]. However, it also allows us to ensure the existence of consensus during the position assignment and enables the convergence to almost arbitrary configurations. Distributed formation control algorithms that require, to some extent, communication between agents and measuring the relative positioning of neighboring agents often rely on situated communication [398; 164]. This type of communication refers to data exchanges using wireless technology and devices capable of estimating the relative position of a message sender. Two-way situated communication enables mutual localization of agents. Other approaches have been proposed for very-large-scale systems where individual positions are not assigned but instead a certain number of agents must be located in a particular area or volume in space. This idea, introduced by Bandyopadhyay *et al.*, uses probability distributions to define the formation configuration [379]. Even though agents do not communicate to achieve consensus regarding their objective positions, they need to be aware of the global spatial distribution of the swarm.

Our work in this section has been partially inspired from previous works from Pinciroli *et al.* [396; 397]. Nevertheless, the differences are significant. First, we propose a method that requires only one-way communication. Second, we also assume that agents share a common orientation reference. This, which can be implemented via inexpensive magnetometers or other kind of inertial sensors or digital compasses, enables us to define configurations with a predefined orientation. This has an important impact when agents should move towards a given direction after or while converging towards the objective configuration. Furthermore, our algorithm does not require of a predefined pair of identified agents which are necessary for assigning the rest of roles in Pinciroli's work. Instead, we propose a method that enables autonomous self-assignment of all roles or positions in the objective pattern.

The main contributions of this section are the following: (i) the definition of an algorithm that enables distributed and autonomous position assignment in multi-robot systems with one-way communication; (ii) the introduction of a control law that ensures convergence and utilizes the information acquired during the position assignment; and (iii) the simulation and analysis of multiple scenarios and pattern configurations showing that our proposed approach is scalable and. A more realistic simulation has been implemented and analyzed using the Robot Operating System (ROS), the Gazebo simulator and the RotorS module for dynamics modelling. These tools enable a more advanced modelling of unmanned aerial vehicle (UAV) dynamics and simulation and have allowed us to demonstrate the efficiency and applicability of our algorithm. Nonetheless, in this section we focus on the theoretical foundations of our algorithm and on analyzing its performance and scalability for different objective patterns and initial distributions of an increasing number of agents.

6.2.1 Problem Formulation

In this section, we use the following notation. We use $[N] = \{k \in \mathbb{Z}^+ : k \leq N\}$ to denote the set of the first N positive integers. Given a set of vectors $x_1, \ldots, x_N \in \mathbb{R}^n$, $\mathbf{x} = [x_1^T, \ldots, x_N^T] \in \mathbb{R}^{nN}$ denotes the stacking of the vectors. Given a set of points in $\mathbb{R}^n$, the convex envelope, hull or closure is the smallest convex set containing all points. We denote by $Conv(\mathbf{q})$ the convex hull of $\mathbf{q}$ and by $\delta Conv(\mathbf{q})$ its boundary. The algorithms presented in this section are formulated for two-dimensional formation control, and therefore we implicitly assume $n = 2$ and all points belong to $\mathbb{R}^2$.

Consider a planar formation configuration defined as a set of points, represented by a set $\mathbf{q} = [q_1, \ldots, q_N] \in \mathbb{R}^{2N}$. Given a set of N agents with positions $\mathbf{p}(t) = [p_1(t), \ldots, p_N(t)] \in \mathbb{R}^{2N}$, we address the problem of achieving a spatial distribution equivalent to $\mathbf{q}$ with respect to a translation.

***Problem* 6.2.1** (Formation Objective). *Given an objective point set $\mathbf{q}$ and a set of agents represented by their positions $\mathbf{p}(t)$, we consider that the formation has been achieved at a time $t = t'$ if a permutation $\sigma : [N] \to [N]$ exists such that*

$$\begin{gathered} \|p_i(t') - p_0(t') + q_{\sigma(i)} - q_{\sigma(0)}\| < \varepsilon \\ \|\dot{p}_i(t')\| < \delta \end{gathered} \tag{22}$$

for predefined constants $\varepsilon, \delta > 0$ that represent the maximum error allowed for positions and speed. We assume that agents are able to measure the position of any other agent in line of sight up to a predefined sensing distance δ_s.

In order to solve Problem 6.2.1, we provide a methodology for uniquely assigning a position in the formation to each agent through the definition of a directed path graph. One-way communication is used to progressively assign positions in the formation. The definition of the path graph requires the following two conditions, where the sensing range of agents is taken into account.

***Assumption* 6.2.1.** *The set of points can be divided in a set of L convex polygons, or layers, $\{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_L\}$ such that the polygonal areas they delimit $\{A_{l_1}, \ldots, A_{l_L}\}$, defined as compact sets in $\mathbb{R}^2$, follow a strict inclusion relation $A_{l_1} \supset A_{l_2} \supset \cdots \supset A_{l_L}$. Any two consecutive points in a given layer are separated by a distance smaller than the agents' maximum sensing distance, $\|l_{ki} - l_{k(i+1 \mod L_k)}\| < \delta_s$. We denote*

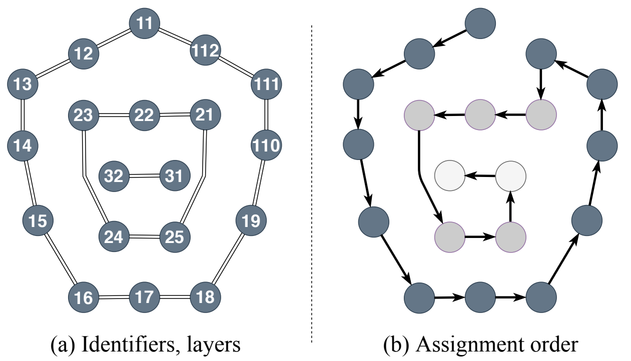


Figure 65. Representation of convex layers in a 2D point set and SDPG generation. Non straight connections between nodes are merely illustrative.

by L_k the number of points if the k-th layer, and define $\boldsymbol{l}_k = \{l_{k1}, \dots, l_{kL_k}\}$ in an order such that any pair of consecutive points $(l_{ki}, l_{k(i+1 \mod L_k)})$ defines an edge of the convex polygon.

***Assumption* 6.2.2.** *For each point l_{ki}, $1 \leq k < L$, there exists $1 \leq j \leq L_{k+1}$ such that $\|l_{ki} - l_{(k+1)j}\| < \delta_s$ and $\|l_{ki} - l_{(k+1)(j+1 \mod L_{k+1})}\| < \delta_s$. Equivalently, we assume that for each point l_{ki}, $1 < k \leq L$, there exists $1 \leq j \leq L_{k-1}$ such that $\|l_{ki} - l_{(k-1)j}\| < \delta_s$ and $\|l_{ki} - l_{(k-1)(j+1 \mod L_{k-1})}\| < \delta_s$. This essentially implies that, for each point, there are at least two points in the previous and next layer within sensing range, unless the point is in the first or last layer. If there is a single point in the innermost layer, then at least three other points must be within sensing range. Intermediate layers cannot have one or two points only, as any three points form a triangle and all triangles are convex polygons.*

The parameter $\delta_s > 0$ is a lower limit of the agents' sensing range. This value is different in each application scenario and should be chosen lower than the real range to ensure agents are able to sense their neighbors consistently through time.

The first layer $\mathbf{l}_1$ is the boundary of the convex hull or convex envelope of the point set $\mathbf{l}$. Therefore, we can easily calculate the points in any layer using the following relations:

$$\mathbf{l}_1 = \delta Conv(\mathbf{q}), \;\; \mathbf{l}_k = \delta Conv\left(\mathbf{q} \setminus \bigcup_{1 \leq k' < k} \mathbf{l}_{k'}\right) \;\forall 1 < k \leq L \tag{23}$$

The convex hull of a finite point set in $\mathbb{R}^2$, or $\mathbb{R}^3$, can be calculated in $\mathcal{O}(n \log h)$ time with Chan's algorithm, where h is the output size, i.e., the number of points defining the convex hull [399].

***Definition* 6.2.1** (Convexly layered set)**.** *Given a set of points in the plane, represented by a stacked vector $\boldsymbol{q} = [q_1^T, \dots, q_N^T] \in \mathbb{R}^{2N}$, and a distance $\delta_s \in \mathbb{R}$, $\delta_s > 0$,* *we define the pair $(\boldsymbol{q}, \delta_s)$ as a convexly layered set with constraint δ_s if Assumption 1 and Assumption 2 hold.*

We should note than given any set $\mathbf{q}$, there always exists a value δ_s large enough such that $(\mathbf{q}, \delta_s)$ is a convexly layered set with constraint δ_s. This can be easily proven from the definition of $\mathbf{l}_k$ in Eq. 23, as the layers can be calculated first and then the minimum sensing range is obtained from the conditions in Definition 6.2.1. Figure 65 (a) shows three convex layers for a set of 19 points. The first digit of each node's identifier references the layer number $\{1, 2, 3\}$ and the rest of digits are unique for each layer, with layer sizes $\{12, 5, 2\}$. This identifiers have been merely chosen for illustration purposes; in a real application, a sequence of increasing natural numbers can be used as agents are given a priori information about the objective configuration. This information can then include the number of agents in each layer.

We can now define a directed path graph on the point layer that uniquely assigns identifiers to each point progressively. The directed path graph is generated as follows. First, a node is chosen as the graph root. Any edge node can be chosen at this point. A node is an edge node if it is a point $q_i = (q_{i_x}, q_{i_y})$ that belongs to the convex hull and there exist constants $m, n \geq 0$, defining a line $f(x) = mx + n$, and constants $s_1, s_2 \in \{-1, 1\}$, such that the edge node q_i belongs to the line, all other points in the set belong to only one of the two half-planes defined by the line, and other points that belong to the line belong to only one of the two half-lines in which q_i divides the line. Mathematically, this means that

$$\begin{aligned} q_{i_y} &= f(q_{i_x}) \\ q_{j_y} &\leq s_1 f(q_{j_x}) \;\; \forall j \in [N],\; j \neq i \\ \forall j \neq i \mid q_{j_y} = f(q_{j_x}) &\Longrightarrow q_{j_x} < s_2\, q_{i_x} \end{aligned} \tag{24}$$

Second, a clockwise or counterclockwise *assignment direction* is chosen. This direction is used in all layers to define the order in which identifiers are assigned to points in the set. Finally, starting at the graph root, the next node in the graph is one of the two points that share an edge with the root in the outer layer, chosen accordingly with the assignment direction. The assignment continues iteratively in the same direction through the outer layer until all points in the layer have been assigned an identifier. When the last point in the first layer has been identified, the assignment continues to inner layers by choosing the closest point in the next layer and repeating the same process as in the first layer. This process continues until all points in the set have been assigned a position.

***Definition* 6.2.2** (SDPG from a convexly layered set)**.** *Given a convexly layered set $(\boldsymbol{q}, \delta_s)$, we define a Spiral Directed Path Graph (SDPG) following the next steps:*

1. Choose an edge node as the graph root, and constants m, n, s_1, s_2 that uniquely define the point within the set $\boldsymbol{q}$. The constants m, s_1, s_2 will be used by *agents to self decide whether they are the root node after deployment. The value n is not necessary because we consider any configuration equivalent with respect to a translation. Therefore, any line from the set of parallel lines defined by m can be used to uniquely identify the root, together with constants s_1, s_2.*

2. Choose an assignment direction, clockwise or counterclockwise.

3. Identify the nodes following the assignment direction, starting from the root node, and following the previous indications when all nodes in a layer have been identified.

A path graph is a tree where only two nodes have degree one, and all other nodes have degree two. Because the SDPG is a directed graph, the root and terminal nodes have degree one. All nodes except the root have a parent node, and all nodes except the terminal have a child node.

To solve Problem 1, we first assign a unique identifier to the positions in a given formation configuration by generating an SDPG. Then agents perform a progressive self-assignment of positions until an equivalent SDPG is generated from the moment they are deployed. This analog process is defined in Section III. At this point, agents actively control their position with respect to the nodes they are connected with in the SDPG according to the displacement defined in the objective formation configuration. The desired pattern is achieved when all agents error are below a predefined threshold. However, agents are not aware of the global error due to the lack of communication, and local errors are used instead to estimate the global system state.

Figure 65 (b) shows the SDPG generated from a given set of points as the desired formation configuration. The edge node is given by constants $m = 0, n = q_{1_y}$, defining a line $f(x) = q_{1_y}$, where q_1 is the point with identifier *11*. The half-plane where all other points lay is defined by $s_1 = 1$. The value s_2 is not relevant in this case because there is no other point in the line, i.e., $q_{j_y} \neq q_{1_y} \forall j \neq 1$. However, the origin choice might affect the graph generation in the agent set when assigning positions in the corresponding SDPG. Therefore, even if not significant, a value for $s_2 \in \{\pm 1\}$ must be chosen. The assignment direction is counter-clockwise in this case. We use the term *spiral* to refer to the directed path graph because of the decreasing distance to the center of mass when considering different layers, even though this does not necessarily happen within a certain layer.

6.2.2 Progressive Position Assignment Algorithm

In this section, we describe a position assignment algorithm that only requires one-way, minimal communication between agents. One-way communication is used to enable the assignment of positions in a random spatial distribution of agents.

Suppose a set of N agents is given with positions represented by $\mathbf{p}(t) = [p_1^T(t), \dots, p_N^T(t)] \in \mathbb{R}^{2N}$ and maximum sensing range δ_s is given. We now define an objective formation configuration by a point set $\mathbf{q} = [q_1^T, \dots, q_N^T] \in \mathbb{R}^{2N}$. We assume that the set $\mathbf{p}(0)$, together with the agent sensing range δ_s, is a convexly layered set $\{\mathbf{p}(0), \delta_s\}$.

During the position assignment process, agents can have one of three different states: **(1)** *Assigned*: agents that know their position and identifier; **(2)** *Known layer:* agents with unassigned position but that are aware of their layer; and **(3)** *Unknown layer*: agents with unknown layer.

All agents in the outer layer are in the *known layer* state immediately after deployment, and all other agents in the *unknown layer* state. Agents continuously transmit their state through a broadcast signal. Situated communication can be used to both transmit the signal and measure the position of other agents [398]. The broadcasted signal is used by neighboring agents to track the position of the transmitting agent while, at the same time, make decisions with respect to their objective position. If an agent is in the *assigned* state, then the broadcast signal includes information about its objective position. We assume that if agent i is able to receive agent j's signal at a certain time, then agent j is able to receive agent's i signal at the same time.

The progressive position assignment (PPA) then proceeds as follows: **(1)** With the same constants m, s_1, s_2 used in the definition of the SDPG from the objective configuration, each agent in position $p_i = (p_{i_x}, p_{i_y})$ checks if the three conditions in Equation 24 hold for $n = p_{i_y} - p_{i_x}m.$ with $f(x) = mx + n$. If an agent i fulfills all three conditions, with $f(x) = m(x - p_{i_x}) + p_{i_y}$, then the agent is the root node and it changes its state to *assigned.* **(2)** The root agent starts broadcasting signal. Then, there are two agents next to it in the outer layer which self-decide whether they are or not the next node in the graph based on the assignment direction. In the case of a positive decision, then the next node starts broadcasting signal as well. After all nodes in the outer layer have a position assigned, then nodes in the next layer can automatically change their state to *known layer.* **(3)** Finally, each node starts to actively control its position immediately after it is in the *assigned state.*

***Theorem* 6.2.1** (Uniqueness of the PPA). *Let $\boldsymbol{p} \in \mathbb{R}^{2N}$ represent the spatial distribution of a set of N agents with sensing range $\delta_s > 0$, and capable of one-way communication by continuously broadcasting a signal over time. If $\{\boldsymbol{p}, \delta_s\}$ is a convexly layered set, then the position assignment process defined by the previous three steps uniquely assigns a position to each agent.*

Proof. Positions are self-assigned in an autonomous way by agents, based on the positions and states of neighboring agents. Therefore, in order to prove the theorem, we first need to demonstrate that a single agent will self-assign itself the role of graph root, and that at every step the appropriate agent, and only that agent, will self-assign the next node in the SDPG.

Suppose there are two agents such that, immediately after deployment, claim that they are the root agent. Let p_i, p_j be the two positions of such agents. Without any loss of generality, we can assume $j > i$. With constants m, s_1, each of the points defines a half-plane. If both half-planes are the same, then p_i and p_j lay on the same line. If agents i and j are within sensing range, then s_2 has different value for each of them and this is a contradiction. If they are not able to sense each other, then there exists another agent in the outer layer with position p_k, $i < k < j$, such that it belongs to the half-plane defined by the p_i and p_j. If it belongs to the boundary, then the same problem arises with the sign of s_2 (and new points can be equally generated if k is not within sensing range of both i and j). If it belong to the interior of the half-plane, then the segment $\overline{p_ip_j}$ is outside of the convex outer layer, and this contradicts the definition of convex polygon and breaks Asumption 1.

If the half-planes are not the same, we can assume without any loss of generality, and based on the value of s_2, that the half-plane generated by p_i is contained within the half-plane generated by p_j. This necessarily means that they do not sense each other. Let k be now the nearest edge to i in the outer layer, such that it is between i and j. Then p_k necessarily belongs to the half-plane defined by p_i based on Eq. 24. However, then $\overline{p_ip_j}$, which lies completely outside of that same half-plane, is not within the outer layer and a contradiction arises.

The above implies that all agents in the external layer can be uniquely identified following the SDPG generation, without two agents claiming to have the same objective position. Now we just need to prove that a single agent in the next layer will claim to have the appropriate objective position. The problem can be reduced to prove that a single agent will claim to be the closest to the last agent identified in the outer layer. Let p_i be the position of the last agent assigned to the outer layer, and assume that two agents in position p_j, p_k claim to be the closest. Both p_j and p_k know that they are in the next layer in the same manner than initially all agents in the outer layer are aware of that, since all these have been already identified. If p_j and p_k are within sensing range, then they can both masure the other's distance to p_i, and only one will claim to be the closest. In case of equivalent distance, the decision is made based on the assignment direction. Therefore, for agents j and k to make the claim, they cannot be within sensing range of each other. Based on Assumptions 1 and 2, this means there is at least one other agent h in the same layer in between. However, because h is farther from i, it is outside the triangular area formed by agents i, j, k. This implies that the segment $\overline{p_jp_k}$ lays outside any convex poligonal area with vertices including p_j, p_k, p_h, that also leaves p_i outside. This contradicts the definition of convex polygon, and we started assuming that p_j, p_k, p_h.

$\square$

As previously noted, all agents except the root have a parent node, and the root can be uniquely identified. This allows us to propose a control law for each agent based on a basic leader-follower formation, where the parent is the leader and the agent itself is the follower. We assume that the parent and child nodes of any agent, which are within sensing range at time $t = 0$, stay within sensing range at any time $t' > 0$. If a new objective configuration is required, then agents loose their assignments and the process is restarted.

Deployment of 3D configurations

The position assignment described for two-dimensional configurations can be leveraged to deploy a group of agents into a three-dimensional configuration. When aerial robots such as UAVs are deployed, their initial positions are a set of points in $\mathbb{R}^2$. Therefore, for three-dimensional deployment of aerial robots, we propose to distributively perform the position assignment before they take-off, and define an equivalent directed path graph on the objective configuration.

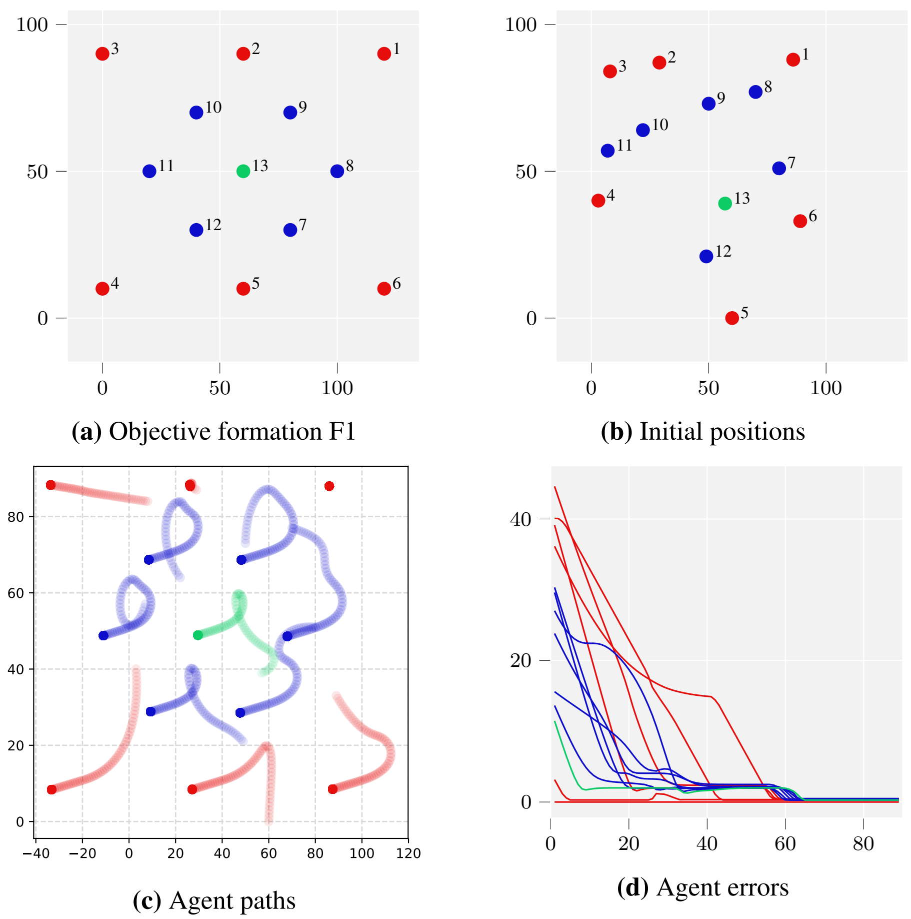
In particular, we reduce the problem to three-dimensional configurations where all points belong to the boundary of their convex hull. This is to avoid dealing with agents passing through the convex hull boundary, which might cause unpredictable behavior because of collision avoidance triggers in the case of a dense objective configuration. In this case, a sufficient condition if that the triangulation of the hull boundary must have at least one Hamiltonian path. In this case, the definitions of the directed path graphs in the two-dimensional initial distribution and the objective three-dimensional configuration are completely independent.

For these 3D formation configurations, we do not provide a methodology for identifying a locally convex directed path graph. Instead, we assume that a Hamiltonian path over the 3D hull boundary has been defined beforehand.

6.2.3 Control inputs

We propose a control law that enables agents to converge to the objective configuration while avoiding inter-agent collisions. A natural solution is for agents to actively control the displacement with respect to their parent node in the path graph. However, this has the disadvantage of increasing the system error with small drifts in the displacement of each parent-child pair. We propose a methodology for reducing the system error after individual agent errors are below a certain threshold, but its analysis is not wihin the scope of this section. The main contribution of this section is on the progressive assignment algorithm that uniquely generates a SDPG given an spatial distribution of agents, and not on the control input that enables convergence. Multiple leader-follower formation control solutions exist in the literature and can be easily adapted to work with our proposed progressive position assignment algorithm [386; 400; 401; 402].

Let $\mathbf{q}$ be a set of N points representing the objective formation, and $\mathbf{p}(t)$ the position of N agents actively trying to converge to a spatial configuration equivalent to $\mathbf{q}$ with respect to a translation $\vec{d_p}(t)$. Without any loss of generality, we assume $\vec{d_p} = q_0 - p_0(t)$ and agents are indexed such that $p_i(t) \xrightarrow[t\to\infty]{} q_i + \vec{d_p}$. Let $p^i_j(t)$



(a) Objective formation F1 **(b)** Initial positions

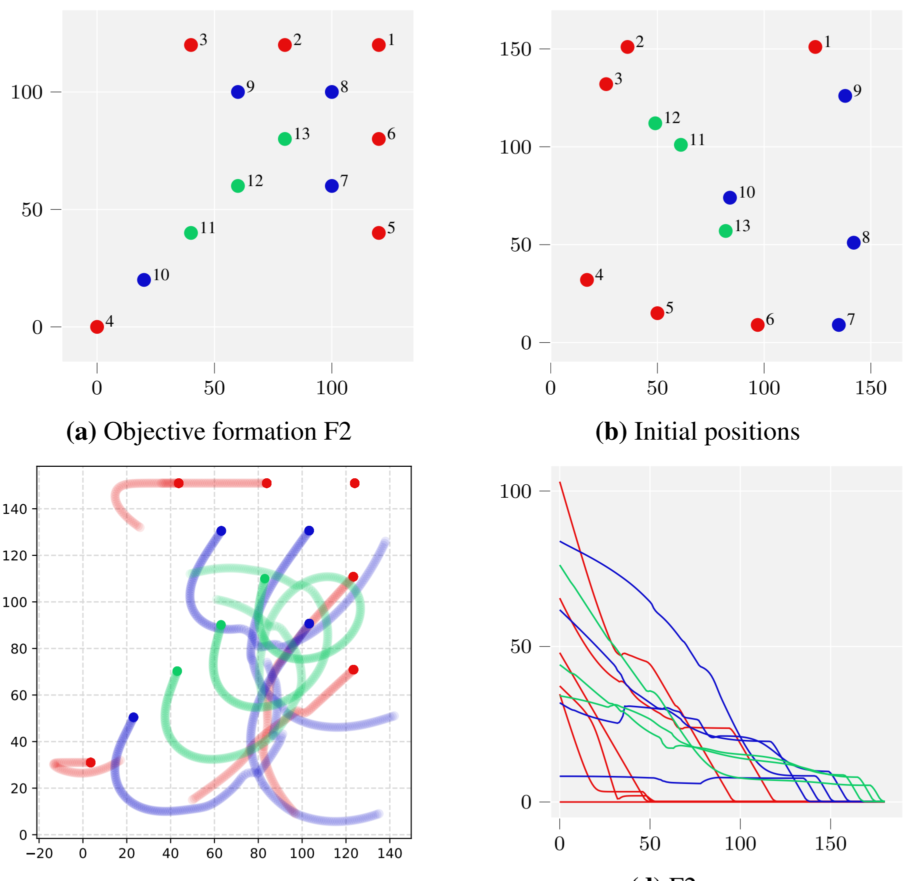
(c) Agent paths **(d)** Agent errors

Figure 66. Illustration of 13 agents in random initial positions converging towards the formation configuration F1 defined in (a).

be the position of agent j measured by agent i in its local reference frame, $p_j^i(t) = p_j(t) - p_i(t)$. Then, agent i's objective is fulfilled at time t', from the point of view of a leader-follower formation, if $p_{i-1}^i(t') = q_{i-1} - q_i$, for $i > 1$. Let $\mathcal{N}_i$ be the set of agents that agent i is able to sense. Necessarily, this set contains at least the parent node, $p_{i-1} \in \mathcal{N}_i \; \forall i > 0$.

In order to test the feasibility of this method, we propose a simple control law based on a single integrator model, $\dot{p}_i = u_i$, where u_i is the control input for agent i. For agent $i = 0$, a trivial solution is to have $u_i = 0\, forallt > 0$. For all other agents, a simple follower equation can be written generically as $u_i = \varphi(p_{i-1}^i(t) - q_{i-1} + q_i)$ such that is ensures asymptotical convergence towards the objective displacement. This function should minimize a cost function such as

$$J_i(t) = \gamma_1 \|p_{i-1}^i(t) - q_{i-1} + q_i\|^2 + \gamma_2 \sum_{j<i,\, j\in\mathcal{N}_i} \|p_j^i(t) - q_j + q_i\|^2 + \gamma_3 \|\dot{p}_i\|^2 \quad (25)$$



(a) Objective formation F2

(b) Initial positions

(c) Agent paths

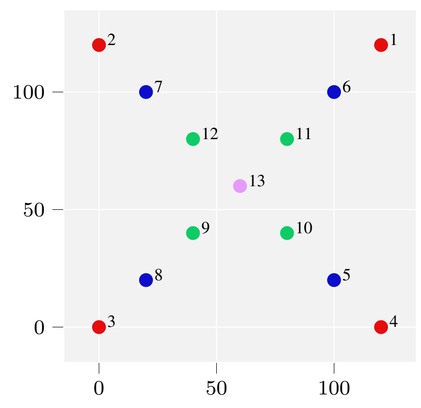
(d) F2 errors

Figure 67. Illustration of 13 agents in random initial positions converging towards the formation configuration F1 defined in (a).

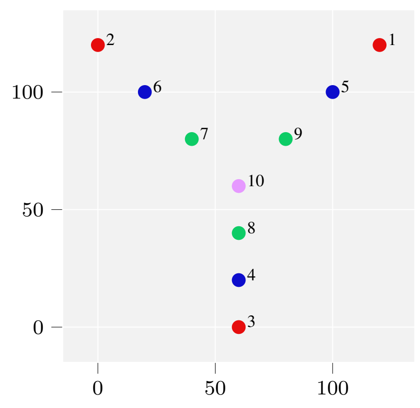
In this section, we use $\gamma_2 = 0$. However, we introduce this term as a proposal to reduce the overall system error when the parent-child displacement is already within a predefined limit. We introduce the constraint $j < i$ because the smaller the index, the smaller the agent position error due to less accumulated drift. Moreover, agents with smaller index converge faster as they are closer to the static root node from the point of view of the directed path graph. Therefore, agents can take as reference any other agents that they can sense, and that precede them in the SDPG.

For collision avoidance, we use a potential such as [403]:

$$V_{ij}(t) = \begin{cases} \left(\min\left\{ 0, \dfrac{\|p_j^i(t)\|^2 - R^2}{\|p_j^i(t)\|^2 - r^2} \right\} \right)^2 & \|p_j^i(t)\| > r \\ \infty & \|p_j^i(t)\| < r \end{cases} \tag{26}$$



(a) Objective formation F3



(b) Objective formation F4

Figure 68. Formation configurations F3 and F4.

where R, r represent the *warning* and *danger* distance, respectively. These constants are defined in a way such that an agent actively tries to avoid another agent when the distance that separates them is smaller than the warning distance, and it must never be below or equal to the danger distance. During the simulations we assume that, for any given agent other than the root, its parent agent is always within sensing range. However, this is unrealistic as even line of sight could be lost when another agent passes by through both. To solve this, since parent agents are also able to measure the position of their child agent, we propose to reduce the parent speed closer to 0 as the distance between parent and child increases towards the sensing range or a predetermined limit.

6.2.4 Point simulations

In order to test the feasibility and effectiveness of the proposed algorithm, we have run a set of simulations to analyze its performance under favorable and unfavorable conditions. Figures 66 and 67 show two example configurations, F1 and F2, with 13 agents each. Both configurations can be divided in a set of three convex layers. In both cases, random initial distributions have been used. These two examples serve as an initial illustration of the feasibility of our proposed algorithm. Figure 70 shows the number of agents that have at least one neighboring agent closer than the warning distance.

Common to all simulation results presented in this section are the following parameters. When distributing agents randomly, a minimum distance of 15 is kept between agents. The inter-agent warning distance to avoid collisions is set to 8, and the danger distance to 4. The maximum speed of agents is 1. The assignment direction is counterclockwise, and the root node is chosen with constants $m = 0, s_1 = 1, s_2 = 1$.

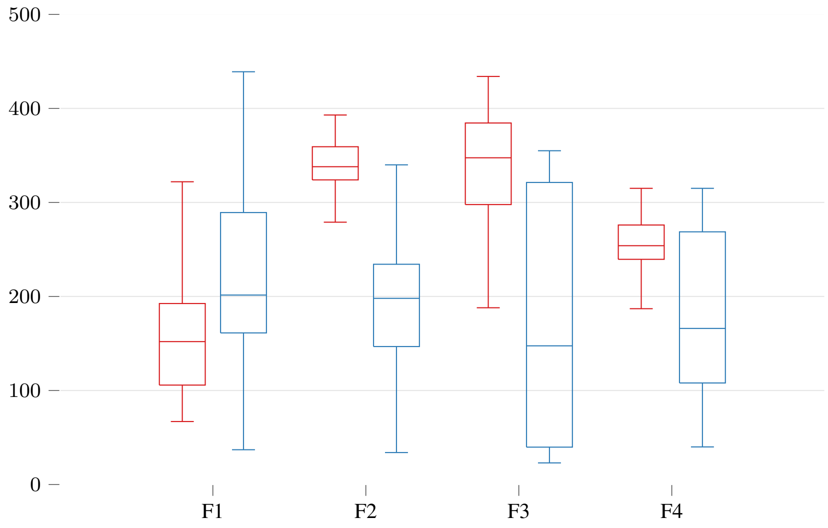
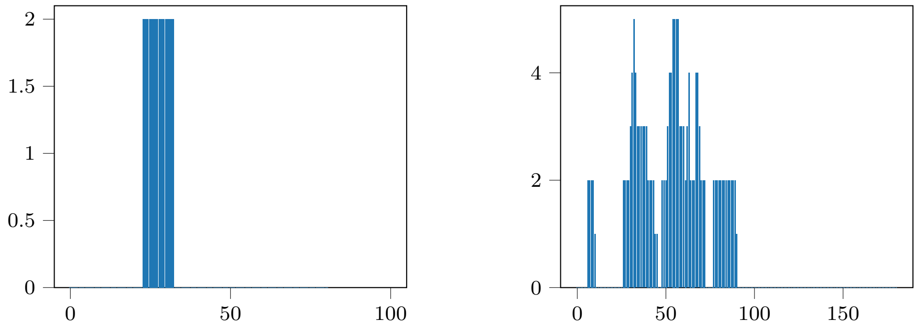


Figure 69. Boxplot illustrating the number of iterations (vertical axis) needed to converge to different objective patterns. In each simulation, the initial distribution of agents is either random (red) or has been generated adding noise to the objective distribution (blue). In the horizontal axis, each of the objective formation configurations introduced earlier.

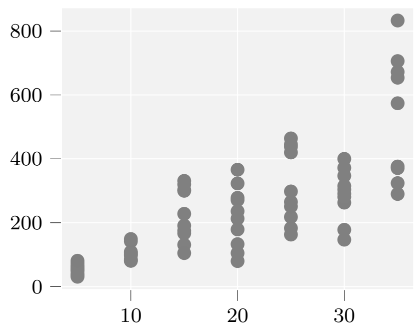


(a) Collision avoidance while converging to F1.

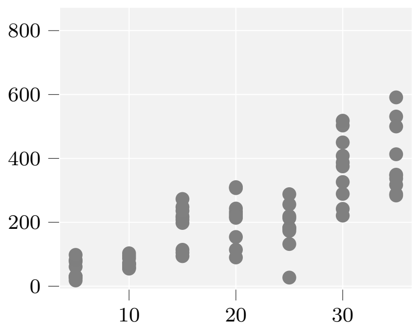
(b) Collision avoidance while converging to F2.

Figure 70. Number of occurrences when collision avoidance potential is triggered while converging to formations F1 (a) and F2 (b).

In order to avoid that agents lose sight of their parent node, we have also successfully tested a modification of the collision avoidance scheme in which roles are assigned a priority based on their distance to the root node in the graph. When the collision avoidance potential is activated, only the agent with lower priority actively avoids the collision. Moreover, if a parent node notices that its follower is getting to a distance near the maximum sensing range, it can reduce its speed until the distance is reduced.



(a) Random initial distributions.



(b) Noise addedd to a fixed initial distribution.

Figure 71. Number of iterations needed to converge to a random configuration.

In Figures 69 and 71, we analyze how the objective configuration and the initial distribution of agents affect the performance of the algorithm in terms of time to convergence. Figure 69 summarizes the number of iterations needed to converge to the objective formations F1, F2, F3 and F4 over 800 simulations, 100 for each configuration and type of initial distribution. In red is shown the results of simulations where the initial distribution of agents is random over an area similar to the objective one. Formation F1 clearly requires less time. This is due to the sparse distribution of agents in space. Formations F2, F3 and F4 show similar complexity, with F4 requiring fewer iterations presumably due to the lower number of agents. The box graphs in blue show the number of iterations needed to converge in the case in which the initial positions of agents are calculated by adding a random drift to the objective positions. In this case, the space sparsity of the formation does not play such a significant role, as the initial distribution is similar. Therefore, very similar complexity is shown by the different formations.

In Figure 71, we show the result of running 280 simulations for 2 different types of initial distributions and 7 different number of agents. The objective configuration is a random distribution of N agents over an area with side length $L_{grid} = 50\sqrt{2}\sqrt{N/5}$. In Figure 71 (a), the initial distribution of agents is also random over an area of the same size. However, in (b), agents are individually deployed nearby the objective distribution, in areas of side length 30. This resembles the idea of adding noise to the objective system. We can see that the complexity of the system rapidly grows in the case of a random distribution, while the number of iterations increases slower when the shapes are more similar. In a real scenario, a person deploying agents in an objective scenario probably has an idea of the final spatial distribution that agents will converge to. Therefore, it is natural to expect that the original distribution is not totally random and is in some way correlated to the objective distribution.

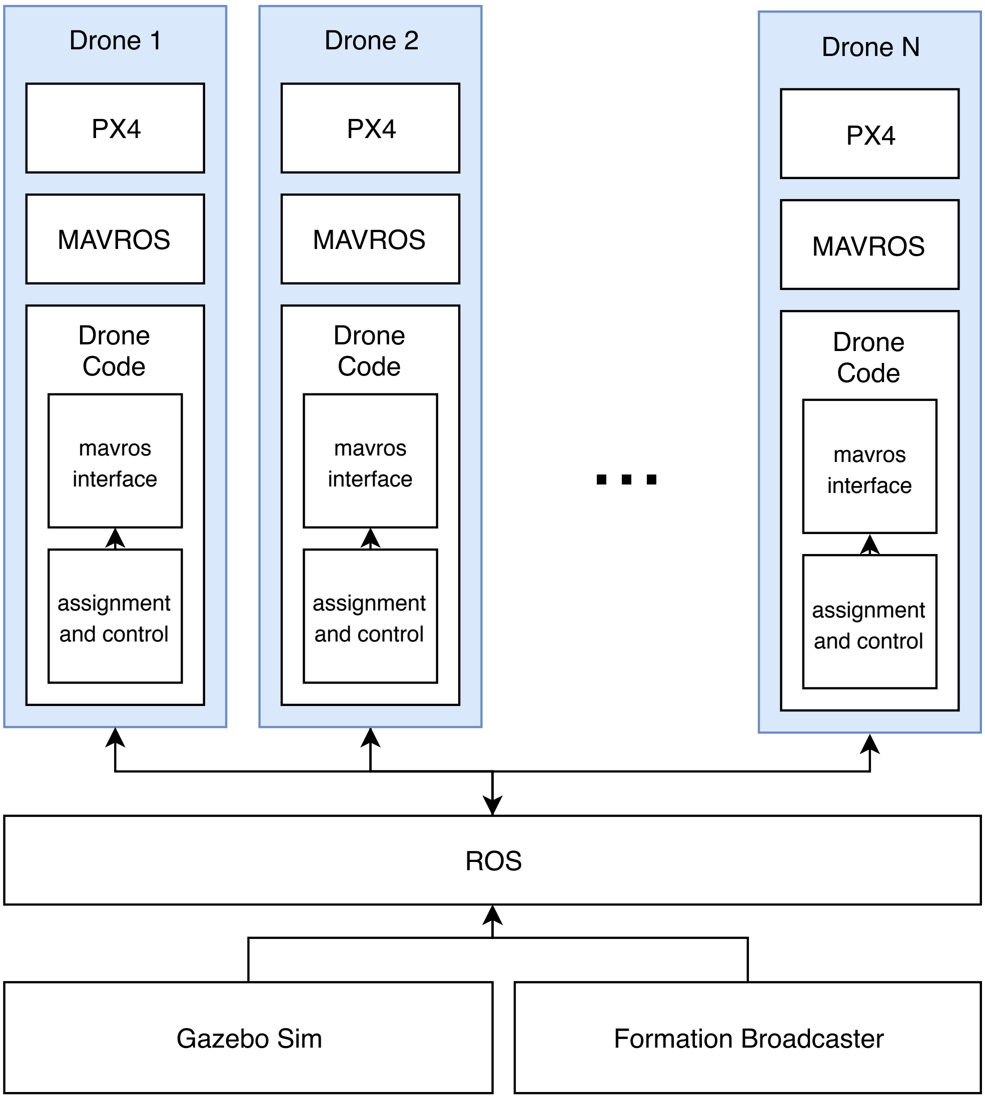


Figure 72. ROS/Gazebo implementation architecture as it appears in [6].

6.2.5 3D simulations with ROS

The previous simulations are relatively simple and use points in space that follow predefined dynamics. We have performed two different simulations with ROS and Gazebo. Traditionally, works introducing formation control algorithms simulate agents with single or double integrator models. In order to show how these models adjust to a real scenario, we have implemented both a single-integrator simulation using Python and a more realistic approach using UAVs in ROS with the Gazebo simulator and RotorS.

During the ROS simulations we assume that, for any given agent other than the root, its parent agent is always within sensing range. However, this is unrealistic as even line of sight could be lost when another agent passes by through both. To solve this, since parent agents are also able to measure the position of their child agent, we propose to reduce the parent speed closer to 0 as the distance between parent and child increases towards the sensing range or a predetermined limit.

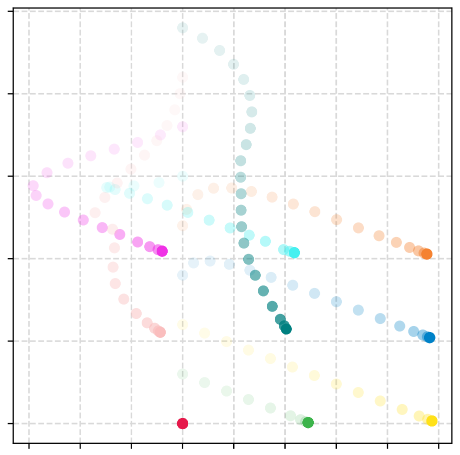
Fig. 72 shows the Gazebo simulation environment consisting of the ROS master process, the Gazebo simulator, a set of processes for each drone in the simulation and a formation broadcaster. Each drone runs a PX4 SITL, MAVROS for communicating with PX4, and the formation control process which has two threads, one for communicating with MAVROS through ROS and the other for assigning and high-level movement planning. PX4 is an open source flight controller. It can also be used for simulation in Gazebo on top of RotorS. The formation broadcaster is responsible for ordering the objective configuration. This process may be computationally expensive and therefore it is computed once and the encoded path graph is transmitted to agents upon deployment. In a real implementation, each drone would only need to run the formation control process and MAVROS since PX4 runs on separate hardware. The formation definition could either be embedded, or broadcast using other protocols. The assignments in the simulation broadcast through ROS.

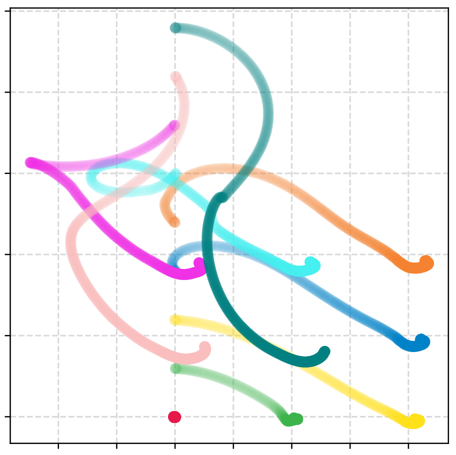
The same environment is used for both 2D and 3D simulations. For the 3D environment, the on-drone code does not require any changes since the assignment is still done in 2D. The formation broadcaster requires the most substantial changes since the algorithms for ordering 2D objective configurations do not easily translate to 3D. In this case, we have precomputed a Hamiltonian path over the triangulation of the convex hull of the objective configuration.

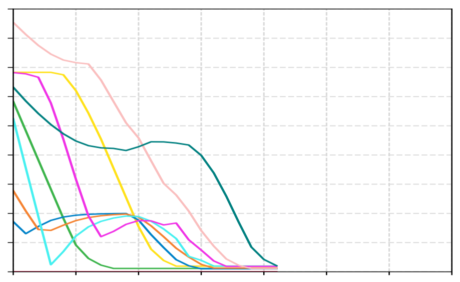
6.2.6 Gazebo simulation results

We have compared the performance of the single integrator simulation and results obtained with ROS/Gazebo in terms of convergence time and agent paths. In order to be able to compare the single integrator implementation in Python with the ROS implementation, we have adjusted the time steps and speeds to match both simulations.

Figure 73 (a) and (b) show the objective configuration and initial distribution of the agents. We chose a line along the y-axis as the initial pattern and a 3x3 grid for the objective configuration. Rather than a random initial distribution, we have chosen a line in order to show the efficacy of our method in a disadvantageous initial distribution. The paths taken by both the single integrator simulation and the Gazebo drone simulation are similar, with moments where collisions were being avoided being more clear in the Gazebo simulation. The single integrator converges much faster, owing to the simpler model used. This is shown in Fig. 73 (c) and (d), where


(a) Single integrator paths.


(b) Paths in Gazebo.


(c) Single integrator errors.

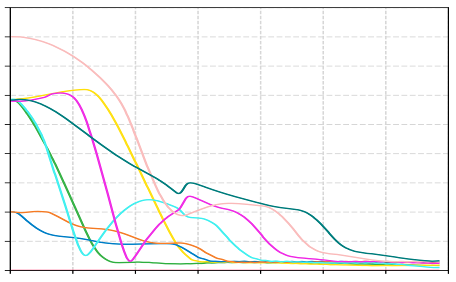

(d) Errors in Gazebo.

Figure 73. Simulation of a 3x3 grid formation in both the Python point simulation and the more realistic Gazebo simulation.

instantaneous errors of individual agents are illustrated. These errors are calculated as the norm of the difference of the current leader-follower displacement and the objective one.

Drones have an approximate size of 55 cm by 55 cm, and the size of the area displayed in Fig. 73 (a) is 9 $m \times 9$ m, (b) is 4 $m \times 25$ m and (c-d) 20 $m \times 30$ m. Although our control input to the drones are velocity that the drone is not capable of instantaneously achieving. The maximum speed was 1 m/s. The Gazebo simulation took 35 seconds to converge, which could be improved with better control. The positions of the drones were sampled at 1Hz for the single integrator simulation and 10Hz for the Gazebo simulation. Figure 74 shows the final results in Gazebo. For the collision avoidance potential, we have utilized are $r = 1.5$ m and $R = 0.75$ m.

To demonstrate the validity of our algorithm for deploying drones into three-dimensional configurations, we chose a pseudo-icosahedron as our objective config-

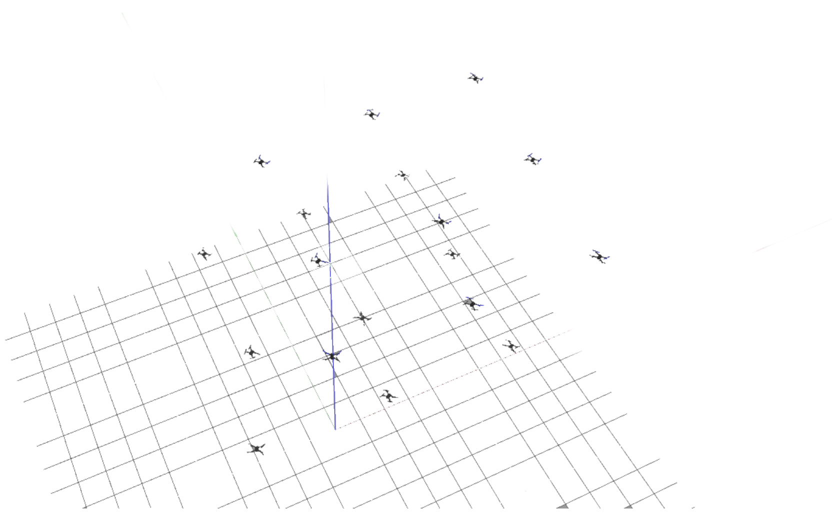


Figure 74. Final 2D configuration in Gazebo with PX4 software-in-the-loop simulations.

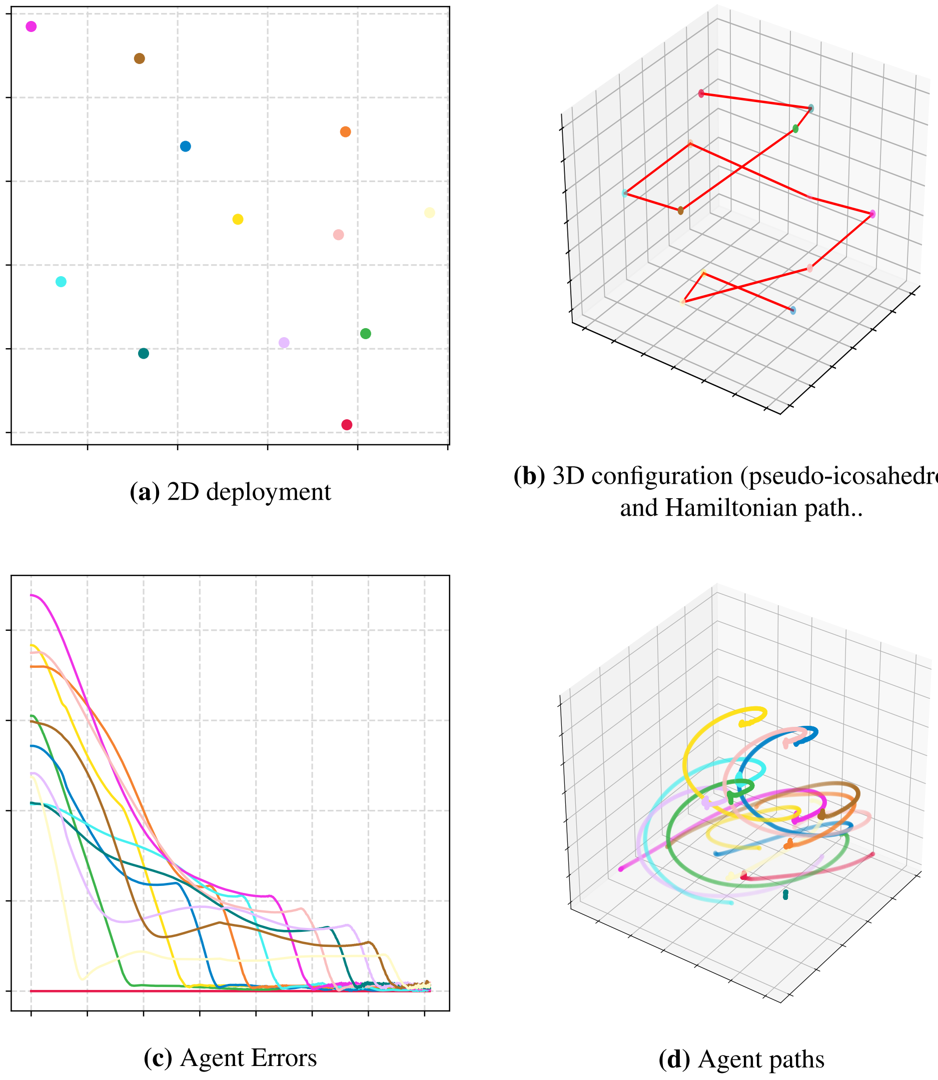
uration, which is illustrated in Fig. 75 (b). The 3D Hamiltonian path was generated manually. The initial configuration was random. It took roughly 70 seconds to converge, with the same agent speed and collision avoidance parameters used in the two-dimensional case.

6.2.7 Remarks

We have presented a distributed progressive formation control algorithm that enables a wide range of formation configurations. Any formation configuration is possible if the sensing range enables the definition of a convexly layered set. Compared to a previous progressive formation control algorithm proposed by Pinciroli *et al.*, our approach requires only one-way communication, and all agents are equivalent and anonymous upon deployment. Therefore, the proposed methodology does not require any agent to have a preassigned role or objective position. Furthermore, we assume that agents share a common orientation reference. This enables convergence to a formation configuration with respect to a translation, keeping the desired orientation. Finally, we propose a control law based on a leader-follower scheme that requires only the same information utilized during the role assignment process. The algorithm is lightweight and can be implemented in resource constrained devices.

6.3 Summary and conclusions

In this chapter, we have been looking at algorithms for formation control in multi-robot systems requiring minimal interaction. This chapter complements the previous chapter, focused on localization, with an extension to spatial coordination. It remains


(a) 2D deployment

(b) 3D configuration (pseudo-icosahedron) and Hamiltonian path..

(c) Agent Errors

(d) Agent paths

Figure 75. Simulation of pseudo-icosahedron

out of the scope of this thesis the integration of these formation control algorithms with the UWB-based localization approaches. However, it is clear that there is significant potential. For instance, UWB ranging and relative localization with basic signaling can directly be applied to the progressive formation control algorithm introduced in the latter part of this chapter. At the same time, anonymized UWB messages, even if noisy, can be used within the earlier formulation of the index-free and communication-free formation control approach. More interestingly, multiple synchronized receivers in a robot, or a receiver with multiple antennas, could be used to estimate both distance and angle of arrival, without two-way ranging. This would provide a direct way to calculate the spherical indicator distribution and enable formation control without explicit communication.

We are exploring the integration possibilities and further researching these topics in other ongoing works. For example, in [374] and [404], we are integrating role allocation, absolute and relative UWB localization, and blockchain smart contracts for building more secure and trustable multi-robot systems.

7 Collaborative Sensing

This chapter extends the collaborative localization approaches introduced in Chapter 5 ensuring uniqueness of the relative localization solution and involving more than two robots, with a graph theory approach that also extends the work in Chapter 6. The main objective in this chapter is to apply UWB-based localization for collaborative multi-robot scene reconstruction. This chapter does not introduce a novel approach to the localization methods themselves, but demonstrates instead their direct use in collaborative sensing. As such, this chapter introduces a novel approach to collaborative localization for dense scene reconstruction in heterogeneous multi-robot systems comprising ground robots and micro-aerial vehicles (MAVs). We solve the problem of full relative pose estimation without sliding time windows by relying on UWB-based ranging and Visual Inertial Odometry (VIO)-based egomotion estimation for localization, while exploiting lidars onboard the ground robots for full relative pose estimation in a single reference frame. During operation, the rigidity eigenvalue provides feedback to the system. To tackle the challenge of path planning and obstacle avoidance of MAVs in GNSS-denied environments, we maintain line-of-sight between ground robots and MAVs. Because lidars capable of dense reconstruction have limited FoV, this introduces new constraints to the system. Therefore, we propose a novel formulation with a variant of the Dubins multiple traveling salesman problem with neighborhoods (DMTSPN) where we include constraints related to the limited FoV of the ground robots. Our approach is validated with simulations and experiments with real robots for the different parts of the system.

Portions of text and a subset of figures in this chapter are reproduced from our previous work [2].

7.1 Context and contributions

In GNSS-denied environments, different approaches to multi-robot cooperative exploration have been showcased during the DARPA Subterranean challenge [49; 51], as we have described in more detail in previous chapters. The participants of the challenge deployed both unmanned ground vehicles (UGVs) and micro aerial vehicles (MAVs). Localization and collaborative sensing represented two of the main challenges. Indeed, solving these problems would allow robotic systems to go out of laboratory settings. In this section, we address the problem of collaborative multi-

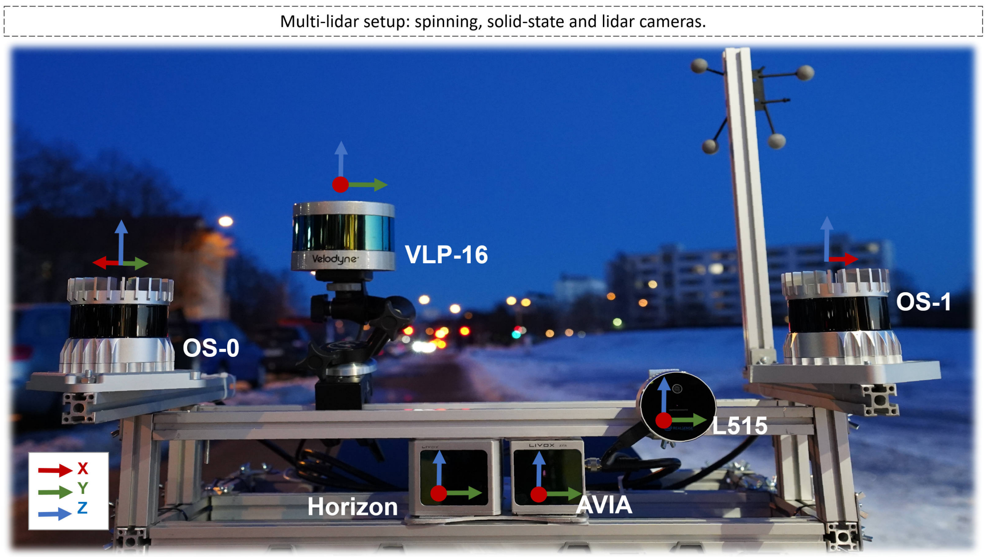


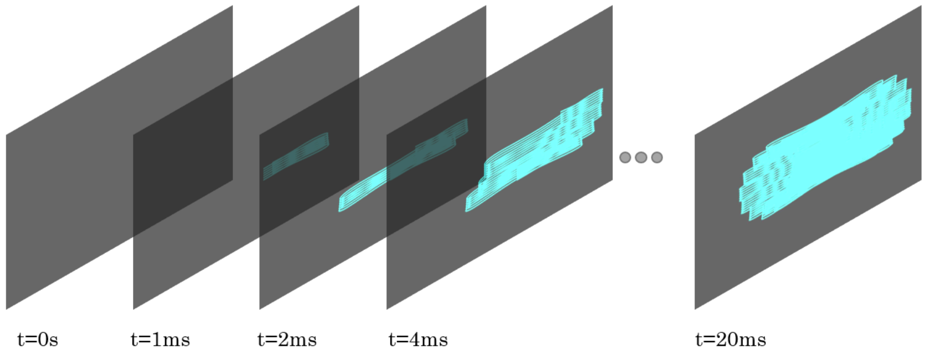
Figure 76. Example setup with spinning and solid state lidars. The Livox Horizon lidar is used in the experiments included in this chapter.

robot 3D dense scene reconstruction involving UGVs and MAVs.

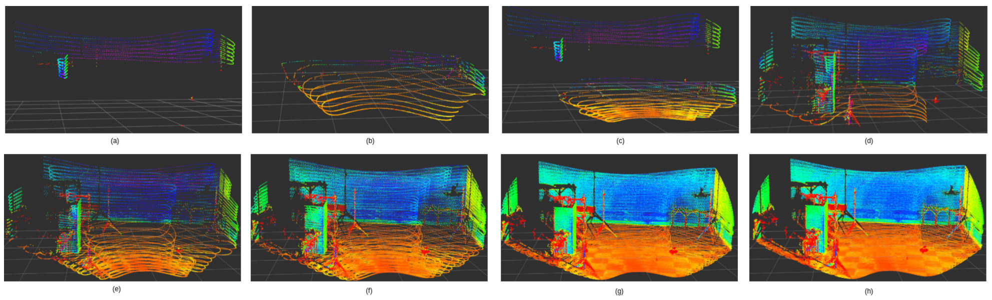
Solid-state lidars represent the state-of-the-art regarding sensors for high-accuracy and long-range dense point cloud scanners, often with limited Field of View (FoV) owing to the lack of rotating parts [405; 406]. However, the stringent payload constraints of MAVs make RGB-D cameras a more viable solution for dense scene reconstruction [407]. Therefore, to achieve collaborative sensing, we chose to embed 3D lidars only on UGVs with MAVs equipped with depth cameras. A sample setup with spinning and solid-state lidars, as well as a lidar camera (with equivalent functionality to RGB-D sensors) is shown in Fig. 76. We also show in Fig. 77 the scanning patterns of the solid state lidar used in this chapter (a) and data samples (b).

From the point of view of MAVs, advances in both monocular and stereo dense reconstruction and navigation have arrived to a point where commercial solutions such as Skydio II are able of high degrees of autonomy and situational awareness [408]. Nonetheless, deployment in GNSS-denied environments with limited visibility and potentially dynamic environments is still challenging [39].

Regarding localization approaches for multi-robot systems in GNSS-denied environments, ultra-wideband (UWB) wireless ranging transceivers have recently emerged as an inexpensive and relatively accurate method for point-to-point ranging [3; 21]. Full pose estimation can be achieved by fusing UWB with visual-inertial odometry (VIO)-based egomotion estimation. However, previous works are not able to provide



(a) Illustration of the field of view (FoV) coverage with different point cloud integration times in a non-repetitive lidar scanning device.



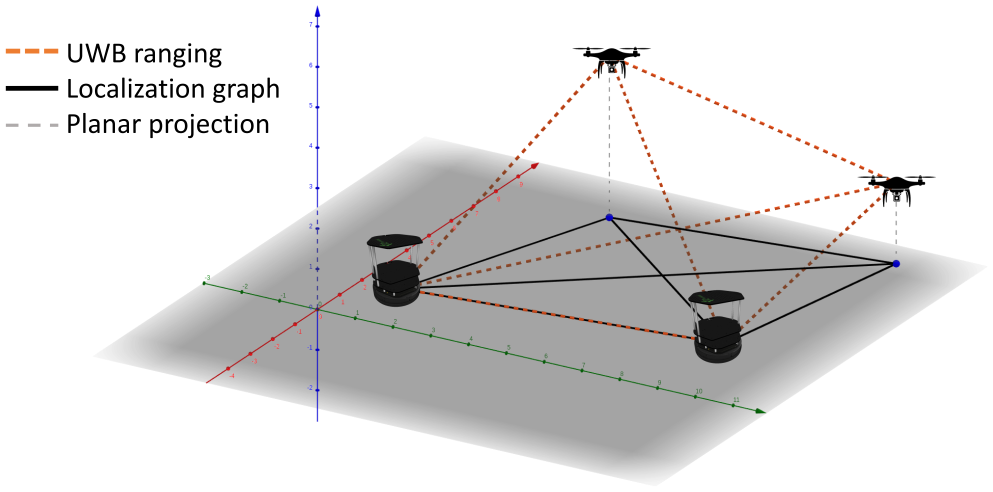
(b) Sample pointclouds obtained with different integration times using a Livox Horizon solid-state lidar.

Figure 77. Sample lidar devices and data from solid-state Livox lidars.

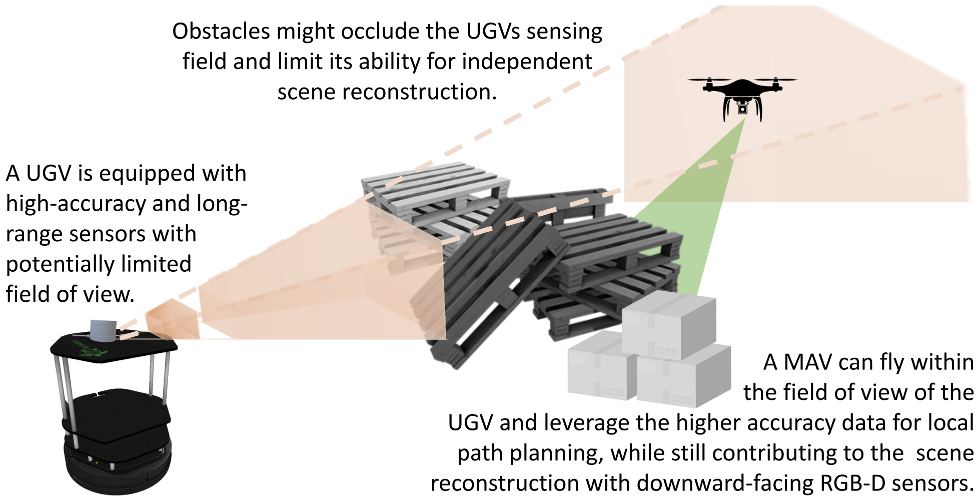
full position estimation within a single common reference unless robots move [359].

At the same time, over the past two decades research in both distance-based [409] and bearing-based [410; 411; 412] rigidity maintenance control have been shown to be robust and efficient methods for distributed collaborative localization in multi-robot systems. Different rigidity maintenance approaches have been proposed to ensure that the collaborative localization problem can be solved. However, up to our knowledge, none of the works of the literature integrate rigidity theory as a feedback to a localization system while, at the same time, achieving a navigation goal given by a high-level planner.

The proposed approach is illustrated in Fig. 78. First, we show conceptually in Fig. 78a how the localization employs a projected 2D graph that is monitored for



(a) Conceptual illustration of collaborative localization in a heterogeneous multi-robot system where the localization graph is globally rigid.


Obstacles might occlude the UGVs sensing field and limit its ability for independent scene reconstruction.
A UGV is equipped with high-accuracy and long-range sensors with potentially limited field of view.
A MAV can fly within the field of view of the UGV and leverage the higher accuracy data for local path planning, while still contributing to the scene reconstruction with downward-facing RGB-D sensors.

(b) Conceptual illustration of collaborative scene reconstruction with a heterogeneous UGV+MAV multi-robot system.

Figure 78. In this chapter, we explore the problem of UWB-based collaborative localization based on graph rigidity (a) for collaborative sensing (b) and dense scene reconstruction.

global rigidity, ensuring uniqueness of the relative localization solution. We then show in Fig. 78b how MAVs can aid in scanning areas occluded to sensors onboard UGVs. In addition, by flying within the FoV of UGVs, MAVs can leverage ground-based data for avoiding obstacles while navigating the environment.

7.1.1 Collaborative scene reconstruction

Over the last decade we witnessed an increasing adoption of inexpensive 2D lidars, 3D scanners, and depth cameras on mobile robots. Also, several advancements have been presented in monocular dense SLAM. These aspects pave the way for the development of multiple approaches for cooperative mapping and collaborative scene reconstruction. Collaborative SLAM and collaborative mapping in general have been widely studied problems. However, the focus is often on the map merging or area coverage distribution among the robots [413; 414]. Relevant to this work is collaborative RGB-D reconstruction [415].

We take a different approach by focusing on path planning with constraints to visit a series of interest regions. In this direction, a recent work by Dong et al. on collaborative dense scene reconstruction that takes an initial map and focuses on task allocation is similar to ours from the formulation point of view [416]. Another related area is next best view (NBV) planning. In [417], Sukkar et al. present a multi-robot region-of-interest reconstruction with RGB-D cameras. Compared to these approaches, we focus on the integration of sensing constraints (e.g., limited FoV) in LoS path planning.

7.1.2 Chapter contribution

Taking the above considerations into account, we present a novel approach to collaborative dense scene reconstruction within heterogeneous multi-robot systems that address several of the aforementioned challenges. First, we propose a UWB-VIO-based collaborative localization framework that exploits sensors onboard UGVs to detect the position of MAVs at startup for a unique localization graph realization. Second, we leverage the UGVs' sensors high accuracy and range for real-time path planning and obstacle avoidance of MAVs. At the same time, we use the collaborative localization framework to extract data from the UGVs' point clouds relative to the MAVs' positions. Third, we propose a different formulation to the collaborative scene reconstruction problem that solves a variant of the Dubins Multiple Traveling Salesman Problem with Neighborhoods (DMTSPN). In this problem the neighborhoods are defined as the parts of the environment which are occluded, due to obstacles, for the UGVs' sensors. Finally, to account for the limited FoV of lidars, we solve the DMTSPN by adding UGV-to-MAV line-of-sight (LoS) constraints.

Compared to previous approaches in rigidity maintenance for collaborative localization, we focus on path planning while ensuring LoS, which, in turn, ensures that the graph is rigid within the UWB range. Compared to previous works on dense scene reconstruction with map merging or point cloud alignment, we focus on teaming the different robots and the distribution of locations to be surveyed. Our results show good alignment of point clouds without further optimization, which can then be improved based on existing algorithms.

In summary, the main contributions of this chapter are the following:

1. A framework for single-shot collaborative localization based on UWB ranging, VIO and lidar fusion for full relative pose estimation, which uses the rigidity eigenvalue of the localization graph as feedback to the system.

2. A path planning algorithm for multi-robot scene reconstruction based on solving a variant of the DMTSPN for UGV-to-MAV LoS maintenance. The algorithm can handle sensing constraints such as limited FoV and it enables MAVs to rely on UGVs for local path planning.

7.2 Background

We use the following notation for the remaining of this section. We consider a group of N robots, or agents, with positions denoted by $\mathbf{p}_i(t) \in \mathbb{R}^3$, with $i \in \{1, \dots, N\}$. Agents are able to measure their relative distance to a subset of the other agents in a bidirectional way, i.e., with both agents calculating a common ranging estimation simultaneously. The set of robots and estimated distances between them are modeled by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\{1, \dots, N\}$ is a set of N vertices and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of $M \leq N(N-1)/2$ edges. We consider undirected graphs, i.e., $(i, j) \in \mathcal{E} \iff (j, i) \in \mathcal{E}$.

We consider robots in three-dimensional space. Nonetheless, for the purpose of collaborative localization, we assume that the position of each agent is given by $\mathbf{p}_i \in \mathbb{R}^2$. The pair $(\mathcal{G}, \mathbf{p})$ with the position vector $\mathbf{p} = [\mathbf{p}_1^T, \dots, \mathbf{p}_N^T]^T$ is a framework. We denote the incidence, degree and adjacency matrices by $E(\mathcal{G}) \in \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$, $\Lambda(\mathcal{G}) \in \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$ and $A(\mathcal{G}) \in \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$, respectively, and the graph laplacian by $\mathcal{L}((G)) = E(\mathcal{G})E(\mathcal{G})^T = \Lambda(\mathcal{G}) - A(\mathcal{G})$. It is worth noting that an important result from algebraic graph theory states that $\mathcal{G}$ is connected if and only if the second smallest eigenvalue of $L(\mathcal{G})$ is positive [418].

The solution of the collaborative localization problem is related to the uniqueness of graph realizations in space. There will be a unique graph realization (except for rototranslations in $\mathbb{R}^2$) if the framework $(\mathcal{G}, \mathbf{p})$ is rigid. Intuitively, a rigid graph is a graph that cannot be deformed without breaking the constraints put over the edges. A sufficient and necessary algebraic condition similar to that of the graph connectivity can be given. We first define an edge constraint function $g_{\mathcal{G}} : \mathbb{R}^{|\mathcal{E}||\mathcal{V}|} \mapsto \mathbb{R}^{|\mathcal{E}|}$ that through $g_{\mathcal{G}}(\mathbf{p})$ defines a constraint over each of the edges $g_{ij}(\mathbf{p}_i, \mathbf{p}_j) \forall (i,j) \in \mathcal{E}$. We will use $g_{ij}(\mathbf{p}_i, \mathbf{p}_j) = \|\mathbf{p}_i - \mathbf{p}_j\|^2$ to use distances as constraints.

Two frameworks that represent realizations of the same graph, $(G, \mathbf{p}_1)$ and $(G, \mathbf{p}_2)$, are said to be equivalent if $g_{\mathcal{G}}(\mathbf{p}_1) = g_{\mathcal{G}}(\mathbf{p}_2)$, and congruent if $g_{\mathbb{K}}(\mathbf{p}_1) = g_{\mathbb{K}}(\mathbf{p}_2)$, where $\mathbb{K}$ is the complete graph with the same vertex set $\mathcal{V}$ as $\mathcal{G}$. A framework is *locally rigid* if $\forall \mathbf{p} \in \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$ $\exists \mathcal{P} \subset \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$, $\mathbf{p} \in \mathcal{P}$ such that $g_{\mathcal{G}}^{-1}(g_{\mathcal{G}}(\mathbf{p})) \cap \mathcal{P} = g_{\mathbb{K}}^{-1}(g_{\mathbb{K}}(\mathbf{p})) \cap \mathcal{P}$ and *globally rigid* if $\forall \mathbf{p} \in \mathbb{R}^{|\mathcal{E}||\mathcal{V}|}$, $g_{\mathcal{G}}^{-1}(g_{\mathcal{G}}(\mathbf{p})) = g_{\mathbb{K}}^{-1}(g_{\mathbb{K}}(\mathbf{p}))$.

Even if a locally rigid graph is achieved, the null space of the transformations is defined by rototranslations of the graph realization in $\mathbb{R}^2$. In our experiments, we will consider the position of one of the agents as the origin of coordinates of the system, and utilize sensors onboard the UGVs to establish the orientation of the graph realization, after rigidity is ensured and MAVs take off. Therefore, all the measurements are relative to the agent chosen as the origin.

We now consider infinitesimal rigid frameworks as those where constraints are met under infinitesimal perturbations $\delta\mathbf{p}$. In order to maintain the constraints over edges, we can compute the Jacobian matrix

$$\dot{g}_{\mathcal{G}}(\mathbf{p}(t)) = \mathbf{0} \Longrightarrow \frac{\delta g_{\mathcal{G}(p)}}{\delta(\mathbf{p})}\dot{\mathbf{p}} = R_{\mathcal{G}}(\mathbf{p})\dot{\mathbf{p}} = \mathbf{0} \tag{27}$$

where $R_{\mathcal{G}}(\mathbf{p}) \in \mathbb{R}^{d|\mathcal{E}||\mathcal{V}|}$ is the rigidity matrix [409], with $d = 2$ because we are considering points in $\mathbb{R}^2$. Translations and rotations in the Cartesian space make up the non-trivial kernel of $R_{\mathbb{K}}(\mathbf{p})$, and therefore we can say that a framework $(\mathcal{G}, \mathbf{p})$ is infinitesimally rigid if the rank of $R_{\mathcal{G}}(\mathbf{p})$ is the same as that of $R_{\mathbb{K}}(\mathbf{p})$: $2N - 3$ in $\mathbb{R}^2$ (equivalently, $3N - 6$ in $\mathbb{R}^3$).

In this chapter, we are considering a heterogeneous multi-robot system comprising both UGVs and MAVs, and therefore their relative positions must be given in a three-dimensional space. While the above conditions for rigidity hold for three-dimensional graphs, the amount of information required is larger (more edges needed in $\mathcal{G}$). Since MAVs are already equipped with relative altitude sensors and capable of VIO estimations, we only consider graph rigidity in 2D and project the ranging information to the plane using data from other onboard sensors. The relative altitude is estimated primarily based on a downward facing single-beam lidar sensor. We model UWB measurements with Gaussian noise:

$$\mathbf{z}^{UWB}_{(i,j),\,(i,j)\in\mathcal{E}} = \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| + \mathcal{N}(0, \sigma_{UWB}) \tag{28}$$

where σ_{UWB} is obtained experimentally from our previous work [19]. VIO egomotion estimations are modeled with

$$\mathbf{z}^{VIO}_{i,\,i\in\mathcal{V}} = \begin{bmatrix} \mathbf{R}_i(t-\delta t)\hat{\mathbf{R}}_i(t) & \|\mathbf{p}_i(t) - \mathbf{p}_i(t-\delta t)\| \\ 0 & 1 \end{bmatrix} + \mathcal{N}(0, \sigma_{VIO}) \tag{29}$$

where we utilize $\sigma_{VIO} < \sigma_{UWB}/10$, estimated experimentally, and δt is the output frequency of the VIO algorithm, $\mathbf{R}_i(t)$ is the orientation matrix for agent i and $\hat{\mathbf{R}}_i(t)$ the relative egomotion estimation in the interval $(t - \delta t, t]$. The relative altitude is estimated based on the lidar

$$\mathbf{z}^{H}_{i,\, i\in\mathcal{V}} = \begin{cases} h_i^{lidar} & \text{if } \Delta(h_i^{lidar}, h_i^{UWB}, h_i^{VIO}) < 1 \\ h_i^{UWB+VIO} & \text{otherwise} \end{cases} \tag{30}$$

where $\Delta(\cdot)$ estimates the mismatch between the different sensors. We use a filter to *smooth* the UWB ranges using the VIO translational estimations at both agents.

UWB-based ranging information provides only the position of the full pose of the agents. Therefore, we rely on VIO-based orientation to estimate the orientation and achieve full pose estimation. We assume that all agents share a common reference. However, in order to match the graph realization with the agents' reference, we need to be able to measure the relative position, and not just distance, of at least one pair of agents. To do so, we assume that each MAV is within at least one UGV's field of view when the mission starts, and that the MAV can be detected from the UGV after taking off. Let $\mathcal{V}_{UGV}$ and $\mathcal{V}_{MAV}$ be the sets of UGVs and MAVs, respectively, with $N = N_{UGV} + N_{MAV} = |\mathcal{V}_{UGV}|+|\mathcal{V}_{MAV}|$. Then, let $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ be the UGV sensing graph with the same vertex set $\mathcal{V}_S = \mathcal{V}$ as $\mathcal{G}$ where $(i,j) \in \mathcal{E}_S \Leftrightarrow i \in \mathcal{V}_{UGV}$, $j \in \mathcal{V}_{MAV}$ and j is *visible* from sensors onboard i (e.g. 3D lidar). We denote by $\mathcal{N}_i$ the set of neighbor nodes in $\mathcal{G}$ and $\mathcal{N}_{S_i}$ the neighbor set in $\mathcal{G}_S$. The actual localization is done by minimizing triangulation errors.

We now assume that the sensors on the UGVs produce accurate and dense point clouds but with limited FoV. Let $\mathcal{P}_i \subset \mathbb{R}^3$ be the point cloud generated at agent i. For a point $p \in \mathbb{R}^3$, we denote by $\mathcal{N}_p^R(\mathcal{P}) = \{q \in \mathcal{P} \mid \|p - q\| \leq R\}$ the set of points in $\mathcal{P}$ within a distance R of p. Finally, given a graph realization orientation θ, we denote by $\hat{p}_i(\theta)$ the estimated position of agent i in the global reference frame, where the position of agent 0 is used as the origin of the frame. We then calculate the orientation $\hat{\theta}$ of the graph realization in the common reference frame by minimizing :

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i\in\mathcal{V}_{UGV}} \sum_{j\in\mathcal{N}_{S_i}} \frac{|\mathcal{N}_{\hat{p}_i(\theta)}^{R_{MAV}}(\mathcal{P})\,|}{|\mathcal{N}_{\hat{p}_i(\theta)}^{2R_{MAV}}(\mathcal{P})\,|+1} \tag{31}$$

where R_{MAV} is the radius of the circumscribed sphere to a MAV point cloud, roughly half the width of a MAV (we consider homogeneous MAVs, otherwise the different sizes must be taken into account). The localization process is summarized in Algorithm 3.

While the assumption of having MAVs within UGVs field of view could be directly leveraged towards measuring relative positions (together with a common orientation reference and VIO estimations at each robot), we still rely on UWB as the main source of localization when the mission starts and during the entire mission.

Algorithm 3: Collaborative localization process.

```
Input:
    UWB Ranges : $\{\mathbf{z}_{(i,j)}^{UWB}\} \in \mathbb{R}^{|\mathcal{E}|}$;
    3D lidar point cloud $\{\mathcal{P}_i\}$;
    Relative altitude of MAVs: $\{\mathbf{z}_i^H\}$;
    VIO odometry: $\{\mathbf{z}_i^{VIO}\}$
Output:
    Full robot poses $\hat{\mathbf{p}}(\hat{\theta}) \in \mathbb{R}^6$
1  while !graph_is_connected($L(\mathcal{G})$) do
2      sleep();
3  while !graph_is_rigid($R_{\mathcal{G}}(\mathbf{z}^{UWB})$) do
4      sleep();
5  $\hat{\mathbf{p}} \leftarrow minimize\_triangulation\_error()$;
6  takeoff_MAVs();
7  Calculate graph orientation $\hat{\theta}$ as follows (init. $\theta = 0$):
8  while not exit_condition($\hat{\theta}$) do
9      $error = 0;\quad theta+ = \Delta\theta$;
10     foreach $i \in \mathcal{V}_{UGV}$ do
11         Generate K-D Tree from point cloud: $kdtree_i \leftarrow \mathcal{P}_i$;
12         foreach $j \in \mathcal{N}_{S_i}$ do
13             $error+ = size\,(kdtree_i.radiusSearch(\hat{\mathbf{p}}_i,\ R_{MAV}))$ /
14             $size\,(kdtree_i.radiusSearch(\hat{\mathbf{p}}_i,\ 2R_{MAV}))$
15     if $error < threshold$ then
16         $\hat{\theta} = \theta$;
17 Calculate full pose:    $\hat{\mathbf{p}}(\hat{\theta}) \mid \hat{\mathbf{p}}_0 = (0,0), \hat{\mathbf{p}}_1 = \left(0, \mathbf{z}_{(0,1)}^{UWB}\right)$;
```

The reason for doing so is twofold: first, UWB ranging is more accurate than estimating the position of an object extracted from a point cloud accounting for its size, shape and orientation; and second, we do not need to consider uncertainties in the point-cloud-based detector (whether it detects MAVs or other similarly sized objects).

7.3 Collaborative scene reconstruction

The main objective of this chapter is to provide methods for collaborative dense scene reconstruction. This serves simultaneously as a validation of the collaborative localization framework owing to the unavailability of high-accuracy tracking systems such as those utilized in the *CoLo* evaluation [419].

In order to formulate the problem, we use the following notation. Let $\mathbf{q} \in \mathbb{R}^{2M}$, $\mathbf{q} = [\mathbf{q}_1^T, \ldots, \mathbf{q}_N^T]^T$ be a stacked position vector for the M locations to be visited. We make the following assumptions: (i) $M \geq N_{MAV} \geq N_{UGV}$; and (ii) if $H(\mathbf{p}_{UGV})$ is the convex hull defined by the positions of the UGVs, then $\mathbf{q} \cap H(\mathbf{p}_{UGV}) = \emptyset$, i.e., all locations to be surveyed by the MAVs lay *beyond* the positions of the UGVs.

We consider the problem of utilizing multiple MAVs to obtain information about the areas of the scene that are occluded to the UGVs, with each MAV always staying within the FoV of one UGV (its *tracker*). The collaborative scene reconstruction process then proceeds as follows (see Algorithm 4). First, the ground robots scan the scene and estimate blind spots based on their movement constraints due to uneven terrain or near obstacles. Second, each of the occluded regions behind obstacles blocking the FoV of UGVs is considered a neighborhood for the DMTSPN problem. We distribute the locations to be surveyed among the MAVs by solving DMTSPN problem where each location represents a neighborhood.

Multiple works have been devoted to solving DMTSPN [420; 421]. However, to the best of our knowledge, the current literature does not consider scenarios where the paths have to meet LoS with limited FoV constraints with respect to a certain point in the environment. In particular, considering limited FoV is a topic largely unaddressed in the multi-robot systems literature [422].

To address this, we assign each MAV to one of the UGVs based on their initial positions, and proceed to assign the neighborhoods to MAVs based on their angular position with respect to their *tracker* UGV when considering their position in polar coordinates from the UGV's local reference. Then, the MTSPN problem is solved based on the constraints defined above, with the neighborhood assignment to MAVs changing until an exit condition is met. Finally, the Dubins paths are generated smoothing turns over the neighborhoods, and the speed of the MAVs is adjusted to ensure LoS maintenance (the DMTSPN solution does not consider time).

Since the different parts of the problem are decoupled (diving neighborhoods among MAVs, solving the TSPN for each MAV, and *smoothing* the paths with Dubns curves), we will obtain a sub-optimal solution. However, this already happens when introducing the LoS with limited FoV constraints, since in general optimal solutions to the MTSPN will not ensure that MAVs can stay within LoS of UGVs. Because our focus is on providing an initial approach that also combines the collaborative localization framework, we leave the optimization of the DMTSP solution to future works.

7.4 Methodology

In this section, we introduce our experimental setup.

Algorithm 4: Collaborative scene reconstruction process.

```
Input:
   Dense UGV lidar point clouds: $\{\mathcal{P}_i^L\}_{i\in\mathcal{V}_{UGV}}$
   Dense MAV depth point clouds: $\{\mathcal{P}_i^D\}_{i\in\mathcal{V}_{MAV}}$
   UGV FoV: $\{(\Delta\theta_i^H, \Delta\theta_i^V)\}_{i\in\mathcal{V}_{UGV}}$
Output:
   UGV+MAV paths $\{\hat{\mathbf{p}}_i(t)\}$
1 $\mathcal{P} = merge\_point\_clouds\left(\{\mathcal{P}_i^L\}\right)$ ;
2 $\mathcal{NBH} = neighborhoods\_from\_blind\_spots(\mathcal{P})$;
3 $\{\mathcal{NBH}_\rangle\}_{i\in\mathcal{V}_{MAV}} = assign\_nbh(\mathcal{NBH}, \mathbf{p}_i^{MAV})$;
4 foreach $i \in \mathcal{V}_{UGV}$ do
5     while !neighborhood_exit_condition() do
6         foreach $j \in \mathcal{N}_{S_i}$ do
7             $run\_tspn\_solver(N_i, \mathbf{p}_i, \Delta\theta_i^H, \Delta\theta_i^V)$;
8             $calculate\_dubins\_path(i)$;
9     $merge\_while\_navigating(\{\mathcal{P}^L\}, \mathcal{P}^D\})$;
```

Heterogeneous Multi-Robot System. In our experiments, we utilize a single ground robot, one MAV, and a set of UWB transceivers (Fig. 79). The ground robot is an EAI Dashgo platform equipped with a Livox Horizon lidar (81.7° $\times$ 25.1° FoV). We also use one custom-built MAV based on the X500 quadrotor frame. The MAV is embedded with a Pixhawk flight controller running the PX4 firmware. A TF Mini Lidar is utilized for height estimation on the MAV, also equipped with Intel RealSense D435 depth camera for sensing and 3D reconstruction. An AAEON Up Square with dual-code Intel Celeron processor is used as a companion computer on both robots. Both robots use RealSense T265 cameras for VIO-based egomotion estimation.

UWB Ranging and Position Estimation. The distance between each pair of robots is estimated using Decawave DWM1001 UWB transceivers. In order to ensure a safe fallback for the MAV autonomous flight, we utilize both an anchor-based localization system as well as ranging between robots for collaborative localization. We also compare the localization from both methods.

Software. The system has been implemented using ROS Melodic, with all robots running the same version under Ubuntu 18.04. All the code utilized in this chapter is open-source and will be made freely available in our GitHub page[1]. Specifically for this chapter we have developed the following ROS packages: *uwb-graph-rigidity*, *uwb-collaborative-sensing* and *offboard-control*. We also use *dashgo-d1-ros*, *ros-*

[1] `https://github.com/TIERS`

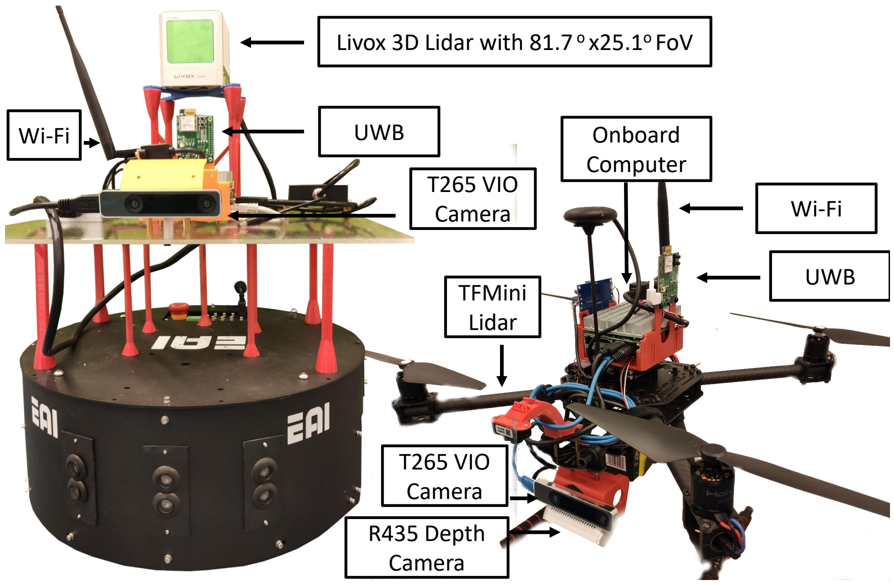


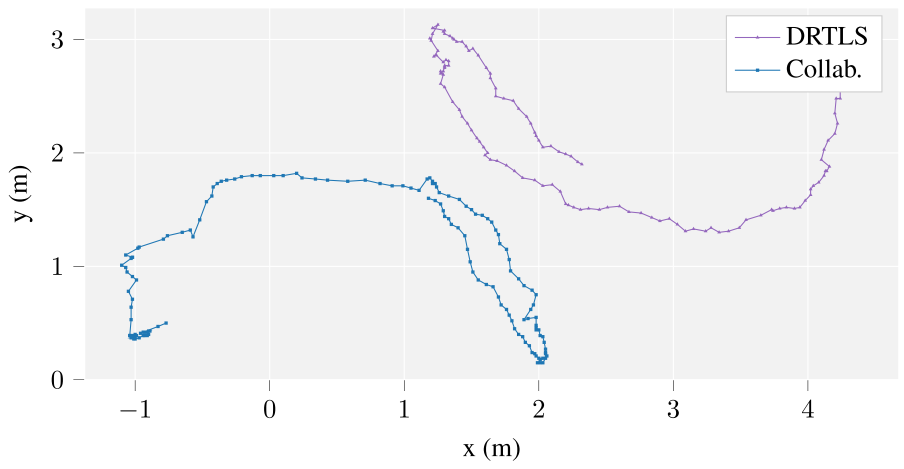
Figure 79. UGV and MAV utilized in the experiments.

dwm1001-uwb-localization and *tfmini-ros* for sensor interfacing. Most of the code has been written in Python or C++. In particular, the point cloud library (PCL) [423] is utilized to extract the position of the MAV for estimating the localization graph orientation, and extracting the point cloud around it. We use MAVROS to interface the onboard computer with the PX4 controller.

7.5 Experimental results

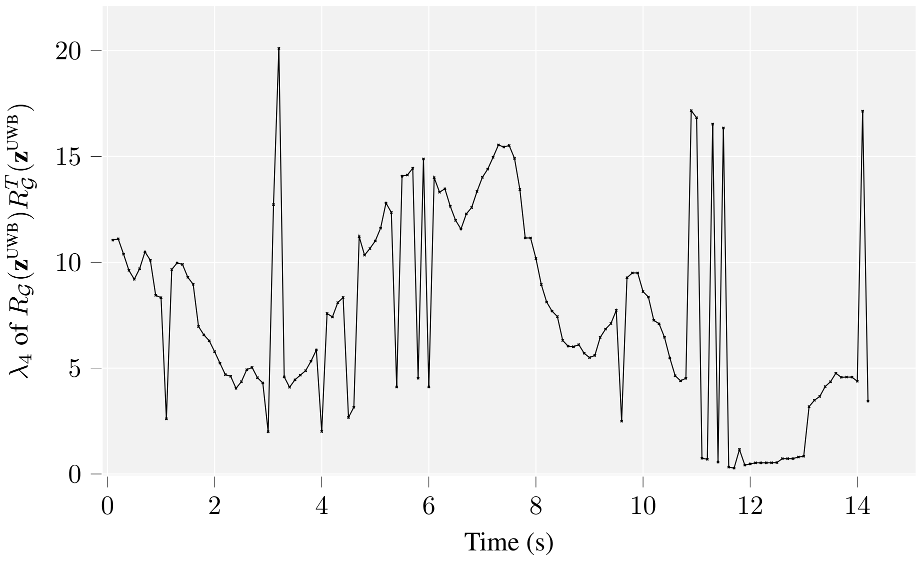
This section reports the results from the simulations and experiments carried out to validate the proposed methods. First, we perform test flights to assess the viability of the collaborative localization framework, using the UGV and MAV and additional UWB transceivers on the ground that are also considered as graph vertexes to ensure rigidity. Second, we introduce simulations to show that an UGV is able to track and maintain in its FoV two MAVs while they are exploring the occluded locations around the UGV (not possible with a basic DMTSPN solver). Then, we show that we achieve good alignment of raw data collected from the UGV and MAV simultaneously for 3D reconstruction, and without further local map merging optimization. Finally, we discuss the system's scalability.

Collaborative Localization Framework. Fig. 80 shows the positions of the MAV based on Decawave's anchor-based localization system (DRTLS) and our collaborative localization framework over a test flight. The UGV is taken as the origin of coordinates in the latter scenario. We also show the evolution of the rigidity eigen-



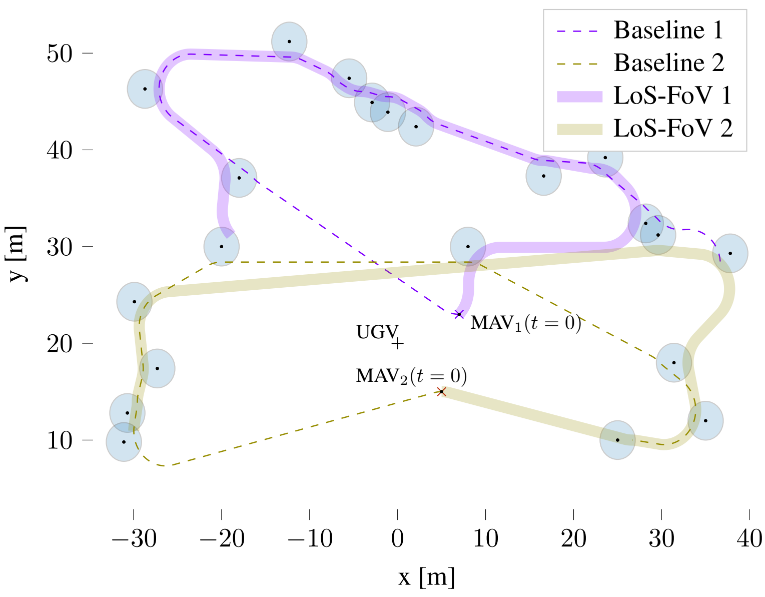
(a) Localization based on Decawave's fixed-anchor DRTLS and the collaborative localization framework (*collab*), using the UGV's position as the origin of coordinates, and therefore giving relative localization only (rototranslation from DRTLS except for measurement errors)

.

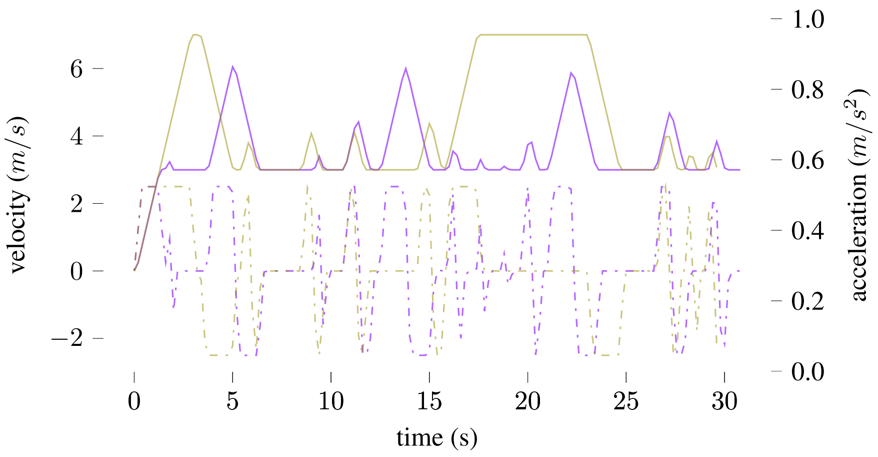


(b) The rigidity eigenvalue is monitored during flight to ensure that the graph is always rigid.

Figure 80. Sample localization estimation and eigenvalue monitoring.



(a) Solution to the DMTSPN (MAV paths). The neighborhoods are shown in blue, representing the locations to be visited.



(b) Velocity and acceleration profiles for each of the MAVs (solution with LoS-FoV constraints only).

Figure 81. Simulations for two MAVs and one UGV with limited FoV. Subfigure (a) shows both a baseline DMTSPN solver, and our solver with LoS constraints, with (b) showing the velocity and acceleration profiles of our solver. This particular example was chosen to illustrate how the MAVs follow an anticlockwise direction from the UGV's reference.

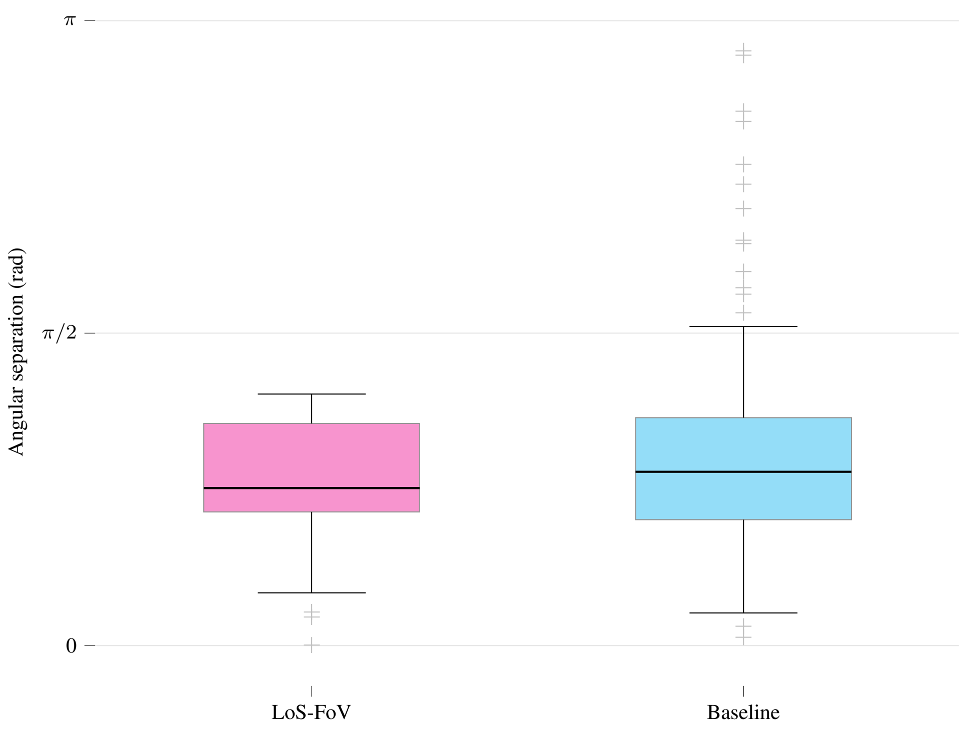
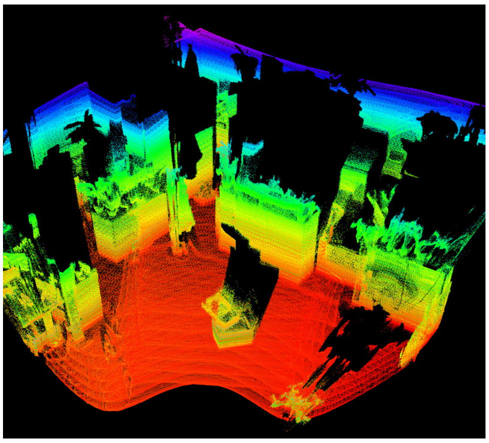


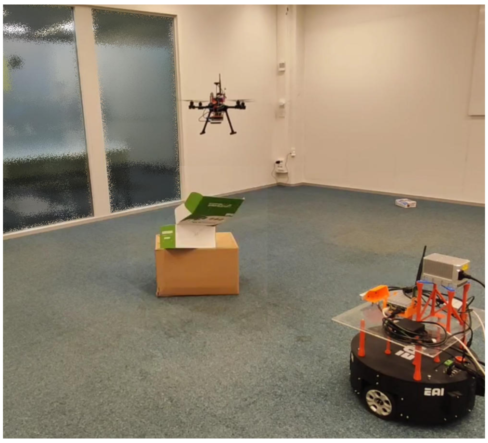
Figure 82. Distribution of angular distance between MAVs. Only with our DMTSPN solver we can ensure that MAVs are always within LoS with limited FoV at the UGV (angular separation always below a predefined FoV of $\pi/2$), with baseline solutions potentially giving any random distribution.

value. Because our framework provides relative localization only, the two paths in the figure are congruent with respect to a rototranslation, except for measurement errors. However, by detecting the MAV from the UGV after take-off, we are able to fix the orientation of the localization graph in the common orientation frame (not necessarily the same than the DRTLS system). The DRTLS localization is based on 8 fixed anchors, and therefore the accuracy can be considered higher. Monitoring the rigidity eigenvalue will play a more important role in applications where the robots operate in a larger environment, and this is, to the best of our knowledge, the first time it is used as a *health* indicator to the system for higher-level planning.

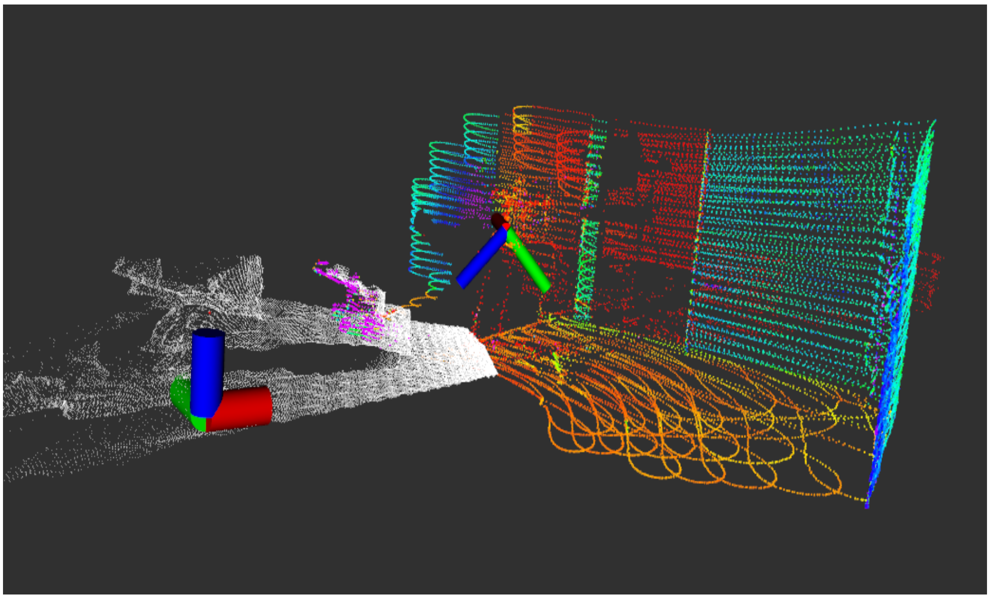
Multi-MAV Path Planning. The multi-robot path planning with UGV-to-MAV LoS constraints in limited FoV scenarios is tested through simulations with random distribution of neighborhoods. The simulations are done with a fixed UGV that can only rotate and two MAVs that have to visit all locations in the map as fast as possible while staying in LoS within the FoV of the UGV. We pick one representative example, and compare in Fig. 81a the difference between the paths calculated by a baseline DMTSPN without constraints, and ours. Fig. 82 shows the angular distance



(a) Global scene with boxes in the center and MAV in the bottom right.



(b) Picture of the test environment while the MAV is hovering.



(c) Aligned point clouds using UWB+VIO for relative localization. The white point cloud is recorded with the RealSense D435 depth camera on the MAV, while the colored point cloud comes from the Livox lidar on the UGV (both poses shown, D345 is inclined 36°).

Figure 83. Validation of the collaborative sensing algorithm. Figures (a) and (b) show the scene from opposite corners. The UGV is not visible in (a) because it is being used to capture the point cloud. Figure (c) shows the limited FoV of the Livox lidar and a single frame alignment of pointclouds from both the lidar and the camera on the drone.

between the two MAVs from the UGV's referece when considering polar coordinates. For the solver with constraints, the FoV is limited to $\pi/2$ rad. We can see that there are multiple times where the UGV would be unable to maintain both MAVs within its FoV (points where the angular distance is larger than $\pi/2$ rad) with the baseline DMTSPN implementation. By introducing the limited FoV constraints, we are therefore able to obtain results that can be better ported to real-world applications. Additionally, we show in Fig. 81b the velocity and acceleration profiles of the two drones, where the speeds are limited by the path curvature as well as the FoV constraints embedded in the DMTSPN solver.

Collaborative Scene Reconstruction. We perform experiments in an indoors facility and show the performance of the collaborative localization and sensing algorithms for matching raw point clouds (see Fig. 83). First, the UGV scans the scene. To create a map, we utilize an implementation of lidar odometry and mapping (LOAM) optimized for limited-FoV lidars: Livox LOAM [406]. We then detect the blind spots to the UGV based on a standard elevation occupancy map. In the experiments, owing to the limited space available, we only use one UGV and one MAV, together with two more UWB transceivers placed on the ground to ensure global graph rigidity. The UGV is set to rotate and move within a limited space to always maintain the MAV within its FoV and in LoS. Fig. 83 shows the aligned point clouds of the MAV and UGV, together with their poses in a given instant. The colored point cloud is obtained from the Livox lidar, with the non-repetitive scan pattern being recognizable by the *waves* in the ground. Two stacked boxes in the middle of the test area are scanned from complementary points of view simultaneously, showing a good point cloud alignment directly with the raw data. This opens the door to further optimization, and providing feedback to the localization framework based on the alignment of local point clouds. The area behind the boxes occluded to the UGV's lidar corresponds to a neighborhood in the DMTSPN formulation. Intuitively, the neighborhoods can be mapped to the UGV lidar'*shadows* as those seen in Fig. 83a.

Scalability. In terms of the system's scalability, the localization framework is based on UWB ranging and the VIO egomotion estimation is done on a separate processor on the T265 camera. The UWB ranging is only limited by the number of robots in terms of the available bandwidth, and therefore the localization frequency must be decreased as the number of robots increases (the current frequency of 10 Hz can accommodate approximately 20 robots within line-of-sight of each other). As to the number of robots involved in scene reconstruction, because the MAVs are assigned to one UGV and the DMTSPN solved from each UGV's reference, the computational load of the path planning algorithm can be maintained even when the number of robots grows. The assignment of MAVs to UGVs has linear complexity.

In summary, we provide an initial implementation of all parts of the proposed system, which can then be leveraged for collaborative scene reconstruction in GNSS-denied environments. In particular, we show that the localization framework has potential to take multi-robot systems out of the lab with good results in local map merging for scene reconstruction, lifting the need for more accurate but significantly less flexible motion capture systems or UWB localization systems based on fixed anchors. Finally, the fact that UGVs are continuously tracking MAVs and keeping them within their FoV in LoS means that the more accurate lidar data they capture can be used for local path planning on the MAVs. This opens the door to operating the MAVs in more complex and dynamic environments even with limited onboard sensing.

7.6 Summary and conclusions

We have addressed some of the challenges in multi-robot dense scene reconstruction, with a focus on (i) collaborative localization, and (ii) path planning with LoS and FoV constraints. We have first presented a framework for collaborative localization with UWB-based ranging and VIO fusion in heterogeneous UGV+MAV systems, exploiting sensors onboard the UGVs to establish a common reference frame for all agents. Then, we have utilized this framework for collaborative scene reconstruction in a heterogeneous multi-robot system comprising UGVs and MAVs. With simulations, we show that the proposed algorithm effectively ensures MAVs are always within the UGVs FoV, and with real experiments we show that the localization framework is accurate enough to provide good alignment of point clouds even at the raw data level.

In future works, we will look into improving the DMTSPN solution by allowing dynamic tracking of a single MAV from different UGVs during the scene reconstruction mission. We will also investigate the possibilities of integrating the navigation of the UGVs within the DMTSPN solver.

8 Concluding remarks

This thesis brings together research in a relatively broad set of topics that converges with their potential application to achieve more efficient *collaborative autonomy* in heterogeneous multi-robot systems. In this chapter, we summarize the key contributions and outline future research directions, some of which we are already actively exploring.

8.1 Summary and contributions

The goal of this thesis was to advance towards more robust and collaborative multi-robot systems. The different studies presented in this thesis have provided new solutions, methods or ideas for such advances from different directions, in varying technical depth.

In Chapter 2, we have proposed different architectures and use-cases for reconfigurable robot swarms and for distributed robotic systems in the edge-cloud continuum. In particular, we explored the potential of computational offloading at the edge within multi-robot systems, with our results showing that offloading computation can lead to more cost-effective robots in some cases. We have also studied the impact that typical image compression techniques have on a standard visual odometry method to assess the potential drawbacks in bandwidth-constrained situations. Finally, we have also shown how specialized hardware allowing for acceleration of computation can bring benefits if available at the robots or shared at the edge.

With some the architecture for reconfigurable swarms proposing the use of DLTs for collaborative decision making, Chapter 3 moves the focus to exploring the integration of DLTs in multi-robot systems. We propose an approach that could leverage new concepts being introduced in Ethereum 2.0 for managing dynamic robot swarms, while providing an initial assessment of the ability of PoW puzzles to model available computational resources. We also propose potential integrations of blockchains for MEC services.

Continuing with the topic of security that DLTs arise, in Chapter 4, and partly inspired by data structures used in blockchains, we present a cryptographic approach to securing single- and multi-robot missions through encoded instruction graphs. Through the chapter, we present a design methodology for encoding robot instructions with cryptographic hashes that are only reproducible by a robot when a series of conditions are met, e.g., when operating in the right environment and in the right way. This allows for validating the robot behaviour while keeping missions secret until they are carried out. We demonstrate the applicability of such approach in both simulation and real-world experiments with a mobile robot.

In Chapter 5, the focus shifts towards localization systems and relative localization in multi-robot systems, a key element necessary to enable more complex multi-robot interaction. We focus our work in UWB-based localization, with an assessment of the viability of off-the-shelf UWB localization systems for autonomous aerial robots. We then explore the potential of relative UWB localization between a ground and an aerial robot in comparison with GNSS sensors in areas where the GNSS signals are degraded, such as urban environments. Our experiments show significantly better accuracy for the UWB solution, which in turn is more reliable than some off-the-shelf visual odometry solutions that diverge for aerial robots at certain height.

Accurate localization between robots in turn enables spatial coordination or distributed formation control. Novel methods requiring minimal to no communication are presented in Chapter 6, where we show that relatively complex shapes can be achieved with only one-way communication or with only mutual sensing, both of which could be achieved with simple UWB-based solutions.

Finally, Chapter 7 closes the gap between planning, localization and sensing with an approach to collaborative 3D mapping, or scene reconstruction, with a heterogeneous team of ground and aerial robots. The experiments show that UWB-based relative localization approaches are accurate enough for merging raw sensor data from lidars in the ground robots and stereo cameras in the aerial robots without further optimization. At the same time, we introduce a cooperative path planning approach solving a multiple traveling salesman problem with neighborhoods under line-of-sight and field-of-view constraints that would enable the aerial robots to rely on more accurate and longer-range data from the ground robots for local obstacle avoidance and situational awareness.

8.2 Future directions and open research questions

This thesis has covered multiple topics in varying depth, with experiments in simulation, with real robots, or with conceptual proofs of concept. In addition, some of the ideas in Chapters 2 and 3 have only been formulated but not implemented. The secure mission definitions introduced in Chapter 4 can be applied to different use cases, defining new types of encoded instruction graphs, e.g., exploring their practical use in human-robot or human-swarm interaction. There are also diverse directions in which to further improve the localization systems presented in Chapter 5, much to study in the integration of DLTs for formation control algorithms with more communication than those introduced in Chapter 6, and endless ways in which all the previous technologies can be leveraged for collaborative sensing in multi-robot systems as shown in Chapter 7. The rest of this section covers some of the potential future research directions and open questions within these topics.

8.2.1 Robots in the edge and DLT integrations

Parts of this thesis have been devoted to introducing architectural designs for future large-scale multi-robot systems or robot swarms, especially in terms of exploiting the potential of the edge-cloud continuum for computational offloading, reconfiguration and overall system elasticity. In terms of relying on DLTs for collaborative decision-making and distributed data management, the field of DLTs itself is relatively immature. The literature is scarce in their integration with distributed robotic systems in the wild, where assumptions to network connectivity do not hold or latency and real-time requirements are hard to meet. While we have presented, for all of these, different proofs of concept and experimental results for specific elements of such architectures (e.g., in terms of computational offloading or studying the potential for PoW-based computational resource estimation), there is still a gap between the theoretical possibilities and the practical reality out of the lab. Our near-future efforts in this area will be targeted at dynamic migration of ROS 2 nodes within containers between edge and cloud, or between different hosts at the edge (e.g., robots or robots and local connected infrastructure). Some recent research has targeted more efficient cloud integration into ROS 2 systems, such as FogRos2 [424].

The vast majority of literature in the area of DLTs and robotics covered until very recently only traditional blockchains, mainly basing the solutions on Ethereum smart contracts. However, we argue that two other type of DLTs have significant potential in the robotics field. Multi-robot systems, managed centrally or in a distributed manner, can often be considered a single entity where, a priori, there is trust between the involved parties (e.g., cloud servers providing services to robots, both deployed or owned by the same party). Therefore, within traditional blockchain architectures, permissioned blockchains such as Hyperledger Fabric have significant advantages in terms of identity management, access control and visibility of shared data to subsets of the network, among others. We have started work in this direction by integrating distributed ROS 2 applications with a Fabric network, with smart contracts for recording data [270], assigning roles [374] or managing multi-robot interaction [404]. Nonetheless, further work is required for enabling more widespread use and integration with ROS 2 systems as well as studying the performance in larger-scale systems.

Another family of DLTs relevant to multi-robot systems is DAG-based technologies, owing to the wider flexibility in terms of network topologies. IOTA is one of the most mature DLTs in this domain. The recent introduction of smart contracts within chains that are anchored to partitions of the DAG enables the design and implementation of distributed byzantine-tolerant decision making processes, as in Ethereum, but that are also tolerant to network partitions under certain conditions. We have also initiated work in this direction, with the integration of ROS 2 and IOTA, and the first byzantine-tolerant and partition-tolerant DLT-based solution for distributed consensus in multi-robot systems [271]. Further study needs to be carried out into the performance of such system in the real world, an specially under dynamic network topologies and evolving latency and bandwidth that are so far simulated.

8.2.2 UWB-based localization and collaborative sensing

Another area of research that started as part of this thesis and we are now continuing with new studies is UWB-based relative localization, specially in terms of designing and developing more scalable solutions. To this end, we have also started research with dynamic role allocation enabling hybrid ToF and TDoA localization schemes based on the spatial distribution of robots [345]. We have also explored recently the integration of DLTs for running the role allocation process within a smart contract [374]. Next steps will explore the integration of a mesh network for achieving situated communication, applicable in many spatial coordination and formation control problems. Multiple open research questions can also be found in the integration of UWB for cooperative state estimation together with other sensors. Recent literature has been delving into various types of sensor fusion approaches [343; 359; 360; 425]. Yet many challenges remain in developing more scalable methods (e.g., without relying in ToF only) and non-complete ranging graphs where some of the links need to be estimated from others.

In terms of collaborative sensing, the approach introduced in this thesis focuses around path planning to meet sensor constraints within a heterogeneous multi-robot system but relies on the localization system alone for merging data from different robots. Introducing optimization techniques and feedback from the data merging into the localization system would allow for higher accuracy in both areas. Therefore, there are multiple possibilities in the design of coupled and distributed localization and perception algorithms and methods.

8.2.3 Multi-robot systems in the wild

Many of the methods introduced in this thesis aimed at deploying robots in the wild. However, the experiments often remained under laboratory conditions. As more robust methods become available for robust operation in dynamic, unstructured and chaotic environments that occur in the real world, more research will move out of the lab. The integration of such uncertainty and the operation under challenging environmental conditions, among others, will require research that focuses more on the robustness of the methods, and the ability of multi-robot systems to adapt, selfheal and reconfigure when necessary. Power-on-and-go methods are also an area to enable higher degrees of autonomy, specially in new or unknown environments. A selection of open problems in the design of resilient multi-robot systems is introduced by Prorok et al. in [426]. These range from achieving efficient collaborative and federated learning to long-term operation, and including evolving world models and their validity, among many others.

List of References

[1] Jorge Peña Queralta, Qingqing Li, Eduardo Castelló Ferrer, and Tomi Westerlund. Secure encoded instruction graphs for end-to-end data validation in autonomous robots. *IEEE Internet of Things Journal*, 2022.

[2] Jorge Peña Queralta, Qingqing Li, Fabrizio Schiano, and Tomi Westerlund. Vio-uwb-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems. In *IEEE International Conference on Advanced Robotics and Mechatronics (ARM)*. IEEE, 2022.

[3] Jorge Peña Queralta, Carmen Martínez Almansa, Fabrizio Schiano, Dario Floreano, and Tomi Westerlund. UWB-based system for UAV localization in GNSS-denied environments: Characterization and dataset. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4521–4528. IEEE, 2020.

[4] Jorge Peña Queralta, Li Qingqing, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Distributed progressive formation control with one-way communication for multi-agent systems. In *2019 IEEE Symposium Series on Computational Intelligence*, 2019.

[5] Jorge Peña Queralta, Cassandra McCord, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Communication-free and index-free distributed formation control algorithm for multi-robot systems. *Procedia Computer Science*, 2019. The 10th ANT Conference.

[6] Cassandra McCord, Jorge Peña Queralta, Tuan Nguyen Gia, and Tomi Westerlund. Distributed progressive formation control for multi-agent systems: 2D and 3D deployment of UAVs in ROS/Gazebo with rotors. In *European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019. doi: https://doi.org/10.1109/ECMR.2019.8870934.

[7] Jorge Peña Queralta, Li Qingqing, Tuan Nguyen Gia, Hong-Linh Truong, and Tomi Westerlund. End-to-end design for self-reconfigurable heterogeneous robotic swarms. In *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 281–287. IEEE, 2020.

[8] J. Peña Queralta, L. Qingqing, Z. Zou, T. Westerlund. Enhancing autonomy with blockchain and multi-acess edge computing in distributed robotic systems. In *The Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2020.

[9] J. Peña Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception and active vision. *IEEE Access*, pages 1–1, 2020.

[10] Yu Xianjia, Li Qingqing, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Cooperative UWB-based localization for outdoors positioning and navigation of UAVs aided by ground robots. In *IEEE International Conference on Autonomous Systems (IEEE ICAS 2021)*. IEEE, 2021.

[11] Li Qingqing, Jorge Peña Queralta, Tuan Nguyen Gia, and Tomi Westerlund. Offloading monocular visual odometry with edge computing: Optimizing image quality in multi-robot systems. In *The 5th International Conference on Systems, Control and Communications*, pages 22–26. ACM, 2019. doi: https://doi.org/10.1145/3377458.3377467.

[12] Jorge Peña Queralta and Tomi Westerlund. Blockchain for mobile edge computing: Consensus mechanisms and scalability. In *Mobile Edge Computing*, pages 333–357. Springer, 2021.

[13] Jorge Peña Queralta, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Collaborative mapping with IoE-based heterogeneous vehicles for enhanced situational awareness. In *IEEE Sensors Applications Symposium (SAS)*. IEEE, 2019.

[14] Jorge Peña Queralta, Tuan Nguyen Gia, Zhuo Zou, Hannu Tenhunen, and Tomi Westerlund. Comparative study of lpwan technologies on unlicensed bands for m2m communication in the iot: beyond lora and lorawan. *Procedia Computer Science*, 2019. Presented at the 14th International Conference on Future Networks and Communications (FNC).

[15] J. Peña Queralta *et al.* Edge-AI in LoRa based healthcare monitoring: A case study on fall detection system with LSTM Recurrent Neural Networks. In *2019 42nd International Conference on Telecommunications, Signal Processing (TSP)*, 2019.

[16] Jorge Peña Queralta, Fu Yuhong, Lassi Salomaa, Li Qingqing, Tuan Nguyen Gia, Zhuo Zou, Hannu Tenhunen, and Tomi Westerlund. Fpga-based architecture for a low-cost 3d lidar design and implementation from multiple rotating 2d lidars with ros. In *IEEE Sensors Conference*. IEEE, 2019.

[17] Tuan Nguyen Gia, Jorge Peña Queralta, and Tomi Westerlund. Exploiting lora, edge, and fog computing for traffic monitoring in smart cities. *LPWAN Technologies for IoT and M2M Applications*, 2020. doi: https://doi.org/10.1016/B978-0-12-818880-4.00017-X.

[18] Li Qingqing, Jorge Peña Queralta, Tuan Nguyen Gia, Zhuo Zou, and Tomi Westerlund. Multi sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments. *Unmanned Systems*, 2020. doi: https://doi.org/10.1142/S2301385020500168. The 9th IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and the 9th IEEE International Conference on Robotics, Automation and Mechatronics (RAM).

[19] C. Martínez Almansa *et al.* Autocalibration of a mobile uwb localization system for ad-hoc multi-robot deployments in gnss-denied environments. *arXiv preprint arXiv:2004.06762*, 2020.

[20] Anum Nawaz, Jorge Peña Queralta, Jixin Guan, Muhammad Awais, Tuan Nguyen Gia, Ali Kashif, Haibin Kan, and Tomi Westerlund. Edge computing to secure IoT data ownership and trade with the ethereum blockchain. *Sensors*, 2020. doi: https://doi.org/10.3390/s20143965.

[21] Wang Shule, Carmen Martínez Almansa, Jorge Peña Queralta, Zhuo Zou, and Tomi Westerlund. UWB-based localization for multi-UAV systems and collaborative heterogeneous multi-robot systems: a survey. *Procedia Computer Science*, 175:357–364, 2020. doi: https://doi.org/10.1016/j.procs.2020.07.051. The 15th International Conference on Future Networks and Communications.

[22] Wenshuai Zhao, Jorge Peña Queralta, Li Qingqing, and Tomi Westerlund. Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning. In *5th International Conference on Robotics and Automation Engineering*. IEEE, 2020.

[23] Wenshuai Zhao, Jorge Peña Queralta, Li Qingqing, and Tomi Westerlund. Ubiquitous distributed deep reinforcement learning at the edge: Analyzing byzantine agents in discrete action spaces. In *The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)*. Elsevier, 2020.

[24] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *IEEE Symposium Series on Computational Intelligence*. IEEE, 2021. doi: https://doi.org/10.1109/SSCI47803.2020.9308468.

[25] Li Qingqing, Jussi Taipalmaa, Jorge Peña Queralta, Tuan Nguyen Gia, Moncef Gabbouj, Hannu Tenhunen, Jenni Raitoharju, and Tomi Westerlund. Towards active vision with UAVs in marine search and rescue: Analyzing human detection at variable altitudes. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020.

[26] Yu Xianjia, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Federated learning in robotic and autonomous systems. In *Procedia Computer Science*. Elsevier, 2021. doi: https://doi.org/10.1016/j.procs.2021.07.041. 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC-2021).

[27] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert

Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science robotics*, 3(14):eaar7650, 2018.

[28] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[29] Laura Lopez-Fuentes, Joost Weijer, Manuel González-Hidalgo, Harald Skinnemoen, and Andrew D. Bagdanov. Review on computer vision techniques in emergency situations. *Multimedia Tools Appl.*, 77(13), July 2018. ISSN 1380-7501.

[30] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, Ahmed Idries, and Farhan Mohammed. Unmanned aerial vehicles applications in future smart cities. *Technological forecasting and social change*, 153:119293, 2020.

[31] J. Tiemann *et al.* Scalable and precise multi-uav indoor navigation using tdoa-based uwb localization. In *IPIN*. IEEE, 2017.

[32] Jiafu Wan, Shenglong Tang, Qingsong Hua, Di Li, Chengliang Liu, and Jaime Lloret. Context-aware cloud robotics for material handling in cognitive industrial internet of things. *IEEE Internet of Things Journal*, 5(4):2272–2281, 2017.

[33] Romulo Gonçalves Lins and Sidney N Givigi. Cooperative robotics and machine learning for smart manufacturing: platform design and trends within the context of industrial internet of things. *IEEE Access*, 9:95444–95455, 2021.

[34] Leijian Yu, Erfu Yang, Peng Ren, Cai Luo, Gordon Dobie, Dongbing Gu, and Xiutian Yan. Inspection robots in oil and gas industry: a review of current solutions and future trends. In *2019 25th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2019.

[35] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, et al. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022.

[36] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[37] Félix Ingrand and Malik Ghallab. Deliberation for autonomous robots: A survey. *Artificial Intelligence*, 247:10 – 44, 2017. ISSN 0004-3702. Special Issue on AI and Robotics.

[38] Changjian Deng, Goumin Liu, and Fucun Qu. Survey of important issues in multi unmanned aerial vehicles imaging system. 2018.

[39] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7: 48572–48634, 2019.

[40] Saqib Mehmood, Shakeel Ahmed, Anders Schmidt Kristensen, and Dewan Ahsan. Multi criteria decision analysis (mcda) of unmanned aerial vehicles (uavs) as a part of standard response to emergencies. In *4th International Conference on Green Computing and Engineering Technologies; Niels Bohrs Vej 8, Esbjerg, Denmark*, 2018.

[41] William Roberts, Kelly Griendling, Anthony Gray, and D Mavris. Unmanned vehicle collaboration research environment for maritime search and rescue. In *30th Congress of the International Council of the Aeronautical Sciences*. International Council of the Aeronautical Sciences (ICAS) Bonn, Germany, 2016.

[42] Bing L Luk, David S Cooke, Stuart Galt, Arthur A Collie, and Sheng Chen. Intelligent legged climbing service robot for remote maintenance applications in hazardous environments. *Robotics and Autonomous Systems*, 53(2):142–152, 2005.

[43] Giacomo Lunghi, Raul Marin, Mario Di Castro, Alessandro Masi, and Pedro J Sanz. Multimodal human-robot interface for accessible remote robotic interventions in hazardous environments. *IEEE Access*, 7:127290–127319, 2019.

[44] Yoonchang Sung. *Multi-Robot Coordination for Hazardous Environmental Monitoring*. PhD thesis, Virginia Tech, 2019.

[45] Luis Merino, Fernando Caballero, JR Martinez-de Dios, and Aníbal Ollero. Cooperative fire detection using unmanned aerial vehicles. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1884–1889. IEEE, 2005.

[46] Sven Brenner, Sebastian Gelfert, and Hendrik Rust. New approach in 3d mapping and localization for search and rescue missions. *CERC2017*, page 105, 2017.

[47] Matthew Spenko, Stephen Buerger, and Karl Iagnemma. *The DARPA robotics challenge finals: humanoid robots to the rescue*, volume 121. Springer, 2018.

[48] Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf, Anant Gupta, Oskar Von Stryk, Felipe Bacim, Doug A Bowman, Alex Goins, Ravi Balasubramanian, and David C Conner. Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials. *Journal of Field Robotics*, 32(3):352–377, 2015.

[49] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurnỳ, et al. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 274–290. Springer, 2019.

[50] Vojtech Spurnỳ Saska, Tomas Báca, Matej Petrlık, Tomas Krajnık, and Tomas Roucek. Darpa subt stix qualification submission: Ctu-cras.

[51] Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176, 2020.

[52] Yi-Wei Huang, Chen-Lung Lu, Kuan-Lin Chen, Po-Sheng Ser, Jui-Te Huang, Yu-Chia Shen, Pin-Wei Chen, Po-Kai Chang, Sheng-Cheng Lee, and Hsueh-Cheng Wang. Duckiefloat: a collision-tolerant resource-constrained blimp for long-term autonomy in subterranean environments. *arXiv preprint arXiv:1910.14275*, 2019.

[53] Anton Koval, Christoforos Kanellakis, Emil Vidmark, Jakub Haluska, and George Nikolakopoulos. A subterranean virtual cave world for gazebo based on the darpa subt challenge. *arXiv preprint arXiv:2004.08452*, 2020.

[54] Jan Faigl, Olivier Simonin, and François Charpillet. Comparison of task-allocation algorithms in frontier-based multi-robot exploration. In *European Conference on Multi-Agent Systems*, pages 101–110. Springer, 2014.

[55] Ahmed Hussein, Mohamed Adel, Mohamed Bakr, Omar M Shehata, and Alaa Khamis. Multi-robot task allocation for search and rescue missions. In *Journal of Physics: Conference Series*, volume 570, page 052006, 2014.

[56] Wanqing Zhao, Qinggang Meng, and Paul WH Chung. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. *IEEE transactions on cybernetics*, 46(4):902–915, 2015.

[57] Jian Tang, Kejun Zhu, Haixiang Guo, Chengzhu Gong, Can Liao, and Shuwen Zhang. Using auction-based task allocation scheme for simulation optimization of search and rescue in disaster relief. *Simulation Modelling Practice and Theory*, 82:132–146, 2018.

[58] Tadewos G Tadewos, Laya Shamgah, and Ali Karimoddini. On-the-fly decentralized tasking of autonomous vehicles. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 2770–2775. IEEE, 2019.

[59] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[60] Alejandro R Mosteo and Luis Montano. A survey of multi-robot task allocation. *Instituto de Investigacin en Ingenierła de Aragn (I3A), Tech. Rep*, 2010.

[61] Heba Kurdi, Jonathon How, and Guillermo Bautista. Bio-inspired algorithm for task allocation in multi-uav search and rescue missions. In *Aiaa guidance, navigation, and control conference*, page 1377, 2016.

[62] Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. Dec-mcts: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, 38(2-3):316–337, 2019.

[63] Graeme Best, Jan Faigl, and Robert Fitch. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42(4):715–738, 2018. ISSN 15737527.

[64] Shubhani Aggarwal and Neeraj Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 149:270–299, 2020.

[65] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Informative path planning for an autonomous underwater vehicle. In *2010 IEEE International Conference on Robotics and Automation*, pages 4791–4796. IEEE, 2010.

[66] Yongyong Wei and Rong Zheng. Informative path planning for mobile sensing with reinforcement learning. *arXiv preprint arXiv:2002.07890*, 2020.

[67] Tauã M Cabreira, Lisane B Brisolara, and Paulo R Ferreira Jr. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1):4, 2019.

[68] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

[69] Susan Hert and Vladimir Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computational Geometry & Applications*, 8(04):437–466, 1998.

[70] JF Araujo, PB Sujit, and João B Sousa. Multiple uav area decomposition and coverage. In *2013 IEEE symposium on computational intelligence for security and defense applications (CISDA)*, pages 30–37. IEEE, 2013.

[71] Junfei Xie, Luis Rodolfo Garcia Carrillo, and Lei Jin. Path planning for uav to cover multiple separated convex polygonal regions. *IEEE Access*, 8:51770–51785, 2020.

[72] Juan Irving Vasquez-Gomez, Juan-Carlos Herrera-Lozada, and Mauricio Olguin-Carbajal. Coverage path planning for surveying disjoint areas. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 899–904. IEEE, 2018.

[73] Yiannis Kantaros, Michalis Thanou, and Anthony Tzes. Visibility-oriented coverage control of mobile robotic networks on non-convex regions. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1126–1131. IEEE, 2014.

[74] Samira Hayat, Evşen Yanmaz, Timothy X Brown, and Christian Bettstetter. Multi-objective uav path planning for search and rescue. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5569–5574. IEEE, 2017.

[75] Paraskevi S Georgiadou, Ioannis A Papazoglou, Chris T Kiranoudis, and Nikolaos C Markatos. Multi-objective emergency response optimization around chemical plants. In *MULTI-OBJECTIVE OPTIMIZATION: Techniques and Application in Chemical Engineering*, pages 355–378. World Scientific, 2017.

[76] M. Garey, D. Johnson, and H. Witsenhausen. The complexity of the generalized lloyd - max problem (corresp.). *IEEE Transactions on Information Theory*, 28(2):255–256, 1982.

[77] Israel Lugo-Cárdenas, Gerardo Flores, Sergio Salazar, and Rogelio Lozano. Dubins path generation for a fixed wing uav. In *2014 International conference on unmanned aircraft systems (ICUAS)*, pages 339–346. IEEE, 2014.

[78] Yulei Liao, Lei Wan, and Jiayuan Zhuang. Full state-feedback stabilization of an underactuated unmanned surface vehicle. In *2010 2nd International Conference on Advanced Computer Control*, volume 4, pages 70–74. IEEE, 2010.

[79] Ertugrul Cetinsoy. Design and flight tests of a holonomic quadrotor uav with sub-rotor control surfaces. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 1197–1202. IEEE, 2013.

[80] Riichiro Damoto, Wendy Cheng, and Shigeo Hirose. Holonomic omnidirectional vehicle with new omni-wheel mechanism. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 773–778. IEEE, 2001.

[81] S Campbell, Wasif Naeem, and George W Irwin. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, 36(2):267–283, 2012.

[82] Zheng Zeng, Lian Lian, Karl Sammut, Fangpo He, Youhong Tang, and Andrew Lammas. A survey on path planning for persistent autonomy of autonomous underwater vehicles. *Ocean Engineering*, 110:303–313, 2015.

[83] Daoliang Li, Peng Wang, and Ling Du. Path planning technologies for autonomous underwater vehicles-a review. *IEEE Access*, 7:9745–9768, 2018.

[84] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, 2013.

[85] Luís C Santos, Filipe N Santos, EJ Solteiro Pires, António Valente, Pedro Costa, and Sandro Magalhães. Path planning for ground robots in agriculture: a short review. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 61–66. IEEE, 2020.

[86] Rachid Alami, Frédéric Robert, Félix Ingrand, and Sho'ji Suzuki. Multi-robot cooperation through incremental plan-merging. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 2573–2579. IEEE, 1995.

[87] Jing Yuan, Yalou Huang, Tong Tao, and Fengchi Sun. A cooperative approach for multi-robot area exploration. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1390–1395. IEEE, 2010.

[88] Ayan Dutta, Anirban Ghosh, and O Patrick Kreidl. Multi-robot informative path planning with continuous connectivity constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3245–3251. IEEE, 2019.

[89] Rajnesh Kumar Singh and Neelu Jain. Comparative study of multi-robot area exploration algorithms. *International Journal*, 4(8), 2014.

[90] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[91] Karl Muecke and Brian Powell. A distributed, heterogeneous, target-optimized operating system for a multi-robot search and rescue application. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 266–275. Springer, 2011.

[92] Farshid Abbasi Doustvatan. *Coverage control for heterogeneous multi-agent systems*. PhD thesis, University of Georgia, 2016.

[93] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.

[94] Md Nazmuzzaman Khan and Sohel Anwar. Paradox elimination in dempster–shafer combination rule with novel entropy function: Application in decision-level multi-sensor fusion. *Sensors*, 19 (21):4810, 2019.

[95] Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.

[96] Tong Meng, Xuyang Jing, Zheng Yan, and Witold Pedrycz. A survey on machine learning for data fusion. *Information Fusion*, 57:115 – 129, 2020. ISSN 1566-2535.

[97] T. Baltrušaitis, C. Ahuja, and L. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[98] Jia Liu, Tianrui Li, Peng Xie, Shengdong Du, Fei Teng, and Xin Yang. Urban big data fusion based on deep learning: An overview. *Information Fusion*, 53:123 – 133, 2020. ISSN 1566-2535.

[99] Inkyu Sa, Stefan Hrabar, and Peter Corke. Inspection of pole-like structures using a vision-controlled vtol uav and shared autonomy. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4819–4826. IEEE, 2014.

[100] Pat Marion, Maurice Fallon, Robin Deits, Andrés Valenzuela, Claudia Pérez D'Arpino, Greg Izatt, Lucas Manuelli, Matt Antone, Hongkai Dai, Twan Koolen, et al. Director: A user interface

designed for robot operation with shared autonomy. In *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, pages 237–270. Springer, 2018.

[101] Carlo Masone, Paolo Robuffo Giordano, Heinrich H Bülthoff, and Antonio Franchi. Semi-autonomous trajectory generation for mobile robots with integral haptic shared control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6468–6475. IEEE, 2014.

[102] Antonio Franchi, Cristian Secchi, Markus Ryll, Heinrich H Bulthoff, and Paolo Robuffo Giordano. Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics & Automation Magazine*, 19(3):57–68, 2012.

[103] Dongjun Lee, Antonio Franchi, Hyoung Il Son, ChangSu Ha, Heinrich H Bülthoff, and Paolo Robuffo Giordano. Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *IEEE/ASME Transactions on Mechatronics*, 18(4):1334–1345, 2013.

[104] Imad Jawhar, Nader Mohamed, Jie Wu, and Jameela Al-Jaroodi. Networking of multi-robot systems: architectures and requirements. *Journal of Sensor and Actuator Networks*, 7(4):52, 2018.

[105] Ashley W Stroupe, Martin C Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No. 01CH37164)*, volume 2, pages 1092–1098. IEEE, 2001.

[106] Kasper Støy. Using situated communication in distributed autonomous mobile robotics. In *Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, SCAI '01, page 44–52, NLD, 2001. IOS Press. ISBN 1586031619.

[107] Joydeep Biswas and Manuela Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *2010 IEEE international conference on robotics and automation*, pages 4379–4384. IEEE, 2010.

[108] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. Spotfi: Decimeter level localization using wifi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 269–282, 2015.

[109] Wei Sun, Min Xue, Hongshan Yu, Hongwei Tang, and Anping Lin. Augmentation of fingerprints for indoor wifi localization based on gaussian process regression. *IEEE Transactions on Vehicular Technology*, 67(11):10896–10905, 2018.

[110] Marco Altini, Davide Brunelli, Elisabetta Farella, and Luca Benini. Bluetooth indoor localization with multiple neural networks. In *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, pages 295–300. IEEE, 2010.

[111] Pavel Kriz, Filip Maly, and Tomas Kozel. Improving indoor localization using bluetooth low energy beacons. *Mobile Information Systems*, 2016, 2016.

[112] Juthatip Wisanmongkol, Ladawan Klinkusoom, Taweesak Sanpechuda, La-or Kovavisaruch, and Kamol Kaemarungsi. Multipath mitigation for rssi-based bluetooth low energy localization. In *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, pages 47–51. IEEE, 2019.

[113] Loizos Kanaris, Akis Kokkinis, Antonio Liotta, and Stavros Stavrou. Fusing bluetooth beacon data with wi-fi radiomaps for improved indoor localization. *Sensors*, 17(4):812, 2017.

[114] Nitesh B Suryavanshi, K Viswavardhan Reddy, and Vishnu R Chandrika. Direction finding capability in bluetooth 5.1 standard. In *International Conference on Ubiquitous Communications and Network Computing*, pages 53–65. Springer, 2019.

[115] Aftab Khan, Stephen Wang, and Ziming Zhu. Angle-of-arrival estimation using an adaptive machine learning framework. *IEEE Communications Letters*, 23(2):294–297, 2018.

[116] Di Tian and Nicolas D Georganas. Connectivity maintenance and coverage preservation in wireless sensor networks. *Ad Hoc Networks*, 3(6):744–761, 2005.

[117] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12):1411–1423, 2013.

[118] Lorenzo Sabattini, Cristian Secchi, Nikhil Chopra, and Andrea Gasparri. Distributed control of multirobot systems with global connectivity maintenance. *IEEE Transactions on Robotics*, 29 (5):1326–1332, 2013.

[119] Li Wang, Aaron D Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 2659–2664. IEEE, 2016.

[120] Hu Xiao, Rongxin Cui, and Demin Xu. Cooperative multi-agent search using bayesian approach with connectivity maintenance. *Assembly Automation*, 2019.

[121] Qian Zhu, Rui Zhou, and Jie Zhang. Connectivity maintenance based on multiple relay uavs selection scheme in cooperative surveillance. *Applied Sciences*, 7(1):8, 2017.

[122] Jacopo Panerati, Marco Minelli, Cinara Ghedini, Lucas Meyer, Marcel Kaufmann, Lorenzo Sabattini, and Giovanni Beltrame. Robust connectivity maintenance for fallible robots. *Autonomous Robots*, 43(3):769–787, 2019.

[123] Cinara Ghedini, Carlos HC Ribeiro, and Lorenzo Sabattini. A decentralized control strategy for resilient connectivity maintenance in multi-robot systems subject to failures. In *Distributed Autonomous Robotic Systems*, pages 89–102. Springer, 2018.

[124] Luca Siligardi, Jacopo Panerati, Marcel Kaufmann, Marco Minelli, Cinara Ghedini, Giovanni Beltrame, and Lorenzo Sabattini. Robust area coverage with connectivity maintenance. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2202–2208. IEEE, 2019.

[125] Koresh Khateri, Mahdi Pourgholi, Mohsen Montazeri, and Lorenzo Sabattini. A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks. *IEEE Robotics and Automation Letters*, 4(2):633–640, 2019.

[126] Francesco Amigoni, Jacopo Banfi, Nicola Basilico, Ioannis Rekleitis, and Alberto Quattrini Li. Online update of communication maps for exploring multirobot systems under connectivity constraints. In *Distributed Autonomous Robotic Systems*, pages 513–526. Springer, 2019.

[127] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. Multirobot exploration of communication-restricted environments: A survey. *IEEE Intelligent Systems*, 32(6):48–57, 2017.

[128] Yicong Tian, Chen Chen, and Mubarak Shah. Cross-view image matching for geo-localization in urban environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3616, 2017.

[129] Bo Zhou, Zhongqiang Tang, Kun Qian, Fang Fang, and Xudong Ma. A lidar odometry for outdoor mobile robots using ndt based scan matching in gps-denied environments. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1230–1235. IEEE, 2017.

[130] Y. Lu *et al.* A survey on vision-based uav navigation. *Geo-spatial information science*, 21(1), 2018.

[131] T. Qin *et al.* Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4), 2018.

[132] T. Qin *et al.* A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

[133] J. Zhang *et al.* Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[134] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[135] Aufar Zakiev, Tatyana Tsoy, and Evgeni Magid. Swarm robotics: remarks on terminology and classification. In *International conference on interactive collaborative robotics*, pages 291–300. Springer, 2018.

[136] Jiaqiang Zhang, Farhad Keramat, Xianjia Yu, Daniel Montero Hern, Jorge Peña Queralta, and Tomi Westerlund. Distributed robotic systems in the edge-cloud continuum with ros 2: a review

on novel architectures and technology readiness. In *The Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2022.

[137] Eduardo Castelló Ferrer. The blockchain: a new framework for robotic swarm systems. In *Proceedings of the Future Technologies Conference*, pages 1037–1058. Springer, 2018.

[138] W. Shi, G. Pallis, and Z. Xu. Edge computing [scanning the issue]. *Proceedings of the IEEE*, 107(8):1474–1481, Aug 2019. doi: 10.1109/JPROC.2019.2928287.

[139] Ana Juan Ferrer, Joan Manuel Marques, and Josep Jorba. Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. *ACM Comput. Surv.*, 51(6):111:1–111:36, January 2019. ISSN 0360-0300. doi: 10.1145/3243929.

[140] Fatemeh Jalali, Olivia J. Smith, Timothy Lynar, and Frank Suits. Cognitive iot gateways: Automatic task sharing and switching between cloud and edge/fog computing. In *Proceedings of the SIGCOMM Posters and Demos*, SIGCOMM Posters and Demos '17, pages 121–123, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5057-0. doi: 10.1145/3123878.3132008.

[141] Davide Calvaresi, Mauro Marinoni, Arnon Sturm, Michael Schumacher, and Giorgio Buttazzo. The challenge of real-time multi-agent systems for enabling iot and cps. In *Proceedings of the international conference on web intelligence*, pages 356–364. ACM, 2017.

[142] Jacques Penders, Lyuba Alboul, Ulf Witkowski, Amir Naghsh, Joan Saez-Pons, Stefan Herbrechtsmeier, and Mohamed El-Habbal. A robot swarm assisting a human fire-fighter. *Advanced Robotics*, 25(1-2):93–117, 2011.

[143] Joachim De Greeff, Nanja Smets, Koen Hindriks, Mark A Neerincx, and Ivana Kruijff-Korbayová. Incremental development of large-scale human-robot teamwork in disaster response environments. In *12th Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI 2017*, pages 101–102. ACM, IEEE Computer Society, 2017.

[144] Ellips Masehian, Marjan Jannati, and Taher Hekmatfar. Cooperative mapping of unknown environments by multiple heterogeneous mobile robots with limited sensing. *Robotics and Autonomous Systems*, 87:188–218, 2017.

[145] Ajay Kattepur, Harshit Dohare, Visali Mushunuri, Hemant Kumar Rath, and Anantha Simha. Resource constrained offloading in fog computing. In *Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets*, page 1. ACM, 2016.

[146] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng. Qos-aware cooperative computation offloading for robot swarms in cloud robotics. *IEEE Transactions on Vehicular Technology*, 68(4): 4027–4041, April 2019. doi: 10.1109/TVT.2019.2901761.

[147] D. Palossi, F. Conti, and L. Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 604–611, 2019.

[148] Li Qingqing, J Peñia Queralta, Tuan Nguyen Gia, Hannu Tenhunen, Zhuo Zou, and Tomi Westerlund. Visual odometry offloading in internet of vehicles with compression at the edge of the network. In *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pages 1–2. IEEE, 2019.

[149] Jorge Peña Queralta and Tomi Westerlund. Blockchain-powered collaboration in heterogeneous swarms of robots. *arXiv preprint arXiv:1912.01711*, 2020. Symposium on Blockchain for Robotic Systems, MIT Media Lab.

[150] Xiao Yan Deng, Phil Trinder, and Greg Michaelson. Autonomous mobile programs. In *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 177–186. IEEE, 2006.

[151] Zehua Zhang and Xuejie Zhang. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, volume 2, pages 240–243. IEEE, 2010.

[152] Soumya Banerjee and Joshua P Hecker. A multi-agent system approach to load-balancing and resource allocation for distributed computing. In *First Complex Systems Digital Campus World E-Conference 2015*, pages 41–54. Springer, 2017.

[153] Murugappan Elango, Subramanian Nachiappan, and Manoj Kumar Tiwari. Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Systems with Applications*, 38(6):6486–6491, 2011.

[154] Avinash Gautam, SP Arjun Ram, Virendra Singh Shekhawat, and Sudeept Mohan. Balanced partitioning of workspace for efficient multi-robot coordination. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 104–109. IEEE, 2017.

[155] Hong-Linh Truong. Asre -towards application-specific resource ensembles across edges and clouds, 2020. Preprint, working paper.

[156] Jason Gregory, Jonathan Fink, Ethan Stump, Jeffrey Twigg, John Rogers, David Baran, Nicholas Fung, and Stuart Young. Application of multi-robot systems to disaster-relief scenarios with limited communication. In *Field and Service Robotics*, pages 639–653. Springer, 2016.

[157] Menglan Hu, Weidong Liu, Kai Peng, Xiaoqiang Ma, Wenqing Cheng, Jiangchuan Liu, and Bo Li. Joint routing and scheduling for vehicle-assisted multidrone surveillance. *IEEE Internet of Things Journal*, 6(2):1781–1790, 2018.

[158] Nuno Mendes, Mohammad Safeea, and Pedro Neto. Flexible programming and orchestration of collaborative robotic manufacturing systems. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 913–918. IEEE, 2018.

[159] Vincenzo Lomonaco, Angelo Trotta, Marta Ziosi, Juan de Dios Yáñez Ávila, and Natalia Díaz-Rodríguez. Intelligent drone swarm for search and rescue operations at sea. *arXiv preprint arXiv:1811.05291*, 2018.

[160] Hong-Linh Truong, Luca Berardinelli, Ivan Pavkovic, and Georgiana Copil. Modeling and provisioning iot cloud systems for testing uncertainties. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 96–105. ACM, 2017.

[161] V. Medel, O. Rana, J. A. Bañares, and U. Arronategui. Modelling performance amp; resource management in kubernetes. In *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pages 257–262, Dec 2016.

[162] Ronan-Alexandre Cherrueau, Adrien Lebre, Dimitri Pertin, Fetahi Wuhib, and João Monteiro Soares. Edge computing resource management system: a critical building block! initiating the debate via openstack. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, 2018. USENIX Association.

[163] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong Linh Truong. Principles of elastic processes. *IEEE Internet Computing*, 15(5):66–71, 2011. doi: 10.1109/MIC.2011.121.

[164] C. McCord *et al.* Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of UAVs in ROS/Gazebo with RotorS. In *ECMR*. IEEE, 2019.

[165] Minh T Nguyen, Hung M La, and Keith A Teague. Collaborative and compressed mobile sensing for data collection in distributed robotic networks. *IEEE Transactions on Control of Network Systems*, 5(4):1729–1740, 2017.

[166] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, 2017.

[167] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.

[168] L Qingqing, F Yuhong, J Pena Queralta, Tuan Nguyen Gia, Hannu Tenhunen, Zhuo Zou, and Tomi Westerlund. Edge computing for mobile robots: multi-robot feature-based lidar odometry with fpgas. In *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pages 1–2. IEEE, 2019.

[169] Jorge Peña Queralta, F Yuhong, L Salomaa, L Qingqing, TN Gia, Z Zou, H Tenhunen, and T Westerlund. Fpga-based architecture for a low-cost 3d lidar design and implementation from multiple rotating 2d lidars with ros. In *2019 IEEE SENSORS*, pages 1–4. IEEE, 2019.

[170] W. Shule *et al.* Uwb-based localization for multi-uav systems and collaborative heterogeneous multi-robot systems: a survey. *arXiv preprint arXiv:2004.08174*, 2020.

[171] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In *AAMAS*, pages 541–549, 2018.

[172] Hong Linh Truong and Schahram Dustdar. Principles of software-defined elastic systems for big data analytics. In *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, IC2E '14, pages 562–567, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-3766-0. doi: 10.1109/IC2E.2014.67.

[173] Stefano Mariani, Hong-Linh Truong, Georgiana Copil, Andrea Omicini, and Schahram Dustdar. Coordination-aware elasticity. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, pages 465–472, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-7881-6. doi: 10.1109/UCC.2014.59.

[174] P. Yun *et al.* Towards a cloud robotics platform for distributed visual slam. In *Computer Vision Systems*. Springer, 2017.

[175] S. Dey *et al.* Robotic slam: A review from fog computing and mobile edge computing perspective. In *MOBIQUITOUS*. ACM, 2016.

[176] Victor Kathan Sarker, Jorge Peña Queralta, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Offloading slam for indoor mobile robots with edge-fog-cloud computing. In *International Conference on Advances in Science, Engineering and Robotics Technology*. IEEE, 2019. doi: 10.1109/ICASERT.2019.8934466.

[177] Aly Metwaly, Jorge Peña Queralta, Victor Kathan Sarker, Tuan Nguyen Gia, Omar Nasir, and Tomi Westerlund. Edge computing with embedded AI: Thermal image analysis for occupancy estimation in intelligent buildings. In *INTelligent Embedded Systems Architectures and Applications, INTESA@ESWEEK 2019*, page 1–6. ACM, 2019. doi: https://doi.org/10.1145/3372394.3372397.

[178] T. N. Gia *et al.* Artificial Intelligence at the Edge in the Blockchain of Things. In *8th EAI Mobihealth*, 2019.

[179] A. Nawaz *et al.* Edge AI and Blockchain for Privacy-Critical and Data-Sensitive Applications. In *The 12th ICMU*, 2019.

[180] T. N. Gia *et al.* Edge AI in Smart Farming IoT: CNNs at the Edge and Fog Computing with LoRa. In *2019 IEEE AFRICON*, 2019.

[181] M. Burri *et al.* The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[182] C. Cadena *et al.* Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on robotics*, 2016.

[183] Raúl *et al.* Mur-Artal. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[184] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.

[185] Satoshi Nakamoto et al. *Bitcoin: A peer-to-peer electronic cash system*. Working Paper, 2008.

[186] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[187] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[188] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1), 2018.

[189] Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE, 2015.

[190] Saide Zhu, Wei Li, Hong Li, Chunqiang Hu, and Zhipeng Cai. A survey: Reward distribution mechanisms and withholding attacks in bitcoin pool mining. *Mathematical Foundations of Computing*, 1(4):393–414, 2018.

[191] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.

[192] Andrew Miller, Ahmed Kosba, Jonathan Katz, and Elaine Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 680–691. ACM, 2015.

[193] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 1:6, 2013.

[194] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2): 122–144, 2004.

[195] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

[196] Serguei Popov. A probabilistic analysis of the nxt forging algorithm. *Ledger*, 1:69–83, 2016.

[197] Nxt Wiki. *Whitepaper: Nxt*. Nxtwiki. org [online] https://nxtwiki. org, 2018.

[198] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.

[199] Vitalik Buterin, Daniel Reijsbergen, Stefanos Leonardos, and Georgios Piliouras. Incentives in ethereum's hybrid casper protocol. *arXiv preprint arXiv:1903.04205*, 2019.

[200] Karl J O'Dwyer and David Malone. *Bitcoin mining and its energy footprint*. IET, 2014.

[201] Alex De Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.

[202] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[203] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, page 4, 2016.

[204] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[205] Joao Sousa, Alysson Bessani, and Marko Vukolic. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 51–58. IEEE, 2018.

[206] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[207] RM Keichafer, Chris J. Walter, Alan M. Finn, and Philip M. Thambidurai. The maft architecture for distributed fault tolerance. *IEEE Transactions on Computers*, 37(4):398–404, 1988.

[208] Everett Hildenbrandt, Manasvi Saxena, Nishant Rodrigues, Xiaoran Zhu, Philip Daian, Dwight Guth, Brandon Moore, Daejun Park, Yi Zhang, Andrei Stefanescu, et al. Kevm: A complete formal semantics of the ethereum virtual machine. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 204–217. IEEE, 2018.

[209] Ethereum Revision 7709ece9. *Solidity Documentation*. Solidity Read The Docs [online] https://solidity.readthedocs.io/en/v0.5.12/. Accessed October 2019., 2016-2019.

[210] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[211] Dejan Vujičić, Dijana Jagodić, and Siniša Randić. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE, 2018.

[212] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer, 2015.

[213] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[214] Deadalnix's den. *Using Merklix tree to shard block validation.* Accessed October 2019 [online] https://www.deadalnix.me/2016/11/06/using-merklix-tree-to-shard-block-validation/, 2016.

[215] Deadalnix's den. *Introducing Merklix tree as an unordered Merkle tree on steroid.* Accessed October 2019 [online] https://www.deadalnix.me/2016/09/24/introducing-merklix-tree-as-an-unordered-merkle-tree-on-steroid/, 2016.

[216] Bo Qin, Jikun Huang, Qin Wang, Xizhao Luo, Bin Liang, and Wenchang Shi. Cecoin: A decentralized pki mitigating mitm attacks. *Future Generation Computer Systems*, 2017.

[217] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3:37, 2014.

[218] Serenity Ethereum Foundation et al. *Ethereum 2.0 Specifications.* Accessed October 2019 [online] https://github.com/ethereum/eth2.0-specs, 2018.

[219] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[220] Ben Edington. *Exploring Ethereum 2.0 Design Goals.* Consensys. Accessed October 2019 [online] https://media.consensys.net/exploring-the-ethereum-2-0-design-goals-fd2d901b4c01, January 2017.

[221] EthHub. *Ethereum 2.0 (Serenity) Phases.* Accessed October 2019 [online] https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/eth-2.0-phases/, 2018.

[222] Ben Edington. *State of Ethereum Protocol #1.* Consensys. Accessed October 2019 [online] https://media.consensys.net/state-of-ethereum-protocol-1-d3211dd0f6, August 2018.

[223] Ben Edington. *State of Ethereum Protocol #2.* Consensys. Accessed October 2019 [online] https://media.consensys.net/state-of-ethereum-protocol-2-the-beacon-chain-c6b6a9a69129, October 2018.

[224] Vitalik Buterin and others. *On sharding blockchains.* Ethereum Wiki. Sharding FAQ. Accessed October 2019. Version April 18th, 2019 [online] https://github.com/ethereum/wiki/wiki/Sharding-FAQ, 2016-2019.

[225] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[226] Mischa Dohler, Toktam Mahmoodi, Maria A Lema, Massimo Condoluci, Fragkiskos Sardis, Konstantinos Antonakoglou, and Hamid Aghvami. Internet of skills, where robotics meets ai, 5g and the tactile internet. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5. IEEE, 2017.

[227] Rongpeng Li, Zhifeng Zhao, Xuan Zhou, Guoru Ding, Yan Chen, Zhongyao Wang, and Honggang Zhang. Intelligent 5g: When cellular networks meet artificial intelligence. *IEEE Wireless communications*, 24(5):175–183, 2017.

[228] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid. Internet of things in the 5g era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, 34(3):510–527, 2016.

[229] Jere Backman, Seppo Yrjölä, Kristiina Valtanen, and Olli Mämmelä. Blockchain network slice broker in 5g: Slice leasing in factory of the future use case. In *2017 Internet of Things Business Models, Users, and Networks*, pages 1–8. IEEE, 2017.

[230] F. Voigtländer *et al.* 5g for robotics: Ultra-low latency control of distributed robotic systems. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, pages 69–72. IEEE, 2017.

[231] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[232] Rami Akrem Addad, Tarik Taleb, Miloud Bagaa, Diego Leonel Cadette Dutra, and Hannu Flinck. Towards modeling cross-domain network slices for 5g. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[233] Syed Husain, Andreas Kunz, Athul Prasad, Konstantinos Samdanis, and JaeSeung Song. Mobile edge computing with network resource slicing for internet-of-things. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2018.

[234] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[235] Dario Sabella, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, 5(4):84–91, 2016.

[236] Li Qingqing, Jorge Pena Queralta, Tuan Nguyen Gia, and Tomi Westerlund. Offloading monocular visual odometry with edge computing: Optimizing image quality in multi-robot systems. In *Proceedings of the 2019 5th International Conference on Systems, Control and Communications*, pages 22–26, 2019.

[237] Ke Zhang, Yuming Mao, Supeng Leng, Quanxin Zhao, Longjiang Li, Xin Peng, Li Pan, Sabita Maharjan, and Yan Zhang. Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. *IEEE access*, 4:5896–5907, 2016.

[238] Ke Zhang, Yuming Mao, Supeng Leng, Yejun He, and Yan Zhang. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Vehicular Technology Magazine*, 12(2):36–44, 2017.

[239] Agata Bareiś, Michał Bareiś, and Christian Bettstetter. Robots that sync and swarm: a proof of concept in ros 2. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 98–104. IEEE, 2019.

[240] Heiko G Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159–162, 2016.

[241] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019.

[242] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015.

[243] Kehua Su, Jie Li, and Hongbo Fu. Smart city and the applications. In *2011 international conference on electronics, communications and control (ICECC)*, pages 1028–1031. IEEE, 2011.

[244] S Kekki *et al.* Mec in 5g networks. *ETSI white paper*, 28:1–28, 2018.

[245] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[246] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[247] 3GPP. Study on architecture for next-generation system rel. 14. *Techical Report*, 23.799, 2016.

[248] N. Alliance. Description of network slicing concept. *NGMN 5G P*, 1:1, 2016.

[249] Fabio Giust, Vincenzo Sciancalepore, Dario Sabella, Miltiades C Filippou, Simone Mangiante, Walter Featherstone, and Daniele Munaretto. Multi-access edge computing: The driver behind the wheel of 5g-connected cars. *IEEE Communications Standards Magazine*, 2(3):66–73, 2018.

[250] Zehui Xiong, Yang Zhang, Dusit Niyato, Ping Wang, and Zhu Han. When mobile blockchain meets edge computing. *IEEE Communications Magazine*, 56(8):33–39, 2018.

[251] MD Abdur Rahman, M Shamim Hossain, George Loukas, Elham Hassanain, Syed Sadiqur Rahman, Mohammed F Alhamid, and Mohsen Guizani. Blockchain-based mobile edge computing framework for secure therapy applications. *IEEE Access*, 6:72469–72478, 2018.

[252] Yueyue Dai, Du Xu, Sabita Maharjan, Zhuang Chen, Qian He, and Yan Zhang. Blockchain and deep reinforcement learning empowered intelligent 5g beyond. *IEEE Network*, 33(3):10–17, 2019.

[253] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Joint computation offloading and content caching for wireless blockchain networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 517–522. IEEE, 2018.

[254] He Zhu, Changcheng Huang, and Jiayu Zhou. Edgechain: Blockchain-based multi-vendor mobile edge application placement. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 222–226. IEEE, 2018.

[255] Li Qingqing, Jorge Pena Queralta, Tuan Nguyen Gia, Zhuo Zou, and Tomi Westerlund. Multi sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments. *The 9th IEEE CIS-RAM*, 2019.

[256] Synced. The golden age of hd mapping for autonomous driving. *Medium*, 2018.

[257] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

[258] Saman Biookaghazadeh, Ming Zhao, and Fengbo Ren. Are fpgas suitable for edge computing? In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[259] T Ohkawa, Y Ishida, Y Sugata, and H Tamukoh. Ros-compliant fpga component technology—fpga installation into ros, 2017.

[260] The European Union Agency for Cybersecurity. Threat assessment for the fifth generation of mobile telecommunications networks (5g). *ENISA THREAT LANDSCAPE FOR 5G NETWORKS*, 2019.

[261] Ronald C Arkin and Tucker Balch. Cooperative multiagent robotic systems. 1997.

[262] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2321–2343, 2003.

[263] Elio Tuci, Muhanad H Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018.

[264] Zeng-Shun Zhao, Xiang Feng, Yan-yan Lin, Fang Wei, Shi-Ku Wang, Tong-Lu Xiao, Mao-Yong Cao, and Zeng-Guang Hou. Evolved neural network ensemble by multiple heterogeneous swarm intelligence. *Neurocomputing*, 149:29–38, 2015.

[265] Zohreh Akbari and Rainer Unland. A novel heterogeneous swarm reinforcement learning method for sequential decision making problems. *Machine Learning and Knowledge Extraction*, 1(2):590–610, 2019.

[266] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[267] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[268] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[269] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[270] Salma Salimi, Jorge Peña Queralta, and Tomi Westerlund. Towards managing industrial robot fleets with hyperledger fabric blockchain and ROS 2. *arXiv preprint*, 2022.

[271] Farhad Keramat, Jorge Peña Queralta, and Tomi Westerlund. Byzantine-tolerant and partition-tolerant decision-making for distributed robotic systems with iota and ros 2. *arXiv preprint*, 2022.

[272] E. Ferrer *et al.* Secure and secret cooperation in robot swarms. *Science Robotics*, 6(56), 2021. doi: 10.1126/scirobotics.abf1538.

[273] L. A. Kirschgens *et al.* Robot hazards: from safety to security. *arXiv preprint arXiv:1806.06681*, 2018.

[274] S. Bragança *et al.* A brief overview of the use of collaborative robots in industry 4.0: human role and safety. In *Occupational and Environmental Safety and Health*. Springer, 2019.

[275] J. Huang *et al.* Rosrv: Runtime verification for robots. In *International Conference on Runtime Verification*. Springer, 2014.
[276] G. W. Clark *et al.* Cybersecurity issues in robotics. In *CogSIMA*, 2017.
[277] A. Akhunzada *et al.* Securing software defined networks: taxonomy, requirements, and open issues. *IEEE Communications Magazine*, 2015.
[278] S. Rivera *et al.* Ros-defender: Sdn-based security policy enforcement for robotic applications. In *Security and Privacy Workshops*. IEEE, 2019.
[279] J. N. K. Liu *et al.* iBotGuard: an internet-based intelligent robot security system using invariant face recognition against intruder. *IEEE Transactions on Systems, Man, and Cybernetics*, 35, 2005.
[280] N. M. Rodday. Exploring security vulnerabilities of unmanned aerial vehicles. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pages 993–994. IEEE, 2016.
[281] J. Wan. Cloud robotics: Current status and open issues. *IEEE Access*, 4:2797–2807, 2016.
[282] J. Miller *et al.* A case study on the cybersecurity of social robots. In *ACM/IEEE HRI*, page 195–196. Association for Computing Machinery, 2018. ISBN 9781450356152.
[283] Py, Frédéric *et al.* Dependable execution control for autonomous robots. In *IEEE/RSJ IROS*, volume 2, pages 1136–1141. IEEE, 2004.
[284] Y. Tang *et al.* Event-based tracking control of mobile robot with denial-of-service attacks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.
[285] S. Tiku *et al.* Overcoming security vulnerabilities in deep learning–based indoor localization frameworks on mobile devices. *ACM Trans. Embed. Comput. Syst.*, 18(6), November 2019. ISSN 1539-9087.
[286] A. H. Rahman *et al.* Evaluation of peer robot communications using cryptoros. *Evaluation*, 10 (7), 2019.
[287] L.V. Legashev *et al.* Monitoring, certification and verification of autonomous robots and intelligent systems: Technical and legal approaches. *Procedia Computer Science*, 150:544 – 551, 2019. ISSN 1877-0509.
[288] A. Yousef *et al.* Analyzing cyber-physical threats on robotic platforms. *Sensors*, 18(5):1643, 2018.
[289] A. Lazanas *et al.* Landmark-based robot navigation. *Algorithmica*, 1995.
[290] C. B. Madsen *et al.* Optimal landmark selection for triangulation of robot position. *Robotics and Autonomous Systems*, 23, 1998.
[291] L. Cheng *et al.* Indoor robot localization based on wireless sensor networks. *IEEE Transactions on Consumer Electronics*, 57(3), 2011. doi: 10.1109/TCE.2011.6018861.
[292] P. Nazemzadeh *et al.* Indoor localization of mobile robots through qr code detection and dead reckoning data fusion. *IEEE/ASME Transactions on Mechatronics*, 22(6), Dec 2017. doi: 10.1109/TMECH.2017.2762598.
[293] A. Zamir *et al.* Accurate image localization based on google maps street view. In *ECCV*. Springer, 2010.
[294] T. Sattler *et al.* Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.
[295] J. Thoma *et al.* Mapping, localization and path planning for image-based navigation using visual features and map. In *IEEE CVPR*, 2019.
[296] Q. Qin *et al.* Sorting system of robot based on vision detection. In *IWAMA Workshop*. Springer, 2017.
[297] M. Gadd *et al.* A framework for infrastructure-free warehouse navigation. In *ICRA*, 2015.
[298] S. Hilsenbeck *et al.* Scale-preserving long-term visual odometry for indoor navigation. In *IPIN*, 2012.
[299] H. Ahmed *et al.* Terrain-based vehicle localization using low cost mems-imu sensors. In *83rd VTC Spring*. IEEE, 2016.
[300] J. Zhang*et al.* Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015.

[301] S. A. S. Mohamed *et al.* A survey on odometry for autonomous navigation systems. *IEEE Access*, 7, 2019.

[302] C. Rizos *et al.* Precise point positioning: is the era of differential gnss positioning drawing to an end? 2012.

[303] S. B. Williams *et al.* Towards multi-vehicle simultaneous localisation and mapping. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 3, May 2002.

[304] N. Michael *et al.* Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Journal of Field Robotics*, volume 92, 07 2012.

[305] B. Kim *et al.* Multiple relative pose graphs for robust cooperative mapping. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010.

[306] H. Lee *et al.* A survey of map merging techniques for cooperative-slam. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Nov 2012.

[307] R. Madhavan *et al.* Distributed cooperative outdoor multirobot localization and mapping. *Autonomous Robots*, 17(1), 2004.

[308] G. Dedeoglu *et al.* *Landmark-based Matching Algorithm for Cooperative Mapping by Autonomous Robots*. Springer Japan, Tokyo, 2000.

[309] N. DeMarinis *et al.* Scanning the internet for ros: A view of security in robotics research. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8514–8521. IEEE, 2019.

[310] R. Amini *et al.* Cryptoros: A secure communication architecture for ros-based applications. *International Journal of Advanced Computer Science and Applications*, 9(10):189–194, 2018.

[311] J. Kim *et al.* Security and performance considerations in ros 2: A balancing act. *arXiv preprint arXiv:1809.09566*, 2018.

[312] B. R. Hilnbrand *et al.* Automated vehicle map localization based on observed geometries of roadways, 2019. US Patent 10,289,115.

[313] H. Sobreira *et al.* Map-matching algorithms for robot self-localization: a comparison between perfect match, iterative closest point and normal distributions transform. *Journal of Intelligent & Robotic Systems*, 2019.

[314] A. S. Huang *et la.* Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research : The 15th International Symposium ISRR*. Springer, 2017.

[315] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611. International Society for Optics and Photonics, 1992.

[316] Martin Lauer, Sascha Lange, and Martin Riedmiller. Calculating the perfect match: an efficient and accurate approach for robot self-localization. In *Robot Soccer World Cup*. Springer, 2005.

[317] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3. IEEE, 2003.

[318] S. A. Rahok *et al.* Odometry correction with localization based on landmarkless magnetic map for navigation system of indoor mobile robot. In *4th ICARA*, 2009.

[319] R. S. Andersen *et al.* Fast calibration of industrial mobile robots to workstations using QR codes. In *IEEE ISR 2013*, Oct 2013. doi: 10.1109/ISR.2013.6695636.

[320] H. Zhang *et al.* Localization and navigation using qr code for mobile robot in indoor environment. In *IEEE ROBIO*, Dec 2015. doi: 10.1109/ROBIO.2015.7419715.

[321] Y. Song *et al.* Uwb/lidar fusion for cooperative range-only slam. In *IEEE ICRA*, 2019.

[322] Anis Koubâa. *Robot Operating System (ROS).* Springer, 2017.

[323] T. Shan *et al.* Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IROS*. IEEE, 2018.

[324] J. Czajkowski *et al.* Quantum preimage, 2nd-preimage, and collision resistance of sha3. *IACR ePrint*, 302, 2017.

[325] P. Sriramya *et al.* Providing password security by salted password hashing using bcrypt algorithm. *ARPN journal of engineering and applied sciences*, 10(13), 2015.

[326] Sahar Salimpour, Farhad Keramat, Jorge Peña Queralta, and Tomi Westerlund. Decentralized vision-based byzantine agent detection in multi-robot systems with iota smart contracts. In *International Symposium on Foundations and Practice of Security (FPS)*. Springer, 2022.

[327] Sahar Salimpour, Jorge Peña Queralta, and Tomi Westerlund. Self-calibrating anomaly and change detection for autonomous inspection robots. In *IEEE Robotic Computing*. IEEE, 2022.

[328] Paavo Nevalainen, Parisa Movahedi, Jorge Peña Queralta, Tomi Westerlund, and Jukka Heikkonen. Long-term autonomy in forest environment using self-corrective slam. *Book Chapter*, 2020. doi: https://doi.org/10.1007/978-3-030-77860-6_5. Presented at FinDrones 2020.

[329] Li Qingqing, Paavo Nevalainen, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Localization in unstructured environments: Towards autonomous robots in forests with delaunay triangulation. *Remote Sensing*, 2020. doi: doi:10.3390/rs12111870.

[330] Li Qingqing, Yu Xianjia, Jorge Peña Queralta, and Tomi Westerlund. Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[331] Yu Xianjia, Li Qingqing, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Applications of UWB networks and positioning to autonomous robots and industrial systems. In *Cyber-Physical Systems of Systems (CPSoS) and Internet of Things (IoT) Conference*. IEEE, 2021.

[332] W. Giernacki *et al.* Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *MMAR*, 2017.

[333] Cirque du Soleil. Zurich, and verity studios. sparked: A live interaction between humans and quadcopters. *ETH*, 2014.

[334] Pozyx Labs. Pozyx accurate positioning. 2018.

[335] M. Contigiani *et al.* Implementation of a tracking system based on uwb technology in a retail environment. In *IEEE/ASME MESA*, 2016.

[336] Robert J Fontana. Ultra wideband precision geolocation system, April 25 2000. US Patent 6,054,950.

[337] Zafer Sahinoglu. *Ultra-wideband positioning systems*. Cambridge university press, 2008.

[338] James D Taylor. *Ultra-wideband radar technology*. CRC press, 2018.

[339] R. Mazraani *et al.* Experimental results of a combined tdoa/tof technique for uwb based localization systems. In *ICC Workshops*. IEEE, 2017.

[340] C. Wang *et al.* Ultra-wideband aided fast localization and mapping system. In *IEEE/RSJ IROS*. IEEE, 2017.

[341] F. J. Perez-Grau *et al.* Multi-modal mapping and localization of unmanned aerial robots based on ultra-wideband and rgb-d sensing. In *IEEE/RSJ IROS*. IEEE, 2017.

[342] V. Magnago *et al.* Robot localization via odometry-assisted ultra-wideband ranging with stochastic guarantees. In *IROS*, 2019.

[343] T. M. Nguyen *et al.* Integrated uwb-vision approach for autonomous docking of uavs in gps-denied environments. In *ICRA*. IEEE, 2019.

[344] N. Macoir *et al.* Uwb localization with battery-powered wireless backbone for drone-based inventory management. *Sensors*, 19, 2019.

[345] Paola Torrico Morón, Jorge Peña Queralta, and Tomi Westerlund. Towards large-scale relative localization in multi-robot systems with dynamic UWB role allocation. *arXiv preprint*, 2022.

[346] Janis Tiemann, Andrew Ramsey, and Christian Wietfeld. Enhanced uav indoor navigation through slam-augmented uwb localization. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018.

[347] T. M. Nguyen *et al.* An integrated localization-navigation scheme for distance-based docking of uavs. In *IEEE/RSJ IROS*, 2018.

[348] K. K. Oh *et al.* A survey of multi-agent formation control. *Automatica*, 53, 2015.

[349] 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*, 2019.

[350] C. Shuqiang *et al.* Uwb based integrated communication and positioning system for multi-uavs close formation. In *MECAE*. Atlantis, 2018.

[351] Damoda News. The 14th moscow uav air show, 2019.

[352] T. M. Nguyen *et al.* Robust target-relative localization with ultra-wideband ranging and communication. In *ICRA*. IEEE, 2018.

[353] L. Qiang *et al.* Formation control of multi robot based on uwb distance measurement. In *CCDC*. IEEE, 2018.

[354] K. Guo *et al.* Ultra-wideband and odometry-based cooperative relative localization with application to multi-uav formation control. *IEEE transactions on cybernetics*, 2019.

[355] R. Liu *et al.* Cooperative relative positioning of mobile users by fusing imu inertial and uwb ranging information. In *ICRA*. IEEE, 2017.

[356] T. Fan *et al.* Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *arXiv preprint arXiv:1808.03841*, 2018.

[357] L. Qiang *et al.* Design and implementation of multi robot research platform based on uwb. In *29th CCDC*. IEEE, 2017.

[358] H. Wei *et al.* Consensus algorithms based multi-robot formation control under noise and time delay conditions. *Applied Sciences*, 9(5), 2019.

[359] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8776–8782. IEEE, 2020.

[360] Hao Xu, Yichen Zhang, Boyu Zhou, Luqi Wang, Xinjie Yao, Guotao Meng, and Shaojie Shen. Omni-swarm: A decentralized omnidirectional visual–inertial–uwb state estimation system for aerial swarms. *IEEE Transactions on Robotics*, 2022.

[361] Enrica Soria. Swarms of flying robots in unknown environments. *Science Robotics*, 7(66): eabq2215, 2022.

[362] J. S. Furtado. Comparative analysis of optitrack motion capture systems. In *Lecture Notes in Mechanical Engineering*. Springer, 2019.

[363] U. Raza *et al.* Dataset: Indoor localization with narrow-band, ultra-wideband, and motion capture systems. In *2nd DATA Workshop*, 2019.

[364] V. Barral *et al.* Multi-sensor accurate forklift location and tracking simulation in industrial indoor environments. *Electronics*, 8, 2019.

[365] J. Li *et al.* Accurate 3d localization for mav swarms by uwb and imu fusion. *arXiv preprint arXiv:1807.10913*, 2018.

[366] K. M. Mimoune *et al.* Evaluation and improvement of localization algorithms based on uwb pozyx system. In *SoftCOM*. IEEE, 2019.

[367] K. Bregar *et al.* Nlos channel detection with multilayer perceptron in low-rate personal area networks for indoor localization accuracy improvement. In *8th Jožef Stefan Int. Conf.*, volume 31, 2016.

[368] K. Bregar *et al.* Improving indoor localization using convolutional neural networks on computationally restricted devices. *IEEE Access*, 2018.

[369] Tuan Li, Hongping Zhang, Zhouzheng Gao, Qijin Chen, and Xiaoji Niu. High-accuracy positioning in urban environments using single-frequency multi-gnss rtk/mems-imu integration. *Remote sensing*, 10(2):205, 2018.

[370] Jae-One Lee and Sang-Min Sung. Assessment of positioning accuracy of uav photogrammetry based on rtk-gps. *Journal of the Korea Academia-Industrial cooperation Society*, 19(4):63–68, 2018.

[371] Kiyoung Kim, Jaemook Choi, Junyeon Chung, Gunhee Koo, In-Hwan Bae, and Hoon Sohn. Structural displacement estimation through multi-rate fusion of accelerometer and rtk-gps displacement and velocity measurements. *Measurement*, 130:223–235, 2018.

[372] Bernhard Groβwindhager, Michael Stocker, Michael Rath, Carlo Alberto Boano, and Kay Römer. Snaploc: An ultra-fast uwb-based indoor localization system for an unlimited num-

ber of tags. In *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 61–72. IEEE, 2019.

[373] Li Qingqing, Yu Xianjia, Jorge Peña Queralta, and Tomi Westerlund. Adaptive lidar scan frame integration: Tracking known MAVs in 3D point clouds. In *20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021. doi: 10.1109/ICAR53236.2021.9659483.

[374] Paola Torrico Morón, Salma Salimi, Jorge Peña Queralta, and Tomi Westerlund. Uwb role allocation with distributed ledger technologies for scalable relative localization in multi-robot systems. *arXiv preprint*, 2022.

[375] Y. Q. Chen and Z Wang. Formation control: a review and a new consideration. In *IEEE/RSJ IROS*, 2005.

[376] Wei Ren and Yongcan Cao. *Distributed coordination of multi-agent networks: emergent problems, models, and issues*. Springer Science & Business Media, 2010.

[377] B. D. O. Anderson *et al.* Rigid graph control architectures for autonomous formations. *IEEE Control Systems Magazine*, 2008. ISSN 1066-033X. doi: 10.1109/MCS.2008.929280.

[378] R. D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9 (4), Oct 2012. ISSN 1545-5955. doi: 10.1109/TASE.2012.2214676.

[379] S. Bandyopadhyay *et al.* Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, 33(5):1103–1123, Oct 2017. ISSN 1552-3098.

[380] M Basiri *et al.* Distributed control of triangular formations with angle-only constraints. *Systems & Control Letters*, 59(2):147 – 154, 2010. ISSN 0167-6911.

[381] A. N. Bishop. A very relaxed control law for bearing-only triangular formation control. *IFAC Proceedings Volumes*, 44(1):5991 – 5998, 2011. ISSN 1474-6670.

[382] A. N. Bishop *et al.* Bearing-only triangular formation control on the plane and the sphere. In *2010 18th MED*, June 2010.

[383] S. L. Smith *et al.* Stabilizing a multi-agent system to an equilateral polygon formation. In *Proc. of the 17th International Symposium on Mathematical Theory of Networks and Systems*, pages 2415–2424, 2006.

[384] D. Morgan *et al.* Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, 2016.

[385] K. K. Oh *et al.* Formation control of mobile agents based on inter-agent distance dynamics. *Automatica*, 47(10), 2011. ISSN 0005-1098.

[386] Sherif A.S. Mohamed *et al.* A survey on odometry for autonomous navigation systems. *IEEE Access*, 2019.

[387] G. Vásárhelyi *et al.* Outdoor flocking and formation flight with autonomous aerial robots. In *IEEE/RSJ IROS*, 2014. doi: 10.1109/IROS.2014.6943105.

[388] S. A. Barogh and H. Werner. Cascaded formation control using angle and distance between agents with orientation control (part 1 and part 2). In *2016 UKACC 11th International Conference on Control (CONTROL)*, pages 1–6, Aug 2016.

[389] M. C. Park, Z. Sun, M. H. Trinh, B. D. O. Anderson, and H. S. Ahn. Distance-based control of k4 formation with almost global convergence. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 904–909, Dec 2016.

[390] S. Kloder *et al.* A configuration space for permutation-invariant multi-robot formations. In *2004 ICRA*, volume 3, April 2004.

[391] N. P. Hyun *et al.* Collision free and permutation invariant formation control using the root locus principle. In *2016 ACC*, July 2016.

[392] M. M. Zavlanos *et al.* Distributed formation control with permutation symmetries. In *2007 46th IEEE CDC*, Dec 2007.

[393] P. Kingston *et al.* Index-free multi-agent systems: An eulerian approach. *IFAC Proceedings Volumes*, 43(19):215 – 220, 2010. ISSN 1474-6670.

[394] N. Shiell *et al.* A bearing-only pattern formation algorithm for swarm robotics. In *Swarm Intelligence*. Springer International Publishing, 2016. ISBN 978-3-319-44427-7.

[395] M. Hoeing *et al.* Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th AAMAS*, pages 280:1–280:8, 2007. ISBN 978-81-904262-7-5. doi: 10.1145/1329125.1329462.

[396] C. Pinciroli *et al.* Decentralized progressive shape formation with robot swarms. In *DARS*, pages 433–445. Springer, 2018.

[397] Guannan Li *et al.* Decentralized progressive shape formation with robot swarms. *Autonomous Robots*, Oct 2018. ISSN 1573-7527. doi: 10.1007/s10514-018-9807-5.

[398] Kasper Støy. Using situated communication in distributed autonomous mobile robotics. In *Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, SCAI '01, pages 44–52. IOS Press, 2001. ISBN 1-58603-161-9.

[399] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 1996. ISSN 1432-0444. doi: 10.1007/BF02712873.

[400] S. A. Barogh and H. Werner. Cascaded formation control using angle and distance between agents with orientation control (part 1 and part 2). In *UKACC 11th International Conference on Control*, Aug 2016.

[401] J. Ghommam *et al.* Cascade design for formation control of nonholonomic systems in chained form. *Journal of the Franklin Institute*, 348(6):973 – 998, 2011. ISSN 0016-0032. doi: https://doi.org/10.1016/j.jfranklin.2011.03.008.

[402] L. Consolini *et al.* Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44, 2008. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2007.09.019.

[403] S. Ahmadi Barogh *et al.* Formation control of non-holonomic agents with collision avoidance. In *American Control Conference*, 2015. doi: 10.1109/ACC.2015.7170825.

[404] Salma Salimi, Paola Torrico Morón, Jorge Peña Queralta, and Tomi Westerlund. Secure heterogeneous multi-robot collaboration and docking with hyperledger fabric blockchain. *arXiv preprint*, 2022.

[405] Kailai Li, Meng Li, and Uwe D Hanebeck. Towards high-performance solid-state-lidar-inertial odometry and mapping. *arXiv preprint arXiv:2010.13150*, 2020.

[406] Jiarong Lin and Fu Zhang. Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3126–3131. IEEE, 2020.

[407] A. S. Huang *et al.* Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*. Springer, 2017.

[408] Evan Ackerman. Skydio's new drone is smaller, even smarter, and (almost) affordable. *IEEE Spectrum*, 2019.

[409] Daniel Zelazo, Antonio Franchi, Frank Allgöwer, Heinrich H Bülthoff, and P Robuffo Giordano. Rigidity maintenance control for multi-robot systems. In *Robotics: science and systems*, pages 473–480, 2012.

[410] Fabrizio Schiano, Antonio Franchi, Daniel Zelazo, and Paolo Robuffo Giordano. A rigidity-based decentralized bearing formation controller for groups of quadrotor UAVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5099–5106. IEEE, 2016.

[411] Fabrizio Schiano and Paolo Robuffo Giordano. Bearing rigidity maintenance for formations of quadrotor UAVs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1467–1474. IEEE, 2017.

[412] Fabrizio Schiano and Roberto Tron. The dynamic bearing observability matrix nonlinear observability and estimation for multi-agent systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3669–3676. IEEE, 2018.

[413] Patrik Schmuck and Margarita Chli. Multi-uav collaborative monocular slam. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3863–3870. IEEE, 2017.

[414] Nesrine Mahdoui, Vincent Frémont, and Enrico Natalizio. Communicating multi-uav system for cooperative slam-based exploration. *Journal of Intelligent & Robotic Systems*, pages 1–19, 2019.

[415] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, 33(4):1–12, 2014.

[416] Siyan Dong, Kai Xu, Qiang Zhou, Andrea Tagliasacchi, Shiqing Xin, Matthias Nießner, and Baoquan Chen. Multi-robot collaborative dense scene reconstruction. *ACM Transactions on Graphics (TOG)*, 38(4):1–16, 2019.

[417] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with dec-mcts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9101–9107. IEEE, 2019.

[418] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[419] Shengkang Chen and Ankur Mehta. Colo: A performance evaluation system for multi-robot cooperative localization algorithms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1458–1464. IEEE, 2019.

[420] Jason T Isaacs, Daniel J Klein, and Joao P Hespanha. Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle. In *American Control Conference*, pages 1704–1709. IEEE, 2011.

[421] Jason T Isaacs and Joao P Hespanha. Dubins traveling salesman problem with neighborhoods: A graph-based approach. *Algorithms*, 6(1):84–99, 2013.

[422] Enrica Soria, Fabrizio Schiano, and Dario Floreano. The influence of limited visual sensing on the reynolds flocking algorithm. In *IEEE International Conference on Robotic Computing (IRC)*, pages 138–145. IEEE, 2019.

[423] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011.

[424] Jeffrey Ichnowski, Kaiyuan Chen, Karthik Dharmarajan, Simeon Adebola, Michael Danielczuk, Vıctor Mayoral-Vilches, Hugo Zhan, Derek Xu, Ramtin Ghassemi, John Kubiatowicz, et al. Fogros 2: An adaptive and extensible platform for cloud and fog robotics using ros 2. *arXiv preprint arXiv:2205.09778*, 2022.

[425] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Thien Hoang Nguyen, and Lihua Xie. Viral slam: Tightly coupled camera-imu-uwb-lidar slam. *arXiv preprint arXiv:2105.03296*, 2021.

[426] Amanda Prorok, Matthew Malencia, Luca Carlone, Gaurav S Sukhatme, Brian M Sadler, and Vijay Kumar. Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems. *arXiv preprint arXiv:2109.12343*, 2021.