Distributed Situational Awareness on a Heterogeneous Multi-Robot System

Department of Electrical and Information Engineering

Electrical and Information Engineering
Ph.D. Program

SSD: ING-INF/03 – Telecommunications

Final Dissertation

Distributed Situational Awareness on a Heterogeneous Multi-Robot System

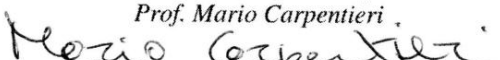
---

by

Matteo Palieri

Referees	Supervisors
Prof. Christoffer Heckman	Prof. Cataldo Guaragnella
Prof. Hsueh-Cheng Wang	Dr. Ali-akbar Agha-mohammadi

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri



Course n. 34, 01/11/2018 – 31/10/2021

Abstract

Multi-robot systems have gained substantial attention in the last decades for their potential in autonomously retrieving and providing situational awareness in environments that might be too far or dangerous for humans to approach. Examples include industrial monitoring, search and rescue, and planetary exploration. To accomplish the mission objective in such scenarios, heterogeneous multi-robot systems are usually employed due to their complementary exploration capabilities (e.g. terrestrial, aerial).

The overall reliability of the situational awareness retrieved by the robot team strongly depends on the accuracy of on-board perception systems, such as localization and semantic understanding modules. In heterogeneous multi-robot systems agents might have different mobility, sensor suites and computational capabilities. For this reason, accuracy of perception systems is greatly challenged by potentially constrained on-board computational resources.

In the localization domain, lidar odometry has gathered considerable attention during the last decade as a robust localization method for extreme terrains, while for semantic understanding, neural networks based modules represent the current state-of-the-art. While great progress has been achieved in these fields, their computational cost is still prohibitive for limited on-board computers of less capable robots, and actions need to be taken to not compromise the overall precision of the situational awareness retrieved by the robot team.

This work is motivated by the aim of enabling robust and accurate situational awareness retrieval on a heterogeneous multi-robot system for extreme operations in GPS-denied and perceptually-degraded environments under severe computation and communication constraints.

We present systems that can contribute in pushing the state-of-the-art boundaries in en-

hancing the overall accuracy of the situational awareness retrieved by an heterogeneous multi-robot system.

First, to provide a reliable ego-motion estimation method for robotic exploration of GPS-denied and perceptually-degraded environments, we present a high-precision lidar odometry system that achieves robust and real-time operation under challenging perceptual conditions. LOCUS (Lidar Odometry for Consistent operation in Uncertain Settings), provides an accurate multi-stage scan matching unit equipped with an health-aware sensor integration module for seamless fusion of additional sensing modalities.

Then, to enhance perception accuracy of computationally constrained platforms in heterogeneous multi-robot systems, we introduce Swarm Manager. The presented approach exploits Distributed Computation and Software Defined Networking paradigms allowing robots to offload heavy computation (e.g. lidar odometry, object detection) to other more resourceful peers in the team under decision of a globally-aware central orchestrator. For each offloading request, Swarm Manager simultaneously identifies the optimum server where is best to execute the task, and the optimum path through is best to route the data given the current system state. The presented approach provides resilience to the failure of the central orchestrator by means of a dynamic leader election mechanism.

We evaluate the performance of LOCUS against state-of-the-art techniques in perceptually challenging environments, and demonstrate top-class localization accuracy along with substantial improvements in robustness to sensor failures. We then demonstrate real-time performance of LOCUS on various types of robotic mobility platforms involved in the autonomous exploration of the Satsop power plant in Elma, WA where the proposed system was a key element of the CoSTAR team's solution that won first place in the Urban Circuit of the DARPA Subterranean Challenge.

Finally, we demonstrate enhancements achievable with Swarm Manager in the overall accuracy of the situational awareness retrieved by the heterogeneous multi-robot system for a multi-level and communication-constrained exploration of a dismissed power plant by a team of four autonomous robots. We demonstrate enhanced localization accuracy, improved semantic detection precision, and increased autonomy time.

Acknowledgements

I deeply thank and express all my gratitude to:

- Dr. Cataldo Guaragnella, Polytechnic University of Bari
- Dr. Ali-akbar Agha-mohammadi, NASA Jet Propulsion Laboratory
- Dr. Benjamin Morrell, NASA Jet Propulsion Laboratory
- Dr. Jeffrey A. Edlund, NASA Jet Propulsion Laboratory
- Dr. Luca Carlone, Massachusetts Institute of Technology
- Dr. Giorgio Parladori, SM-Optics
- Dr. Marco Mussini, SM-Optics

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation aims at enhancing situational awareness accuracy of an heterogeneous multi-robot system involved in the autonomous exploration of GPS-denied and perceptually-degraded environments under severe computation, communication, and energy constraints.

We introduce the concept of situational awareness on a multi-robot system in Section 1.1 and outline potential applications in Section 1.2. We discuss open challenges for robotic operation in extreme environments in Section 1.3, and refer to the DARPA Subterranean Challenge (SubT) in Section 1.4 as a remarkable opportunity to push boundaries of autonomous robotic exploration in extreme settings. We describe the goal of the work in Section 1.5 and outline content of the dissertation in Section 1.6.

1.1 Multi-Robot Systems

Situational Awareness Situational awareness [1] is the perception of environmental elements and events in relation to time and space, the comprehension of their meaning, and the projection of their future status. Situational awareness represents a vital, yet often elusive, foundation for both manned and unmanned systems to enable effective decision-making in a wide variety of situations.

Multi-Robot System A multi-robot system [2] is a system consisting of multiple robots that might interact with each other to accomplish a common mission-level objective. Research on multi-robot systems began in 1980: up to that point in time

research was mainly focused on single robot solutions. [3] provide an extensive overview of ongoing research efforts and advancements in the field of distributed mobile robot systems.

Situational Awareness on a Multi-Robot System Multi-robot systems offer enhanced efficiency in performing spatially-distributed tasks in time-constrained operations, and are therefore particularly suited to retrieve distributed situational awareness by complex data gathering and mapping operations in large-scale environments. For these reasons, multi-robot systems have gained substantial attention in the last decades for their potential in autonomously retrieving and providing situational awareness in environments that might be too far or dangerous for humans to approach.

1.2 Applications

Rapid advancements in robotic research are enabling a wide range of applications in increasingly complex environments ranging from industrial monitoring [4, 5] to search and rescue [6, 7] and planetary exploration missions [8, 9, 10, 11].

Industrial Monitoring In industrial monitoring applications, a team of robot can be deployed to provide situational awareness in hazardous industrial facilities after a nuclear disaster [12] or to autonomously explore and inspect the industrial site to provide distributed and real-time situational awareness updates to a remote human supervisor to ease the task of monitoring and maintenance of the infrastructure of interest.

Search And Rescue In search and rescue operations after natural disasters (e.g. post-earthquake scenarios, collapsed mines, etc.), an autonomous team of robots might provide situational awareness updates with locations of found survivors, and enable a remote operator to schedule a rescue plan accordingly. An outstanding example is represented by the Tham Luang cave rescue (Figure 1.1), in which the international community aimed at rescuing thirteen members of a football team trapped inside a 4 km partially flooded cave. A team of drones equipped with thermal cameras and an

underwater robot were used to provide information on the condition of the environment and the depth of the water to a remote base station: however, at that time, no technology was available to autonomously reach the people, map the cave, and scan for potential survivors in the deep underground.

Figure 1.1: Tham Luang cave rescue mission. Image adapted from [13]

Planetary Exploration The research community identified more than 200 Lunar and 2000 Martian cave-related features [14, 15]. Characterized by stable temperature profiles and sheltered from cosmic radiation, caves and sub-surface voids constitute both an ideal candidate for development of microbial life, as well as a potential habitat for future human space missions [16, 17, 8]. The distributed situational awareness collected by a team of robots could therefore provide scientists with extremely precious information to understand where to search for life, or where to potentially build future shelters for humans [18, 19]. Moreover, autonomous multi-robot systems might be sent on reconnaissance tours to perform rapid search and mapping missions to support follow-on operations of advance service personnel such as astronauts.

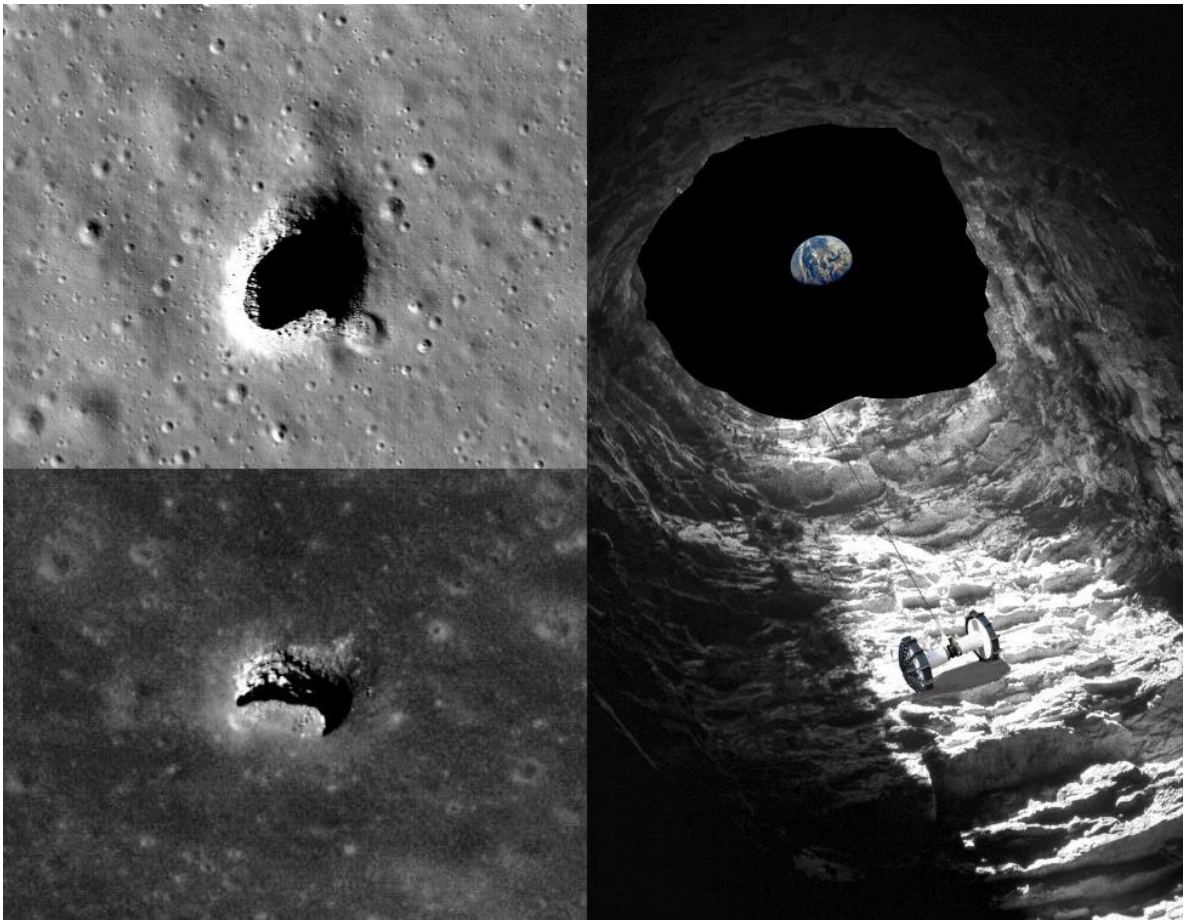


Figure 1.2: Example of robotic sub-surface void explorations of lunar caves. Image adapted from [20]

Situational Awareness Accuracy In all these applications, the accuracy of the situational awareness retrieved by the robot team is of fundamental importance to provide a reliable understanding model of the environment and enable decision making at any level. For example, in search and rescue operations, accuracy of localization reports of found survivors would be a critical key-requirement for a successful rescue plan as the injured people may have very limited mobility.

1.3 Challenges

Despite the outstanding growth of robotic technologies, many challenges have still to be addressed to enable reliable operation and situational awareness retrieval in *extreme environments* like the ones mentioned in the application scenarios described above. In

such environments, robotic operation is challenged by a wide range of factors which are briefly described below.

Hazardous Traversability Environments might be characterized by structured or unstructured geometries, involve multiple floors, and present traversability-challenging elements such as constrained passages or sharp turns. Different types of terrain might be encountered during the exploration with elements including rocks, mud, sand and water. All these factors represent great challenges for the robot's mobility capabilities.

Figure 1.3: Example of hazardous traversability in extreme terrains. On the left, a wheeled platform involved in autonomous subterranean exploration. On the right, a tracked vehicle stuck on a obstacle within hazardous traversal of train rails.

Degraded Perception and Sensing Environments might often be GPS-denied, leading to the need of robust and accurate localization systems on-board. In these settings, perception systems have to operate in perceptually-degraded conditions including darkness, sudden changes in illumination, presence of obscurants (e.g. fog, dust, smoke), lack of prominent perceptual features in texture-less and geometrically self-similar areas, along with jerky sensory motion induced by navigation over rough terrains or high-rate motions in agile aerial agents. All these features pose a significant threat to the accuracy of on-board perception systems and could lead to significant localization errors over the duration of an extended run. Furthermore, on-board sensors can fail at any time without notice in such high-risk missions, making therefore ro-

bustness and resilience to potential failures of sensors a fundamental requirement for reliable operation.

Constrained Communication Extreme subterranean environments are usually carachterized by the lack of a previously available communication infrastructure and pose numerous impediments to reliable networking including limited opportunities for line-of-sight communications and challenges in radio frequency propagation in varying geology profiles.

Heterogeneity To enable such applications, heterogeneous multi-robot systems are usually employed to accomplish the mission objective due to their complementary exploration capabilities (e.g. terrestrial, aerial). Wheeled ground robots can be deployed to cover floor-accessible regions, legged platforms might be suitable to traverse rough terrains or conduct multi-level explorations, while aerial agents could instead access vertical shafts, or in general areas that are not accessible with ground-robots.

While the heterogeneity of the multi-robot system might be a *necessity* to accomplish the mission-objective, this introduces further challenges on the overall reliability of the situational awareness retrieved by the multi-robot system as agents might have different mobility, sensor suites, and computational capabilities: while ground robots might be able to carry large batteries and computational resources, legged and aerial agents are usually able to carry substantially smaller batteries and computers.

Despite the tremendous advancements in localization and semantic understanding domains, these on-board perception systems working on large-data volumes still require computational expenses that result prohibitive for computationally-constrained platforms: this can greatly degrade the accuracy of the situational awareness provided by less capable agents forced to trade-off accuracy to pursue real-time operation, and might have catastrophic impacts on the overall decision making process at higher levels.

Conclusions Overall, further work is needed to push the state-of-the-art in robotics to enable systems that can robustly and consistently address the above mentioned challenges for accurate situational awareness retrieval in extreme settings.

1.4 DARPA Subterranean Challenge (SubT)

An outstanding example that summarizes all the challenges encountered when deploying multi-robot systems in extreme settings is the DARPA Subterranean Challenge (SubT) [21]. The SubT Challenge is an international robotic competition that aims to foster technological advancements for autonomous robotic operations in extreme and underground environments which notoriously pose significant complications for manned and unmanned operations due to limited situational awareness. This competition spans over a period of three years and encourages research teams from all around the world to develop innovative solutions to rapidly navigate, map, search, and exploit complex underground environments where is either impossible or too risky to send human personnel in, including human-made tunnels, urban undergrounds, and natural cave networks.

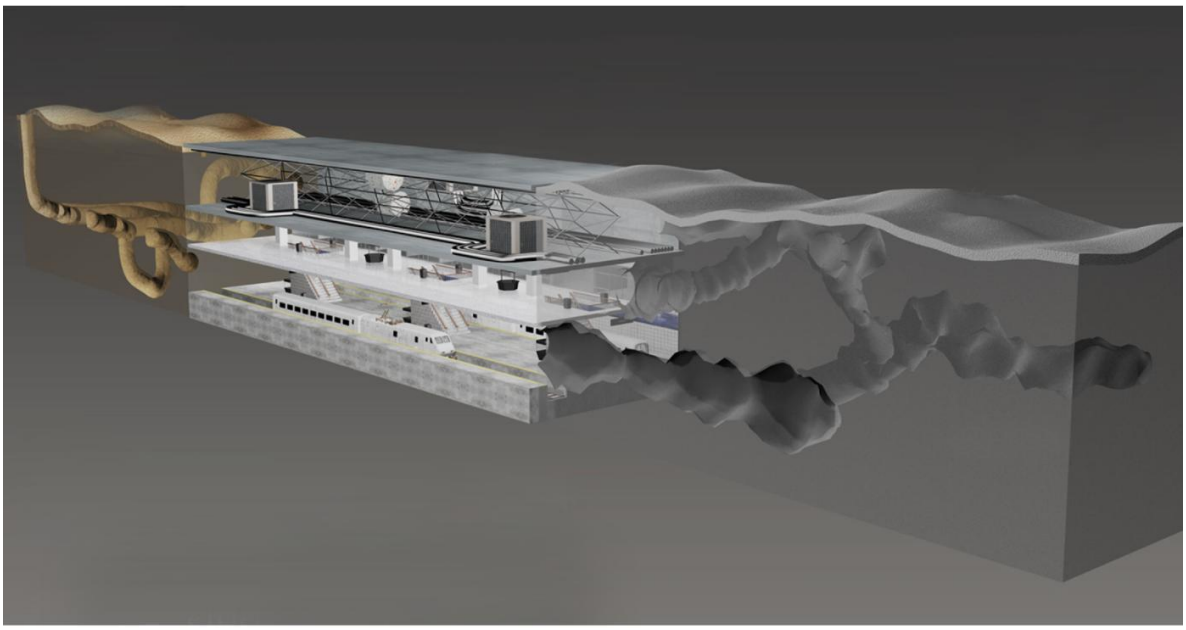


Figure 1.4: The three subdomains of the DARPA Subterranean Challenge: tunnel systems, urban underground, and natural cave networks. Image taken from [22]

A successful development of this technology could enable a wide range of high-impact applications, including the ones outlined in Section 1.2. The SubT Challenge therefore provides us with the unprecedented opportunity of focusing this dissertation on a *real* world problem.

Structure Teams involved in the challenge deploy their robots to provide rapid situational awareness through mapping of the unknown GPS-denied environment and localization of specific objects of interest which are referred to as *artifacts* in this dissertation. Artifacts of interest include survivors, evidence of survivors (e.g. backpack, rope, cellphone), CO_2 gas source, electrical boxes and more. Each team dispose of a fixed 1 hour time window to successfully accomplish the mission: as the robotic team explores the unknown environment, situational awareness updates need to be rapidly communicated back to a remote base station, usually placed at the entrance of the course outside the challenging area. In the competition, there is no prior map of the environment, no team member is allowed to inspect the course prior or during the competition, and only a single human supervisor outside the course is allowed to monitor the data communicated by the robots and interact with them for high-level decision making purposes when a communication link is established (e.g. initiating an autonomous stair-climbing operation for a legged-platform that detected the presence of a stair-case).

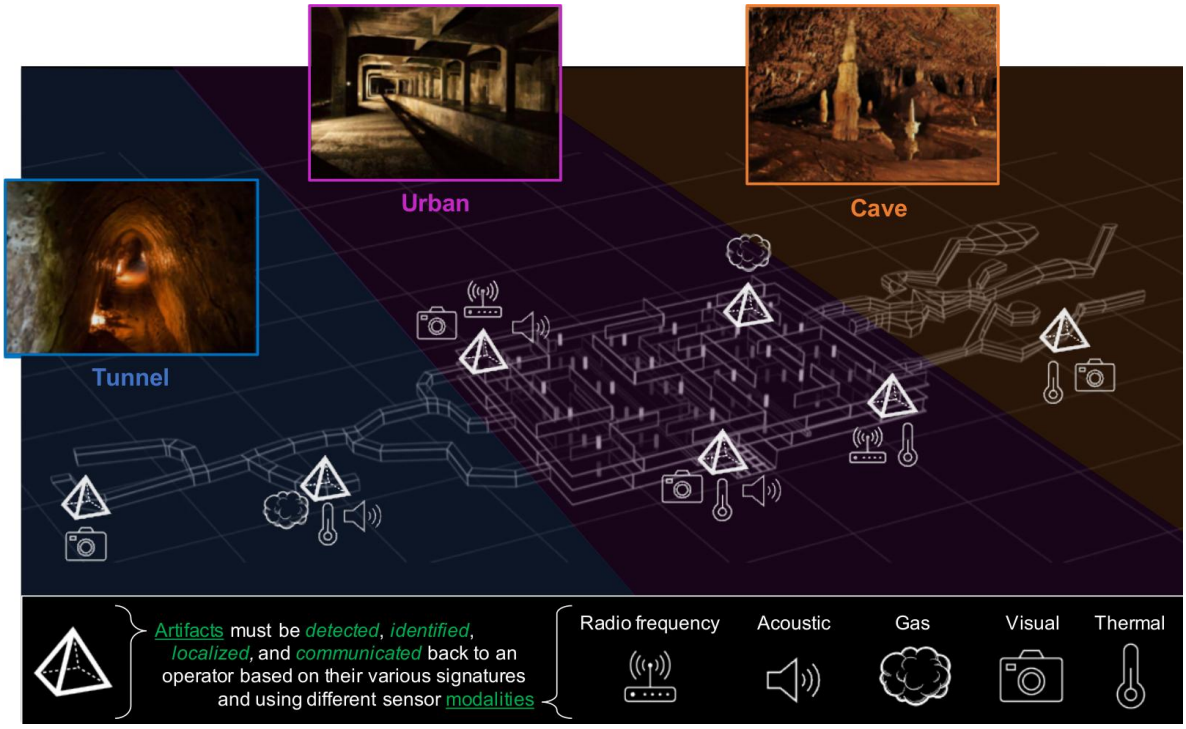


Figure 1.5: The final event of the DARPA Subterranean Challenge aggregates elements from all subdomains into a single course to test the performance and versatility of the developed autonomy solutions. Robotic teams in the challenge score points by reporting type and location of artifacts found during the autonomous exploration. Image taken from [23]

Scoring To score a point in the competition, the robotic team needs to be able to: *i)* reach, detect and recognize a given artifact, *ii)* localize the artifact in global coordinates with less than 5 meters error, and *iii)* successfully communicate the retrieved situational awareness update to the remote base station for artifact submission purposes to the DARPA Server before the end of the mission time.

NeBula The SubT Challenge offers a great opportunity to test capabilities of the broader NASA's NeBula framework for robotic autonomy in real world representative scenarios. NeBula (Networked Belief-aware Perceptual Autonomy) is an uncertainty-aware architecture for resilient and modular robotic autonomy in extreme environments which aims at providing a technological foundation for future robotic exploration missions of outer planets, such as the Moon or Mars, to look for the answer to one of humanity's oldest question: *are we alone in this universe?* NeBula represents the current state-of-the-art solution for autonomous robotic exploration of extreme environments, and in the context of the SubT Challenge exploits an heterogeneous multi-robot system along with perception and uncertainty aware paradigms to rapidly map unknown environments, locate artifacts within it, and provide rapid situational awareness updates via a dynamically extended wireless mesh network to a remote human operator.

Figure 1.6: NeBula-powered robots. Image taken from [23]

Table 1.1: Heterogeneous NeBula-powered Mobility Modes. Table taken from [23].

Robot Type	Deployed In	Energy capacity	Payload capacity	Comm	Speed	Mobility	Endurance
Legged robots	Urban	Mid	Mid	Mid	Mid	Mid	Mid
Hybrid (ground/aerial)	STIX	Low	Low	Low	High	Mid-High	Low-Mid
Wheeled	STIX, Mine, Urban	High	High	High	Low	Low	High
Drones	STIX, Mine	Low	Low	Low	High	High	Low
Tracked	STIX	High	Mid	Mid	Low	Low-Mid	Mid-High
Fast small rovers	Mine	Mid	Mid	Low	High	Low	Mid
Aggregated robot team	All events	Shared/Synergistic	Shared/Synergistic	Shared/Synergistic	Aggregated	Aggregated	Aggregated

Table 1.2: Heterogeneous NeBula Sensors. Table taken from [23].

	Exteroceptive						**Non-navigational**			**Proprioceptive**		
Sensors	Lidar	Vision	Radar	Thermal	Sonar	IR Depth	CO2/Gas	Wi-Fi	Sound	Contact/Force	Encoder	IMU
Accuracy	High	Mid	Low	Low	Low	High	Low	Low	Low	Low	Mid	High
Power efficiency	Low	High	High	Mid	Mid	High	High	High	High	High	High	High
Size/weight efficiency	Low	High	High	Low	Mid	High	High	High	Mid	Mid	Mid	High
Range and FOV	Mid	High	Low	High	Low	Low	Low	High	Mid	-	-	-
Dark/fog/smoke/dust	Mid	Low	High	High	Mid	Mid	-	-	-	-	-	-

Table 1.3: NeBula Processors. Table taken from [23].

Processors	**Micro-controllers**	**Snapdragon**	**Intel NUC**	**Nvidia Xavier**	**AMD**
Compute	Low	Low	Mid	High	High
Power consumption	Low	Low	Mid	High	High
Size efficiency	High	High	Mid	Low	Low

NeBula's heterogeneous robot capabilities are summarized in tables 1.1, 1.2, 1.3 from mobility, sensory, and computing perspectives. Typically, robots with a larger payload capacity can accommodate for a larger sensory suite and can carry larger batteries and more powerful computing resources with respect to robots with constrained payloads.

All robots mount on-board the NeBula payload described in [23] which includes the NeBula Sensor Package (NSP), the NeBula Power and Computing Core (NPCC), and the NeBula Comm Deployment System (NCDS). The NSP gathers real-time sensory information from a subset of the following sensors, depending on what is available on the robot: lidars, monocular, stereo, and thermal cameras, IMUs, encoders, contact sensors, ultra high lumen LEDs, radars, gas sensors, UWB and wireless signal detectors. The NPCC provides power to all sensors and computers mounted on-board and comprises two high-power computers for sensing and autonomy: to enhance semantic understanding functionalities, on some robots the NPCC is equipped with an on-board GPU. To construct and expand a backbone network throughout the exploration, ground robots are equipped with NCDS which allows them to carry and deploy radio communication nodes and static assets, which are better described in Section 2.4.1.

We refer the reader to [23] for a complete description of NeBula's system architecture and concept of operations.

1.5 Goal of the work

With the objective of enhancing situational awareness accuracy on heterogeneous multi-robot systems, this dissertation aims at reaching two main research goals: *i)* developing a robust and accurate mapping and positioning system for robotic agents operating in extreme, GPS-denied, and perceptually-degraded environments, *ii)* realizing a distributed computation framework capable of enhancing situational awareness accuracy of computationally-constrained platforms of heterogeneous multi-robot systems operating under severe computation, communication, and energy constraints.

Being the PhD program conducted within the INTENTO project which aims to investigate Software Defined Networking paradigms to enhance performance of distributed systems, in the pursuit of the joint optimization of the INTENTO and NeBula project goals, we adopt the SubT Challenge as a unique test-bed to push forward boundaries of Software Defined Networking research for complex distributed applications in dynamic wireless multi-robot mesh-networks.

1.6 Organization of the work

The dissertation is organized as follows. Chapter 2 provides an overview of background and related work for the different sub-domains of the problem of interest. After a review of the literature, Chapter 3 outlines the open challenges and identifies current gaps in the state-of-the-art of robotics at enabling accurate and distributed situational awareness on a heterogeneous multi-robot system to then state contributions of the work. Chapter 4 describes the system functioning of LOCUS, our first main contribution in achieving robust and accurate ego-motion estimation in perceptually-degraded settings for extreme robotic explorations. Chapter 5 describes instead the conceptual operation and ROS [24] implementation of Swarm Manager, our second main contribution in pursuing enhancement of situational awareness accuracy in heterogeneous multi-robot systems under sever computation and communication constraints. Experimental results of the fully integrated system are presented in Chapter 6. Finally, conclusions and future research directions are discussed in Chapter 7.

Chapter 2

Background and Related Work

This chapter introduces background and related work for the sub-domains involved in the problem under investigation, namely distributed situational awareness on a heterogeneous multi-robot system.

First, we discuss the perception layer with particular focus on localization and object detection aspects, as these represent the core components of the situational awareness retrieved by the agent. Second, we discuss how distributed computation might enhance the accuracy of the overall situational awareness retrieved by the robot team, when the performance of the perception layer is challenged by limited computational capabilities of heterogeneous robots, and how this could be further exploited to extend autonomy time of power-constrained robots. Finally, we discuss multi-robot networking in communication-degraded environments, and introduce the emerging paradigm of Software Defined Networking for its potential to orchestrate network traffic in a globally-aware and centrally-supervised approach to enhance communication performance in a distributed computation framework.

Given the multidisciplinary nature of the thesis and the vast body of literature available in each sub-domain, we provide high-level details for the different sub-topics, and refer the reader to dedicated surveys for an extensive review of the state-of-the-art in each field.

2.1 Localization

In this section, we first introduce the Simultaneous Localization And Mapping (SLAM) problem, categorize the main types of SLAM solutions, and discuss related work for SLAM systems in extreme environments. Then, we focus the attention on lidar odometry modules that are commonly employed in lidar-based SLAM to outline current gaps in accurate and robust ego-motion estimation in perceptually-degraded conditions.

2.1.1 Simultaneous Localization And Mapping (SLAM)

When operating in GPS-denied environments with no availability of a prior-map, robots need to process on-board sensors readings to solve a self-localization problem commonly known as Simultaneous Localization And Mapping (SLAM). Through repeated observations of fixed environmental features, the robot can simultaneously estimate its own movement, construct a map of the unknown environment, and use this map to keep track of its position within it. Solving the SLAM problem enables global localization of the robotic agent rendering an accurate SLAM solution a key requirement for reliable situational awareness updates of autonomous robots operating in extreme environments.

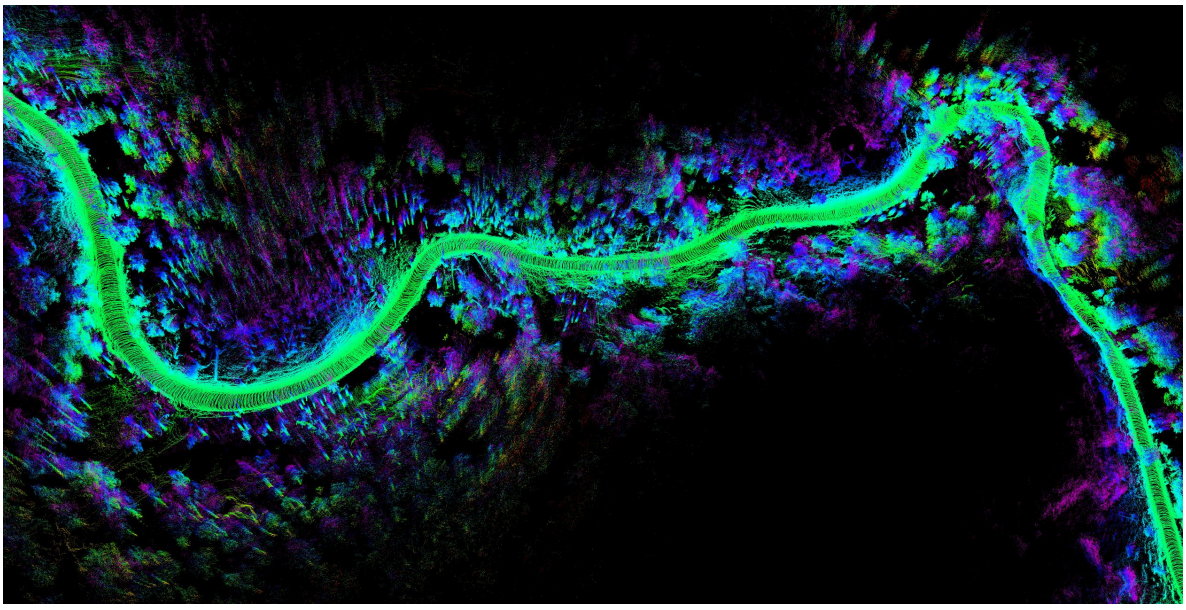


Figure 2.1: Example of a 3D map constructed by a lidar SLAM system: a robot running SLAM on-board exploits self-perception to construct a trajectory and a map that is dynamically updated during the exploration.

SLAM is a well-established research area with numerous applications in robotics and industry [25]. Cadena et al. [26] provide the definition of the SLAM problem, along with a comprehensive overview on modern SLAM algorithms.

Originally addressed by Smith and Cheeseman et al. [27], and Leonard and Durrant-Whyte et al. [28], the SLAM problem finds its current formulation in the work of Lu and Milios [29] where authors study the problem of consistent registration of multiple range scans by modeling the spatial relationships between measurements as random variables and using a maximum likelihood criterion to globally optimize the different spatial relations and reduce the error introduced by constraints.

Different approaches to the SLAM problem have been proposed over the years including: Kalman Filters [30, 31, 32], Particle Filters [33, 34, 35, 36], and Graph-based algorithms [37, 38, 39, 40, 41]. Graph-based algorithms have gained increasing popularity in the last decade for their efficiency in maintaining and processing large-scale maps: these approaches often use factor graphs [42] to model interdependence among variables and formulate SLAM as a maximum a posteriori estimation problem where robot poses and environment landmarks are the nodes in the graph representing the state variables to be optimized, and edges in the graph represent the observation constraints existing between two interconnected variables.

Structure of a SLAM system

The structure of modern graph-based SLAM systems can be divided into two main logical components: a *front-end* and a *back-end*.

Front-End The SLAM front-end process raw sensor data to estimate the robot motion between consecutive measurements: this is commonly referred to as *odometry*, and it represents one of the core components of any SLAM system. By registering observations to the incrementally computed robot poses, a map of the environment can be progressively constructed. However, as each relative pose estimate contains some error, the integrated trajectory can considerably diverge over time from the true path in large-scale explorations.

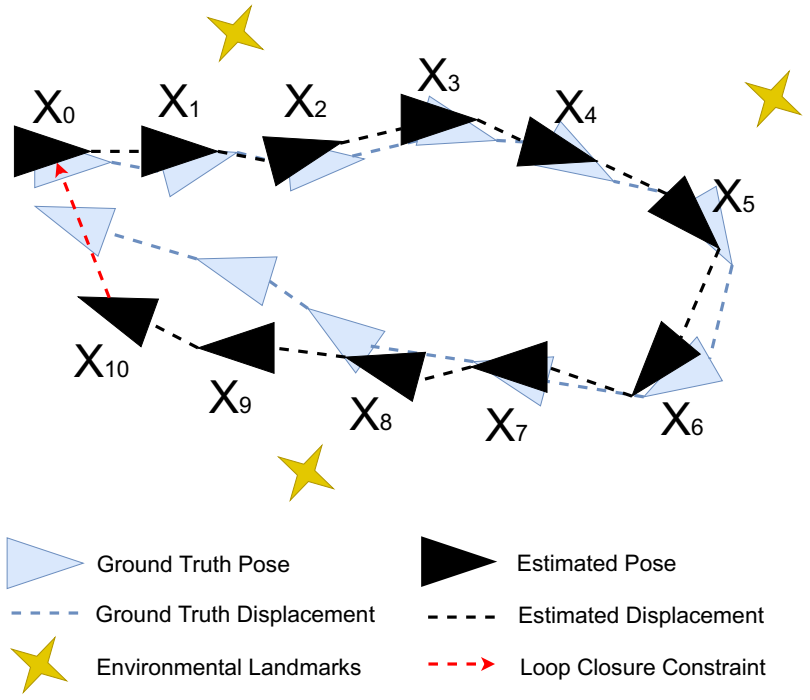


Figure 2.2: Example of a graph-based SLAM system. The front-end computes relative pose estimates between consecutive sensor acquisitions. The back-end periodically instantiates key-nodes in a graph after a minimum odometric displacement is occurred. When loop-closures are detected, a loop closure constraint is applied between different pose nodes of the pose-graph. Pose-graph optimization techniques then reshape the pose-graph and correct for the odometric drift accumulated in the front-end trajectory to achieve global localization consistency.

Back-End The SLAM back-end receives odometry measurements produced by the front-end and periodically instantiates new key-nodes with associated key-measurements in a pose-graph structure after a certain odometric displacement has occurred. It then uses place-recognition techniques to detect loop closures when the robot has returned to a previously visited location. When a loop closure is detected, a new constraint is added between two key-nodes in the graph, and the pose graph is optimized accordingly using iterative nonlinear optimization techniques such as Levenberg-Marquardt [43] to get the updated best estimate of robot poses: this reduces the cumulative error accumulated by the front-end trajectory over the loop, enabling therefore localization with global consistency in the world frame over long term explorations. In multi-robot systems, loop closure detection also enables merging maps obtained by individual robots into a consistent global map of the environment by finding the correspondences between the maps.

Types of SLAM system

[44] provide an extensive survey on different types of SLAM systems. Depending on the main sensor used to solve the SLAM problem, SLAM systems can be grouped into three main categories: *i)* visual, *ii)* lidar, *iii)* hybrid.

Visual SLAM With camera sensors becoming cheaper and increasingly lightweight, the last decade has assisted to the fast rise of visual SLAM [45]. Visual SLAM solutions might rely on different types of cameras as exteroceptive sensor including monocular, stereo, RGB-D, thermal, and event-based. Popular visual SLAM solutions include VINS-Mono [46], Maplab [47], Kimera [48] and ROVIO [30].

For odometry estimation, visual methods [49, 50] might exploit feature detection and tracking techniques [51, 52, 53] or optical flow based approaches over sequential camera frames to estimate the robot ego-motion. In monocular systems, the sensor motion can be recovered only up to a scale ambiguity [54], therefore, to address the problem of absolute scale recovery, visual information is usually combined with inertial measurements from an Inertial Measurement Unit (IMU) to realize Visual Inertial Odometry (VIO) approaches [55, 56, 39, 57, 58] which currently represent a popular solution for a wide range of robotic applications. Combining vision with inertial information not only recovers the absolute scale of monocular visual odometry systems, but also enhance the performance of the ego-motion estimation during abrupt motions or image blur scenarios [59]. Methods using multiple cameras (e.g. stereo) are also capable of scale recovery when the extrinsic calibration between sensors is known, but suffer from tedious calibration processes and might need to be often re-calibrated. For place recognition purposes, visual SLAM systems might usually exploit Bag Of Words approaches [60] or NETVlad descriptors [61].

General challenges for visual-based systems include: *i)* problematic feature tracking in degraded observability conditions (e.g. darkness or sudden changes in illumination, smoke, fog, dust), *ii)* potential ego-motion estimation spikes in abrupt motions or motion blur scenarios, *iii)* lack or ambiguity of information in texture-less or texture-repetitive environments. Moreover, while thermal cameras could be useful in some

conditions [62, 63], they quickly become unsuitable for SLAM purposes in thermally flat environments.

Lidar SLAM Lidar sensors do not rely on external light sources and emit pulsed light waves to estimate range of different points in the environment through time-of-flight based techniques. The active illumination of lidar represents an essential advantage over visual methods as it reduces the sensitivity to ambient lightning variations and provides accurate long-range 3D measurements leading odometry systems based on a single lidar to not suffer from the problem of scale ambiguity. Moreover, lidars usually provide 360° horizontal field of view and high sampling rate. For these reasons, 3D lidar SLAM has become a popular solution for mapping complex environments, from early work [64] to more recent systems [65, 66, 67, 68, 69, 70, 71]. In laser-based SLAM systems, by relying on estimated robot poses and integration of a sequence of 3D scans, 3D maps of the environment can be created.

For odometry estimation, lidar SLAM systems rely on lidar odometry algorithms which typically estimate the ego-motion of the robot by comparing and registering consecutive lidar acquisitions, and are discussed in more depth in the next subsection. For place recognition purposes, lidar SLAM systems can rely on different techniques including PointNetVLAD [72], SegMatch [73], OverlapNet [74], among others [75].

General challenges encountered in pure lidar-based systems include limited motion observability in geometrically-featureless environments (e.g. long corridors or pipes) [76], and spurious loop closure detections in ambiguous and self-similar areas.

Hybrid SLAM Hybrid solutions often combine cameras and lidars due to their complementary nature, along with inertial measurements, to enhance the performance of the overall system. VLOAM [77] present a general framework for combining visual odometry and lidar odometry, while VI-SLAM [78] propose a system that combines an accurate laser odometry estimator, with vision-based place recognition techniques to achieve loop closure detections. [79] provide an abstract and modular architecture to deploy SLAM solutions over a different number of sensors, front-end and back-end solvers.

Related Work

When deployed in perceptually-degraded environments, many SLAM systems result in unsatisfactory performance as sensors must operate in off-nominal conditions, and different factors of the course can stress different sensing modalities in various ways. Darkness, sudden changes in illumination, presence of obscurants (e.g. dust, smoke, fog) and potential visual aliasing phenomena in texture-less or texture-repetitive environments make feature-tracking problematic, decreasing the overall reliability of vision-based odometry and rendering visual centric SLAM approaches unreliable [80, 81, 82, 83]. Geometrically featureless environments such as long corridors without salient features make lidar centric SLAM approaches ambiguous and prone to drift, and the lack of prominent perceptual features in self-similar areas in environments with repetitive appearance can result in inaccurate loop closure detections which can greatly damage the overall mapping result. Moreover, lidar centric SLAM approaches employing a radius-based search for loop closure detection [66] can fail if the front-end odometry drift is excessive. All these challenges contrast sharply with the need to build high-fidelity 3D maps of the environment and support accurate situational awareness updates from the robot team.

More recently [84] introduce a factor-graph based lidar centric SLAM solution with adaptable odometry inputs and multi-modal loop closure detection modules which achieves low drift, multi-robot, multi-sensor SLAM over large scales in perceptually-degraded conditions. [75] exploit 2D semantic and 3D geometric features extracted from lidar data to achieve a pose-invariant and drift-resilient loop closure detection method in perceptually-degraded and self-similar underground environments.

While tremendous progress has been made over the past decades in the field of SLAM, extending these approaches to heterogeneous multi-robot systems remains an open challenge, as the accuracy of SLAM algorithms can be greatly challenged in the case where robotic platforms have to operate under sever computational constraints.

2.1.2 Lidar Odometry

If the robot trajectory does not contain loops, the overall SLAM performance is highly reliant on the accuracy of the front-end odometry, making accurate and robust ego-motion estimation a key-requirement for autonomous robotic operation in extreme environments. Lidar odometry has gathered considerable attention during the last decade as a robust localization method for extreme terrains, leveraging the high-fidelity long range 3D measurements from lidar sensors, and the robustness to illumination variations.

The goal of the lidar odometry module is to estimate the the 6-DOF motion $T \in SE(3)$ of the robot between consecutive lidar acquisitions by means of scan-registration: by computing the rigid body transformation that best aligns consecutive 3D observations, it is possible to retrieve an estimate of the agent's ego-motion, and use the computed incremental transform to progressively update the trajectory followed by the robot.

The Iterative Closest Point (ICP) algorithm [85] is a popular method for finding the transformation between two lidar scans and represents a key building block of most SLAM algorithms. The ICP algorithm consists of two main stages: *i)* data association, *ii)* transformation computation. Given two sets of points $A = \{a_1, a_2, ..., a_n\}$ (source) and $B = \{b_1, b_2, ..., b_m\}$ (target) with a_i, $b_i \in \mathbb{R}^3$ the data association stage computes correspondences between points in the two point clouds by retrieving for each point in the source point cloud its nearest Euclidean neighbour from the target point cloud: this produces a set of correspondances $C = \left\{ \{a_x, b_x\}_1, \ldots, \{a_y, b_y\}_k \right\}$ such that points in each pair are the same point in the environment when expressed in a common frame. Once correspondences have been obtained, the second stage of the algorithm tries to find the rigid body transformation $T \in SE(3)$ that better aligns the two point clouds A and B so that the sum of the squared errors between corresponding points in the two set is minimized as illustrated in Eq. 2.1 where i indexes each pair of correspondences.

$$E(T) = \sum_{i=0}^{k} \|Ta_i - b_i\|^2 \tag{2.1}$$

These operations are performed iteratively until a convergence criteria is met, such as the solution changing less than a threshold. At each iteration, points in A are transformed with the best estimate of T and correspondences are retrieved by finding for each point the nearest Euclidean neighbour in B. The optimal transformation is the one that minimizes the error.

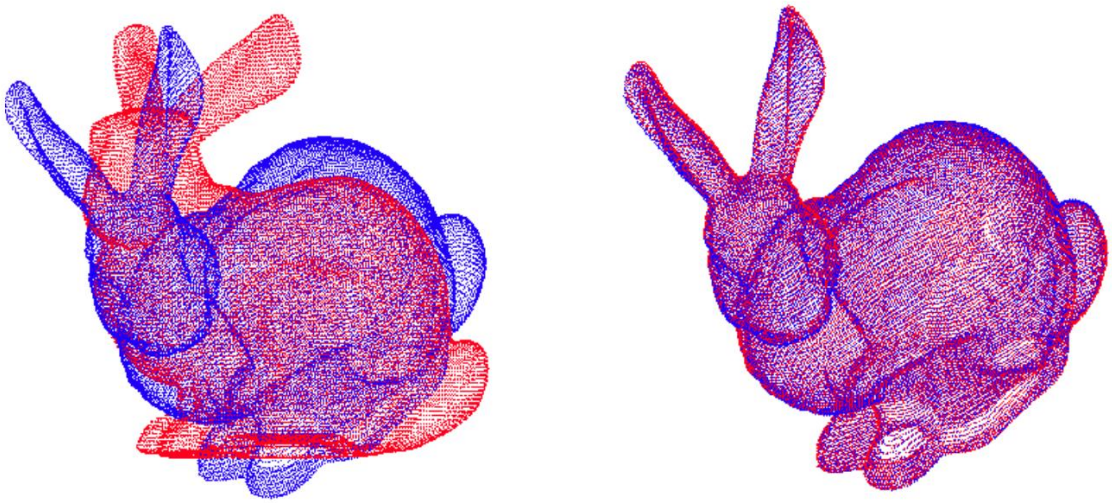


Figure 2.3: Example of registration of two point clouds with the Iterative Closest Point (ICP) algorithm. Image taken from [86]

The perfect correspondence assumption at the point-wise level of ICP can introduce systematic error in the ego-motion estimation process, for this reason, many variants of the original algorithm have been proposed over the years to relax the perfect correspondence assumption and enhance accuracy and efficiency [87]. [88] assumes points are sampled from a surface and presents a point-to-plane ICP variant that matches points to local planar patches. The Generalized Iterative Closest Point (GICP) [89] instead match local planar patches in both scans, and represents one of the currently most widely adopted solutions in lidar-based SLAM. As all these approaches rely on a proximity-based data association, they are commonly known to be non-convex optimization problems highly susceptible to local minima if an accurate initial guess of the transformation between the scans is not provided: while a correct convergence might be achievable if the starting positions of the two point clouds are close enough, an accurate initial guess quickly becomes critical when the robot experiences high-rate motions, rapid rotations, and strong platform vibrations when navigating over rough terrains.

Related Work

Lidar Odometry (LO) algorithms can be categorized by the representation type and the number of points (or features) used to align lidar-scans, including *(i)* feature-based methods, *(ii)* grid-based methods, and *(iii)* dense methods.

Feature-based methods Feature-based methods rely on extracting and matching salient environmental features across consecutive lidar-scans to estimate the ego motion of the robot. Possible features can include planar and edge features [67, 77, 90, 71], ellipsoidal surfels [91] and ground features [66]. These features can be matched with different approaches including proximity [84], type [67], or descriptor [91], depending on the algorithm and feature type used.

Grid-based methods In probability grid-based methods the key idea is to abstract lidar observations into a number of Gaussian distributions by mapping lidar-scans into grids to compute occupancy probability densities. These probability distributions can then be matched by using the Newton's method [92] to minimize the distribution-to-distribution distance and find the optimal transformation that describes the robot motion across consecutive observations.

Dense methods Dense methods work instead with a large subset of lidar-scan points. As using the full point cloud can be computationally expensive for real-time operation, most approaches select a subset of points for scan matching. The Generalized Iterative Closest Point (GICP) [89] is a common dense point-based scan matching method, where local surface normal information is used to better address the measurement noise in scan matching by using both point-to-point and point-to-plane matching, with planes being evaluated in local neighborhoods.

Scan-to-scan alignment The computation of the optimal alignment between two scans can be cast as a non-linear optimization problem, addressed by various solvers including Levenberg-Marquardt (e.g. [67]), iterative gradient descent (e.g. [84, 89, 90]), optimization tools such as Ceres [93], least-squares solvers (e.g. [94]), or in a sliding-window, pose-graph structure (e.g. [95, 65]) with GTSAM [96]. Recently, [97] present

a fast and certifiable algorithm for the registration of 3D point clouds that can achieve remarkable accuracy also in the presence of large amounts of outlier correspondences.

Scan-to-map alignment To enable global consistency across the history of scans, the computed pose is refined by aligning the current scan and the existing map. The scan-to-map alignment step has been proved to considerably enhance the overall accuracy of the lidar odometry solution [84]. For various map representations, ranging from feature-based (e.g. [67, 98, 95, 66]), to grid-based maps (e.g. [69]) and point-based maps (e.g. [84]), one can adopt different scan-to-map alignment methods. This includes point-based alignment methods (e.g. [84]), Normalized Distributions Transform (e.g. [92, 99]) or smoothing function alignment (e.g. [69]).

Sensor fusion While pure lidar-based methods are powerful, their performance can significantly degrade when it comes to perceptually-challenging conditions, including environments with geometrically self-similar patterns, partial observability [76], or agile robots with high-rate motions. To address these challenges, it is important to fuse lidar with additional sensing modalities to improve ego-motion estimation accuracy, such as an inertial measurement unit (IMU) or visual-inertial odometry (VIO). The IMU can provide rotational estimates that are tightly integrated (e.g. [69, 98, 65, 71]) or loosely integrated (e.g. [91, 95, 67, 90]) with the scan matching process. When the drift is translational (referred to as *lidar-slip* in this dissertation), VIO, wheel-inertial odometry (WIO) and kinematic-inertial odometry (KIO) can complement IMUs by providing a full 6-DOF transform estimate. Tight integration (e.g. [77]) and loose integration (e.g. [95, 69]) of these methods with lidars have shown significant improvements over individual use of either one of these modalities. Tightly coupled approaches are typically threatened by potential failures of fused sensing modalities, whereas loosely coupled methods could be designed to tolerate a loss. Initial approaches integrating additional sensor measurements have been demonstrated in caves [100], mines [101] and urban environments [102, 77, 66]. While multi-sensor fusion represents a vital step in achieving accurate operation under challenging perceptual conditions, potential failures in different sensing modalities can degrade, or drastically compromise the odometry performance if not properly handled.

2.2 Object Detection

When operating in unknown environments, semantic understanding of the surrounding space is a primary capability required to enable autonomous robots in providing meaningful situational awareness updates to a remote human operator who can rapidly gain knowledge of salient objects within the area of interest. When paired to an accurate location estimate, the successful detection of a significant object or phenomena in the environment constitutes the foundation of the overall situational awareness that the multi-robot system is tasked to provide.

The problem of object detection consists in: *i)* determining where objects are located in a given visual observation (object localization), *ii)* and determining to which category or class each object belongs to (object classification). The last decade has seen the rapid rise of Convolutional Neural Networks (CNNs) based object detection approaches as they can achieve remarkable performance over a wide range of different settings including change in scale, pose, and illumination.

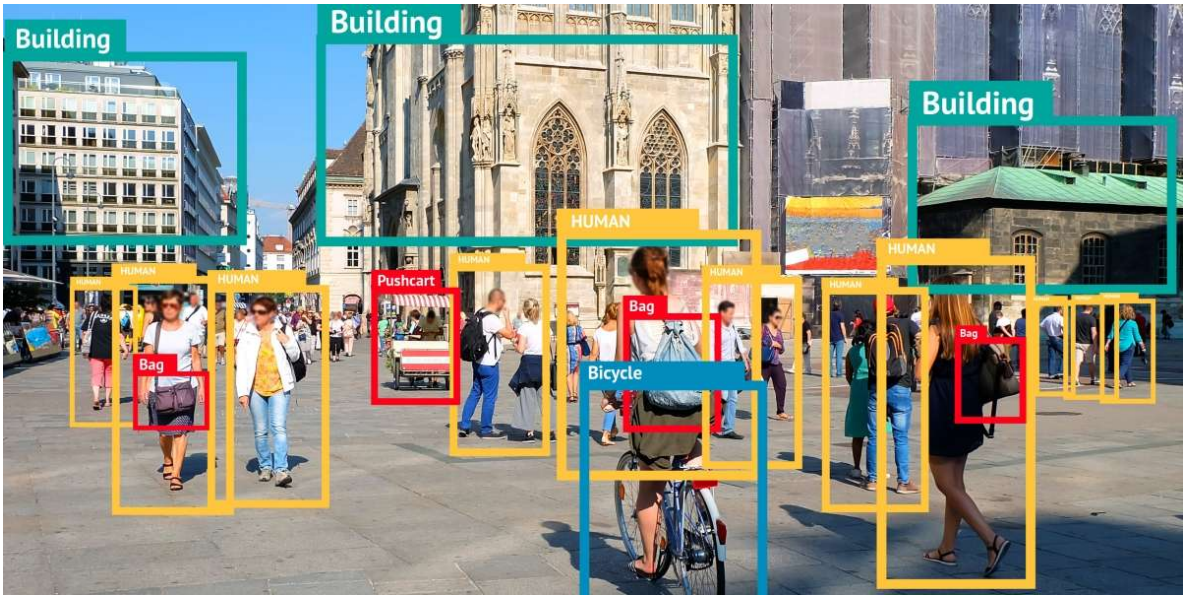


Figure 2.4: Example of YOLO's object detection on a sample image: multiple labelled bounding-boxes provide semantic understanding of the visual information. Image taken from [103]

Related Work

The object detection problem has been widely studied on both 2D [104, 105, 106, 107, 108, 109] and 3D [110, 111, 112, 113, 114] data. The reader is referred to [115] for an extensive review of the state-of-the-art techniques in object detection.

YOLO [106, 107, 108] provides a single shot architecture for object detection capable of computing bounding boxes and class probabilities directly from full images in a single evaluation and currently represents the state-of-the-art solution due to its balance between accuracy and speed. Yet, the approach struggles in low-illumination settings and its computational cost is not negligible leading computationally-constrained platforms needing to downsample the input image before the detection stage to pursue real-time operation: a possible shortcoming of this is that a downsampled version of the image might result in degraded object detection performance as the more the image is downsampled, the harder it is for YOLO to successfully detect objects of interest.

[116] present a cascaded and modular pipeline for the detection, localization and visualization of objects of interest that is adaptable to heterogeneous robots with arbitrary sensor configurations. For the detection of visually-observable objects, authors process visual data coming from on-board cameras and thermal cameras with a state-of-the-art machine learning framework where a pretrained convolutional neural network is fine-tuned over a specific set of objects. The detection networks produce 2D bounding boxes within the image that are combined with depth measurements to compute the position of the artifact relative to the robot. [23] extend the work presented in [116] and focus on multi-modal recognition of man-made objects (e.g. fire extinguisher, drill, rope, helmet, survivor manikin, backpack, vent, and cell phone) by leveraging visual cameras, depth measurements, and thermal cameras. To adapt to various processing capabilities, they rely on different CNN implementations using: a YOLO Tiny [108, 109] on ground robots leveraging GPU hardware, a MobileDet [117] on drones modified to run on a Google EdgeTPU. The artifact reports from each robot are then processed with a larger and more performant detection network (YOLOv4 [109]) on a remote base station to reduce the number of false positives, and update the detection confidence of each report.

2.3 Distributed Computation

To address the problem of degraded situational awareness accuracy on computationally-constrained platforms, the concept of distributed computation can be used to distribute heavy perception-related computation (e.g. lidar odometry, object detection) across the multi-robot system and enhance the overall performance of the robot team.

2.3.1 Approaches

Different approaches for distributed computation have been proposed over the years in the field of networked robots including cloud, edge, and fog.

Cloud [118] extend the computation and information sharing capabilities of networked robotics by proposing a cloud based architecture where cloud technologies (e.g. computing, storage) benefit robotics applications by providing supplemental computation to computationally limited platforms. Rapyuta [119, 120] is a popular open source cloud robotics platform which enables robots to offload heavy computation by providing secured customizable computing environments in the cloud. The major disadvantage of cloud-based solutions is represented by the dependence on persistent connection to an external network, scenario which is unlikely to be achievable in disaster relief or planetary exploration missions.

Edge Edge computing introduces the idea of processing the raw data at the local network level instead of the cloud to decrease application latency and optimize the network load [121]. While installing powerful edge-devices at the perimeter of a robotic warehouse might be an attractive solution to provide robots with supplemental computation capabilities, this quickly becomes impractical if robots have to operate in extreme environments that are too dangerous for humans to enter. Moreover, while single base-station solutions can be very helpful, they still constitute a single point of failure for the entire system as robots might not take advantage of additional computational assets in case of failure of the single highly-resourceful edge-device.

Fog The fog paradigm detaches from potential lack of connectivity to cloud or edge services in extreme operations by moving the computation within the self-contained robotic cluster connected via a local network and exploiting peers and intermediary network devices to share resources to execute intensive computations. In these settings, robots might assist their teammates in computationally demanding tasks, and offer more localized and low-latency services. The solution designed is independent of connections to an outside network to avoid possible communication bottlenecks and enables team scalability. [122] introduce the concept of robotic cluster by empowering heterogeneous robots with the ability of sharing their processing resources when solving complex collective problems and applies the proposed concept to the problem of topological map merging in a distributed multi-robot system communicating through a wireless ad-hoc mesh network.

2.3.2 Fields

This subsection provides a high-level overview of the main different fields of Distributed Computation.

Multi-Robot Task Allocation Multi-Robot Task Allocation (MRTA) is the process of assigning a set of robots to a set of task so that the overall system performance is optimized subject to a set of constraints. The most popular approach to MRTA is the market-based mechanism where each robot checks its eligibility, computes the cost of performing a task, and places a bid for the task: the task is then auctioned off to a robot bidding with the least cost [123].

Multi-Robot Task Scheduling Multi-Robot Task Scheduling (MRTS) [124] is a subset of MRTA problems when the tasks have time dependencies, namely, some tasks can be executed only if their predecessors have been executed. In its general form, the MRTS problem entails assigning tasks to appropriate workers and ordering task executions on each worker so that task-precedence requirements are satisfied and a minimum overall completion time is reached.

Computation Offloading Computation offloading alleviates restrictions of resource-constrained mobile systems (e.g. limited computation, limited battery life) by sending heavy computation to more resourceful servers to obtain back the result of the requested computation without on-board processing. Aspects that need to be taken under consideration when designing a computation offloading mechanism include: *i)* why to offload (e.g. improve application performance, save energy), *ii)* when to offload (e.g. static offloading where the tasks to be offloaded are pre-defined and do not depend from the environment condition, and dynamic offloading approaches that adapt to different run-time conditions such as fluctuating network bandwidth and latency constraints [125]), *iii)* where to offload (e.g. cloud, edge, fog) [126].

2.3.3 Related Work

Within the broad domain of distributed computation, we aim to review related work addressing offloading of computationally-expensive perception-related tasks over the local cluster of robots to enhance perception accuracy of computationally-constrained platforms and increase the overall reliability of the situational awareness retrieved by the heterogeneous multi robot system. If possible, we investigate whether the routing problem is also addressed when designing the distributed computation architecture.

[127] investigate the problem of MRTS in a representative heterogeneous multi-robot system for planetary exploration missions. A common clock synchronizes all agents acting in a 5 s broadcast, 10 s plan, and 30 s execute cycle. In the broadcast stage, agents flood their status update to let other agents update their shared view of the world model. In the planning stage, a solution for the task scheduling problem is computed on-board of every robot with a deterministic solver: while computing a solution, the status of inter-agent communication links in the execution stage is assumed to be known a-priori as movements of the agents are already scheduled. In the execution stage, scheduled tasks are performed on-board of each robot as decided by the mission scheduler. A first possible shortcoming of this approach is that assumption of a-priori knowledge of communication links status for 30 s execution windows does not hold in extreme operations in communication-degraded and extreme subterranean environments, as robots move freely in non line-of-sight settings and communication links

can go down at any moment. A second possible problem is that potential failures of robotic agents during the execution stage are not modeled during the planning stage: in extreme operations this represents a real possibility and could challenge the overall mission throughput as a robot would not be able to address a given obligation, introducing further troubles if scheduled tasks exhibit chained dependencies. Authors conclude that the main limitation of this work is that the presented approach is not robust to failures of the broadcasting synchronization mechanism.

[128] provide a middleware for distributed computing in heterogeneous mobile robotic networks which allows easy integration of the scheduler presented in [127] into existing autonomy executives. The proposed architecture consists of two processes responsible for orchestrating task sharing across different nodes: a dispatcher and a resource-obligation matcher, where a controller provides the resource-obligation matcher with a global and consistent world view. The goal is to enable autonomous agents to share computational resources for computationally expensive tasks such as localization and path planning. Authors conclude that the proposed algorithm is not suitable for networks with time-varying, non periodic connectivity.

[129] provide an overview of the motivations, techniques, technological enablers, and architectures for computation offloading for mobile systems. [130] study the trade-off in computation and communication to maximize perception accuracy of a dynamic phenomenon observed by multiple agents by trying to balance the on-board data preprocessing, the network delay, and the post-processing components.

[131] study the problem of finding the best access point for computation offloading to the cloud in a multi-robot system composed of static agents, without addressing the data routing aspect. [132] investigate multi-hop cooperative computation offloading to a remote cloud infrastructure for Quality of Service (QoS) improvements, and treat the routing aspect with a hierarchical chained strategy. [133] study the problem of computation offloading in a UAV edge-cloud environment. The main limitation of this work are the assumption of a non-varying UAV topology and stable communication link status over time.

[134] develop a strategy to offload SLAM related computation-intensive tasks to cloud

centers from an indoor environment. [135] demonstrate that offloading computationally intensive tasks within SLAM algorithms to edge-computing devices can bring significant enhancements to localization and positioning algorithms with low latency and high reliability services. [136] focus on visual odometry offloading to edge-computing devices to study the impact of image compression rate on the overall accuracy and round trip latency without addressing data routing aspects. [137] explore the use of FPGAs at the edge for 2D lidar odometry offloading at the local network level: in this approach a single gateway can perform odometry calculations for multiple robots in real-time, but authors do not provide insights on how to optimally route the multi-robot data traffic in the network for edge-processing purposes. [138] present an offloading decision mechanism for multi-robot systems to minimize the execution time of a SLAM algorithm in a scenario where only one robot performs SLAM, while the other robots are merely used as computational assets: a main limitation of this approach is that distributed computation capabilities would be rendered unavailable on failure of the central decision maker. [139] exploit the concept of robotic cluster to allocate computational resources available in the multi-robot system for solving computationally expensive SLAM tasks and demonstrate gains in localization precision and map accuracy, while assessing the impact of network bandwidth on the performance. However, in the proposed approach only one robot executes SLAM in a leader-follower paradigm, and broadcast is used for data exchange at the inter-robot level. [140] propose a computation offloading framework where the system contains three main components: the robot cluster, the local coordinator and the remote cloud. In this approach, a local coordinator handles the execution of centralized scheduling algorithms and management of computation offloading requests. Robots can offload their tasks in conditions if they need others assistance, or execute tasks offloaded by peers on demand. The local coordinator can be a computer in the local network or a robot chosen from the robot cluster and provides offloading management, resource management and task scheduling functions. A first limitation is that the work only optimize the performance for a single offloading request, not addressing dynamic multi-client settings, nor addressing data routing aspects. A second main limitation of the work is that the scenario of failure of the central orchestrator is not addressed.

2.4 Networking

Reliable networking is a key-prerequisite to enable successful operations of multi-robot systems as the networking infrastructure can be used to exchange data between robots, support computation offloading purposes, and provide situational awareness updates to a remote base station that maintains a coherent understanding of the mission status. However, extreme environments are usually characterized by the lack of a previously available communication infrastructure, and might present a wide range of challenges for inter-robot wireless communication due to complex propagation of radio signals in unstructured environments and limited line-of-sight opportunities for data transmission.

2.4.1 Mobile Ad Hoc Network

A mobile ad hoc network (MANET) is a peer-to-peer, self-forming and self-healing wireless network, where the network is ad hoc as it does not rely on a pre-existing communication infrastructure. In a MANET each node, either static or mobile, participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is dynamically made basing on the network connectivity and the routing algorithm in use. For these reasons, MANET are usually exploited to provide a wireless communication backbone for inter-agent communication in multi-robot systems: while WiFi can work reliably in the short range, radio communication is preferred for long range communication between robots.

2.4.2 Related Work

[141] provide a concept of operations for supervised autonomy of a robotic team in communication-degraded settings. The robot team makes use of mesh networking technology, where the wireless mesh network is formed by the base station node, robot nodes, and droppable communication nodes which are carried and deployed by carrier robots. Robots are designed to autonomously drop radio communication nodes on strategic locations throughout the exploration to dynamically extend and maintain a backbone network for inter-agent communication purposes, and for situational awareness updates to a remote base station. Communication nodes can be dropped at junctions, or if the

signal-to-noise ratio to the base station goes below a threshold due to distance or sharp turns, or by criteria focusing on the estimated available bandwidth between each radio [142].

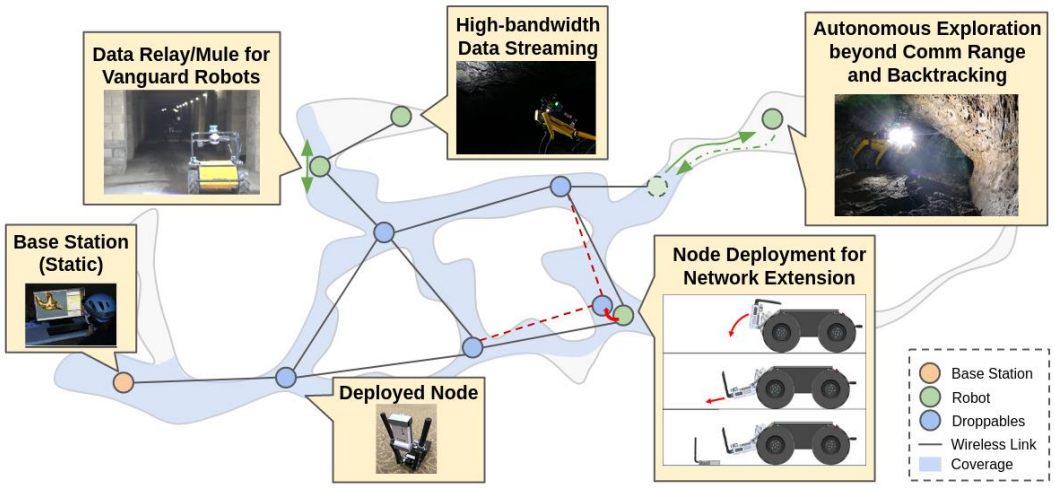


Figure 2.5: Example of mesh network extension into a subterranean environment by means of autonomous communication node dropping behaviours. Image taken from [23].

[143] present CHORD (Collaborative High-bandwidth Operations with Radio Droppables), a communication system for operation in communication-degraded subterranean environments. CHORD maintains high-bandwidth links to multiple robots for efficient commanding, autonomous operation, and data gathering in complex, large-scale subterranean environments. Each agent communicates by means of a wireless mesh network using commercial off-the-shelf radios. ROS 1 is used for intra-robot communications, while ROS 2 is used for inter-robot communications.

[144] investigate energy-aware data routing strategies for subsurface exploration of a robot team with limited lifetime. As flooding techniques commonly employed to increase chance of data delivery are expensive in terms of battery and network congestion, authors address this problem by proposing an energy aware contact graph routing solution that finds paths of minimal energy over a time-varying topology.

2.4.3 Software Defined Networking

The Software Defined Networking (SDN) paradigm abstracts network control functions (control plane) from network forwarding functions (data plane) disclaiming networking devices from the responsibility of taking routing decisions and centralizing intelligence in a separate component. In a SDN based approach the knowledge of the status of the network is centralized in the SDN controller which maintains a coherent global view of the network, and is responsible to take dynamic routing decisions to optimally manage the network traffic and meet QoS requirements at the application level. In such settings, the control plane decides where to send traffic, while the data plane of the networking hardware devices just purely forwards data where is needed upon decision update of the central orchestrator, achieving therefore optimum and globally-aware traffic management in complex distributed systems.

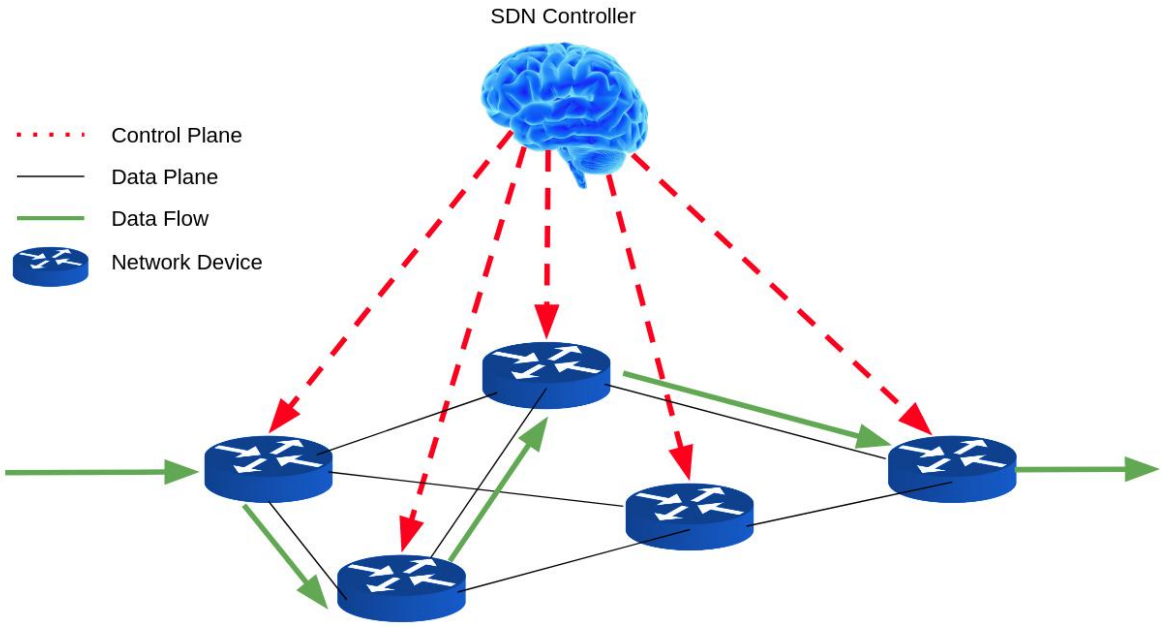


Figure 2.6: The Software Defined Networking (SDN) paradigm. A central controller, globally-aware of the state of the network, computes optimal routing decisions for all types of traffic and sends routing rules updates to networking devices.

While SDN paradigms have been mainly studied over wired static networks, we investigate their integration in dynamic multi-robot wireless mesh networks for their potential in enhancing high-volume perception related traffic in computation offloading frame-

works dealing with multi-client scenarios to support situational awareness accuracy enhancement purposes over an heterogeneous multi-robot system.

[145] survey the state-of-the-art in SDN technology. A typical SDN infrastructure consists of a multi-layer architecture composed of: *i)* Application Layer, *ii)* Control Layer, *iii)* Data Layer. To handle communication between layers two interfaces are defined: *i)* North-Bound APIs (NBAPIs), *ii)* South-Bound APIs (SBAPIs). The SDN Controller keeps track of QoS and bandwidth requirements from the Application Layer through NBAPIs to formulate routing decisions. The SDN Controller then makes use of SBAPIs to force updated routing rules to the Data Layer where pure forwarding devices are instructed to route traffic where the globally-aware SDN Controller wants. OpenFlow [146] is a commonly used protocol for SBAPIs of SDN systems that allows access to the forwarding plane of the network device, and that can be therefore used to update forwarding rules of the network devices upon decision of the central orchestrator. The placement of SDN Controllers in a network has been a widely studied field of research including centralized, distributed, and hybrid approaches. In the centralized approach, a single controller manages all the devices in the network, and is commonly seen as a single point of failure or bottleneck for the entire system.

2.4.4 Related Work

[147] investigate the possibility of applying SDN paradigms to a 5G mobile network. [148] present an hybrid design exploiting advantages of centralized SDN controllers (global network view, control programmability) and distributed protocols (robustness): in such approach the network control is split between the SDN controller and the mobile devices, allowing therefore mobile devices to make their own forwarding decision in case of failure of the central orchestrator. [149] propose an SDN-enabled UAV-assisted vehicular computation offloading optimization framework to minimize the system cost of vehicle computing tasks. The work does not cover network routing aspects, and only one UAV is deployed. Overall, very few works investigate usage of SDN paradigms in dynamic wireless multi-robot mesh networks.

Chapter 3

Contributions

This chapter outlines the gaps identified in the current state of the literature to then introduce contributions of this work. Investigation domains are two-fold: *i)* accurate and robust lidar odometry in perceptually-degraded environments, *ii)* enhancement of distributed situational awareness of an heterogeneous multi-robot system by means of computation offloading of perception tasks (e.g. lidar odometry, object detection) within the robot cluster aided by SDN paradigms.

3.1 Gaps

Gap 1 As for lidar odometry, while many multi-sensor methods that have been proposed can achieve great accuracy, they do not take into account potential failures of the fused sensing modalities, scenario which is likely to be observed in real world deployments, and that can result in catastrophic degradation of the odometry performance, if not robustly handled.

Gap 2 As for the enhancement of distributed situational awareness of an heterogeneous multi-robot system, most works focus on offloading perception-related computation to cloud or edge targets, destinations that are hardly suitable for robotic operations in extreme environments. Most works never address the dynamic routing problem, and either assume a stable network topology, a direct and persistent connection, or a-priori known inter-agent connectivity profiles. The very few works addressing computation offloading of perception tasks to other robot peers in the team, either model only one computation offloading request not targeting multi-client scenarios, or assume the pres-

ence of a central orchestrator, without addressing the problem of its potential failure. To the best of author's knowledge, no existing work in literature *simultaneously* address the aspects of: *i)* enhancing perception accuracy of computationally-constrained platforms of heterogeneous multi-robot systems by means of computation offloading to other robot peers, *ii)* managing high-dimensional perception-related traffic in distribuited computation infrastructures by means of SDN paradigms within a globally-aware central supervisor, *iii)* designing of a comprehensive framework for simultaneous management of computation and network resource allocation in the heterogeneous multi-robot system that can survive on failure of the central orchestrator.

3.2 Contributions

Robust ego-motion estimation in extreme settings As first contribution, we present a high-precision lidar odometry system to achieve robust and real-time operation under challenging perceptual conditions: LOCUS (Lidar Odometry for Consistent operation in Uncertain Settings), provides an accurate multi-stage scan matching unit equipped with an health-aware sensor integration module for seamless fusion of additional sensing modalities. LOCUS enables accurate, robust and real-time odometry in perceptually-stressing settings and is fail-safe to drops or loss of one or more sensor channels by relying on a loosely-coupled switching scheme between sensing modalities. Furthermore, LOCUS can be adapted to heterogeneous robotic platforms with diverse sensor inputs and computational capabilities. We present an extensive field demonstration of LOCUS. In particular, we provide results and insights from deploying LOCUS as part of the CoSTAR team's solution that won the Urban Circuit of the DARPA Subterranean Challenge. We present an ablation study on LOCUS and then compare the performance with six state-of-the-art methods using the data acquired in the field tests. We evaluate the performance of the proposed system against state-of-the-art techniques in perceptually-challenging environments, and demonstrate top-class localization accuracy along with substantial improvements in robustness to sensor failures. We then demonstrate real-time performance of LOCUS on various types of robotic mobility platforms involved in the autonomous exploration of the Satsop power plant in

Elma, WA where the proposed system was a key element of the CoSTAR team's solution that won first place in the Urban Circuit of the DARPA Subterranean Challenge.

Communication-aware and perception-oriented distribution of computation As second contribution, to address the problem of degraded situational awareness accuracy on computationally-constrained platforms of an heterogeneous multi-robot system we present Swarm Manager. The proposed framework exploits Distributed Computation and Software Defined Networking paradigms to enhance perception accuracy of computationally constrained platforms in heterogeneous multi-robot systems, by allowing robots to offload heavy computation (e.g. lidar odometry, object detection) to other more resourceful peers in the dynamic multi-robot mesh network under decision of a globally-aware and dynamically eligible central orchestrator. Without relying on remote cloud or edge utilities, for each offloading request generated by clients, Swarm Manager not only identifies where is best to execute the requested computation (optimum server) in a task-aware fashion in the robotic cluster, but also computes through SDN paradigms the best path in the network (optimum route) that maximizes communication performance given the current system state. The proposed approach does not make assumptions on a-priori knowledge of inter-agent communication link profiles and can work in dynamic settings while also exploiting on the fly potential failures of robotic agents in the team (e.g. a ground rover stuck in the mud, a fallen legged robot) to then use them as computational assets for other robot needs. A main novelty of the proposed approach consists in overcoming the well-known point of failure of centrally-supervised systems, where the orchestration mechanism provided by the central supervisor would become unavailable on its failure: to address this problem, we formulate a dynamic manager election mechanism, that makes the framework self-healing and resilient to potential failures of the central supervising entity. We realize the framework in ROS and demonstrate situational awareness accuracy enhancements for a team of four heterogeneous robots involved in the autonomous exploration of a multi-level and communication-degraded dismissed power plant in Elma, WA with the data gathered during the Urban Circuit of the DARPA Subterranean Challenge.

Chapter 4

LOCUS

A reliable odometry source is a prerequisite to enable complex autonomy behaviour in next-generation robots operating in extreme environments. In this chapter, we present a high-precision lidar odometry system to achieve robust and real-time operation under challenging perceptual conditions. LOCUS (Lidar Odometry for Consistent operation in Uncertain Settings) [150], provides an accurate multi-stage scan matching unit equipped with an health-aware sensor integration module for seamless fusion of additional sensing modalities. We evaluate the performance of the proposed system against state-of-the-art techniques in perceptually-challenging environments, and demonstrate top-class localization accuracy along with substantial improvements in robustness to sensor failures. We then demonstrate real-time performance of LOCUS on various types of robotic mobility platforms involved in the autonomous exploration of the Satsop power plant in Elma, WA where the proposed system was a key element of the CoSTAR team's solution that won first place in the Urban Circuit of the DARPA Subterranean Challenge. In this chapter, we present LOCUS (Lidar Odometry for Consistent operation in Uncertain Settings), a lidar odometry system that *(i)* enables accurate real-time operation in extreme and perceptually-challenging scenarios, and *(ii)* is robust to intermittent and faulty sensor measurements. LOCUS has been a key element of the CoSTAR team's solution [151] that won first place at the Urban Circuit of the DARPA Subterranean Challenge (SubT Challenge), where robots are tasked to autonomously explore complex GPS-denied underground environments (e.g. Fig. 4.1).

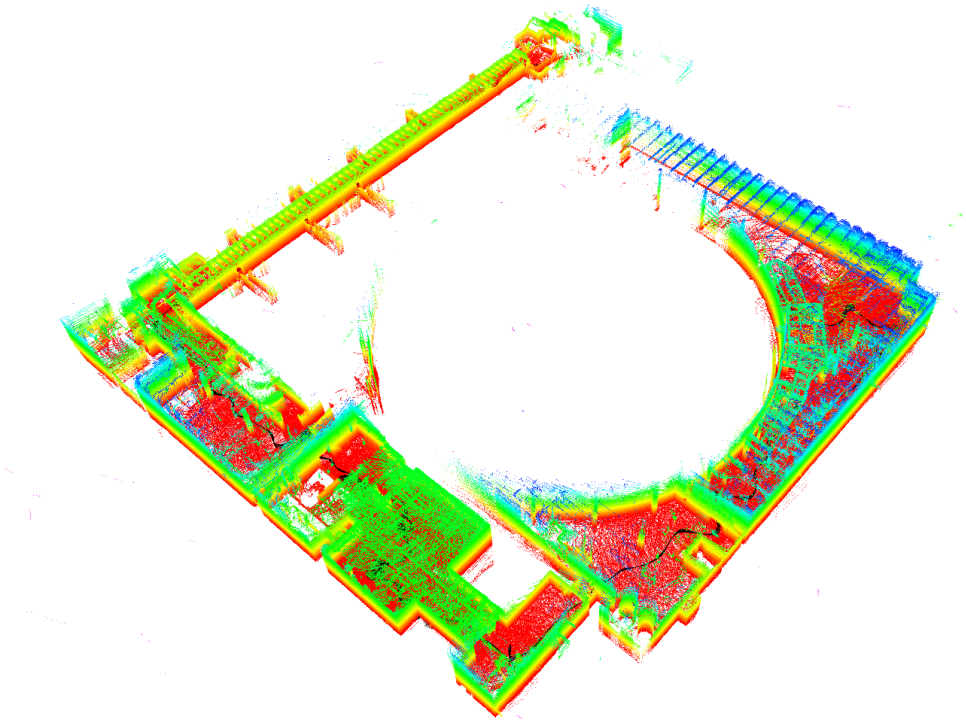


Figure 4.1: 3D map produced by LOCUS on a husky robot during the exploration of the Beta 2 run of the Urban Circuit of the DARPA Subterranean Challenge.

4.1 System Description

In this section, we describe the system architecture of LOCUS reported in Fig. 4.2 and provide explanation and details of each submodule.

4.1.1 Point Cloud Preprocessor

The point cloud pre-processor is responsible for the management of multiple input lidar streams (e.g. syncing, motion-correction, merging, filtering) to produce a unified 3D data product that can be efficiently processed in the scan matching unit.

Motion Distortion Correction We assume information from one or more 360 degree lidar sensors, such as the Velodyne Puck or Ouster lidars. The raw information coming from the lidar is fed into a motion distortion correction (MDC) unit which corrects the Cartesian position of each point to account for the motion of the robot while

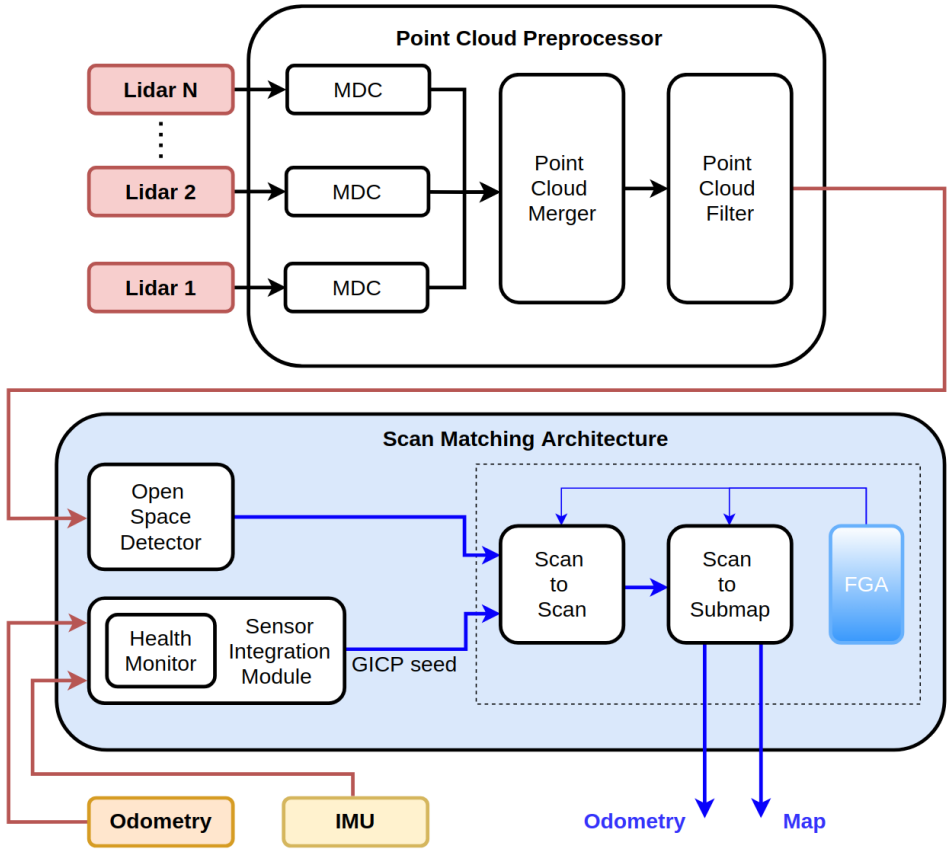


Figure 4.2: Architecture of the proposed lidar odometry system.

a single scan of the lidar is completed[1]. This correction is particularly important for points at large range, when high-rate rotations are experienced by the robot, and is a commonly employed step [98, 95]. The correction is informed by either an IMU, or an odometry source (e.g VIO, WIO, KIO) where the chosen source depends on what is reliably available and calibrated on a given robot.

In our setup, MDC is implemented at the driver level, which provides a simple yet effective solution to combine multiple motion corrected acquisitions from different sensors into a unified point cloud in the merging stage: other algorithms do not offer this flexibility and make assumption on the structure of the incoming point cloud, for example by means of ring processing (e.g. LOAM, LIO-Mapping, LIO-SAM) leading therefore to decreased adaptability of operation when working with multiple lidar sensors.

Point Cloud Merger For robots with multiple lidars, to enlarge the overall robot field-of-view and compensate for complementary perception blind-spots, the point cloud

[1]The Velodyne Puck lidar requires 0.1 s to complete one scan

merger (Fig. 4.2, top and middle) combines each motion-corrected point cloud into a single one, using the known rigid body transformation between sensors. This step is implemented so the rest of the pipeline is consistent for robots with one or multiple lidar sensors. Combining multiple sensors readings can provide with a denser point-cloud, which greatly enhances ego-motion estimation and mapping accuracy: this however at the cost of increased computational load, due to the greater number of points to process.

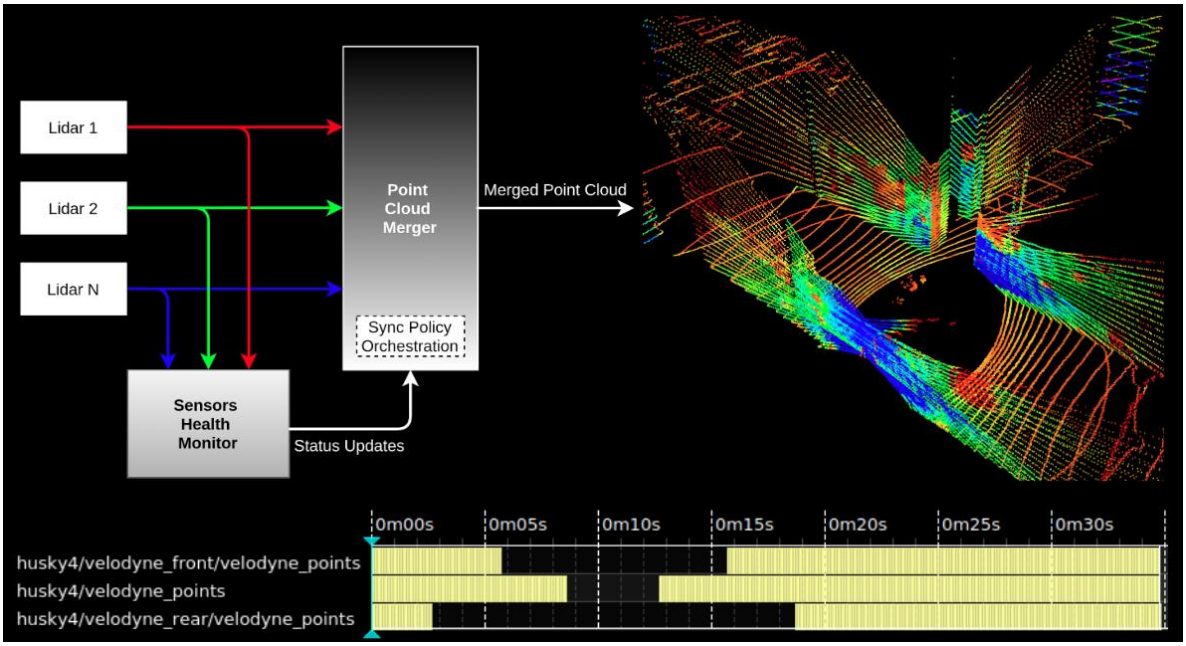


Figure 4.3: Failure-aware multi lidar merger. Top left) an intermediate health monitor layer watches the health of multiple lidar feeds to dynamically update the synchronization policy of the merger to adapt to the number of currently available lidar feeds. Bottom) Lags in lidar data for three Velodynes mounted on-board of a wheeled platform. Top right) Beside for the time interval where *no* lidar data is available, the point cloud merger always produce an output point cloud no matter which failures are experienced by individual feeds.

We design the Point Cloud Merger to be failure-safe on lags/failures of individual lidar streams: an intermediate lidar health monitor layer keeps track the actual availability of the different lidar sources, to then dynamically update the time synchronization policy of the Point Cloud Merger to the number of actual available feeds. This is an essential feature in the development of a failure-aware framework for the exploration of extreme

terrains where sensors can break at any moment greatly compromising the performance of the downstream processing.

If a common clock is available for multiple on-board lidar sensors, instantaneous acquisitions could be merged into a unified information by just using the *static* extrinsic transformation between them. In real world practice, a common clock might not be available. Therefore, to account for the motion of the robot between slightly different timestamps of lidar acquisitions, we use an external odometry source to inform the point cloud merger for relative motion offsets in order to merge multiple unsynced lidar acquisitions into a unified, spatially and temporal coherent frame of reference (the merged point cloud is expressed in base link frame, at the timestamp of the main lidar acquisition).

Point Cloud Filter The resulting point cloud is then processed in the point cloud filter to remove noise and out-of-range points, manage the volume of data, and reduce computational load. The point cloud filter is composed by a sequential combination of a: *i)* body filter, *ii)* 3D voxel grid filter, *iii)* random downsampling filter. Each component can be individually tuned, activated and deactivated. The body filter is a box-filter that removes occlusions of the robot body and on-board carried communication nodes. The voxel grid filter takes the average of the points in each 3D volume (a voxel) to decrease the data size while still capturing the dominating structure of the environment: we use a voxel size of 0.1 m in the tests presented in this work. The random downsampling filter randomly samples points with a uniform probability: in the results present in this chapter, we use an implementation of [152] with a downsampling percentage of 90%. For all filters we use the implementation in the Point Cloud Library [153].

4.1.2 Scan Matching Unit

The scan matching unit (light blue box in Fig. 4.2) performs a cascaded GICP-based scan-to-scan and scan-to-submap matching operation to estimate the 6-DOF motion of the robot between consecutive lidar acquisitions.

Notation We denote with $\mathcal{R}$ the coordinate system of the robot and with $\mathcal{W}$ the coordinate system of the world, that coincides with $\mathcal{R}$ at the start of the test. We therefore address the problem of determining the poses of $\mathcal{R}$ with respect to $\mathcal{W}$ by means of consecutive lidar acquisitions to incrementally reconstruct a trajectory and a map of the explored environment. We denote with L_k the lidar scan collected at time k and with L_{k-1} the lidar scan collected at time $k-1$. All lidar scans are expressed in the robot frame. We denote with $\mathbf{X}_k \in SE(3)$ the robot pose in $\mathcal{W}$ at time k and with $\mathbf{X}_{k-1}$ the robot pose in $\mathcal{W}$ at time $k-1$. We denote with $\mathbf{T}_k^{k-1} = \mathbf{X}_{k-1}^{-1}\mathbf{X}_k$ the rigid body transformation between two consecutive robot poses, where $\mathbf{T}_k^{k-1} \in SE(3)$ is the transform between $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$.

Sensor integration module

In the joint optimization of accuracy and robustness, a key component of the pipeline is the sensor integration module. In robots with multi-modal sensing, when available, we use an initial transform estimate from a non-LiDAR source in the scan-to-scan matching stage to ease the convergence of the GICP , by initializing the optimization with a near-optimal seed that improves accuracy and reduces computation, enhancing real-time performance.

Health monitoring Multiple sources of odometry (e.g VIO, KIO, WIO) and raw IMU measurements available on-board are first transformed into $\mathcal{R}$, and then fed into a sensor integration module which selects the output from a priority queue of the inputs that are deemed healthy by a built-in health-monitor, which prioritizes the order based on the expected accuracy of the methods. (see bottom left of Fig. 4.2). The system is designed to take in a variety of sources of health metrics to evaluate the health of input sources. For example, ongoing work is looking to integrate with the Heterogeneous Robust Odometry (HeRO) system [154] that employs custom health analysis (such as feature counts and observability analysis) on each odometry source, as well as rate and covariance checks. For our implementation presented below, we use a simple rate-check: if input messages are at a sufficient rate (> 1Hz), then the source is healthy.

Priority queue The priority queue is intend to always select the highest accuracy source, based on previous testing for a given robotic system in similar environments [2]. If the robot enters an area where the highest priority source is degraded, it is intended for this to be reflected in the health metric, that would trigger a transition to the next highest, healthy input. With this health-metric-driven dynamic switching, the priority queue is static. The priority queue for our legged robot is: VIO, KIO, IMU, no input, and for our wheeled robot is: VIO (if present), WIO, IMU, no input.

We define the pose estimate (with respect to $\mathcal{W}$) of the highest priority source that is found to be healthy as $\mathbf{Y}$. To reduce operations, we buffer only $\mathbf{Y}$ and interpolate the buffered data at lidar timestamps, t_{k-1}, and t_k to get $\mathbf{Y}_{k-1}$ and $\mathbf{Y}_k$: the pose of highest priority healthy source at times t_{k-1} and t_k, respectively. We denote with $\mathbf{E}_k^{k-1} = \mathbf{Y}_{k-1}^{-1}\mathbf{Y}_k$ the rigid body transformation of the sensor integration module output between $\mathbf{Y}_{k-1}$ and $\mathbf{Y}_k$ in the $[t_{k-1}, t_k]$ time interval where $\mathbf{E}_k^{k-1} \in SE(3)$. Each odometry source provides a rotation and translation, whereas for the IMU we only use the rotation measurements.

Scan-to-scan

In the scan-to-scan matching stage, GICP computes the optimal transformation $\hat{\mathbf{T}}_k^{k-1}$ that minimizes the residual error $\mathcal{E}$ between corresponding points in L_{k-1} and L_k.

$$\hat{\mathbf{T}}_k^{k-1} = \arg\min_{\mathbf{T}_k^{k-1}} \mathcal{E}(\mathbf{T}_k^{k-1} L_k, L_{k-1}) \tag{4.1}$$

If the sensor integration module is successful, we initialize the GICP with $\mathbf{T}_k^{k-1} = \mathbf{E}_k^{k-1}$. If all sensors fail, the GICP is initialized with identity rotation and zero translation and the system reverts to pure lidar odometry.

[2] Different characteristics of the environment can challenge different sensing modalities: *i)* textureless and texture-repetitive areas along with dust, fog and smoke, are elements that can challenge the accuracy of VIO, *ii)* harsh surfaces can challenge the accuracy of KIO, *iii)* bumpy and unstructured terrains can challenge the accuracy of WIO. Ongoing work is looking into integrating additional health metrics on individual sensing modalities beyond the default rate-check to let the system always select the most reliable source given the current state of the environment. Please refer to [155, 154] for further details.

Figure 4.4: Visualization of the LOCUS priority queue. a) all external measurements are available, VIO is picked. b) VIO fails, so KIO is picked, c) Neither VIO or KIO are available, WIO is picked. d) No external odometry sources are available, IMU is picked. e) no measurements are available from on-board sensing modalities, GICP is therefore initialized with the identity pose. For the sake of brevity, we don't graphically model all other possible configurations (e.g. VIO available, KIO unavailable, WIO unavailable, IMU available, etc.)

When using an IMU, a Maximum Correntropy Criterion Kalman Filter (MCCKF) like then one presented by [156] can be used to enhance accuracy of attitude estimation in the presence of non-gaussian noise and outliers in raw IMU measurements (e.g. acceloremeter, gyroscope, magnetometer) which might occur during the navigation of harsh terrains. This enhances the quality of the motion prior provided to the GICP in the scan-matching stage of LOCUS resulting in increased accuracy as demonstrated in [156].

Scan-to-submap

The motion estimated in the scan-to-scan matching stage is further refined by a scan-to-submap matching step. Here L_k is matched against a local submap S_k which is taken from the local region of the global map M_k given the current estimate of the robot pose in $\mathcal{W}$.

$$\tilde{\mathbf{T}}_k^{k-1} = \arg\min_{\mathbf{T}_k^{k-1}} \mathcal{E}(\mathbf{T}_k^{k-1} L_k, S_k) \tag{4.2}$$

In this optimization, $\mathbf{T}_k^{k-1}$ is initialized with the $\hat{\mathbf{T}}_k^{k-1}$ result from Eqn. 4.1. The global map is a point cloud stored in an octree format that is an accumulation of point clouds after every t meters of translation, or r degrees of rotation: for our results, we use $t = 1$, $r = 30^o$. We use an octree with a minimum resolution of 0.001m to store the map, which usually maintains all points in an easily searchable format.

Output After scan-to-scan and scan-to-submap matching, the final estimated motion $\tilde{\mathbf{T}}_k^{k-1}$ between consecutive lidar acquisitions is used to update the robot pose in $\mathcal{W}$: the generated odometry is therefore the integration of all computed incremental transforms.

Both accuracy and computational speed are improved by the incremental estimation from input odometry to scan-to-scan and finally scan-to-map (shown, for example, in [84]). A good initial estimate in both Eqn. (4.1) and Eqn. (4.2) reduces the chances of converging in a sub-optimal local minima, and reduces the number of iterations needed to converge, lowering computation time.

Notes on multi-threading

The computational speed of the scan-to-scan and scan-to-submap matching is greatly increased through a multi-threaded GICP approach modified from the PCL implementation [153]. The multi-threading utilizes a user-specified number of cores for the normal computation stage in GICP, which represents over 70% of the computation time in the process. For the evaluations performed in this work, we use 4 threads unless otherwise stated.

4.1.3 Environment Adaptation: Flat Ground Assumption

In human-made environments there are many areas with flat grounds, which if known prior, could be utilized to aid odometry algorithms. When detected or known, the flat ground assumption (FGA) can be activated to limit drift in Z and error in roll and pitch (lower-right blue box in Fig. 4.2). FGA operates on the output of both scan-to-scan and scan-to-submap alignment, by zeroing any Z movement, roll or pitch, all in a global, gravity aligned reference frame.

FGA activation modalities The system provides two ways to detect a flat ground and activate FGA: context-based, and sensor-based. The first approach relies on prior knowledge of the environment that can be acquired by a human supervisor, for instance in single floor exploration of urban environments. For stair-climbing robots, the initiation of a stair mission can be used to deactivate FGA, and then reactivate it when the stair mission is complete through the input of a human operator (see [155] for an example). In the second approach, an IMU monitor can be used to detect periods when the robot has near-zero roll and pitch over a sufficient time period to activate FGA, and then deactivate it when this condition is no longer met.

4.1.4 Adaptation for Different Platforms

The system includes adjustable components to adapt to heterogeneous robotics platforms with different computing power and sensors. These adaptations are primarily in: the number of lidars, the filtering, and number of threads for GICP and the measure-

ments used for the initial transform estimate. Sec. 4.2.3, demonstrates the flexibility of LOCUS through application to two different robotic platforms.

4.2 Field Experiments

In this section we present the experimental results obtained from tests in the Tunnel and Urban circuits of the SubT Challenge. We first use three datasets from a Clearpath Husky ground rover (Fig. 4.1a-b) to perform an ablation study on LOCUS, and compare it with state-of-the-art lidar odometry solutions. We then showcase results achieved during live operations in field tests across heterogeneous robotic platforms. See `https://youtu.be/5QQkkQ_YrbU` for visualization of the results.

Dataset description Each dataset comprises 3D lidar scans coming from 2 on-board VLP16 lidars (one flat, one pitched forward 30^o), along with IMU (Vector Nav 100) and WIO measurements for a 60-minute run. Each dataset is selected to contain components that are challenging for lidar odometry. The Urban datasets (Alpha Course, Fig. 4.1f and Beta Course, Fig. 4.1e) are collected in a dismissed power plant located in Elma, WA that presents many challenges for robot perception such as long feature-poor corridors and large open spaces (the test area dimensions are 100x100x20 m). The Tunnel dataset (Safety Research Course, Fig. 4.1c) is recorded in the Bruceton Research Mine in Pittsburgh, PA that is characterized by self-similar and self-repetitive geometries (the test are dimension are 200x200x10 m). All datasets have substantial vibrations and large, sudden accelerations as is characteristic of a skid-steer wheeled robot traversing rough terrain and rubble. See Fig. 4.1 for sample images of the environments.

Lidar scans are recorded at 10Hz. WIO and IMU are recorded at 50Hz in the Urban datasets, while a higher-rate IMU recording (100Hz) is available for the Tunnel dataset. Both motion corrected and raw points are available for the Urban datasets, whereas the Tunnel dataset only has raw points available. We use LOCUS to do scan matching on the ground-truth map provided by DARPA to estimate the ground-truth reference of the robot trajectory.

Figure 4.5: On the left, the husky robot used for the autonomous exploration and data collection. On the right, pictures of the circuits where the robot has been deployed within the context of the Tunnel and Urban circuit of the DARPA Subterranean Challenge.

4.2.1 Ablation Study

To investigate the impact of each component of LOCUS on the overall pipeline accuracy, we evaluate the Absolute Position Error (APE) of the robot-trajectory in the Urban Alpha dataset[3]. The results are summarized in Fig. 4.7. The study confirms that the use of motion-corrected points is essential, and highlights the effectiveness of the filtering approaches that limit the data volume and reduce computational load. Feature-based filtering (e.g. LOAM-type features of edges and planes) can result in greater accuracy, yet with a higher CPU load (25% more than the baseline configuration), leading to non-real-time performance on our system. Environmental knowledge of a flat ground is not essential, but can help to improve accuracy for exploration of human-made buildings. The loose sensor integration of WIO or IMU results in minor improvements, however, we use this approach as baseline to robustly operate in scenarios with high-rate motions or low lidar observability.

[3]While these results are for one dataset, we observed similar trends from tests on the Urban Beta dataset.

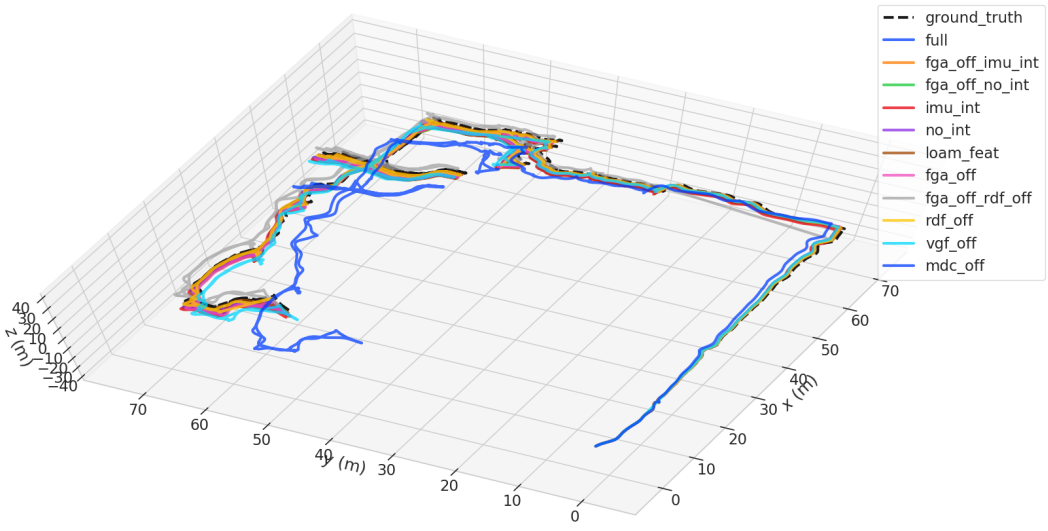


Figure 4.6: Visualization of LOCUS estimated trajectories in the Alpha course of the SubT Challenge for different processing configurations.

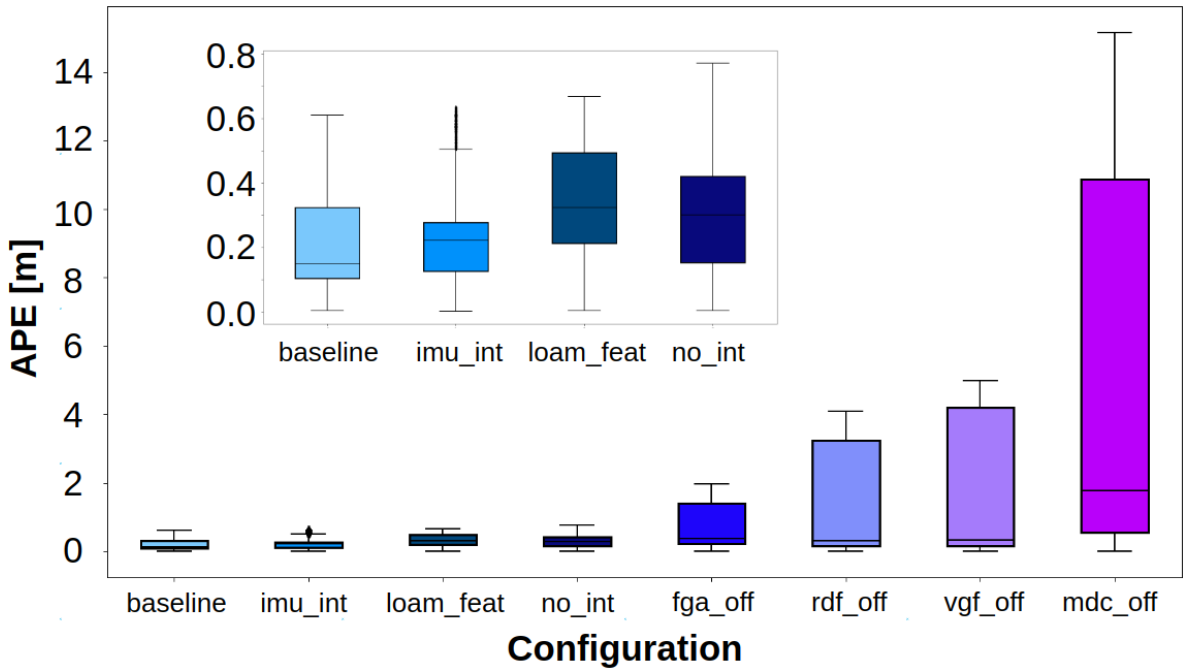


Figure 4.7: Evolution of the Absolute Position Error (APE) of the proposed method for different processing configurations in the Alpha course of the SubT Challenge. The inset gives more detail on the four best configurations. baseline: all features in Section locus. imu_int: no WIO integration, only IMU integration, no_int: neither WIO or IMU integration, loam_feat: using LOAM feature extraction instead of filtering, fga_off: no FGA, rdf_off: no random downsample filter, vgf_off: no voxel-grid filter, mdc_off: no MDC.

Table 4.1: Summary of State-of-the-Art, Open-Source Algorithms

Algorithm	Align.	Opt.	IMU	Odom.	MDC*
LOCUS	Dense	GICP	Loose	Loose	Yes
BLAM [157] [4]	Dense	GICP	None	None	No
ALOAM [67] [5]	Features	Ceres	None	None	No
FLOAM [67] [6]	Features	Ceres	None	None	No
Cartog. [69] [7]	Grid	Ceres	Tight	Loose	Yes
LIO-Map. [98] [8]	Features	Ceres	Tight	None	Yes
LIO-SAM [65] [9]	Features	GTSAM	Tight	None	Yes

* See section IIIB for more details.

4.2.2 Evaluation Against the State-of-the-Art

We compare the proposed algorithm against a variety of the state-of-the-art open-source lidar odometry systems, selected to cover the range of aligment methods and sensor integration methods, as summarized in Table 4.1. While we would like to compare against systems integrating with visual odometry (e.g. [77], [95]), or more recent multi-sensor released systems such as [158], these algorithms do not have open source implementations to readily test. FLOAM and ALOAM are two modern implementations of LOAM aimed to simplify the code structure and increase computational speed, respectively. Cartographer is an LIO algorithm distinguished by its use of grid-based matching. LIO-Mapping is a more recent LIO algorithm that combines the IMU pre-integration approach from VINS-Mono [46] with LOAM-type feature alignment. LIO-SAM is yet more recent, and builds from LeGo-LOAM [66], adding in IMU pre-integration in a smoothing-and-mapping approach.

Comparison criteria We aim to compare the lidar odometry systems holistically, hence we use three criteria: i) Accuracy, ii) Robustness and iii) Efficiency.

Comparison setup Each algorithm is setup for the best performance, yet with minimal parameter tuning, only input adjustment (number of lidars, motion corrected

[4] github.com/erik-nelson/blam_slam
[5] github.com/HKUST-Aerial-Robotics/A-LOAM
[6] github.com/wh200720041/floam
[7] github.com/cartographer-project/cartographer
[8] github.com/hyye/lio-mapping
[9] github.com/TixiaoShan/LIO-SAM

points). WIO is the odometry input, if used. Loop closures are disabled to focus on the lidar odometry performance. Variations in the input for each algorithm are summarized below.

LOCUS, BLAM, and ALOAM each use two lidars in all datasets, with motion-corrected points in the Urban datasets. FLOAM only succeeded with one lidar in the Urban Alpha dataset and otherwise uses two lidars, in each case (with motion correction in Urban datasets). Cartographer can do internal motion correction, but only with the input of points as individual UDP packets from a single lidar. We elected to instead feed pre-corrected scans from two lidars to Cartographer. LIO-Mapping and LIO-SAM are both set up to do internal motion-correction on the point clouds, leveraging the integrated IMU data, hence uncorrected scans are used as inputs. For LIO-Mapping, 2 lidars were used, except for Urban Alpha, where only a 1 lidar test was successful.

LIO-SAM was only able to run with a single lidar input, as there are internal assumptions of a structured point cloud of rings for motion correction and feature extraction. We were not able to get LIO-SAM working on the Urban datasets, likely due to the IMU rate, at 50Hz being lower than the recommended 200Hz for LIO-SAM.

Accuracy Evaluation

We evaluate the accuracy with two metrics: position error and map error.

Position error To evaluate the localization accuracy, we use evo [159] to compute the absolute position error (APE) of the trajectories estimated by the different methods against the ground-truth reference. We report in Fig. 4.14 a boxplot visualization of the APE results for the Urban and Tunnel datasets and summarize these results in Table 4.3.

The results show that LOCUS is equal to or better than the state-of-the-art in all datasets. FGA does help to improve LOCUS performance, but is not essential for LOCUS to perform well. LIO-Mapping has similarly low error, as expected with tight integration of IMU and lidar, yet with a large delay from a mean processing time of 1s

per scan. The larger optimization being performed with pre-integration, scan alignment and extrinsic estimation all together likely leads to the longer computation times.

LIO-SAM performs the best in the Tunnel dataset, and with feasible computational speeds, yet the strict requirements on appropriate input data limit the range of platforms and datasets it can be applied to. Cartographer and ALOAM perform relatively well in Urban, showing the effectiveness of edge and plane features as well as grid methods in human-made environments.

For dense alignment methods, BLAM perfoms adequately in the Urban datasets, but poorly in the Safety Research dataset, suggesting the WIO integration employed by LOCUS is an important component in the self-similar and low lidar observability conditions seen in that dataset.

Map error As second quantitative evaluation, we compare the maps obtained with each algorithm to the DARPA provided ground-truth map to compute the overall cloud-to-cloud error. To account for potential calibration misalignments, we run Iterative Closest Point (ICP) between the reconstructed map and ground-truth map before performing error analysis. We report in Table 4.3, a numerical summary of the RMSE values of the map error (ME) computed for each algorithm on each relevant dataset. The results show similar trends to the position error, with some negligible differences in the order of the algorithms.

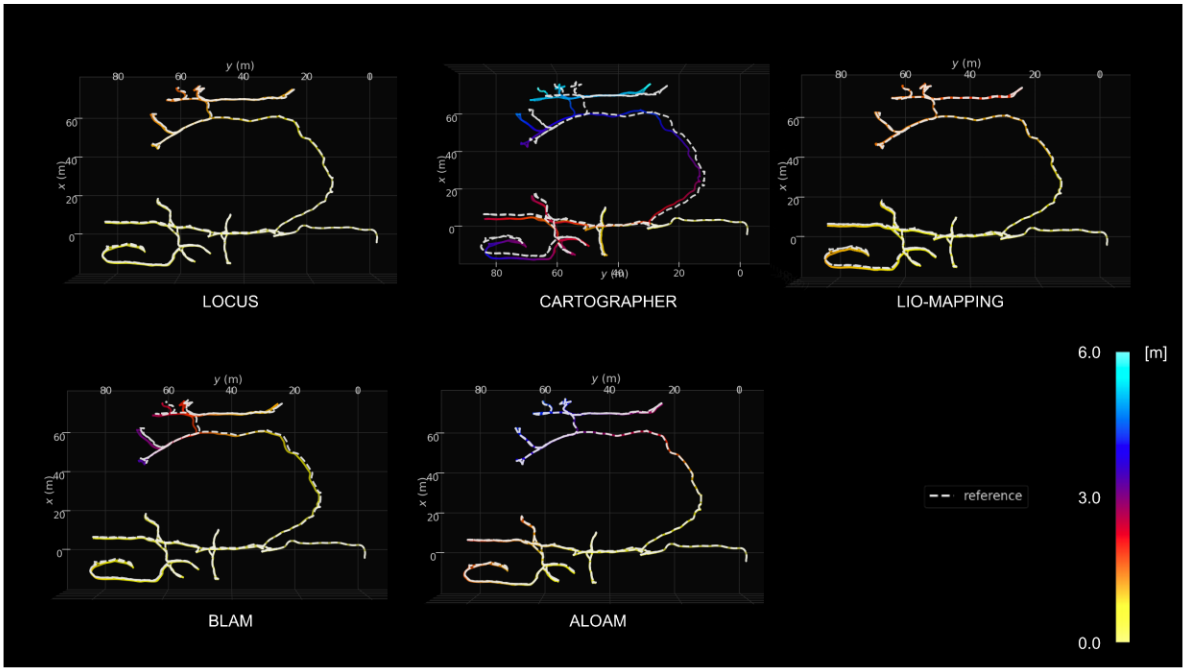


Figure 4.8: Absolute Position Error visualization for different algorithms running on the Alpha one dataset from the Urban Circuit of the DARPA Subterranean Challenge.

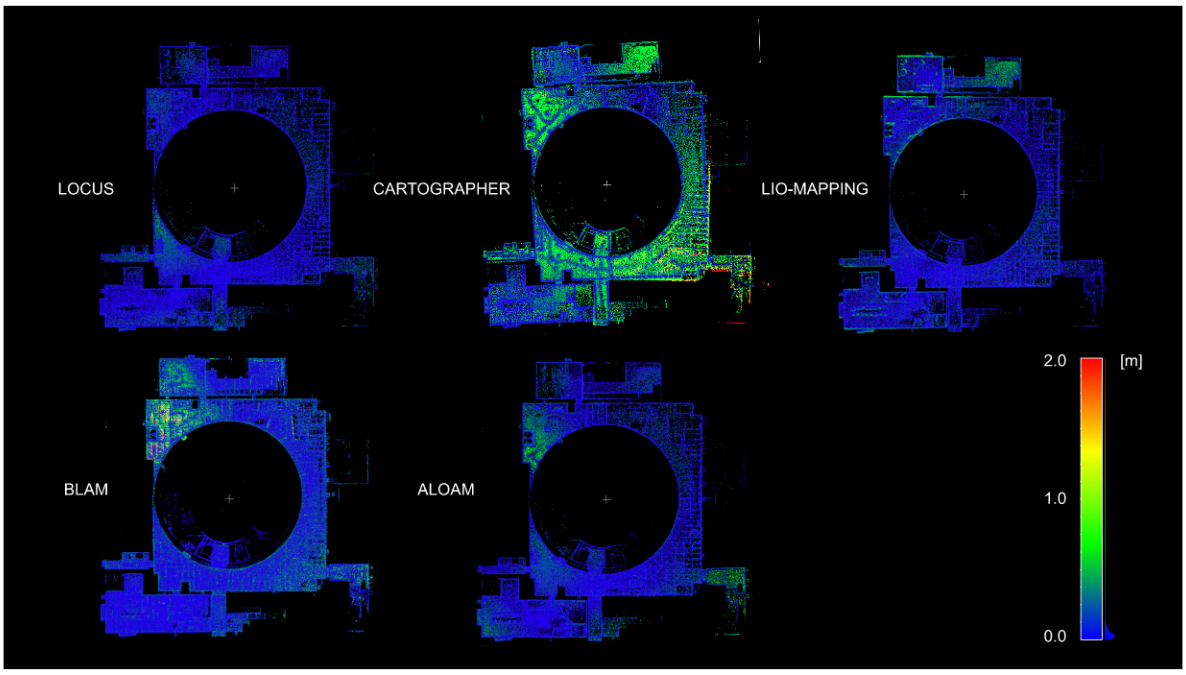


Figure 4.9: Map Error statistics for different algorithms running on the Alpha one dataset from the Urban Circuit of the DARPA Subterranean Challenge.

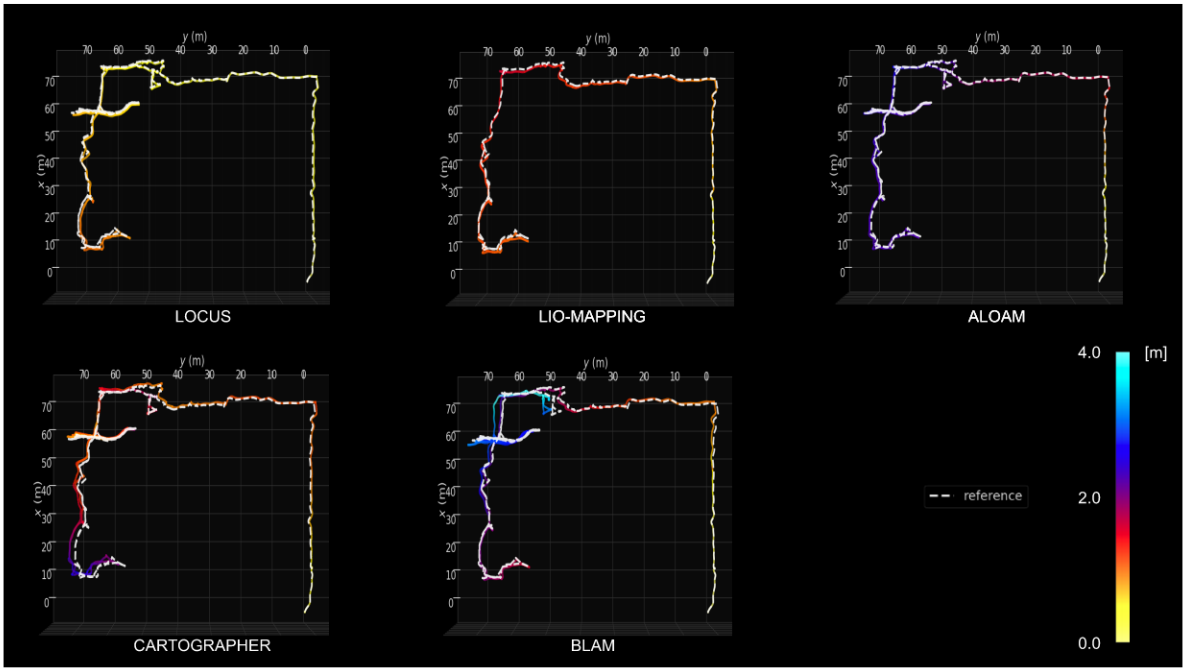


Figure 4.10: Absolute Position Error visualization for different algorithms running on the Beta two dataset from the Urban Circuit of the DARPA Subterranean Challenge.

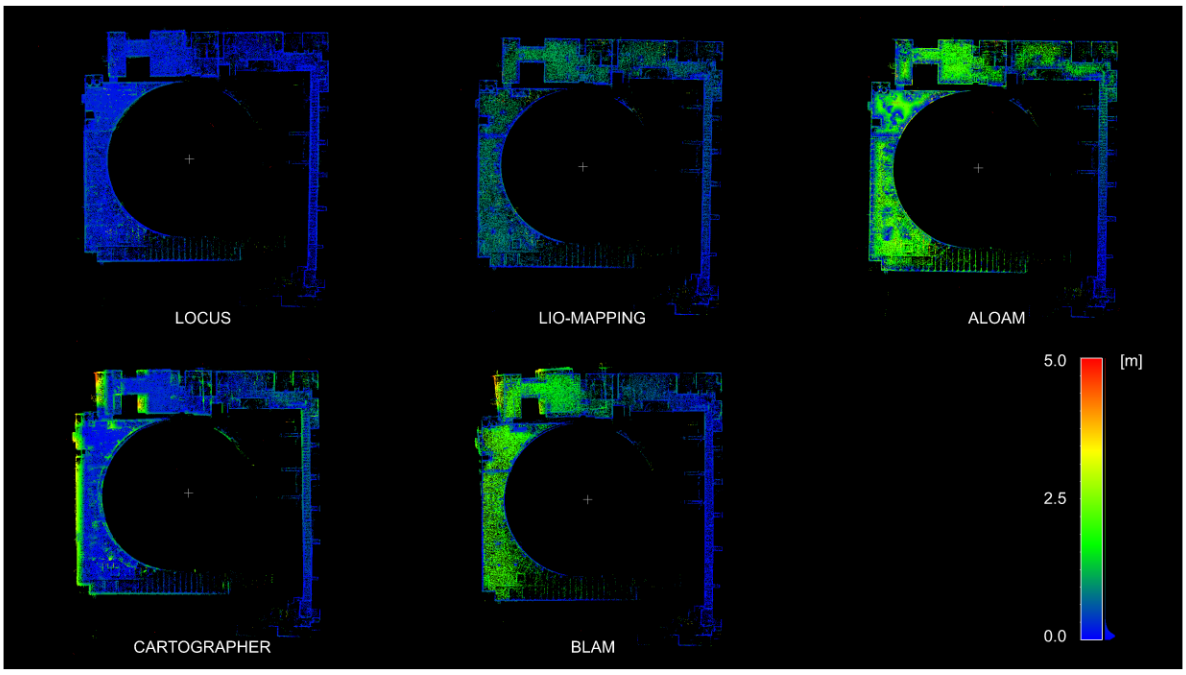


Figure 4.11: Map Error visualization for different algorithms running on the Beta two dataset from the Urban Circuit of the DARPA Subterranean Challenge.

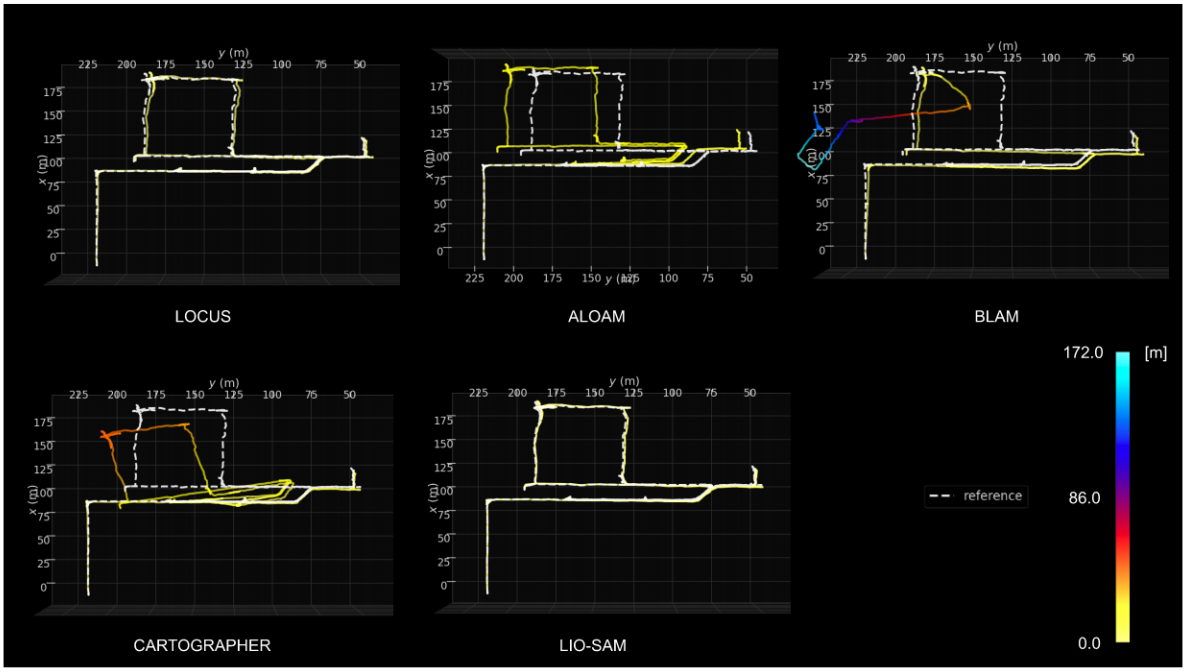


Figure 4.12: Absolute Position Error visualization for different algorithms running on the on the Safety Research dataset from the Tunnel Circuit of the DARPA Subterranean Challenge.

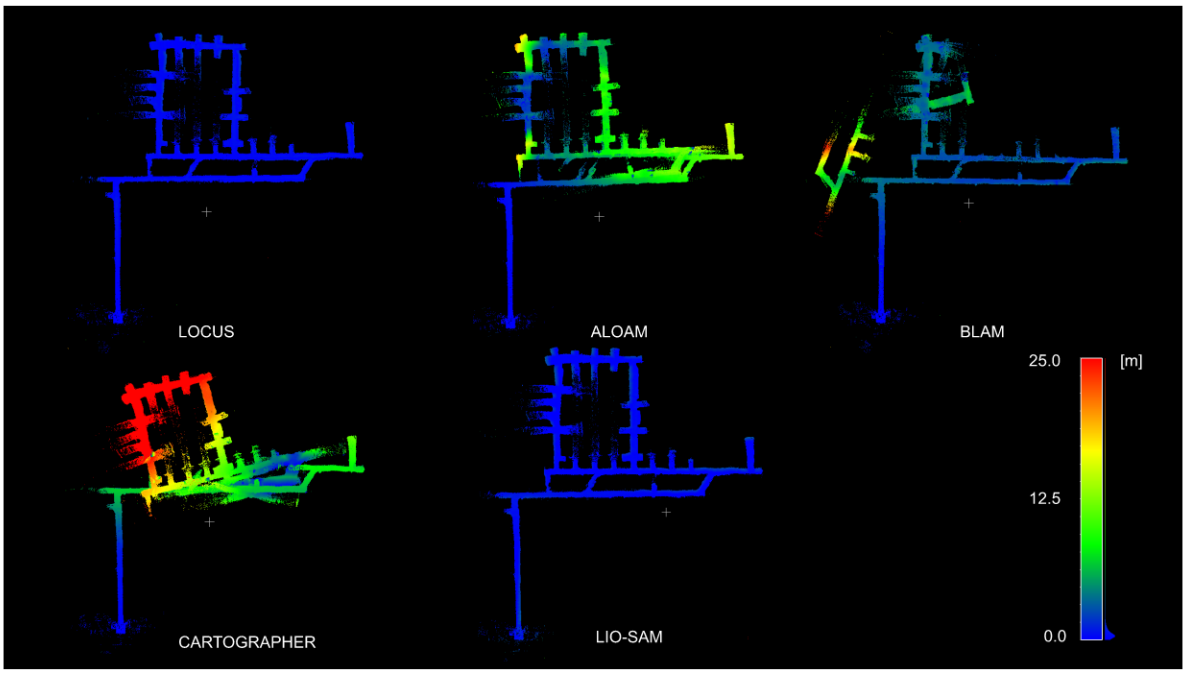


Figure 4.13: Map Error visualization for different algorithms running on the Safety Research dataset from the Tunnel Circuit of the DARPA Subterranean Challenge.

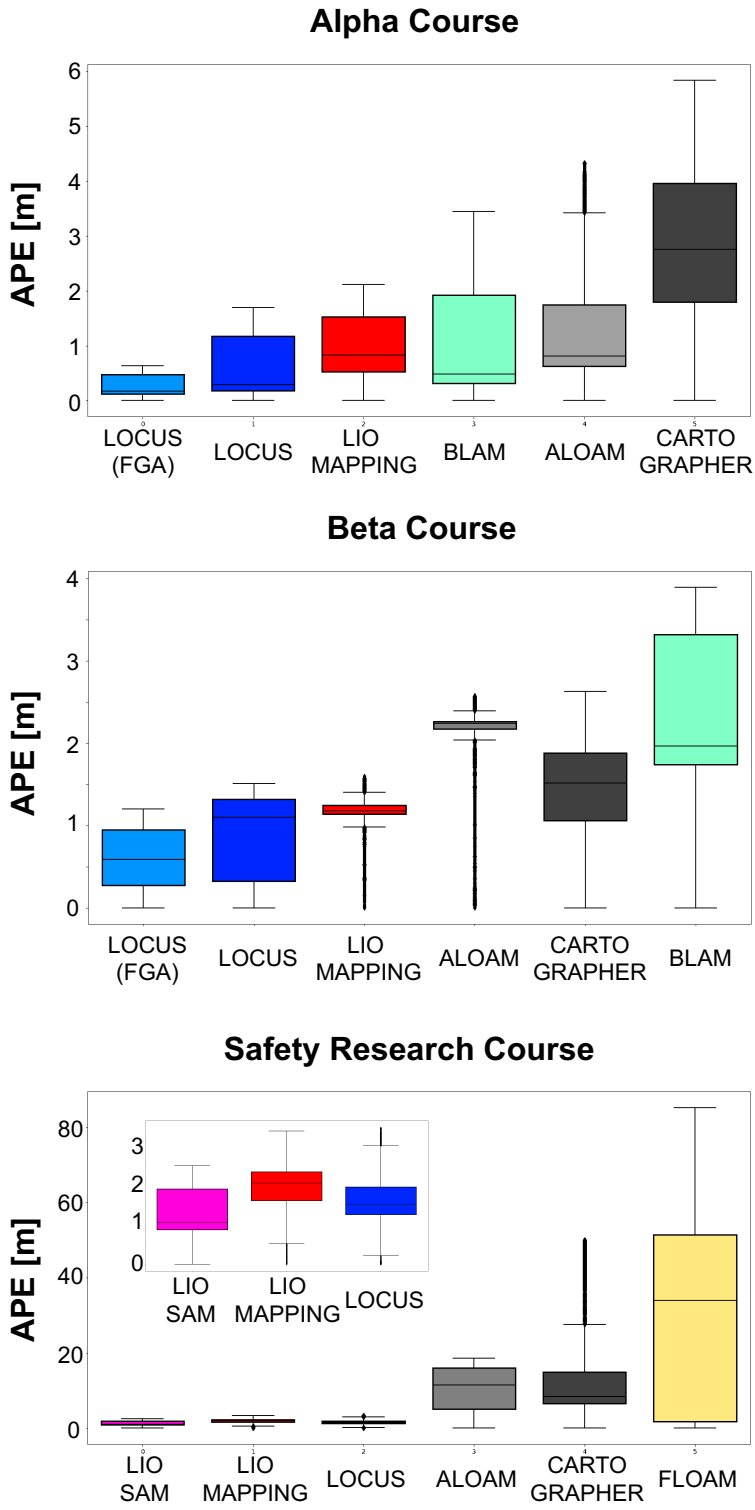


Figure 4.14: Boxplot visualization of the Absolute Position Error (APE) computed for the different methods on the test datasets. For clarity, only the best six algorithms in each dataset are shown.

Table 4.2: Summary of Accuracy Analysis results on Alpha and Beta Courses from Urban Circuit

Algorithm	**Alpha Course**				**Beta Course**			
	APE [m]			ME [m]	APE [m]			ME [m]
	max	mean	std	RMSE	max	mean	std	RMSE
LOCUS	1.69	0.62	0.57	0.29	1.51	0.88	0.51	0.69
LOCUS_FGA	**0.63**	**0.26**	**0.18**	**0.28**	**1.20**	**0.58**	0.39	0.48
BLAM	3.44	1.01	0.94	0.43	3.89	2.27	0.89	1.27
ALOAM	4.33	1.38	1.19	0.60	2.58	2.11	0.44	0.99
FLOAM	29.49	9.19	8.96	1.73*	40.64	3.94	8.42	3.73*
Cartographer	5.84	2.91	1.60	1.05	2.64	1.37	0.67	**0.31**
LIO-Mapping	2.12	0.99	0.51	0.45	1.60	1.18	**0.22**	0.61

Table 4.3: Summary of Accuracy Analysis results on Safety Research Course from Tunnel Circuit

Algorithm	**Safety Research Course**			
	APE [m]			ME [m]
	max	mean	std	RMSE
LOCUS	3.39	1.67	0.76	0.63
BLAM	171.34	35.45	51.91	5.37
ALOAM	18.61	10.01	6.01	6.11
FLOAM	85.31	32.49	25.73	20.16
Cartographer	50.05	14.31	13.45	14.25
LIO-Mapping	3.31	1.99	**0.55**	0.76
LIO-SAM	**2.45**	**1.26**	0.58	**0.52**

Robustness Evaluation

The previous section highlighted the robustness to low lidar observability, substantial vibrations, large accelerations and self-similar environments through the accuracy results on the datasets. In this section we focus on another aspect of robustness: the ability to handle a sudden failure of an input source. Specifically, we test the following scenarios: *i)* failure of WIO/IMU, *ii)* failure of WIO, *iii)* failure of lidar. Each of these failure scenarios have been experienced in real field tests in preparation for the SubT Challenge. We artificially create these failures in our datasets to have a controlled way of isolating the source of the failure, and the resulting impact on the algorithms. The results are summarized in Table 4.4, with example maps resulting from different failure modes shown in Fig 4.15.

Table 4.4: Summary of Robustness Test Results

	Robustness Test Result		
Algorithm	a) WIO/IMU Fail	b) WIO Fail	c) Lidar Drop
LOCUS	OK	OK	OK
BLAM	NA	NA	Errors
ALOAM	NA	NA	Errors
FLOAM	NA	NA	Errors
Cartographer	Stops	Stops	Errors
LIO-Mapping	Stops	NA	Errors
LIO-SAM	Stops	NA	Errors

OK: negligible degradation in accuracy. **NA:** Not Applicable - the algorithm does not use that sensor source. **Errors:** substantial errors in accuracy. **Stops:** no more odometry output after failure.

WIO/IMU failure We simulate sensor failure after 1200s, by shutting down WIO and IMU streams for the rest of the run. This failure only affects those algorithms that use IMU, where the algorithms cease to run, either relying on a synced callback with WIO and IMU (e.g. Cartographer) or relying on pre-integrated IMU to provide odometry updates between scans as well as initial scan to map alignment estimates (LIO-Mapping, LIO-SAM). In contrast, LOCUS processes the input data separately, and hence can automatically switch from lidar odometry with WIO integration, to lidar odometry with IMU integration, to pure lidar odometry, demonstrating robust handling of sensor failures in a cascaded fashion. This behavior is highly desirable to accommodate the unforeseen challenges posed by rough terrains in real-world applications where hardware failures are likely to happen, or sensors sources can become unreliable (e.g. dark areas with no visual texture for VIO).

WIO failure We simulate a loss of WIO after 1200s. Cartographer and LOCUS are the only algorithms affected, with the same result as the WIO/IMU case.

Lidar failure We stress the systems by subtracting the most fundamental data source: lidar. More specifically, we simulate a 10s gap in lidar data while the robot is in motion. There are three responses to this failure. The first response is that the algorithm stops running until the lidar returns, resulting in large map errors (BLAM, ALOAM, FLOAM, and Cartographer due to the synced callback). The second response is that the algorithm runs purely on IMU integration, leading to an accumulation of

drift before the lidar returns (LIO-Mapping, LIO-SAM). The final response is only seen by LOCUS, where the loose coupling allows WIO to accumulated over the 10s of no lidar data to produce an accurate initial transform when the lidar returns.

Overall, LOCUS consistently achieves reliable ego-motion estimation and mapping, demonstrating efficient handling of sensor failures in a cascaded fashion, behaviour that is desirable to accommodate the unforeseen challenges posed by real-world operations where hardware failures are likely to happen, or sensor sources can become unreliable.

Efficiency Evaluation

We profile the time needed from the different algorithms to process a single lidar scan when running the algorithms on an Intel Hades Canyon NUC8i7HVKVA (4x1.9 GHz, 32 GB RAM) running Ubuntu 18.04 LTS. Fig. 4.16 shows the resulting times per scan with scans at 10Hz (LIO-Mapping is omitted as the processing time, 1s per scan, is too large). Additionally, Table 4.3 shows the CPU loads for each algorithm. All values are from the Urban Beta dataset, except for LIO-SAM, which is on the Tunnel Safety Research dataset.

LOCUS, Cartographer and BLAM can all maintain real-time processing, whereas ALOAM and LIO-SAM can only stay real-time with a lower rate of lidar scans. LIO-SAM can use the IMU pre-integration to cope with a lower IMU rate, and by using features, ALOAM can also handle a lower rate for certain datasets. Both FLOAM and LIO-Mapping do not appear to be feasible for real-time operation. Cartographer has both the quickest processing time and the lowest CPU load, yet the accuracy is not as strong as the other algorithms. LOCUS produces the best accuracy, with real-time performance, yet requires the largest CPU load.

4.2.3 Real-Time Operation Across Different Platforms

In this section, we demonstrate real-time field operation of LOCUS on different robotic platforms during the Urban Circuit of the SubT Challenge and provide statistics from logged online operation. Results come from the four competition runs, two in Alpha course and two in Beta course.

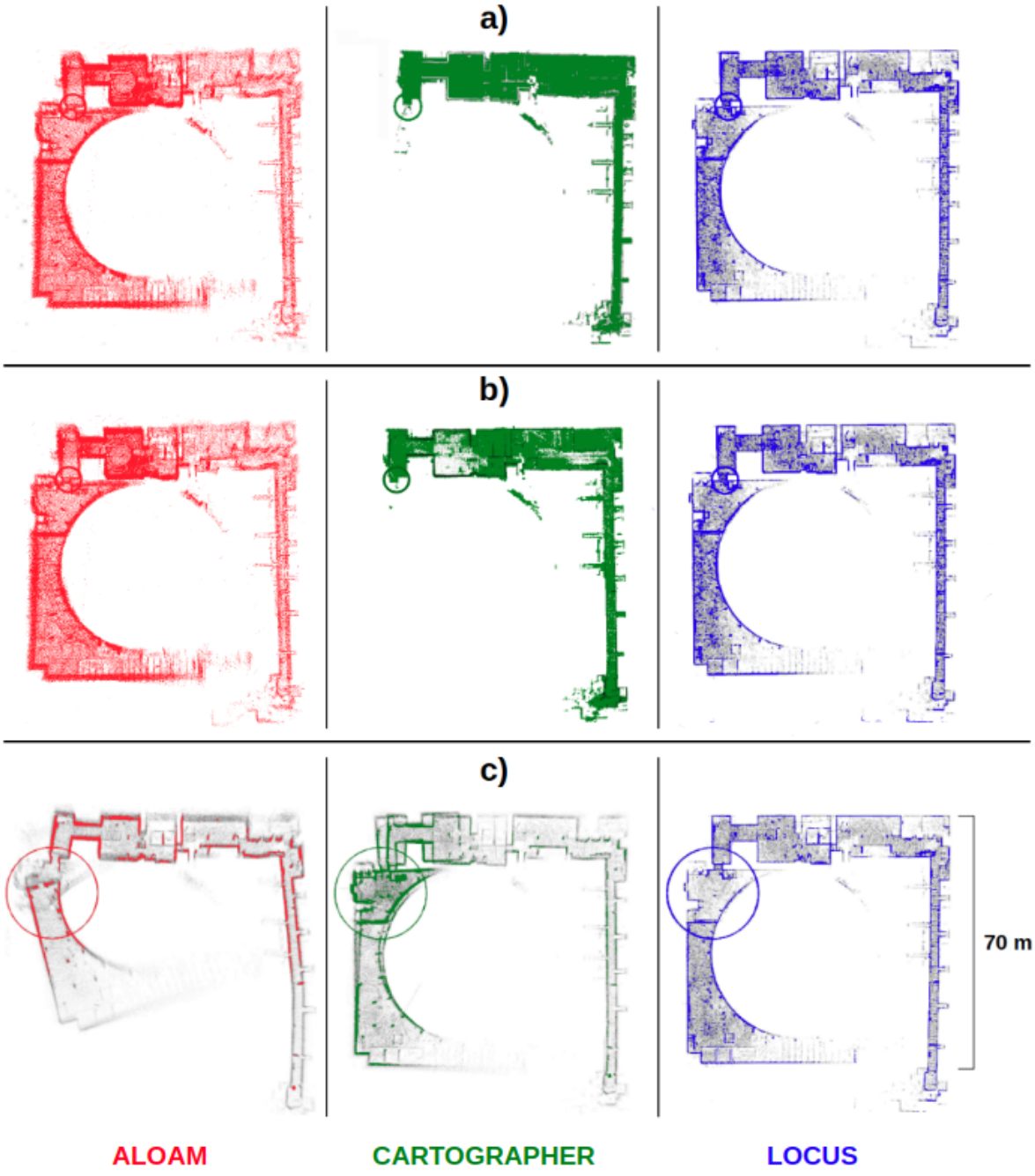


Figure 4.15: Robustness test in Beta course: a) results on WIO/IMU failure, b) results on WIO failure, c) results on Lidar failure. The failure locations are circled in all cases.

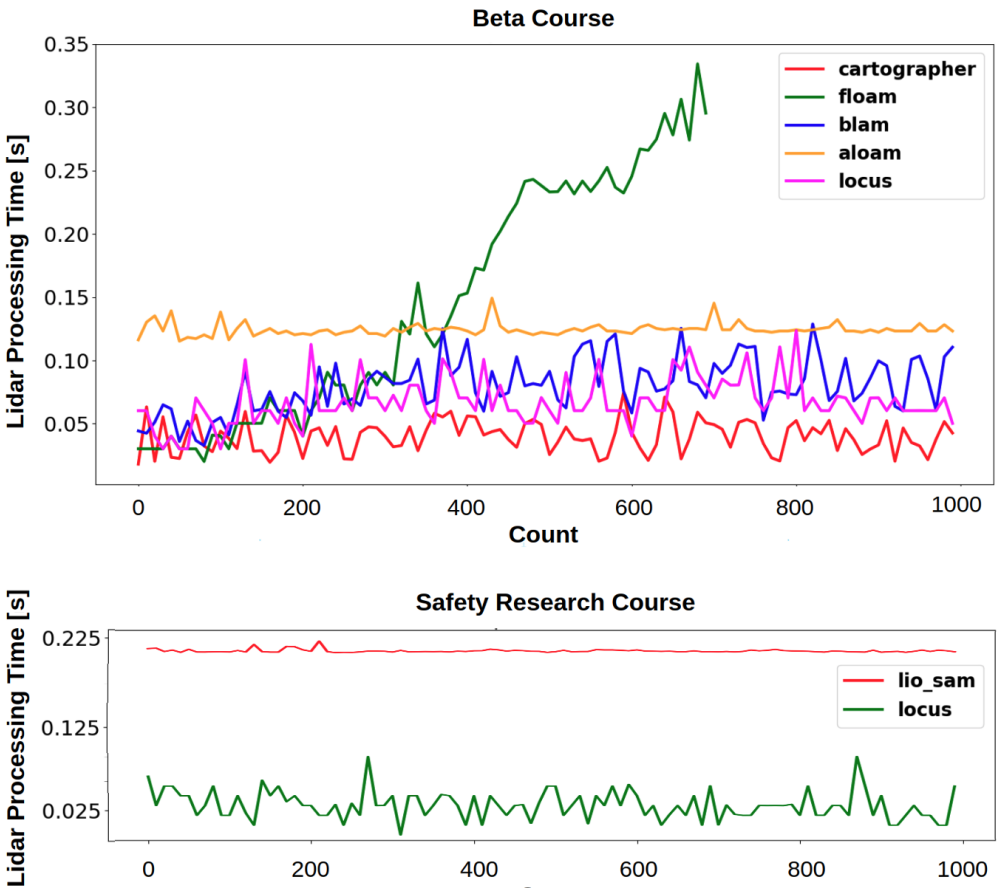


Figure 4.16: Comparison of lidar processing time across the different lidar odometry algorithms. The times are the duration for processing a single scan. Top - Urban Beta dataset, Bottom - Tunnel Safety Research dataset. A processing time of 0.1 s indicates realtime performance (10 Hz scans).

Hardware and Tuning

During the competition, we deployed LOCUS on two very different robots: *i)* the Husky from the datasets used above (see Fig. 4.1.b), and *ii)* Spot from Boston Dynamics (see Fig. 4.1.d).

Husky In addition to the sensors described in Sec. 4.2, Husky carries an AMD RYZEN 9 3900X 12-Core 3.8 GHz for computation.

Spot A legged robot that is equipped with 1 VLP16 and an Intel NUC7i7DN 4-Core 1.9 GHz for computation. Both VIO and KIO are available from the Boston Dynamics API, and can be used for integration into LOCUS. We choose VIO as it was shown to be more accurate than KIO in our tests.

Adaptation The paramaters of LOCUS are tuned to achieve accurate and robust real-time operation on both platforms while accounting for differences in computational capabilities and hardware configurations. Table 4.5 summarizes the configurations used during the competition.

Table 4.5: Summary of LOCUS settings on different robots

Parameter	**Husky**	**Spot**
Number of lidars	2	1
Voxel Grid Filter leaf size (m)	0.1	None
GICP iterations in scan-to-submap	20	25
GICP number of cores	4	1
Sensor Integration	WIO	VIO

Performance

We report in Table 4.6 the average value of the number of lidar scans dropped per second by each robot in each course of the competition during real-time operation. Point clouds are subscribed to at 10hz and we do not buffer any lidar scans, to minimize the delay of the computed odometry. Therefore, the number of dropped scans per second represent how frequently the lidar processing time exceeded 0.1s. Spot drops 2 scans a second, due to the less powerful computer onboard. However, Spot has a more accurate additional

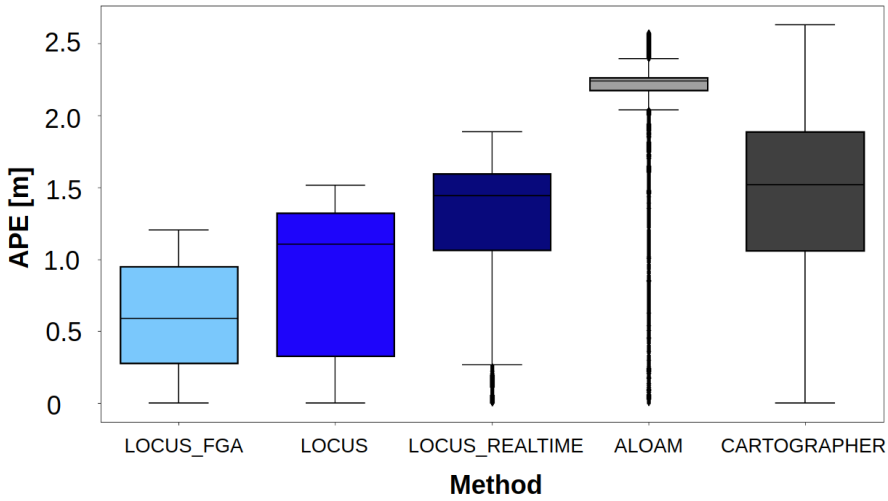


Figure 4.17: Absolute Position Error (APE) of the trajectories estimated by the different methods against ground-truth in Beta course, including the performance when running live in the SubT Challenge (LOCUS_REALTIME).

odometry source than Husky, with VIO. The reliable initial transform from VIO enables Spot to process fewer scans per second, and still perform acceptably.

Table 4.6: Dropped lidar scans from real-time on-robot tests

-2*Robot	Number of dropped scans / s				
	Alpha 1	Alpha 2	Beta 1	Beta 2	Average
Husky	0	0	0	0	0
Spot	2.082	2.205	1.833	2.016	2.034

Real-time accuracy profiling LOCUS performed accurately for both Husky and Spot in the competition, as evident in the overall team's performance, winning first place[10]. Fig. 4.17, shows the live performance of LOCUS with FGA on the Husky in Beta 2, with a slightly larger error than the post-processed results (on a different computer), yet still highly competitive.

For Spot, LOCUS was run live in a multi-level exploration in the Urban Alpha 2 course. In this run, the mean APE was 0.586 m, and the maximum APE was 2.599 m, which was sufficiently small for scoring in the competition.

[10]The LOCUS output was integrated with a robust odometry aggregator [154], and then fed to a back-end SLAM algorithm [84].

4.2.4 Discussion

Overall, fusion of additional sensing modalities is crucial to enable accurate operation in extreme robotic explorations: by relying on a loosely-coupled mechanism, LOCUS is robust to potential failures of sensors, and can achieve improved performance with respect to tightly-coupled approaches in settings where the extrinsic sensors calibration is not ideal. While being very accurate and robust, a possible avenue for future improvements of LOCUS is represented by decreasing its computational cost, as it currently represents a challenging bottleneck for its deployment over platforms operating under severe computation constraints.

4.3 Conclusions

Achieving accurate lidar odometry in perceptually-challenging conditions can be difficult due to the lack of reliable perceptual features, presence of noisy sensor measurements, and high-rate motions. While integrating additional sensing modalities can help address these challenges, potential sensor failures can have dramatic impacts on the mission outcome if not robustly handled. We present a lidar odometry system to enable accurate and resilient ego-motion estimation in challenging real-world scenarios. The proposed system, LOCUS, provides an accurate multi-stage scan matching unit equipped with an health-aware sensor integration module for seamless loose integration of additional sensing modalities. The proposed architecture is adaptable to heterogeneous robotic platforms and is optimized for real-time operation. We compare LOCUS against state-of-the-art open-source algorithms and demonstrate top-class accuracy in perceptually-challenging real-world datasets, top-class computation time and superior robustness to sensor failures, yet with greater CPU load. Finally, we demonstrate field-proven real-time operation of LOCUS on two different robots involved in fully autonomous exploration of Satsop power plant during the Urban Circuit of the DARPA Subterranean Challenge, where the proposed system was a key component of CoSTAR team's solution that achieved first place.

4.4 Ongoing Work

This section provides details of ongoing work not included in [150].

4.4.1 Open Space Detector

With voxel-based filtering the size of the output point cloud changes with the size of the environment surrounding the robot: for example, when moving from closed space to open space, the number of post-filtered points will greatly increase. Different cross sections of the environment can considerably affect the total lidar processing time in the scan-matching unit as illustrated in Fig. 4.18 and this might impose challenges for computationally-constrained platforms pursuing real-time operation.

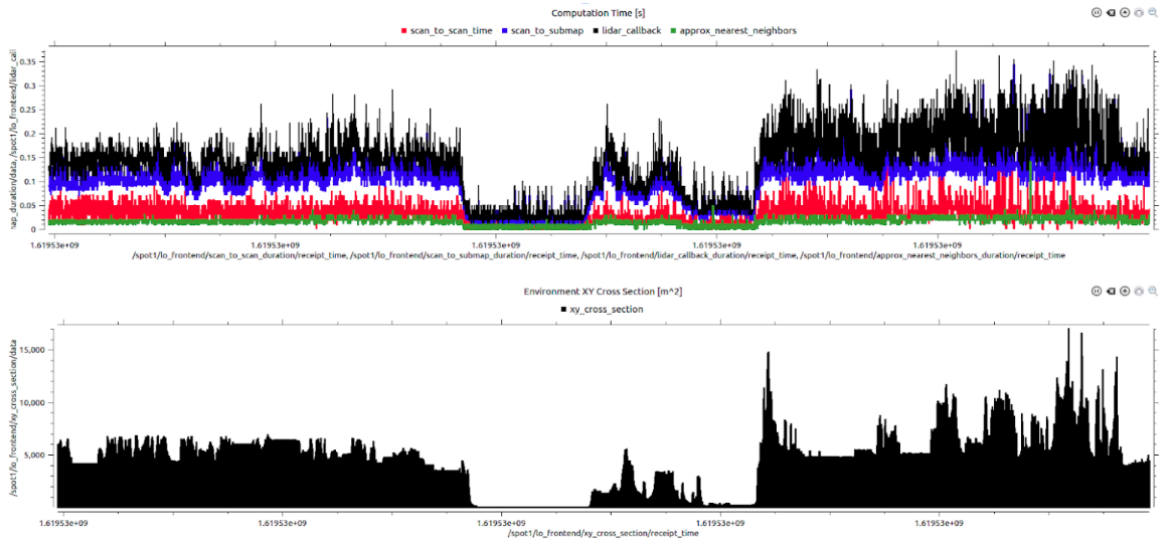


Figure 4.18: Influence of the environment x-y cross-section (bottom) on the total lidar processing time in LOCUS (top) for the multi-level exploration of spot1 robot in LA Subway. When entering open spaces, the greater number of points lead to greater processing times as the system needs to register a wider amount of information. Closed spaces instead are easier to handle, and just challenge the system on the observability layer. On the bottom plot is possible to distinguish four main areas of the exploration: an open-space, a corridor, a medium-space, a corridor, and finally another large open-space.

The Open Space Detector analyzes the size of the environment by looking at the x-y cross section of the current lidar observation: if this value exceeds a user-defined threshold and an open space is detected, this information can be used to apply additional downsampling to reduce the number of points before the first GICP operation, and cope with limitated computational capabilities on board to achieve real-time opera-

tion. Furthermore, detection of open space can also be exploited to dynamically adjust the keyframe policy addition, by for example adding points to map less frequently in open space, and more frequently in closed space. The presented capability becomes then a tuning parameter to trade off between accuracy and real-time requirements.

4.4.2 Map Sliding Window

In lidar based systems, a fairly unexplored question is how to store the map during long-term operations. If the global map is stored in memory for localization purposes, its size will monotonically increase over large-scale explorations leading to possible catastrophic consequences at the mission-level as the on-board RAM will not be able to store further data. This subsection presents two approach developed to keep under control the map size in memory and enable long-term explorations of large-scale environments on memory-constrained platforms. More specifically, we will discuss: *i)* a stationary-triggered map sliding window approach, *ii)* a multi-threaded map sliding window approach.

Stationary Approach The first attempt in enabling large-scale explorations under memory constraints consists in the development of a stationary-triggered map sliding window approach. This method monitors the translational and rotational velocity of the robot and waits to detect stationary conditions. When the stationary-status is detected, the map is refreshed with a robot-centered box-filter to retain only the robot-centered sub portion of the map in memory. When the point cloud map has been refreshed, the octree is reset, and reinitialized with the updated map. This approach results in a blocking operation for the lidar callback, and the robot needs to stand still until the refresh operation has finished.

From preliminary tests on the dataset collected in the Louisville Mega Cavern, KY, the stationary map sliding window approach decreases memory usage by 40 GB with respect to the classic mapper as shown in Fig. 4.20 with no evident consequences on accuracy. A possible shortcoming of this approach is that potential issues would be still faced if the robot never stops during the long-term exploration as the map would never be refreshed.

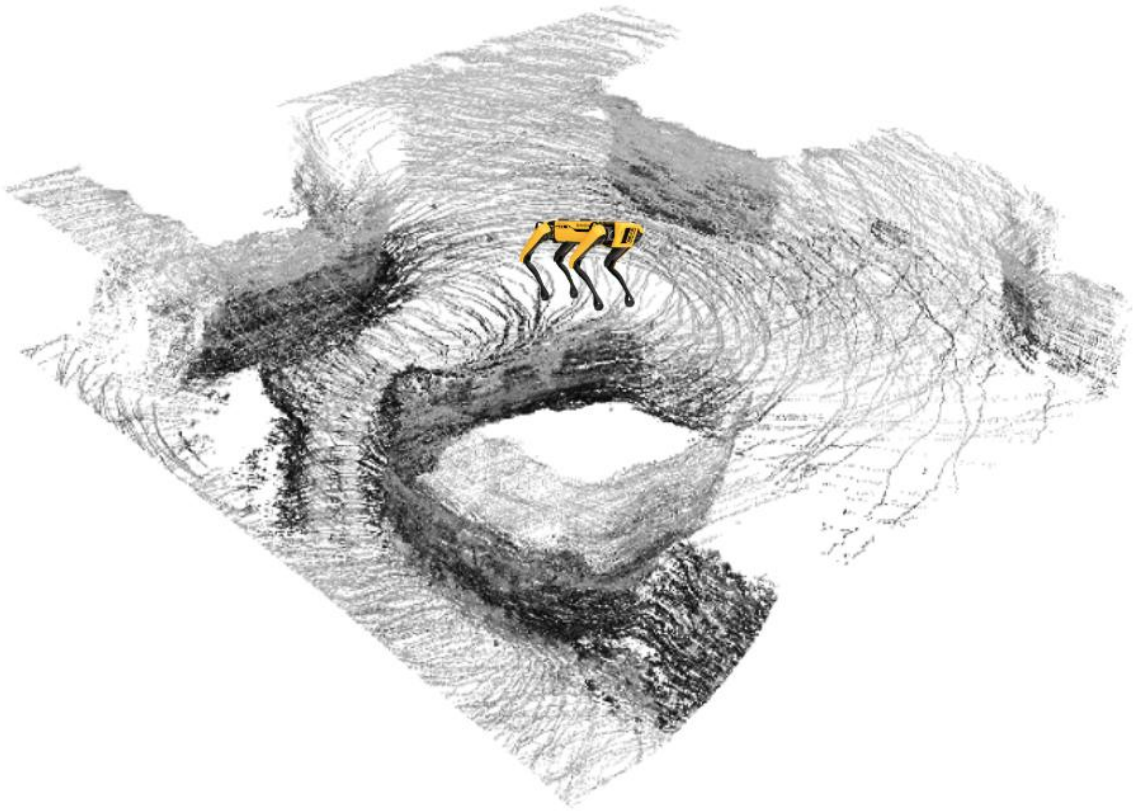


Figure 4.19: Stationary-triggered map sliding window testing on the Mega Cavern dataset. When the robot is standing stationary, the map/octree is refreshed to keep in memory only a robot-centered submap of the environment for front-end localization purposes.

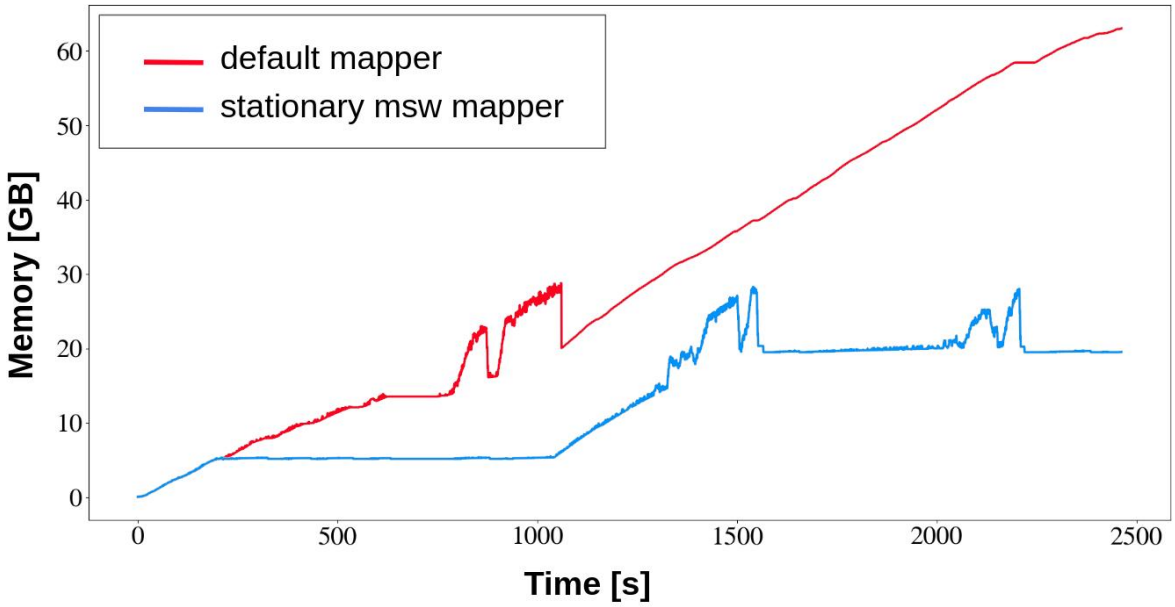


Figure 4.20: Size of map in memory for a large-scale exploration of the Mega Cavern dataset. In red, the classic mapper keeps the full map in memory resulting in 60 GB RAM usage at the end of the run. In blue, the stationary-triggered map sliding window approach refresh the maps every time the robot stops, leading to decreased overall memory usage (20 GB).

Multi-Threaded Approach Our second attempt in keeping under control the map size in memory consists in developing a multi-threaded map sliding window approach that: *i)* does not constitute a blocking operation for the lidar callback, *ii)* can be exploited while moving, without needing the robot to stop. In this approach, we exploit multi-threading techniques to always have two parallel working threads ($thread_a$ and $thread_b$) working on dedicated data structures for the maps (e.g. map_a, map_b) and for the octrees (e.g. $octree_a$, $octree_b$) to dynamically refresh the map size in memory while accounting for robot motions between parallel worker processes. Fig. 4.22 demonstrates the effectiveness of the proposed approach in limiting the map size in memory always under 1 GB throughout the large-scale exploration of the Mega Cavern dataset.

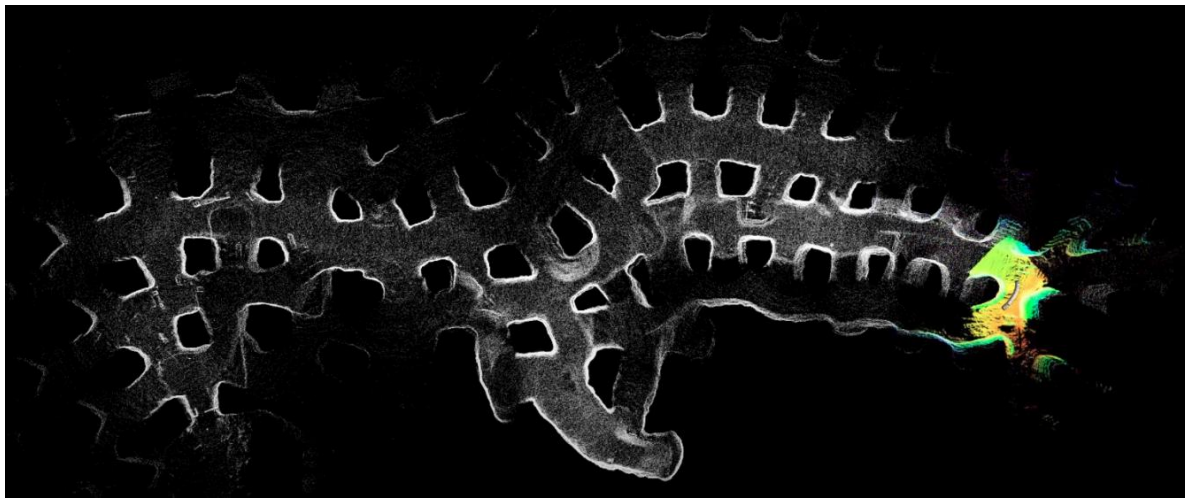


Figure 4.21: Multi-threaded map sliding window testing on the Mega Cavern dataset. Only a robot-centered submap (colored) is kept in RAM memory for front-end localization purposes.

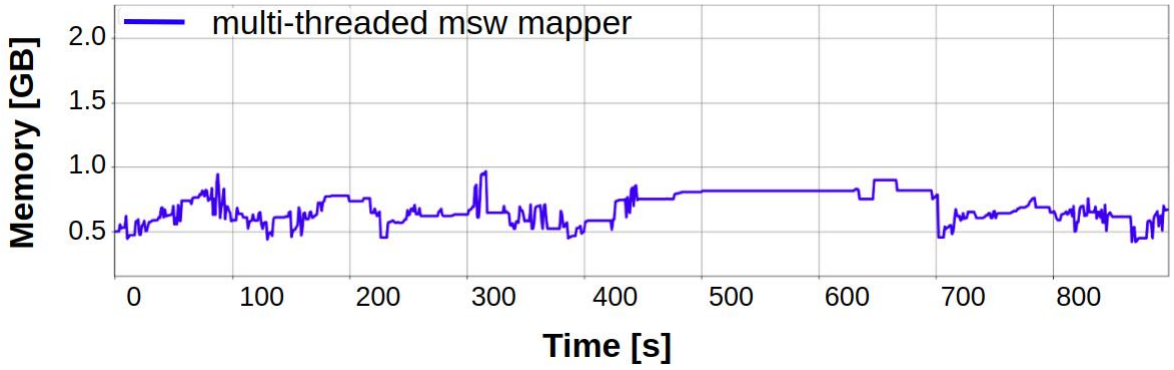


Figure 4.22: Multi-threaded map sliding window test in Mega Cavern dataset. From 60 GB when keeping the full map as in the standard published version of the system, this approach decreases the overall front-end memory usage to only 1 GB throughout the exploration.

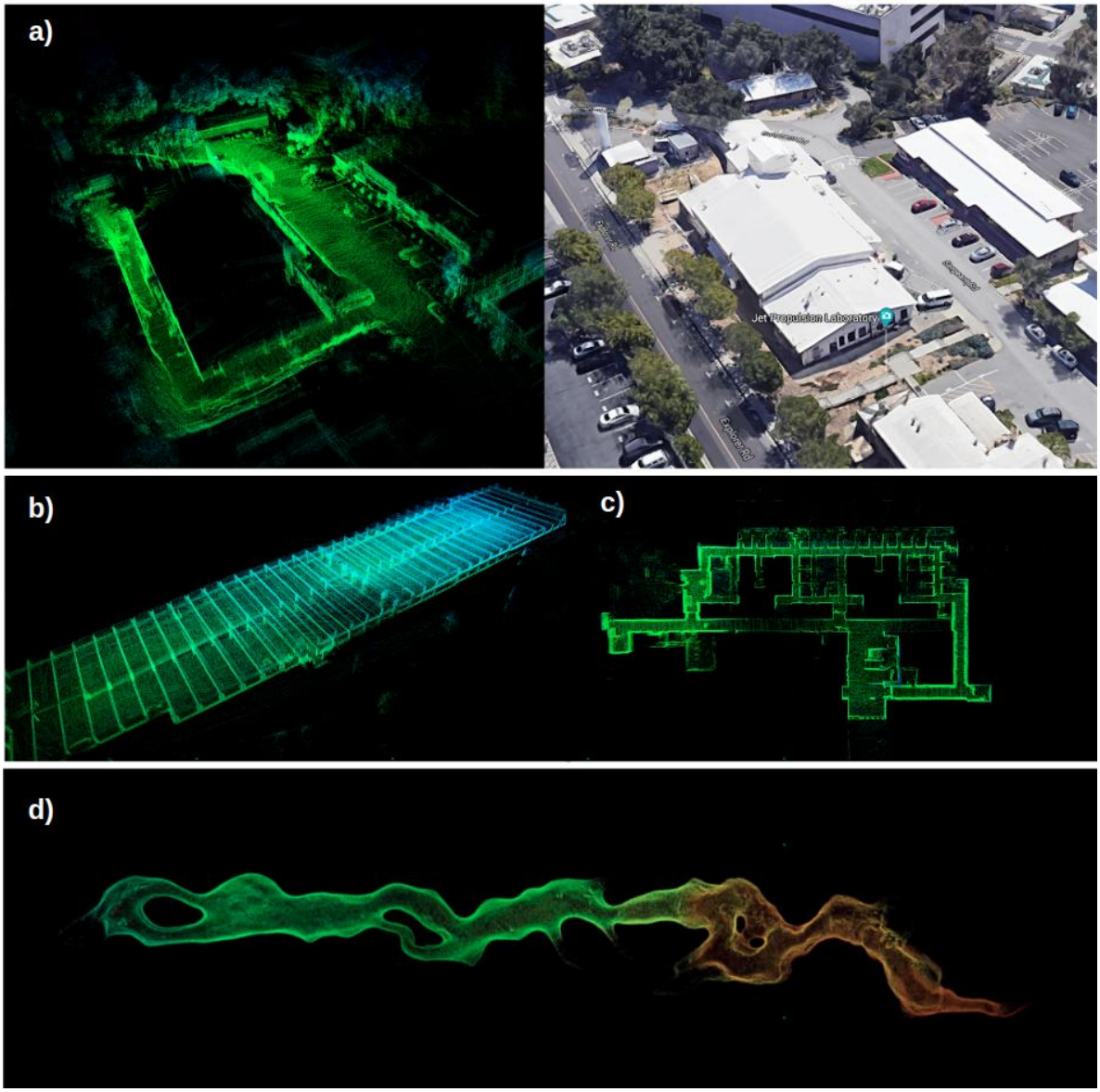


Figure 4.23: Example of high-definition 3D maps produced by LOCUS during autonomous robotic exploration of heterogeneous environments. a) Outdoor area at the NASA Jet Propulsion Laboratory. b) A multi-level parking lot at the NASA Jet Propulsion Laboratory. c) and indoor office at the NASA Jet Propulsion Laboratory. d) A natural cave network.

4.5 LAMP back-end

LOCUS is a sub-component of the broader NeBula's SLAM solution referred to as LAMP (Large-scale Autonomous Mapping and Positioning) [84]. LAMP is a factor-graph based SLAM solution, with: a) an adaptable odometry input that can process individual (e.g. LOCUS, VIO) or fused odometry sources (e.g. HeRO), b) a multi-modal loop closure module, based on lidar [84], visual [48] or semantic features [75], c) an outlier-resilient optimization of the factor graph, including multi-sensor inputs. LAMP achieves low drift, multi-robot, multi-sensor SLAM over large scales in perceptually-degraded conditions, producing a consistent global representation of an unknown environment by fusing information retrieved by individual robots into a comprehensive map with globally-referenced semantic information as showed in Fig 4.24. The resulting 3D semantic map constitute the critical output data product for the overall situational awareness retrieved by the multi-robot system. We refer the reader to [23] for a detailed description of LAMP.

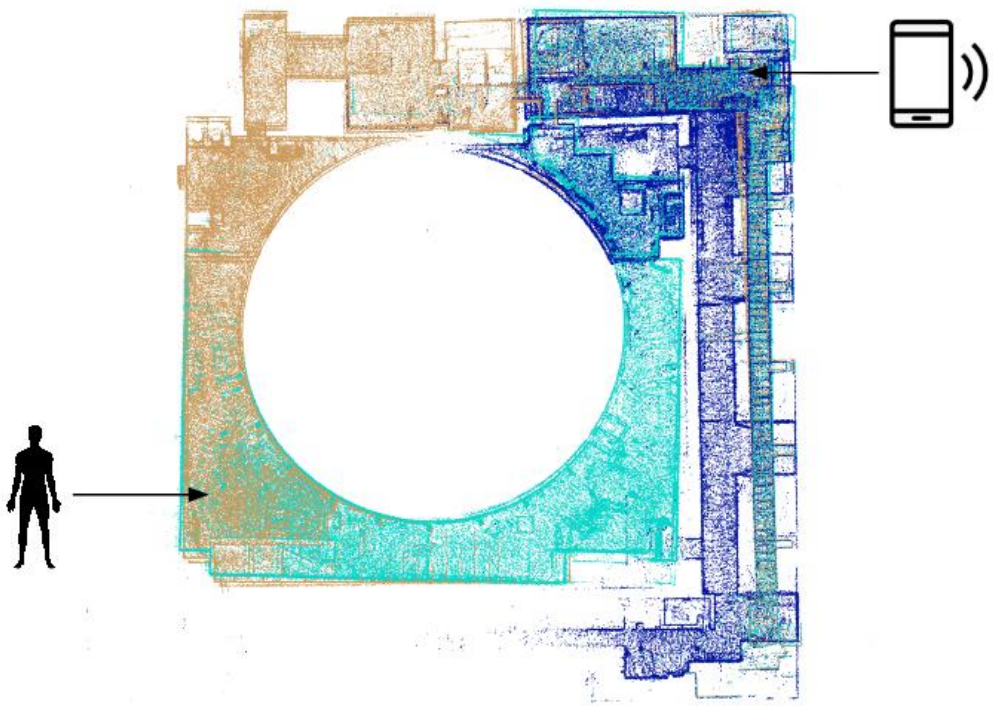


Figure 4.24: Multi-robot LAMP map with robot-specific coverage information. In clear blue husky1, in brown husky4 and in dark blue spot1. Examples of artifact positions in the global map are reported for detections of phone and survivor.

Chapter 5

Swarm Manager

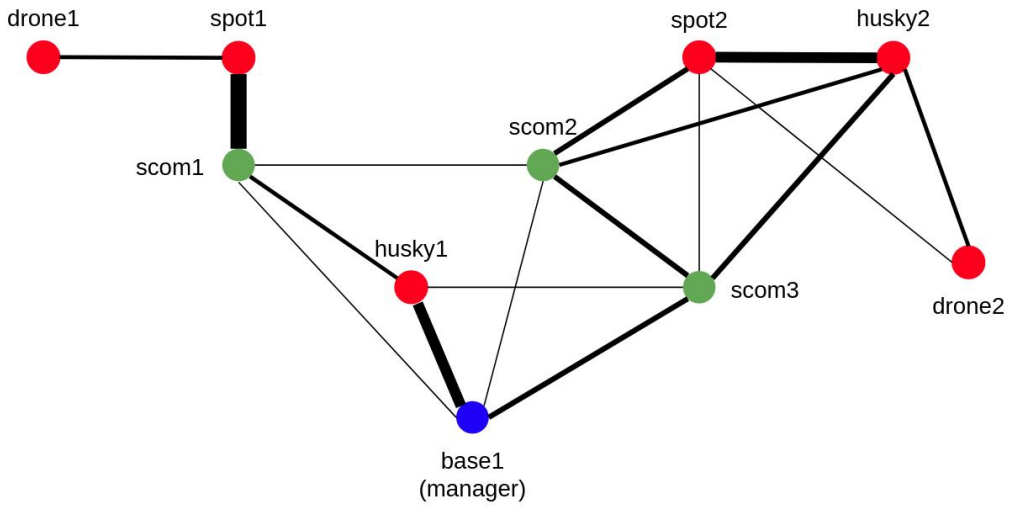


Figure 5.1: Swarm Manager's graph formulation: the world is composed of a set of nodes and edges. Nodes can be mobile agents (red) or static assets (green) such as the base station or communication nodes. Edges represent communication links between nodes in the world, and they are graphically rendered with line thickness proportional to the available bandwidth on the communication channel. A node in the world can be initialized to be Manager at mission start (blue).

To address the problem of degraded situational awareness accuracy on computationally-constrained platforms in heterogeneous multi-robot systems operating in unknown environments, we introduce Swarm Manager. The proposed framework exploits a combination of Distributed Computation and Software Defined Networking paradigms to enhance the performance of robotic teams. Swarm Manager provides a globally-aware and centrally-supervised management strategy to handle computation offloading requests generated by computationally-constrained robots in the team while optimally allocating computation and communication resources.

The underlying idea is to offload computationally-heavy perception tasks (e.g. lidar odometry, object detection) from computationally-constrained platforms to more powerful peers in the robotic cluster: a more powerful machine will process more information and in a more accurate way, enhancing therefore the overall reliability of the situational awareness retrieved by the agent which requested the offloading.

At a fixed rate r (1 Hz in our setup), all agents in the network communicate status updates to the central entity so the Manager can update its knowledge of the world model. When receiving a computation offloading request generated by a robot in the team, Swarm Manager exploits its updated knowledge of the world model to identify, among all machines in the system, the optimum server available for computation. Once the optimum server is identified, Swarm Manager exploits Software Defined Networking techniques to choose among all possible paths in the network, the optimum route to instantiate the client-server communication. This mechanism is operated sequentially for all requests received at time t, achieving therefore optimal computation and communication allocation in multi-client scenarios.

An essential feature of the proposed framework consists in overcoming the well-known point of failure of centrally-supervised systems, where the orchestration mechanism provided by the central supervisor would become unavailable in the event of its failure. To address this problem, a dynamic Manager election mechanism is provided, which makes the system resilient to potential failures of the central entity as other dynamically elected machines can take care of the supervision role on failure of the central supervisor.

While the proposed framework could potentially be exploited for *generic* computation offloading, this work focuses on the management of perception-related computation offloading requests as the goal of the work is to enhance three main aspects of the performance of computationally-constrained platforms: *i)* localization accuracy, *ii)* detection precision, *iii)* autonomy time.

We assume to have some ground robots equipped with powerful computers and batteries that are specifically deployed in the mission to serve as computational assets for other robots needs, however, the proposed framework can work also in the absence of the above mentioned assumption.

In the following sections, we describe the fundamental components of the proposed framework, and pair them with their respective ROS [**ros**] implementation to provide a global overview of the system. First, we describe the entities of interest (e.g. nodes, edges) and present the types of messages that standardize inter-agent interactions. Then, we provide an overview of the different type of interactions between the entities in the system that are designed to achieve the overall mission goal.

5.1 Entities

We formulate the problem of multi-robot computation and network management by means of graph theory. The world is modeled as a graph G composed of a set of N nodes (agents) and M edges (communication links).

5.1.1 Nodes

A node represents any device capable of doing computation. A node in the world can be both a static asset (e.g. base-station, dropped communication node) or a mobile agent (e.g. ground-rover, legged-platform, aerial-robot). Each node might have different computation capabilities (e.g. high/medium/low), be equipped with CPUs or GPUs of different nature, and generally be characterized by a set of other generic user-definable features (e.g. power-efficiency, hardware-specifications etc.). Each node i in the graph G is characterized by a state S_i^t at time t given by:

$$S_i^t = \{P_i^t, B_i^t, C_i^t\} \tag{5.1}$$

where: *i)* $P_i^t \in SE(3)$ represents the pose (x, y, z, roll, pitch, yaw) of the node i at time t in the world frame $\mathcal{W}$, *ii)* B_i^t represents the battery level of the node i at time t, *iii)* whereas C_i^t represents the list of communication links and associated communication bandwidths to other nodes in the network that are available on the node i at time t. The computational capability of an agent is instead constant and does not vary over time, with wheeled platforms having *high* computation, legged platforms having *medium* computation, and aerial platforms having *small* computation.

By targeting mobile multi-robot systems, the position of nodes in the network associated to mobile agents dynamically evolves over time, and we do not make any assumption on a-priori known robots motion. From a communication point of view, all nodes in the graph act as logical switches and are therefore capable of forwarding data to other nodes in the MANET.

Roles A node in the graph can have different *roles* at different times, and at any given point in time the node acts accordingly to what its role is. The different roles are briefly described below:

- **Default** A node in the graph that only does its own processing. This is the initial role for all agents in the graph at mission start.
- **Client** A node in the graph that generates a computation offloading request to be handled by the Manager. Offloading requests might be generated for different kinds of computation (e.g. lidar odometry, object detection), at different times in the mission, whenever a mission-level criteria is met (e.g. constrained computation on-board limiting accuracy of situational awareness reports, critical battery level, degraded observability)
- **Server** A node in the graph that does its own processing, and also serves as computational asset for further processing required by computation offloading requests generated by a set of clients.
- **Manager** A node in the graph that does its own processing, and serves as central orchestrator for the multi-robot system. The Manager owns a centralized knowledge of the global world model, namely the status of agents and communication links at time t. This enables the Manager to handle computation offloading requests by taking decisions that are globally-aware. The Manager might also act as Server for some computation offloading requests. At mission start, the Manager could be either: *i)* elected by a human operator, *ii)* elected by the autonomous swarm by broadcasting unit status (e.g. computation capability, battery level, agent position, connections) to retrieve the most connected and powerful machine in the graph to be the Swarm Manager.

Battery Drain The battery drain of mobile agents is modeled with two main components: *i)* background-drain, *ii)* foreground-drain. The background drain represents the battery drain component caused by background purposes such as powering up computers and on-board sensors, along with controlling on-board motors for motion: for the sake of simplicity, we assume this sub-component to be constant over time. The foreground-drain instead represents the battery drain component caused by foreground processes running on-board of the robot (e.g. lidar odometry, object detection, traversability analysis, motion planning). Every second, we model the foreground battery drain caused by a generic process running on-board of the robot to result in $X\%$ drain of the battery level where X is proportional to the CPU usage needed by that process over a 1 s time window. The battery drain of static assets (e.g. base-station, communication node) is not modeled as the base-station might usually exploit a wired electricity distribution system, while communication nodes are equipped with long-lasting batteries whose power consumption can be neglected.

ROS Implementation Beside their roles, all nodes in the graph are implemented as instances of the *Agent* class with each class instance running in a dedicated ROS node. At mission start, all agents start with Default role, and only a machine is constructed with Manager role. Computational capabilities are encoded through integer numbers with 3 representing high, 2 representing medium, and 1 representing small.

5.1.2 Edges

Edges in the graph represent communication channels between nodes in the network. A communication edge E_{ij}^t exists only if there is a communication link from node i to node j at time t. The communication edge E_{ij}^t is characterized by bandwidth BW_{ij}^t which expresses the available bandwidth in Mb/s for data communication from node i to node j at time t. Same concept applies for communication in the opposite direction.

We do not assume stability of communication links, nor assume a-priori known evolution of network bandwidths over time. This makes the proposed framework suitable to be deployed in multi-robot systems where inter-agent communication bandwidths can rapidly vary when operating in extreme and communication-degraded environments.

Graphically, we encode the available bandwidth over a communication link with line thickness information, with high-bandwidth communication edges appearing very thick, and low-bandwidth communication edges appearing very thin.

ROS Implementation We implement inter-agent communication links through dedicated ROS topics. We set up dedicated unidirectional communication edges between all Agents to ensure that peer-to-peer communication is enabled between all ROS entities in the system. Following this formulation, husky1 will communicate directly to husky2 through the `/husky1_husky2_edge` topic, while husky2 will directly communicate to husky1 through the `/husky2_husky1_edge` topic. While *all* communication links exist at the ROS-network level, only a subset of them are actually used in the simulation for inter-agent communication, depending on which communication links are actually available at time t from real communication data.

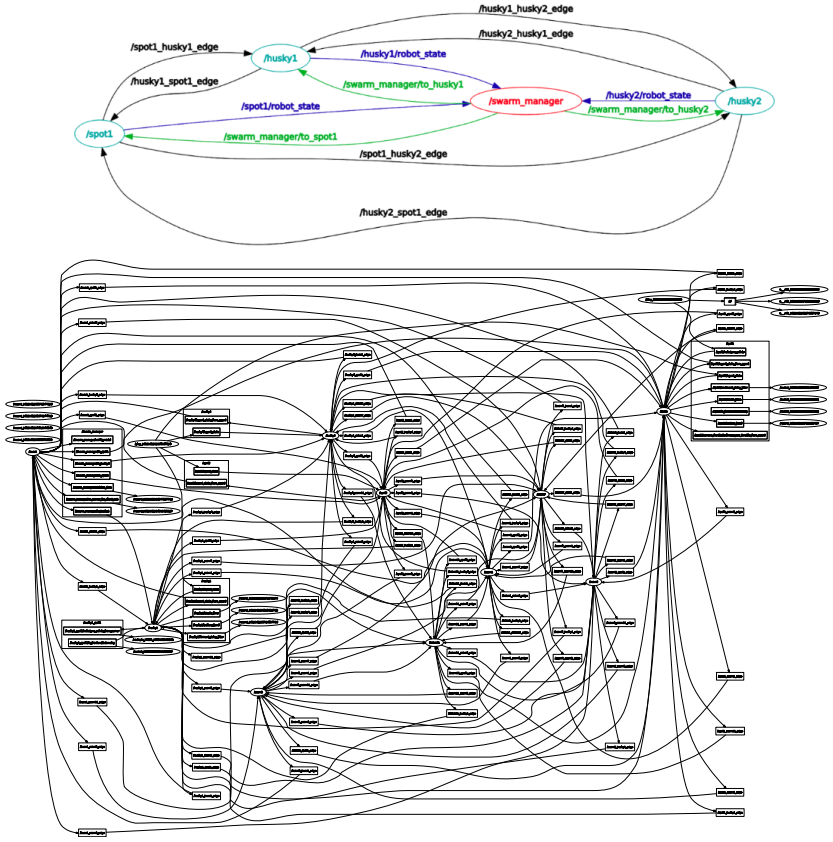


Figure 5.2: Overview of the ROS rqt graph evolution of the system.

5.1.3 Messages

We standardize inter-agent interactions by means of custom ROS messages as follows:

- **AgentState** The AgentState message standardizes the way an Agent (any node in the graph with non-Manager role) can provide its status update to the Manager in order to update its world model. This message encloses information on the agent's status S_i^t at time t along with: *i)* a string subfield to specify the name of a potentially dropped communication node in P_i^t at time t, *ii)* a request subfield specifying whether a computation offloading is requested, for which task (e.g. lidar odometry, object detection), and for which size (e.g. size in Mb of the current lidar scan, size in Mb of the current camera image). Agents can ask for computation offloading of multiple tasks while transmitting a single AgentState message to the Manager.

- **ManagerResponse** The ManagerResponse message standardizes the way the Manager responds to agents in the network at each world model update. Once AgentState messages from all agents have been successfully received by the Manager, the Manager uses this set of synced received messages to update its knowledge of the world model which is then exploited to handle computation offloading requests generated by clients accordingly.

  For agents requesting computation offloading, this message provides: *i)* a computation offloading solution if it was found (in terms of the identified optimum server and optimum route), *ii)* a no solution response if the Manager was unable to successfully produce a solution for the offloading request (e.g. no servers available, no routes available). If a solution is found for an offloading request, this is provided to the client in form of a simple string subfield of the ManagerResponse message, where the string is composed by a set of agent names separated by slash symbols: the first substring encapsulates the name of the client that generated the offloading request, the last substring represents the name of the identified optimum server, and all substrings in the between instead represent names of the communication hops involved in the optimum route chosen by the Manager for client-server communication (e.g. `/spot1/scom1/husky1`). For clients requesting

offloading of *different* computational tasks, the Manager provides, if available, a solution for each offloading request type.

Independently whether an agent expresses a computation offloading request or not, the ManagerResponse message received by all agents from the Manager also encapsulates information on: *i)* the name of the next backup Manager to be used in case of failure of the current central orchestrator, *ii)* the list of dropped communication nodes up to this point in time in the mission, along with their location. These information can be used by a dynamically elected machine that becomes Manager on original Manager's failure to have a comprehensive knowledge of the world status and take routing decisions accordingly. Such approach therefore realize a "passing of the baton" mechanism to handle sudden and unpredictable failures of the central coordinator, and is described in more detail in section 6.2.9.

- **DataWithRouteInfo** This message probably represents the most important message type in the framework as it enables a Software Defined Networking routing mechanisms in absence of OpenFlow enabled communication elements. More specifically, this message abstracts a given data that has to travel across the network (e.g. a lidar scan, a camera image) and pairs it with a routing information (the optimum route chosen by the Manager for the flow of that specific data stream) which is encapsulated as a subfield in the message definition.

  In this way, the Swarm Manager's SDN Controller does *not* need to send routing table updates to all networking elements, as the optimum route chosen for a given computation offloading request is communicated *only* to the client asking for it. This ensures that the sensor data offloaded by the client will reach the optimum server destination chosen by the Manager through the optimum route chosen by the Manager as all network elements will simply act as logical switches, directing the data on the route that is specified in the DataWithRouteInfo message route subfield. More specifically, the DataWithRouteInfo message has two route fields: *route* which expresses the remaining route at each node of communication, *full route* which expresses the full route chosen from the Manager for client-server communication, and never gets modified. An in-depth description of the

forwarding mechanism of DataWithRouteInfo messages on network elements is provided in section 6.2.5.

The DataWithRouteInfo message can transport data both in the forward direction (input data from client to server) and in the backward direction (output result of the computation from server back to client).

5.2 Interactions

In this section we present the different interactions at the inter-agent level that constitute the principle of operation of the proposed framework.

5.2.1 Updating Manager's World Model

To take advantage of the orchestration capabilities provided by the Manager (e.g. computation and communication allocation for dynamically generated offloading requests), the robot team needs to provide the Manager with a comprehensive overview of the world model by transmitting status updates in form of Agent States messages to the central entity at a fixed rate r. When all updates from all agents have been received by the Manager, they can be used to construct and/or update the knowledge that the Manager has of the world model at time t: the status of the nodes represents the status of the agents (e.g. mobile, static) while the status of the edges represent the status of the inter-agent communication links (e.g. radio or wifi) at that point in time. At each world model update, mobile agents might have moved from their position in the previous update, and communication links might have abruptly changed their status. The result is that the graph G is dynamically updated at rate r, which, given the system dynamics of the mission of interest (e.g. robots moving at medium-low velocity) is set to 1 Hz.

ROS Implementation Agents in the network transmit AgentState messages to the Manager at fixed rate. The Manager receives all of these messages, exploits a message filters ApproximateTimeSync policy to synchronize updates coming from different agents, and uses the resulting information to update its knowledge of the world model.

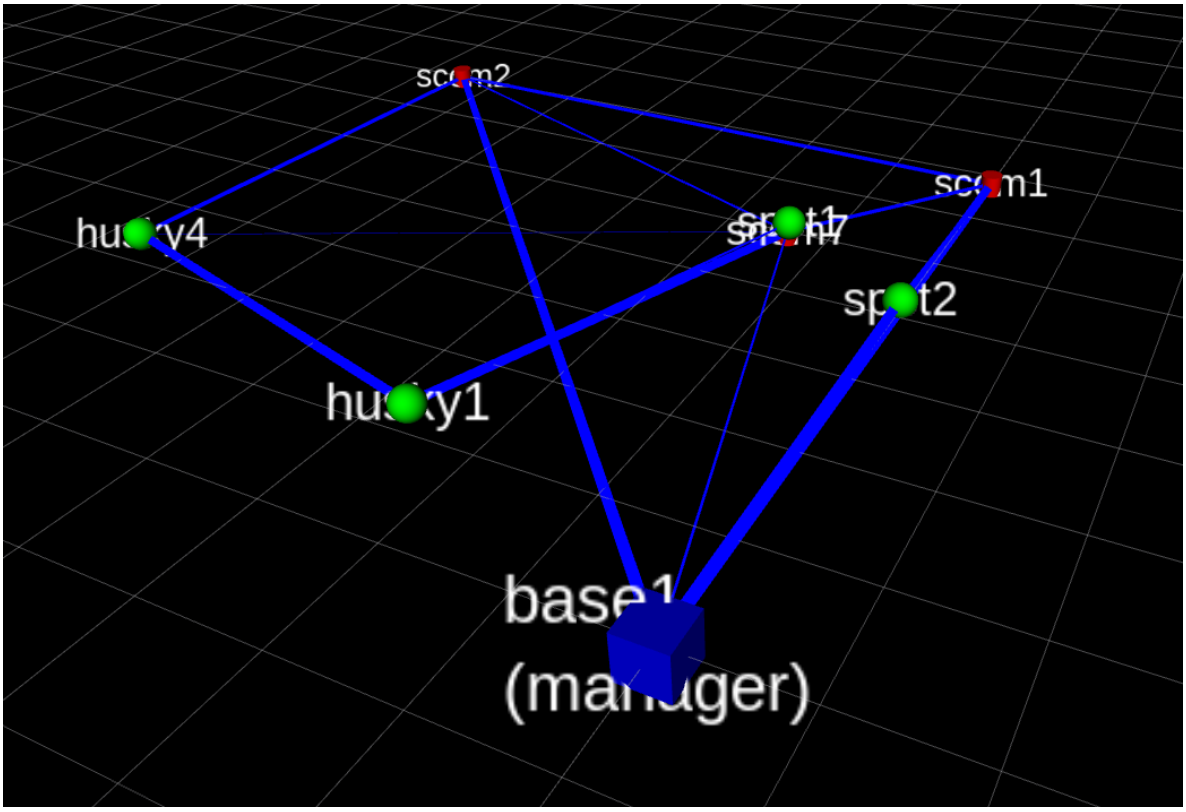


Figure 5.3: RVIZ visualization of the world model known by the Manager at time t, after a successful synchronization of the AgentState messages from all agents.

Synchronization is needed as messages emitted by different agents might arrive with different latencies on the Manager side: in our settings, a synchronization time-window of 0.7 s is enough. The framework is also robust to scenarios where an agent goes out of communication range and does not further provide status updates: if such condition is detected, the Manager quickly updates its synchronization policy to keep managing the rest of the robotic team.

We use a graph data structure from the networkx python library [160] to store the Manager's world model as a set of nodes and edges.

When a communication node dropping event is specified in the AgentState message received by an agent, the Manager adds a fixed node in the graph to represent the static communication asset dropped by that robot, at that time, in that specific location. Fig. 5.3 reports a RVIz visualization of the world model constructed and published by the Manager after successful reception and synchronization of the different status updates provided by the different agents. The Manager is indicated with a blue cube,

mobile agents are indicated with green spheres, and the dropped communication nodes are indicated with red cylinders.

5.2.2 Computation Offloading Request Generation

A robot can ask the Manager to offload a given computation if in need as illustrated in 5.1.1. If perception related tasks (e.g. lidar odometry, object detection) need to be always runned on more powerful servers, computationally-constrained platforms might offload perception-related computation from mission start.

ROS Implementation An agent in the team willing to offload computation can express this in the request subfield of the AgentState message that is constantly provided to the Manager to update its world model specifying the name of the task he wants to offload (e.g. LOCUS, YOLO), along with the size expressed in Mb of the current sensor reading (e.g. lidar scan, camera image). As stated earlier, the AgentState message can host multiple offloading requests of different natures in the same message.

5.2.3 Computation Offloading Request Handling

At each world model update at time t, we define with R_i^T the computation offloading request of the agent i for the task T with data size L, and represent with $\mathbb{R}$ the set of all computation offloading requests received by clients at the current iteration.

When receiving computation offloading requests from agents, the Manager makes use of its updated world model to process requests accordingly in a sequential fashion. For each client request, the Manager tries to take two decisions: *i) where* to execute the task (e.g. identify the optimum server), and *ii) how* to route the data for that client/server communication given the current global status of the network (e.g. identify the optimum route). The two fundamental steps needed to handle a computation offloading request are described in the following, while Algorithm 1 provides a high-level schematization of the computation offloading requests handling logic.

Optimum Server Computation

When a computation offloading request is received from a client, the Manager exploits its updated world model to understand where is best to execute the task of interest given the current status of the system. First, the Manager extracts the subset of nodes in the graph that are directly or indirectly connected to the client. Then, only a subset of the nodes that are potentially available for computation (e.g. servers) are retained. Finally, for each server, a server score is calculated where the metric is general and can be defined by the user. For example the score of a server machine increases with increasing computational capability, connectivity to the client and battery level. In the current implementation the battery level of a machine is not taken in consideration when computing server scores: the score of a server is decreased only when its battery level goes below a certain user-definable threshold. After scores have been computed for all potential servers, the optimum server is identified as the machine that maximizes the server score. If no servers are available, the client needs to execute the task on-board, until a server becomes available again.

Task-Aware Server Score Computation An important feature of the framework is that the resource allocator logic of the Manager is task-aware meaning that it will autonomously adapt to different server score computation metrics depending on what is the nature of the task that the client is willing to offload. For example, for a client requesting LOCUS offloading, any machine equipped with a powerful CPU will be considered during server score computation, while for a client requesting to offload YOLO, the server score will be greatly increased only for the subset of machines equipped with on-board GPUs. Overall, the goal of the resource allocator logic of the Manager is to identify the absolute best suitable machine for the specific computation offloading requested by the client.

Load Balancing To avoid overloading individual machines, the Manager keeps track of decisions taken for other client requests within the same world model update: this information is used to decrease the score of servers that are already serving other clients and avoid Denial of Service (DoS) scenarios. In the current implementation, we set to

zero the score of a server machine if this machine is already serving a maximum number of clients indicated with DoS_T (to prevent computationally-constrained platforms from being potential server candidates, all is needed is to set DoS_T to zero when instantiating that specific instance of the Agent class). This therefore translates into an implicit mechanism of Load Balancing at the resource allocation level.

Exploiting Failures In multi-robot exploration missions of extreme environments, failures of on-board sensors and/or mobility systems represent a real possibility. For example, a lidar sensor might fell from the rigid mounting on-board of a robot, making the agent "blind" if no other exteroceptive sensors are available. As another example, a robot might get stuck on muddy areas, or a legged-platform might fall on a negative-obstacle. In all these scenarios, by exploiting on-board health monitoring modules, it is possible to enclose in an additional sub-field of the AgentState message, a health status flag that can be set to false when a critical failure is experienced by the platform. When such event is detected on the Manager side, the Manager can exploit the failure of this agent, send commands to shut down on-board sensors and redundant processes to save power, and increase the score of this machine during the server computation stage: by doing this, the "dead" machine will actually be used to serve as a computational asset for the needs of other robots, enhancing therefore the overall performance of the robotic team and improving the level of cooperatively behaviours in the swarm.

Optimum Route Computation

Once an optimum server has been identified, the Manager exploits its globally-aware knowledge of the network status to identify the optimum route that minimizes communication latency between client (source) and server (destination) by maximizing the average communication bandwidth and minimizing the number of communication hops. To enhance the processing time needed for the optimum route computation, the Manager analyzes, among all possible paths in the network connecting source to destination, only a subset of paths whose number of communication hops does not exceed a predefined threshold.

Traffic Management Accordingly to what happens when computing server scores, the Manager keeps track of the size in Mb/s of streams already instantiated for other client-server communications in the current world model update, to account for the actual available bandwidth on different communication edges while computing the optimum route for another client request. This therefore enables the Manager to achieve optimum and globally-aware traffic management in multi-client scenarios and decrease network congestion.

```
ResetResourceAllocationFromPreviousIteration()
for T in ["lidar_odometry","object_detection"] do
    for R_i^T in ℝ do
        data_size ← R_i^T.L
        server_scores_i ← {} get optimum server;
        optimum_server_i ← None
        for {j in N | j ≠ i and j.clients < DoS_T and ∃ path_i,j in G} do
            server_score_j ← computation_j/hop_wise_distance_ij
            if T is "object_detection" then
                server_score_j ← server_score_j * j.has_GPU
                server_scores_i.insert(server_score_j)
            end
        end
        optimum_server_i ← argmax(server_scores_i)
        if optimum_server_i ≠ None then
            path_scores_i ← {}
            best_path_i ←None
            all_paths ← GetAllPaths(G, i, optimum_server_i, cutoff)
            for k in all_paths do
                path_score_k ← ∑_Eink E.BW / number_of_hops_k
                path_scores_i.insert(path_score_k)
            end
            best_path_i ← argmax(path_scores_i)
            ProvideSolutionTo(R_i^T, optimum_server_i, optimum_path_i)
            for E in optimum_path_i do
                E.BW ← E.BW - data_size
            end
            optimum_server_i.number_of_clients++
        end
    end
end
```

Algorithm 1: Computation Offloading Requests Handling

Solution Generation

When a solution for the client request has been successfully computed, the Manager aggregates the resulting information into a unified ManagerResponse message that can be communicated to the client to instantiate the client-server communication for offloading purposes.

ROS Implementation As stated earlier, the computed solution is constructed by means of string concatenation (e.g. `spot1/scom1/husky1`) where the first string represents the client, the last string represents the server, and every string in the between represent the intermediate communication hop of the route chosen by the Manager for client-server communication.

5.2.4 Computation Offloading

When a valid solution for the computation offloading request is returned by the Manager, the client can use it to offload its sensory stream.

ROS Implementation To start the offloading mechanism, the client needs to operate the following steps. First, it has to create an empty DataWithRouteInfo message and enclose in the *full route* field the route chosen by the Manager. Then, it has to wrap the sensory data coming from on-board drivers (e.g. lidar scan, camera image) into the data field of the DataWithRouteInfo message. In order to ensure that this data will travel across the network through the route chosen by the Manager, the client first needs to remove its name from the route provided by the Manager, and then encapsulate the so called remaining route in the *route* field of the DataWithRoute info message. Once the above steps have been performed, the client can send the data to the next hop of communication. Then, later in the network each node in the graph acts as a logical switch on reception of a DataWithRouteInfo message as always, by removing its name from the route information, and forwarding the data with the updated remaining route to the next hop of communication: more details on the forwarding mechanism are provided in 6.2.5.

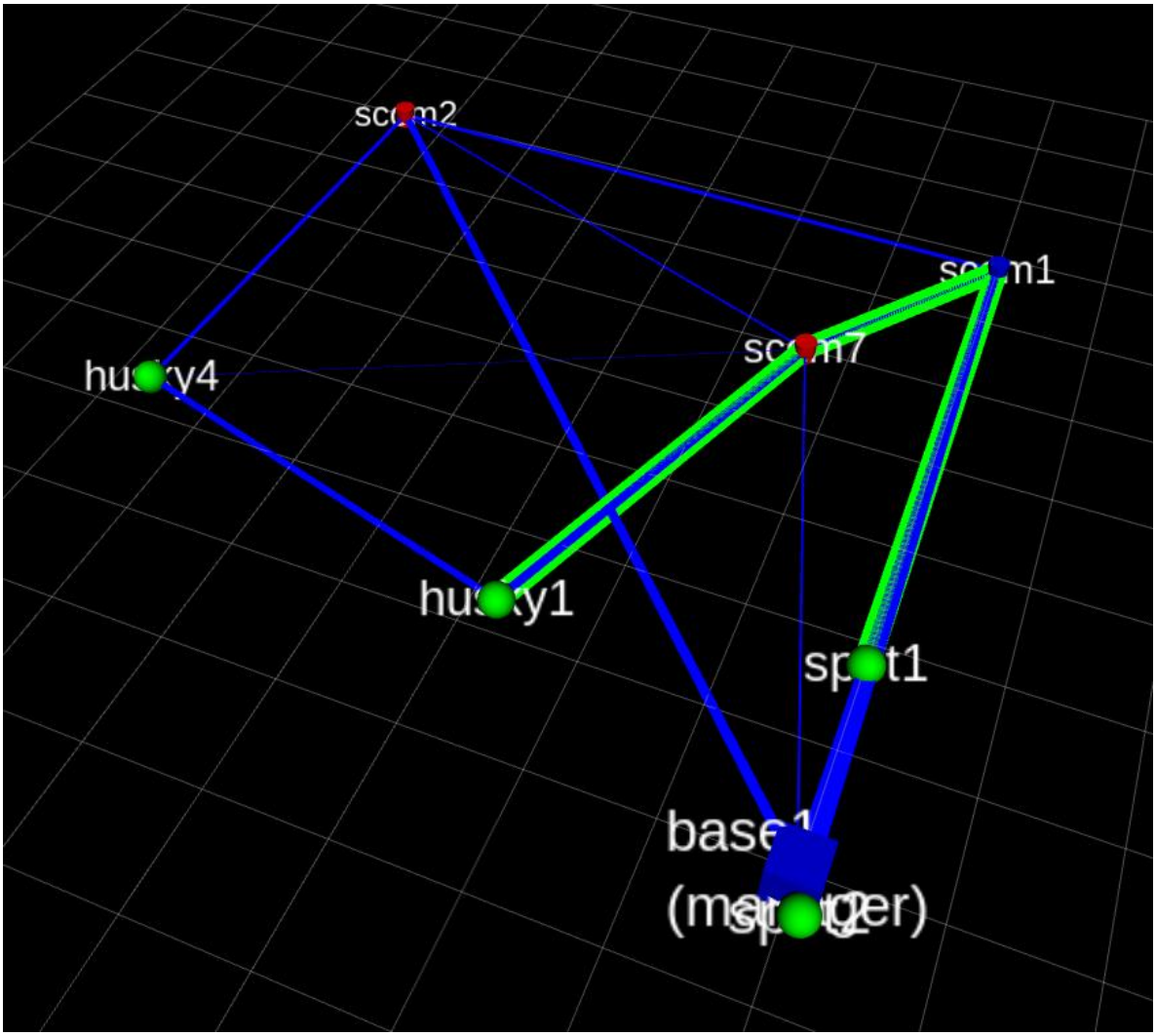


Figure 5.4: Spot1 offloads its lidar stream to the optimum server husky1 for high-definition ego-motion estimation purposes through the optimum route chosen by the Manager displayed in green. The lidar data offloaded by spot1 pass through scom1 placed at the end of the straight corridor, to then go to scom7, to finally reach the husky1 server identified as optimum server by the Manager.

This process is automatically triggered for each sensor message coming from the on-board drivers within the 1 s time window that separates consecutive transmissions of AgentState message updates to the Manager (which then translate into updated offloading solutions provided by the Manager at 1 Hz). For example, if the input sensory data rate is 10 Hz, 10 messages will be offloaded through the route and to the destination specified in the latest available Manager's solution that is stored on-board of each agent. Given the moderate system dynamics, we assume that communication links maintain their status unchanged during the 1s time window: this assumption can be however relaxed by increasing the Manager's update rate to higher values, assuming

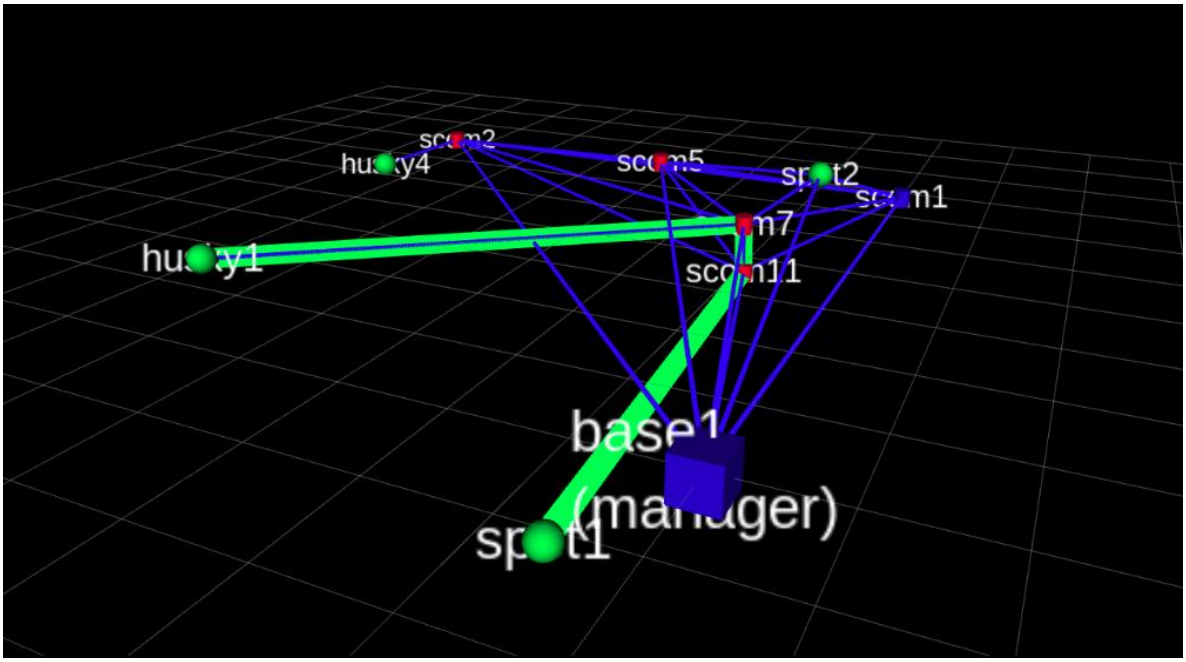


Figure 5.5: Handling of computation offloading request from a single client. In this case spot1 is exploring the lower floor after autonomous stair descent operation, and offloading its lidar stream to husky1 for ego-motion estimation. The optimum route chosen by the Manager to instantiate robot to robot communication and high-volume lidar data exchange after the perception offloading request is displayed in green: it starts from spot1, then pass through scom11, scom7, to finally reach the husky1 server identified as optimum server by the Manager.

high-rate profilements of communication link status are provided by mission logging utilities.

5.2.5 Forwarding Mechanism

A key idea of the presented framework is to realize dynamic data routing behaviours by encapsulating the route information inside the message that is travelling across the network itself. When a DataWithRouteInfo message arrives on a communication hop, to exploit a Software Defined Networking paradigm in absence of OpenFlow enabled devices, each communication hop executes the abstract forwarding procedure described in the next paragraph.

ROS Implementation The first thing performed by a network element receiving a DataWithRouteInfo message is removing its own name from the route specified in the

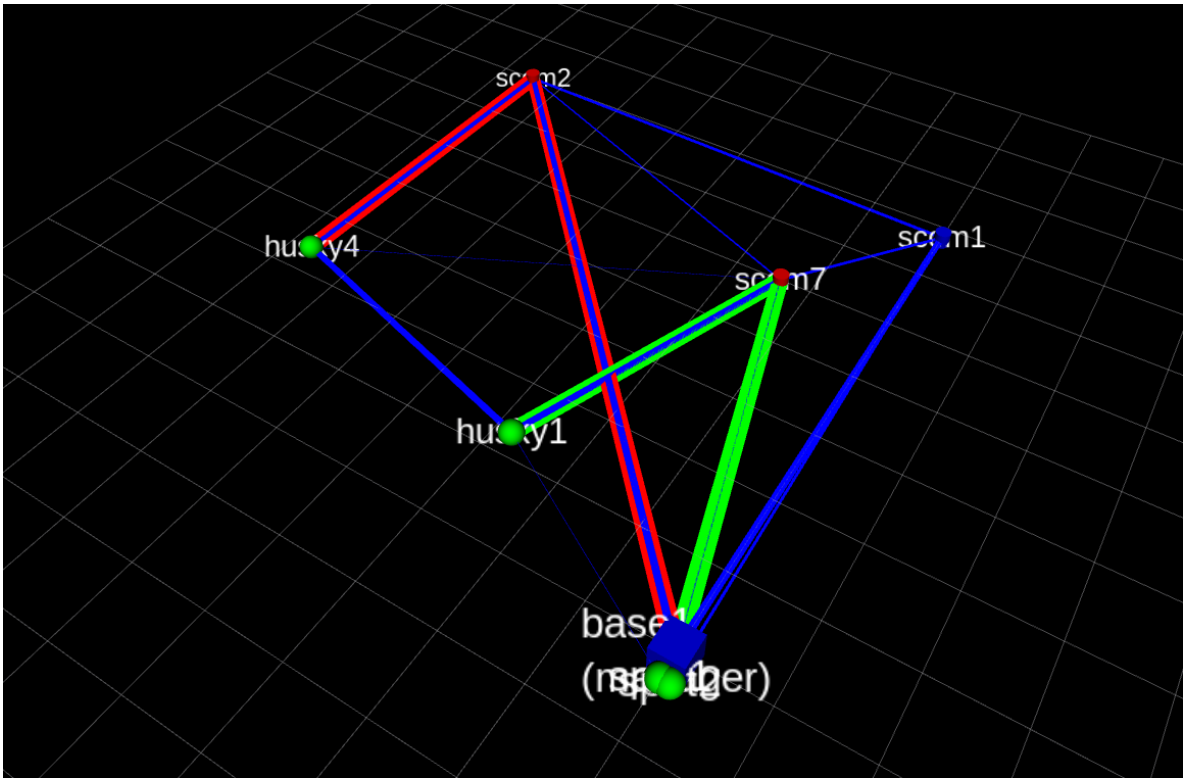


Figure 5.6: Examples of multi-client management. For the computation offloading request generated by spot1 the Manager chooses husky1 as optimum server and spot1/scom7/husky1 as optimum route (green). For the computation offloading request generated by spot2 the Manager chooses husky4 as optimum server and spot2/scom2/husky4 as optimum route (red).

route field of the DataWithRouteInfo message to obtain the remaining route, while leaving the *full route* subfield of the DataWithRouteInfo message unchanged. At this point two main situations can occur: *i)* a next-hop of communication exists in the remaining route, *ii)* no next-hop of communication exists in the remaining route. In the case where a next-hop of communication exists in the remaining route, the network element overwrites the route field of the received DataWithRouteInfo with the updated remaining route and forwards the resulting data to the next hop through the dedicated communication channel. In the case where no next-hop of communication exists in the remaining route: *i)* if the received DataWithRouteInfo contains "input data" (e.g. sensory data travelling in the forward direction from client to server) then the data has to be processed on-board as the current recipient machine is actually the optimum server chosen by the Manager to execute the computation required by the client, *ii)* if the received DataWithRouteInfo contains "output data" (e.g. output of server-side

computation travelling in the backward direction from server to client) then the client has successfully received the result of the offloaded computation.

This therefore realizes a Software Defined Networking communication paradigm as the route for the client-server communication is dynamically chosen and updated by the Manager at the update rate r.

5.2.6 Server Side Processing

When the sensory data offloaded by the client successfully reaches the optimum server through the optimum route chosen by the Manager, the server-machine has to operate the computation required by the client on the received data. For example, if the received DataWithRouteInfo message contains lidar sensory data, the input lidar message is processed by a dedicated LOCUS process running on the server machine. If the received DataWithRouteInfo message contains image sensory data, the input image message is processed by a dedicated YOLO process running on the server machine. Obviously, powerful server-side machines can allocate much larger computational capabilities to perform the task of interest with respect to what could be achievable on the computationally-constrained client platform. The high-definition server-side processing is therefore a key component in the overall enhancement of the distributed situational awareness retrieved by the heterogeneous multi-robot system.

ROS Implementation Each Agent in the team comes equipped with dedicated ROS nodes for LOCUS and YOLO processing that can be used on the robot with intra-robot calls as sensory messages arrive from on-board drivers. Server machines, not only dispose of dedicated ROS nodes for proprietary processing purposes, but can also instantiate additional dedicated LOCUS server and YOLO server nodes that can be used to perform computation on the data received by the clients. Multiple ROS nodes can be instantiated on the same server machine for server computation related purposes of different received client streams, depending on the available computational resources on-board.

On the receiver side it is important to implement a sorting layer that takes care of processing received messages in sequential order to cope with potential out-of-order

arrivals of messages caused by sudden changes in network routes dynamically chosen by the Manager, as these could translate into potential different time of arrivals on the receiver side, despite being sent in a correct sequential order from the computation offloading client.

For each message received by each client, the server machine, populates a dedicated queue of messages each time a new message arrives. Then, with multi-threading techniques, the server goes over the queue of each type of data of each client, and process it accordingly. A rate-limiter publisher of the received data to server-dedicated ROS nodes has been also implemented to avoid performance degradation in case of communication bursts.

5.2.7 Sending back the result

Once the sever-side computation on the input data provided by the client is completed, the output product of the computation needs to be delivered back to the client. The result of the server-side computation is delivered back to the offloading client through the inverse route used to transfer the sensory data in the forward direction (e.g. from client to server), which is stored on server-machines upon reception of clients data. Generally speaking, the backward route to deliver outputs from server back to client can simply be the latest available stored backward route obtained from the latest message received by the specific client whose computation request has just been fulfilled.

ROS Implementation After the server-side processing is completed, the ouput data product (e.g. Odometry output from LOCUS, Image detection output from YOLO), is retrieved by the server through its dedicated intra-robot subscribers to output product topics. Once the output message of interest is retrieved, the server is responsible to wrap this data in the output data subfield of a freshly created DataWithRouteInfo message, that is then paired with the backward route information (which is computed by inverting the full route field of the DataWithRouteInfo message received by the client) to deliver back the result of the computation to the client. The client therefore successfully receives the result of the offloaded computation: for example, for LOCUS offloading this translates into a computationally-constrained platform knowing a high-

definition odometry estimate of its position without having performed any computation on board.

5.2.8 Network Delay Simulation

At each communication hop, the time needed to transmit a message with data size L (in Mb) over a communication link E_{ij} with bandwidth BW_{ij}^t (in Mb/s) at time t is expressed by $d = L/BW_{ij}^t$. We model the end-to-end delay from a source to a destination node in the graph as the sum of the communication delays introduced by each communication hop in the route from source to destination.

The total offloading time can be therefore decomposed into the following sub-components:

- **Forward Network Delay:** amount of time in seconds needed by the DataWithRouteInfo message transmitted by the client to reach, through the communication route chosen by the Manager, the destination server.
- **Processing Time:** amount of time in seconds needed by the server to process the client request, namely the time needed by the server to operate a desired computation on the client's input data.
- **Backward Network Delay:** amount of time in seconds needed by the DataWithRouteInfo message transmitted by the server to reach, through the communication route chosen by the Manager, the destination client to deliver back the result of the offloaded computation. When offloading lidar odometry computation, the backward network delay is much smaller than the forward one, as the output of the server-side computation (e.g. Odometry message) is much more lightweight then a lidar scan message.

The *total offloading time* is therefore modeled, for each message, as the sum of Forward Network Delay, Processing Time and Backward Network Delay.

5.2.9 Backup Manager

The proposed framework implements automatic ways of monitoring the health of the central orchestrator, along with automatism to dynamically elect the backup Manager

machine to act as central orchestrator in case of failures of the previous one. As mentioned earlier, at each update of the world model, the Manager provides all agents with information on: *i)* the name of the next backup Manager (computed as the most connected node in the graph with computational capabilities), *ii)* a list of the dropped communication nodes in the world (these constitute essential knowledge requirements about the world status that a newly elected Manager needs to satisfy in order to make globally aware resource allocation decisions). When no responses from the Manager are received by agents after a pre-estabilished timeout threshold, the Manager's replacement stage is entered.

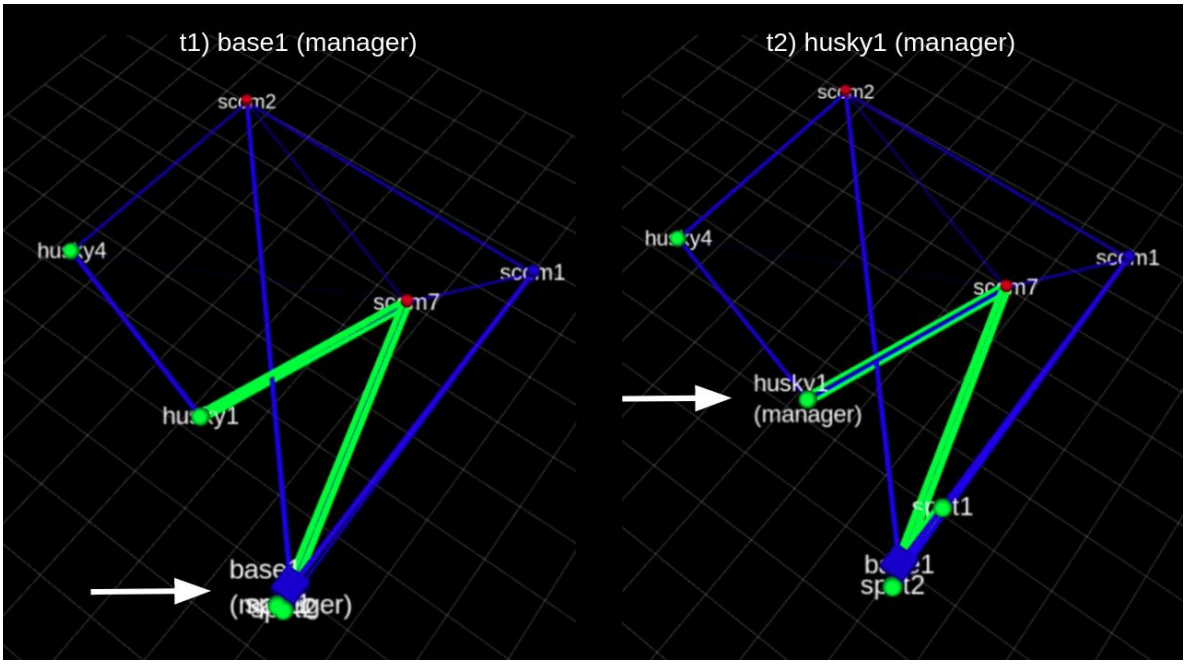


Figure 5.7: Demonstration of recovery from central orchestrator's death. At time t1, base1 is the Manager of the robot team, but fails afterwards. At time t2, husky1 is elected as new Manager of the robot team and keeps handling the decision making at the computation and network allocation level.

ROS Implementation Each agent reads the name of the successor Manager from the latest stored ManagerResponse message. At this point, a given agent can find himself in two conditions: *i)* being the successor, *ii)* not being the successor. If the first case occurs, the agent immediately proceeds to take the place of the Manager and instantiates all the resources needed to keep the orchestration going. In this case, the ROS publisher/subscriber paradigm simplifies the implementation, as the successor

only needs to register as a subscriber to the topics already directed to the supervising entity. Furthermore, this allows the Manager replacement process to be completely masked from the eyes of all other agents, as they will continue to communicate with the controller through the same channels. Video demo: `https://youtu.be/O412vszjI-U`

Chapter 6

Integrated System Performance

In this chapter we test the integrated system performance on a real multi-robot dataset collected by NeBula's robots during the Beta 2 Course of the Urban Circuit round of the DARPA Subterranean Challenge. We demonstrate initial results on enhancements achieved through the use of Swarm Manager on the situational awareness accuracy retrieved by a team of four heterogeneous robots involved in the autonomous exploration a multi-level and perceptually-degraded dismissed power plant in Elma, WA, under severe computation and communication constraints.

First, we describe the nature of the dataset gathered during the multi-robot mission, and briefly describe how this has been interfaced to Swarm Manager. Then, we present enhancements achieved in localization accuracy of a computationally-constrained legged platform by means of LOCUS offloading. Finally, we present improvements in object detection precision of a computationally-constrained legged platform by means of YOLO offloading. For each type of task offloading, we showcase decreased battery-usage onboard of robot clients which could therefore translates in extended autonomy time and potential increased information gain.

6.1 Dataset Description

The heterogeneous robotic team is composed of two Cleopatra Husky ground vehicles, and two Boston Dynamics Spot legged platforms, whose hardware description is reported in Fig. 6.1. Namely, the team is made of husky1, husky4, spot1, spot2.

The robots communicate with the base through a layer-2 mobile ad-hoc mesh network

Topology	Wheeled Vehicles Husky	Legged Platform Spot
		
High-level Computers	**SLAM:** Multiple, High-power CPU **Object Recognition:** NVIDIA GPU	**SLAM:** Mid-power CPU **Object Recognition:** NVIDIA GPU
Low Level Microcontrollers	**Safety features:** Atmel MCU **Comm Dropper:** STM MCU	**Safety features:** Atmel MCU **Comm Dropper:** STM MCU
Sensors	LiDAR, depth and thermal cameras, gas and signal sensors, external IMU.	LiDAR, depth and thermal cameras, gas and signal sensors, external IMU.
Comm Dropper	Yes	Yes
Mass	50kg	40kg
Battery	Lead acid	Lithium ion
Operational time	3 hours	1.5 hours

Figure 6.1: Hardware specifications for husky and spot NeBula robots.

constructed and expanded with commercial off-the-shelf MANET radios from Silvus Technologies (SC4240E-235-BB) and Persistent Systems (MPU5).

The dataset contains information of available communication links in the wireless mesh network along with their bandwidth status, which are logged on base-station from Silvus nodes at 1 Hz throughout the 1 hour mission. The dataset also contains data logged on-board of each robot including: *i)* lidar stream from a Velodyne VLP16 at 10 Hz, *ii)* throttled 424x240 px camera stream from an Intel RealSense D435i at 10 Hz, *iii)* vehicle state updates at 1 Hz with information on vehicle position.

During the operation, multiple radio communication nodes are deployed by robots at different times to maintain a backbone wireless mesh network: scom1, scom11, scom7, scom5, scom3. Agents in the dataset include therefore: base1, husky1, husky4, spot1, spot2, scom1, scom11, scom7, scom5, scom3.

Figure 6.3 reports location of communication nodes dropped during the mission by spot1, which is the robot we focus on as offloading client: in the multi-level exploration

Figure 6.2: Silvus radios used in the mission. Mobile agents can carry multiple communication nodes that are autonomously dropped when a mission-level criteria is met (e.g. low bandwidth and/or signal-to-noise ratio with respect to the base station)

spot1 drops a communication node before descending stairs, and one communication node when reaching the lower floor, creating therefore 7 m vertically-displaced high-bandwidth communication link to maintain communication with the team.

We think this dataset represents a formidable opportunity to test performance of SDN-inspired paradigms in dynamic multi-robot mesh networks under severe communication constraints as the dataset presents challenging conditions for inter-robot communication including constrained radio bandwidths, sudden loss of communication links, and evolves over a multiple-floor exploration. Currently, this dataset is not open source, but it may be in the future.

We parse the multi-robot dataset into a readable format for the Swarm Manager framework accumulating AgentState message updates for each agent (e.g. position, communication logs). From mission start, all agents in the world provide their status updates through the wireless mesh network to the Manager which is elected to be base1 at start. At 1Hz, the Manager receives status update messages from agents in the world and updates its world model. The updated world model is then exploited to handle computation offloading requests generated by clients. For each client request, the Manager tries to compute the optimum server and the optimum route to satisfy the client's offloading need. If a solution is found, that is communicated to the client which will

Figure 6.3: Location of communication node dropping for the autonomous exploration of spot1 robot during the Beta 2 round of the Urban Circuit of the DARPA Subterranean Challenge. Top) spot1 drops a communication node before stair descent. Bottom) spot1 lands on the lower floor, 7 m underground and drops a communication node to maintain a vertical communication channel with the above wireless mesh network. Images taken from DARPA matterport.

handle it accordingly to instantiate the offloading mechanism as described in section 5.3.

6.2 LOCUS Offloading Results

We demonstrate the effectiveness of the proposed framework by proving that offloading the computationally-heavy LOCUS task from a computationally-limited platform (spot1) to a heavily-equipped ground-server through the high-speed route chosen by the Manager, results in greater accuracy of the localization estimate of the agent, and therefore in enhanced distributed situational awareness accuracy retrieved by the heterogeneous multi-robot team.

In the results reported here, lidar scans provided by a Velodyne VLP16 at 10 Hz are preliminary downsampled with a voxel-grid filter with 0.1 m leaf size and expressed in base link frame. We benchmark pure lidar odometry performance with no integration of additional sensing modalities (e.g. VO, IMU). The average size of single lidar messages is 1.3 Mb with substantial variations in closed and open spaces. As shown later, lidar streaming results to be achievable through the high-speed routes chosen by the Manager with acceptable bandwidth requirements, translating into an overall LOCUS offloading time that is suitable for global localization enhancement purposes (e.g. odometry updates at the LAMP-robot level).

In the comparison, spot1 only has 1 thread allocated for scan-registration, and needs to apply an additional RDF with 90% decimation percentage to *try* to keep-up with the lidar processing at sensor rate if executing LOCUS on-board. On the other hand, husky1 dispose of 4 threads for scan-registration and can downsample less heavily the input lidar data: in this test a 80% RDF decimation percentage was chosen for husky-side processing. We compare the Absolute Position Error (APE) of the trajectories generated by LOCUS on the spot1 robot when not offloading, and on the server when the client offloads under orchestration of Swarm Manager against the ground-truth information (Fig. 6.11) to assess accuracy enhancements. The ground-truth trajectory of the robot is retrieved as explained in 4.5.

As demonstrated below, the edge ground-server can exploit its advanced computational capabilities to operate ego-motion estimation at a higher level of fidelity on the received lidar data, enhancing therefore the overall localization estimate of the computationally-

constrained offloading platform with acceptable latencies (Fig. 6.15). As a consequence, the offloading client does not perform heavy computation on-board, saves more battery, and can therefore aim for a longer exploration.

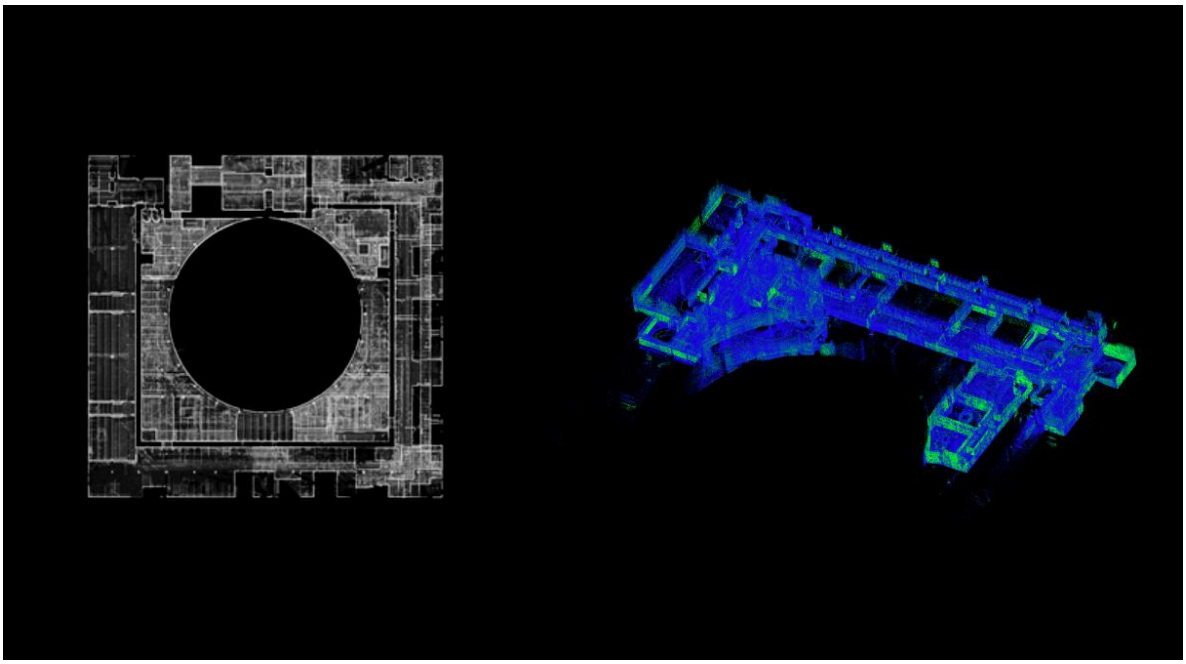


Figure 6.4: On the left, full ground-truth map of the Urban Circuit of the DARPA Subterranean Challenge. On the right, the ground-truth map of spot1 specific exploration.

While the framework is designed to dynamically choose the optimum server at each client request (e.g. at 1 Hz), to demonstrate the proof of concept of increased localization accuracy under orchestration of Swarm Manager, the results presented in this section are obtained with husky1 being statically set to be the optimum server for the computation offloading request of spot1. As for the network route, the spot1 client offloads its lidar stream to the husky1 server through the optimum route which is dynamically chosen by the Manager given the current status of the wireless mesh network as the exploration evolves. Video demo: `https://youtu.be/9-u0CZ6dA3o`

The results presented below demonstrate how offloading LOCUS computation to powerful robot peers through the high speed routes chosen by the globally-aware Swarm Manager can enhance the localization accuracy of computationally-constrained platforms (Fig. 6.11) of heterogeneous multi-robot systems, enhancing therefore the overall accuracy of the retrieved distribuited situational awareness.

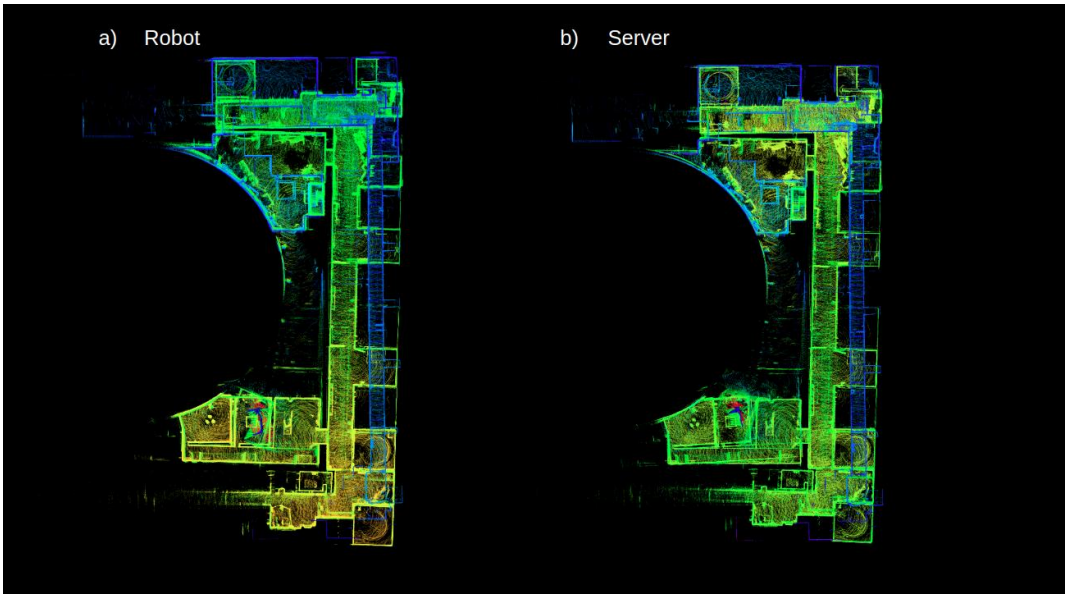


Figure 6.5: Top-view of the maps produced by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2. The server-side computed LOCUS on client's data results in substantially greater accuracy.

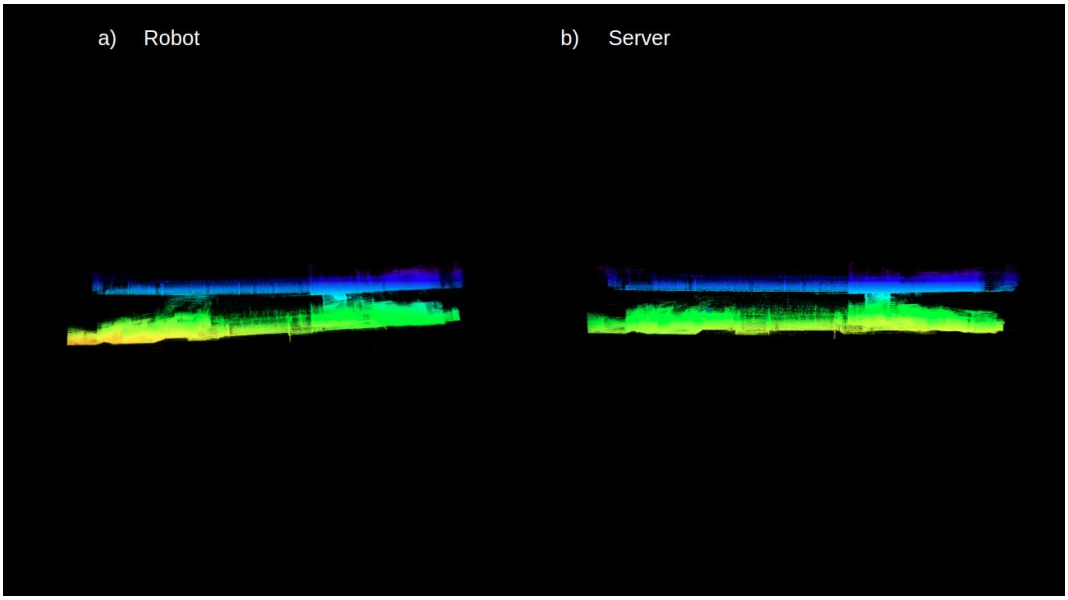


Figure 6.6: Side-view of the maps produced by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2.

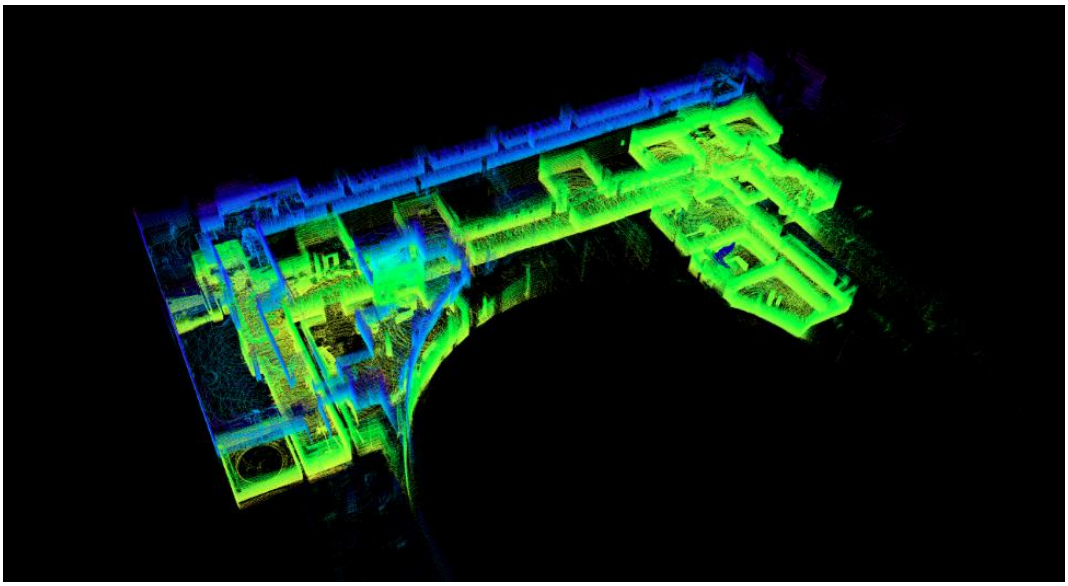


Figure 6.7: Orbit-view of the map produced by LOCUS on husky1 server on the lidar stream offloaded by spot1 client during the exploration of spot1 in Urban Beta 2.

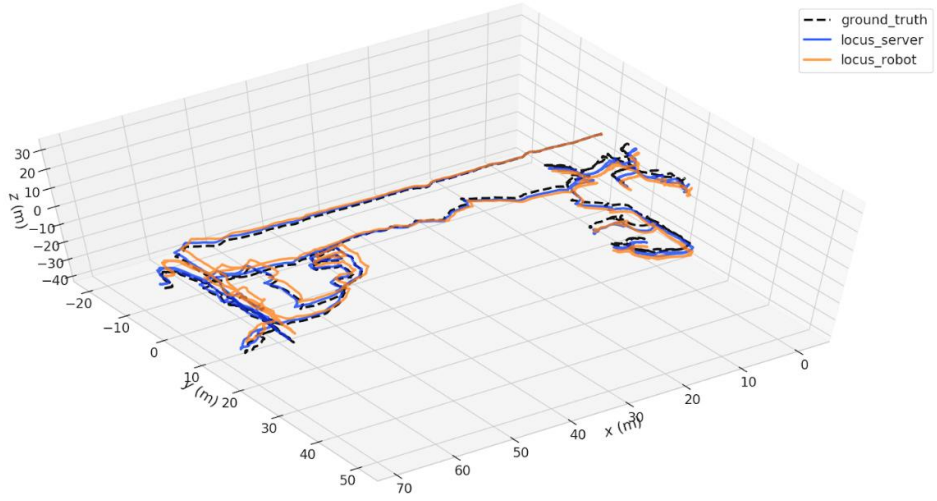


Figure 6.8: Trajectories estimated by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2.

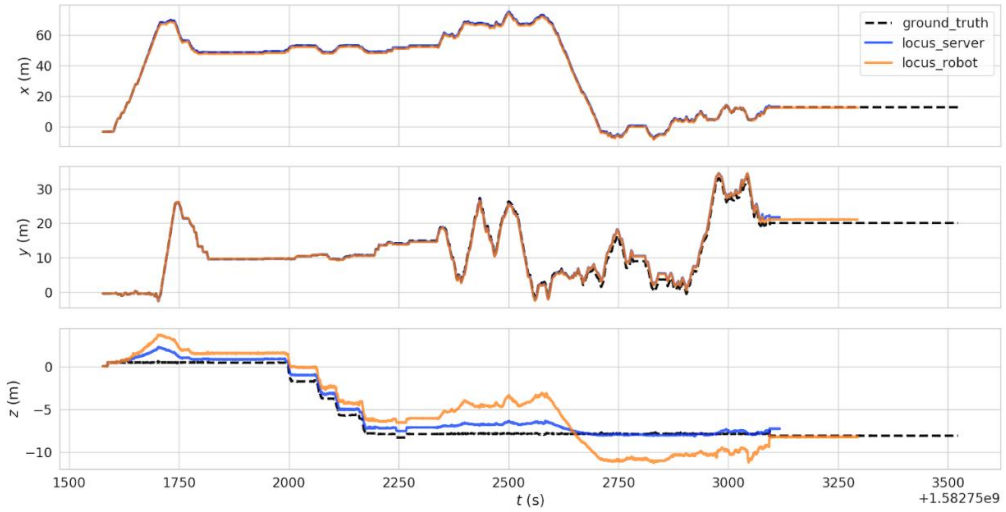


Figure 6.9: Profiling of the x,y,z components of the odometries generated by LOCUS on client and server against the ground-truth information. A lower downsampling percentage on the server side and a faster computation capability makes edge-processing of the ground-server achieve substantially better estimates with the respect to the performance achievable on the robot, expecially for the z component.

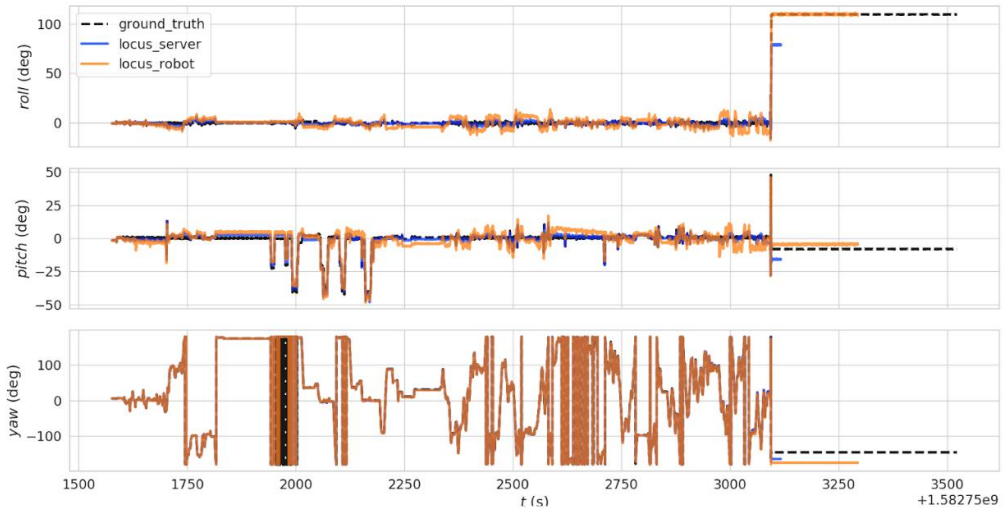


Figure 6.10: Profiling of the roll, pitch, yaw components of the odometries generated by LOCUS on client and server against the ground-truth information. A lower downsampling percentage on the server side and a faster computation capability makes edge-processing of the ground-server achieve sensibly better estimates with the respect to the performance achievable on the robot.

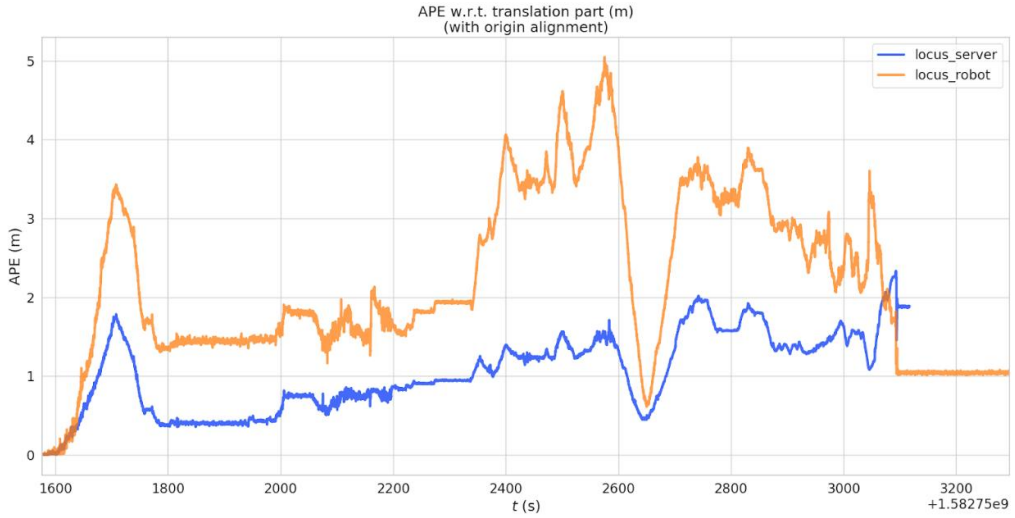


Figure 6.11: Absolute Position Error (APE) of the trajectories produced by LOCUS on the robot client (spot1) and on the server (husky1) for the exploration of spot1 in Urban Beta 2. This is the key result of the thesis work as we demonstrate higher perception accuracy of computationally-constrained platforms with a centrally-managed offloading mechanism. Computing high definition LOCUS on the server side results in greater accuracy of the front-end information, which then translates into greater consistency of the global localization at the back-end level. Deploying the proposed framework on a real multi-robot systems could potentially enable the performance of the multi-robot team as discussed.

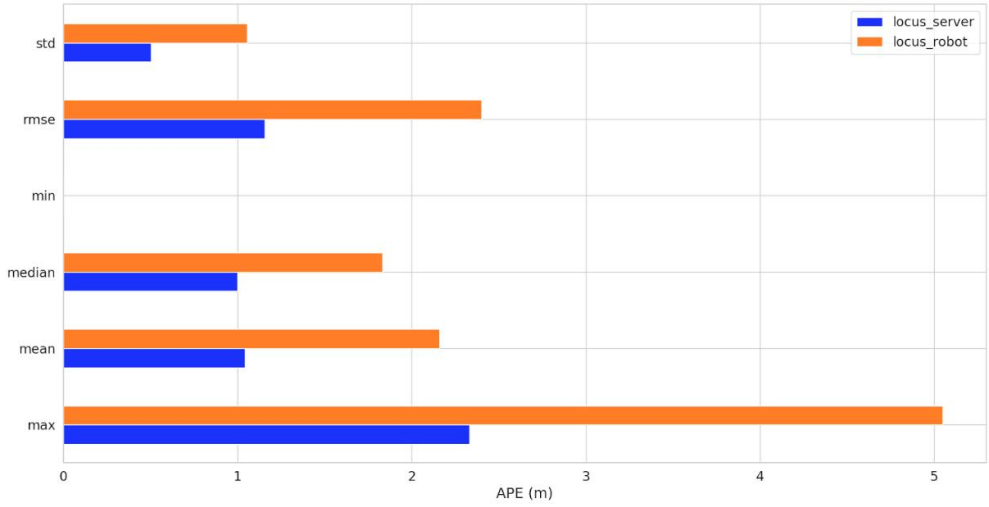


Figure 6.12: Absolute Position Error (APE) statistics of the trajectories produced by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2. The server side processing achieves lower max and mean error, along with lower standard deviation.

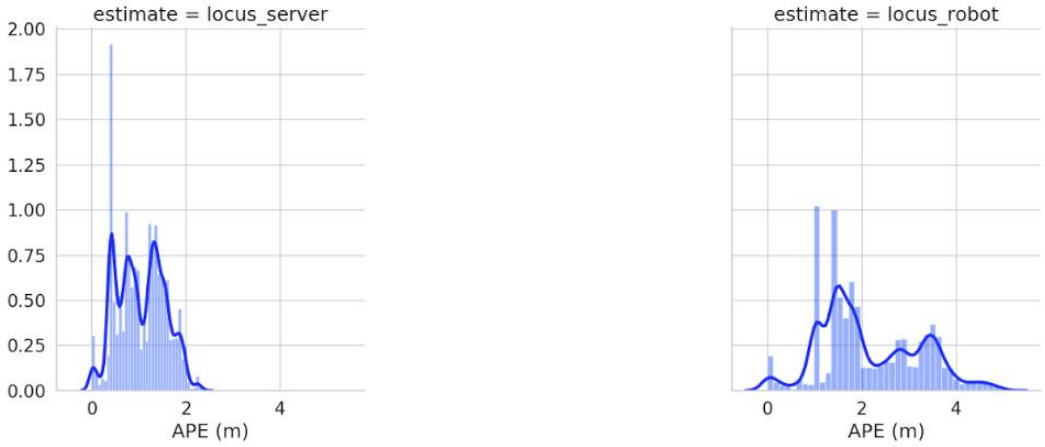


Figure 6.13: Absolute Position Error (APE) distribution of the trajectories produced by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2.

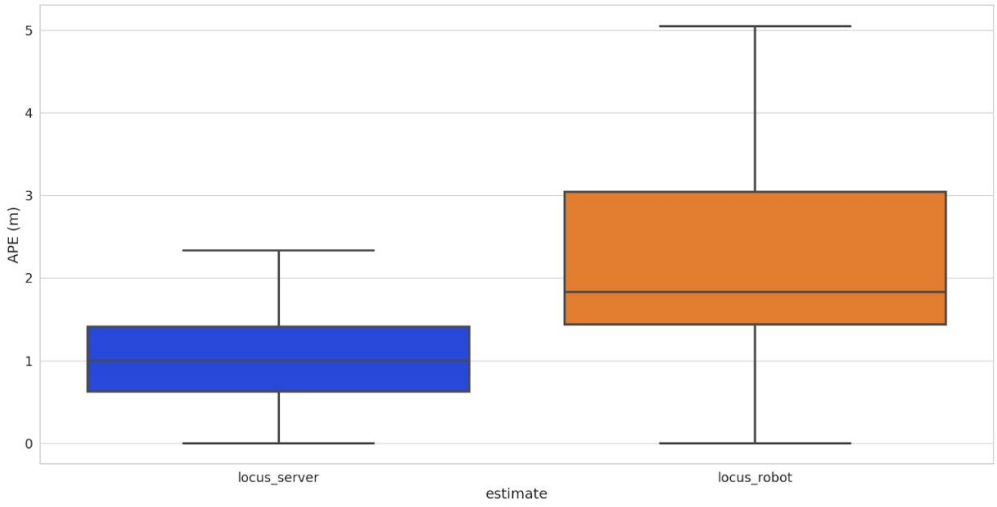


Figure 6.14: Box plot visualization of the Absolute Position Error (APE) of the trajectory estimated by LOCUS on client (spot1) and server (husky1) for the exploration of spot1 in Urban Beta 2.

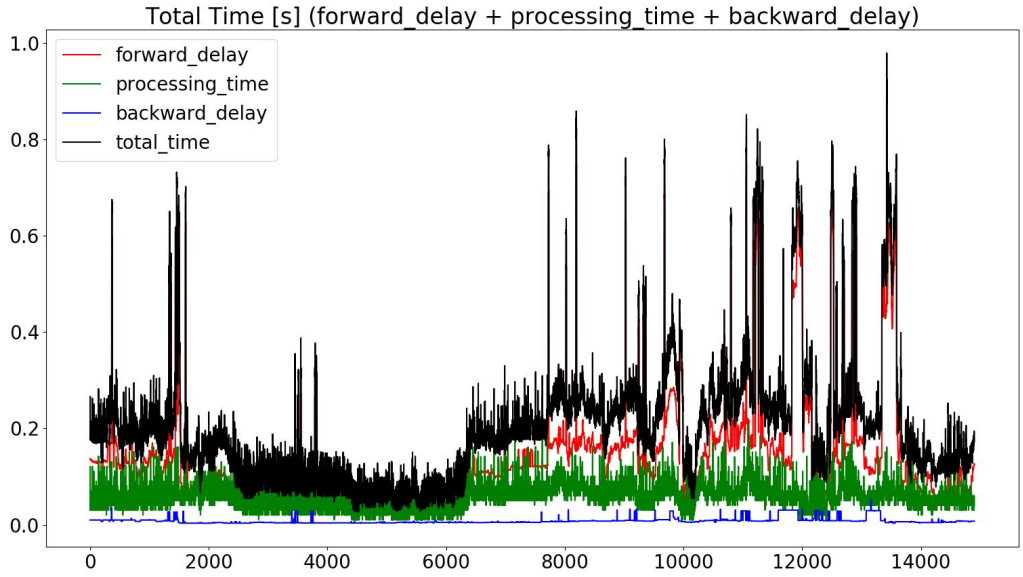


Figure 6.15: Evolution of the total processing time (transmission, edge-computation, re-transmission) for LOCUS offloading of spot1 robot during the exploration of the Beta 2 Course of the DARPA Subterranean Challenge on husky1 server. Operation time is comparable to on-board performance. Sudden drops in communication bandwidths to other hop of communication when exploring the lower floor result in the lidar message being delivered to the server side with some delay. Individual sub-components of the total offloading time are represented in red (forward delay), green (processing time), and blue (backward delay).

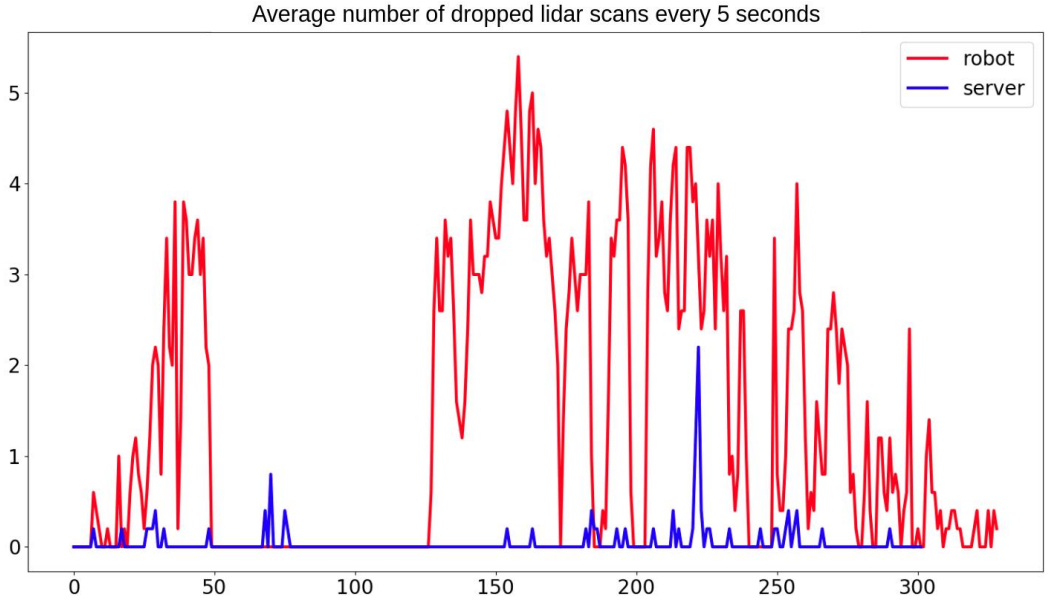


Figure 6.16: Average number of lidar scans dropped every 5 seconds. Husky1 acts as server for Spot1 LOCUS offloading. By using 4 threads for scan registration, the server is able to process much more information with respect to the client, despite the lower RDF percentage (80%). If LOCUS is executed on-board of spot, that results in a higher number of not processed information, and therefore in a lower ego-motion estimation accuracy.

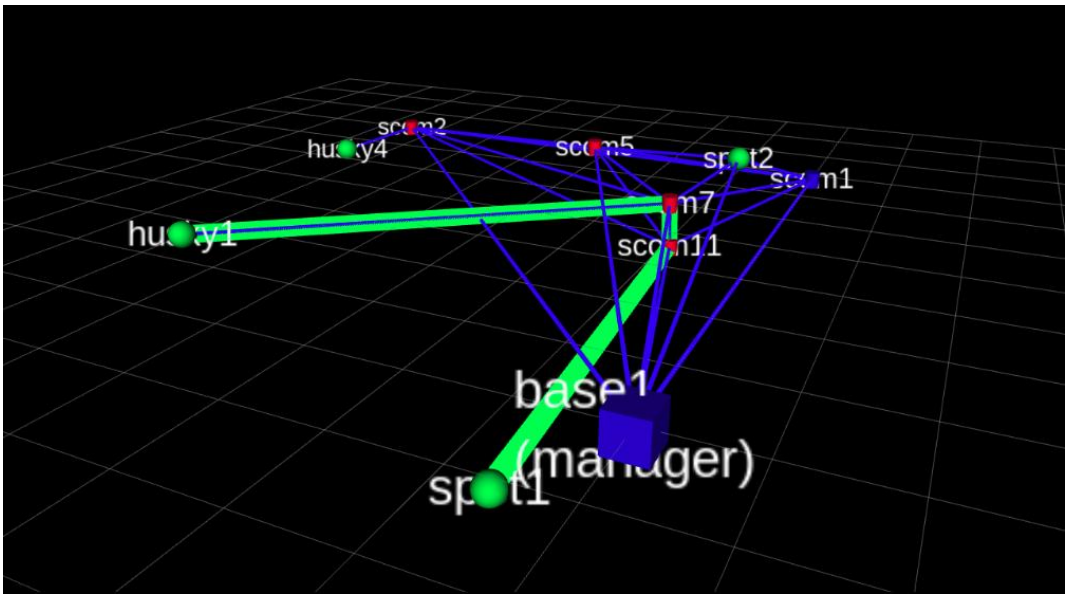


Figure 6.17: Snapshot of the optimum network route chosen by the Manager for LOCUS offloading of spot1 to the optimum server husky1. At this moment, spot1 is exploring the lower floor and continuously offloading its lidar stream to husky1 through scom11 and scom7 bridge; husky1 is receiving the data, processing it, and sending the computed high-definition odometric result back to the client. Spot1 therefore receives a high-definition estimate of its position in the environment at the front-end level, from the machine dynamically chosen by the Manager.

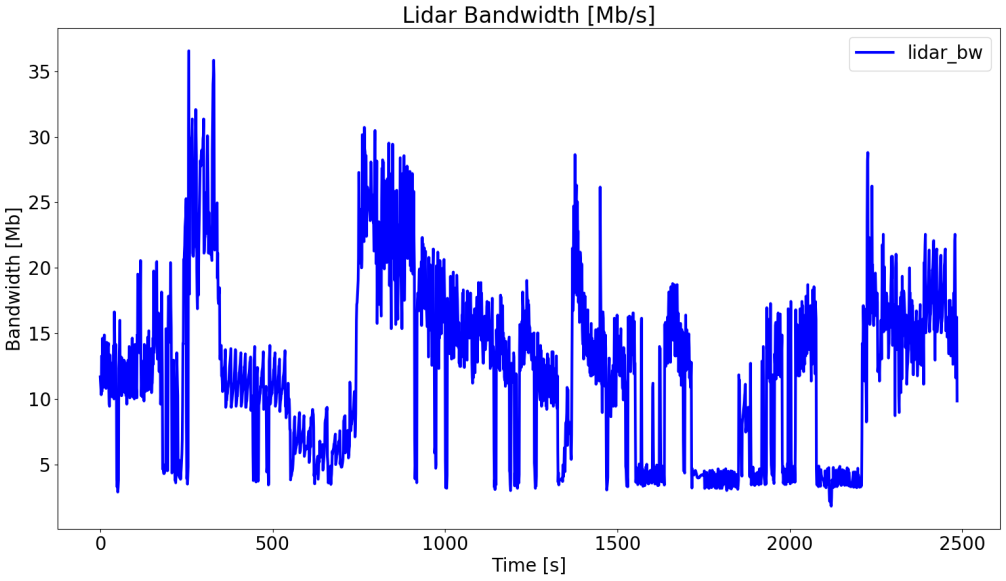


Figure 6.18: Evolution of the bandwidth usage for the offloaded lidar stream of spot1 robot during the exploration of hte Urban Beta 2 Circuit.

6.2.1 Autonomy Time

Fig. 6.19 shows how offloading LOCUS computation decreases the battery usage of the agent increasing its autonomy time. When not offloading LOCUS computation, spot1 experiences both background and foreground battery drain, arriving at end of mission with approximately 60% battery level. When offloading LOCUS computation spot1 saves power on-board, reaching end of mission with 85% battery level. Results are obtained with the method outlined in Section 6.1.1, where for the sake of simplicity we ignore foreground drain components introduced by other processes of the autonomy stack running on-board (e.g. traversability analysis, object detection, etc). Battery drain of server machines are not profiled as we assume availability of heavily equipped platforms whose energy resources are enough to prove this overall proof concept.

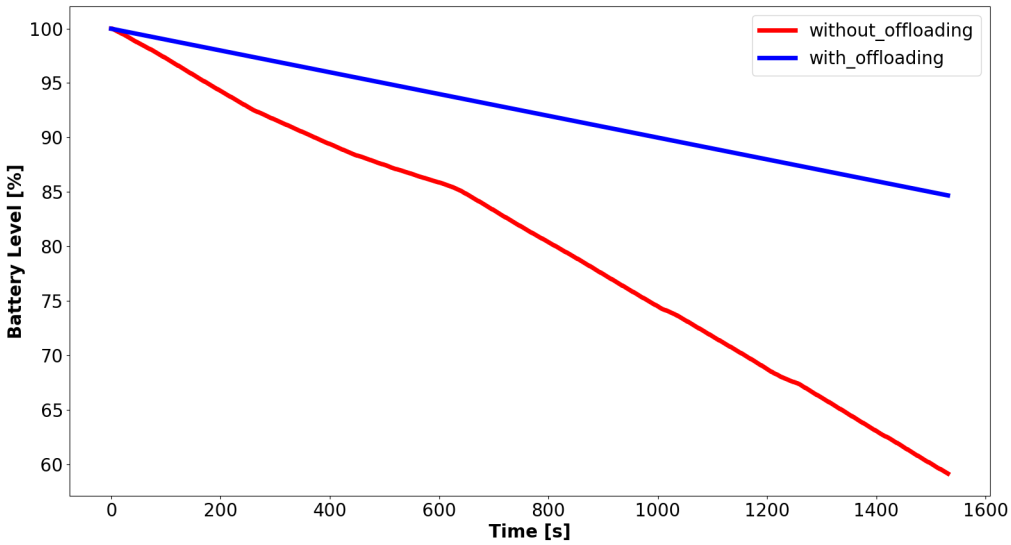


Figure 6.19: Comparison of spot1 battery level from start to end of mission in Urban Beta 2 with (blue line) and without (red line) LOCUS offloading. Offloading LOCUS computation results in decreased battery usage on the agent and increased autonomy time. Please note that while the full mission duration is 1 hour, spot1 exploration starts late at 1300s and ends early at 2800s: for this reason, the profiled battery drain is reported only for the exploration time of interest.

As expected, offloading computationally-expensive perception-related tasks results in a substantial decrease of battery usage in the robot, leading to increased autonomy, exploration time, and potential information gain.

6.3 YOLO Offloading Results

In this section we demonstrate enhancements on the situational awareness accuracy retrieved by computationally-constrained platforms of heterogeneous multi-robot systems on the semantic understanding level by means of YOLO offloading for a single client (spot1) to a single server (husky1) through the use of Swarm Manager. Video demo: `https://youtu.be/ZOpzOUx-XaQ`

In the comparison, when not offloading, spot1 needs to downsample the camera inputs logged at 424x240 px resolution at 10 Hz rate by a 50% factor to cope with constrained computation on-board and pursue real-time operation. As stated earlier, a possible shortcoming of this approach is that a downsampled version of the image might result in degraded YOLO detection performance as the more downsampled the image, the harder is for YOLO to successfully find the object of interest. Otherwise, when offloading, spot1 can stream its full resolution image feed (the average size of single image message is 2.4 Mb) to the optimum server and through the optimum route dynamically chosen by the Manager to take advantage of the server side computation at full-resolution, which runs YOLO on the full received image and provides back the client with a response if an artifact of interest was detected in the camera stream with suitable bandwidth requirements leading to near real-time operation at the communication level.

We analyze the number of detections produced by YOLO in the two cases and evaluate enhancements in object detection precision. As demonstrated below, the edge ground-server can exploit its advanced computational capabilities to run YOLO at full resolution, achieving therefore increased number of detections and enhanced precision increasing the overall reliability of the distribuited situational awareness retrieved from the heterogeneous multi-robot system in the unknown environment. While in the real mission ground-robots run YOLO and legged robots run Tiny-YOLO, we fix YOLO as single object detector module across all robots to benchmark enhancements achievable for a single system.

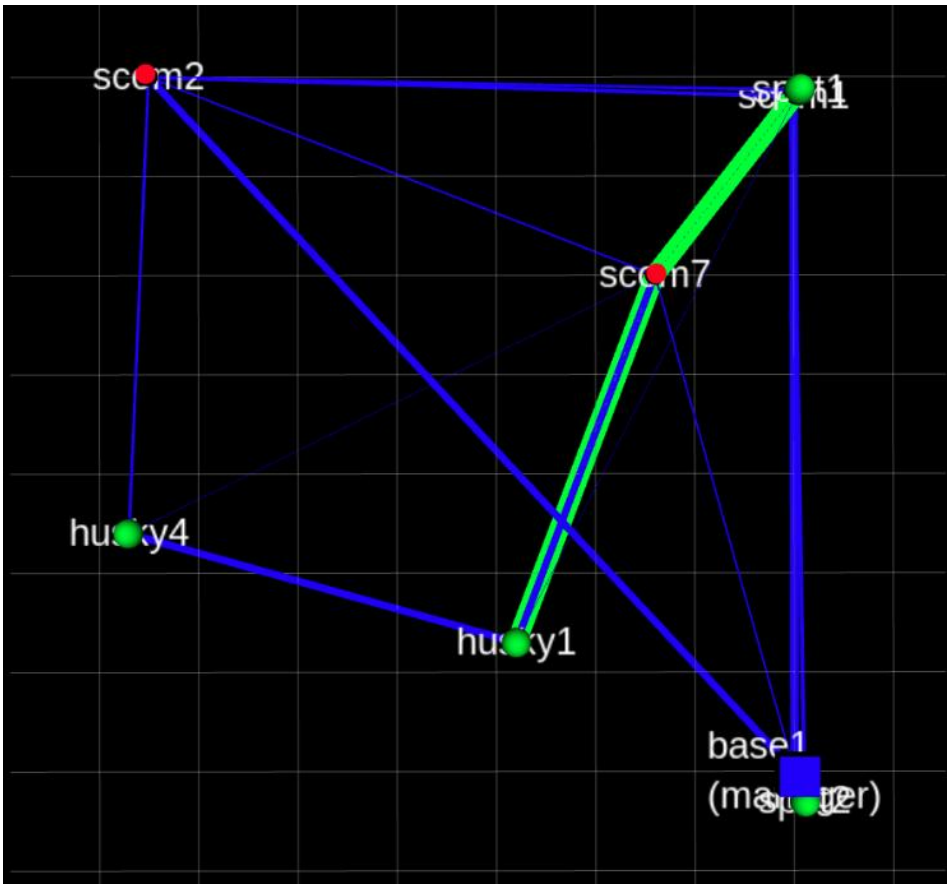


Figure 6.20: Snapshot of the optimum network route chosen by the Manager for YOLO offloading of spot1 to the optimum server husky1. At this moment, spot1 is entering an open-space area located on the first floor.

Figure 6.21: YOLO detections on spot1 camera stream when running on-board of the robot with a 50 % downsampling factor.

Figure 6.22: Evolution of the detection confidence of a backpack in the spot1 camera stream when running YOLO on husky1 server receiving the full-resolution image offloaded by spot1 through the route chosen by the Manager.

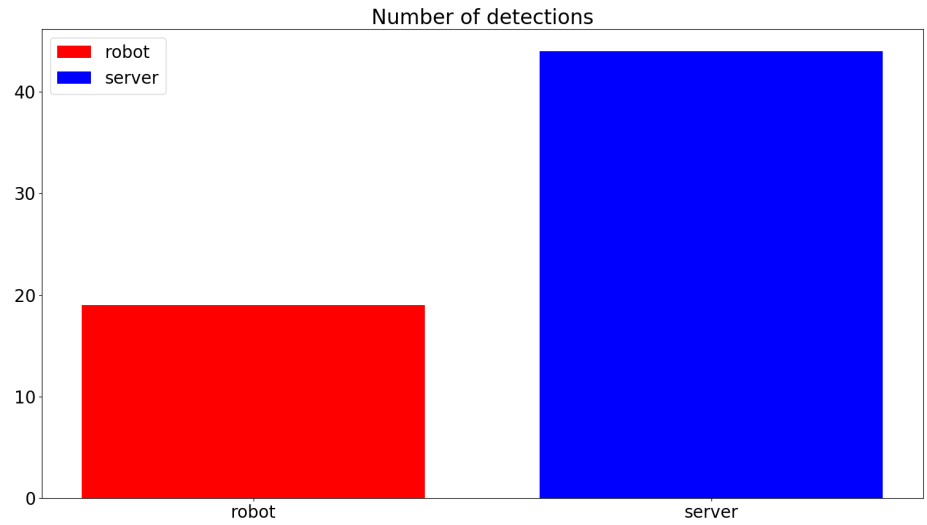


Figure 6.23: Barplot representing the total number of YOLO detections performed on the image stream for robot and server. When processing on-board of the robot, the image is downsampled of 50 % factor to cope with the constrained available resources: this results in a total of 19 total detections in this test. When offloading, the client can provide the server with the full-resolution image: in this case the server runs full YOLO on the received image stream, resulting in more detections, a total of 44 in this case. Among all the detections, in this run only a single TP is encountered (backpack): for this artifact the server-side detection provides higher detection confidence (96%) against the robot-side (85%)

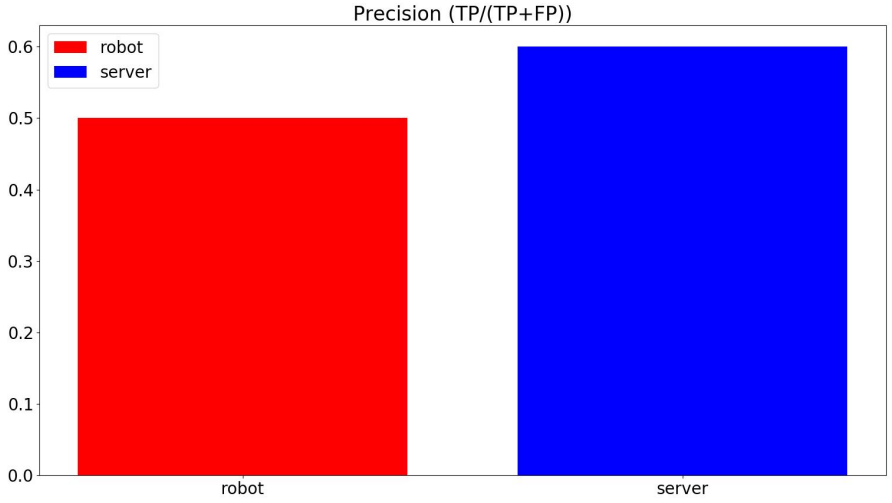


Figure 6.24: Comparison of YOLO detection precision (TP/(TP+FP)) computed with a 80% confidence filter when executed on client (red), and on the server (blue). The server side achieves higher precision than the robot side. As specified in Section 6.3, the same version of YOLO (YOLOv4) is used in the benchmarking on both robot and server.

6.3.1 Autonomy Time

Fig. 6.25 shows how offloading YOLO computation decreases the battery usage of the agent increasing its autonomy time. When not offloading YOLO computation, spot1 experiences both background and foreground battery drain, arriving at end of mission with approximately 80% battery level. When offloading YOLO computation spot1 saves power on-board, reaching end of mission with 86% battery level. Results are obtained with the method outlined in Section 6.1.1, where for the sake of simplicity we ignore foreground drain components introduced by other processes of the autonomy stack running on-board (e.g. traversability analysis, lidar-odometry, etc).

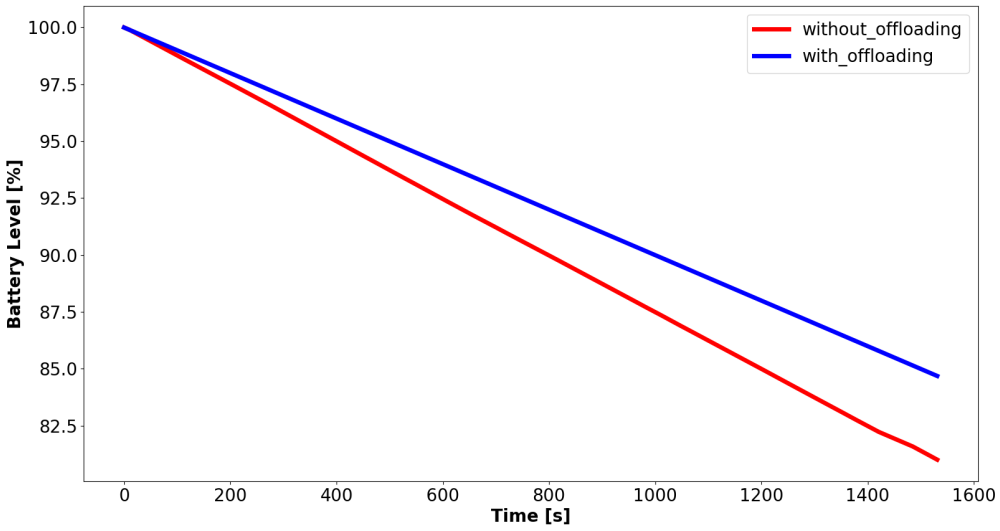


Figure 6.25: Comparison of spot1 battery level from start to end of mission in Urban Beta 2 with and without YOLO offloading. Offloading YOLO computation results in decreased battery usage on the agent and increased autonomy time. Please note that while the full mission duration is 1 hour, spot1 exploration starts late at 1300s and ends early at 2800s: for this reason, the profiled battery drain is reported only for the exploration time of interest.

As expected, offloading semantic understanding tasks results in a substantial decrease of battery usage in the robot, leading to increased autonomy, exploration time, and potential information gain.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Conclusions of this dissertation are two-fold, at the intersection of perception and networking.

First, achieving accurate lidar odometry in perceptually-challenging conditions can be difficult due to the lack of reliable perceptual features, presence of noisy sensor measurements, and high-rate motions. While integrating additional sensing modalities can help address these challenges, potential sensor failures can have dramatic impacts on the mission outcome if not robustly handled. In this dissertation, we present a lidar odometry system to enable accurate and resilient ego-motion estimation in challenging real-world scenarios. The proposed system, LOCUS, provides an accurate multi-stage scan matching unit equipped with an health-aware sensor integration module for seamless loose integration of additional sensing modalities. The proposed architecture is adaptable to heterogeneous robotic platforms and is optimized for real-time operation. We compare LOCUS against state-of-the-art open-source algorithms and demonstrate top-class accuracy in perceptually-challenging real-world datasets, top-class computation time and superior robustness to sensor failures, yet with greater CPU load. Finally, we demonstrate field-proven real-time operation of LOCUS on two different robots involved in fully autonomous exploration of Satsop power plant during the Urban Circuit of the DARPA Subterranean Challenge, where the proposed system was a key component of CoSTAR team's solution that achieved first place.

Second, retrieving accurate distributed situational awareness on a heterogeneous multi-robot system can be difficult due to potential constrained computational capabilities of less-capable agents (e.g. legged, aerial) limiting the accuracy of on-board localization and semantic understanding modules. While inter-agent offloading mechanisms supervised by a central entity could benefit the robotic team with distributed computation capabilities, the failure of the central orchestrator might translate into catastrophic consequences. In this dissertation, we present Swarm Manager, a novel framework that exploits Distributed Computation and Software Defined Networking paradigms to enhance perception accuracy of computationally constrained platforms in heterogeneous multi-robot systems, by allowing robots in the team to offload heavy computation (e.g. lidar odometry, object detection) to other more resourceful peers under decision of a globally-aware and dynamically eligible central orchestrator. For each offloading request, Swarm Manager simultaneously identifies the optimum server where is best to execute the task, and the optimum route through is best to route the data given the current knowledge of the system state. The presented approach provides resilience to the failure of the central orchestrator by means of a dynamic leader election mechanism. We demonstrate successful operation of Swarm Manager on the data collected by a team of four autonomous robots exploring the Satsop power plant during the Urban Circuit of the DARPA Subterranean Challenge and showcase improvements in localization accuracy by means of LOCUS offloading, and improvements in object detection precision by means of YOLO offloading.

We think that the concept of Swarm Manager opens up for new avenues of advanced cooperative behaviours in multi-robot teams. For example, a server performing ego-motion estimation and mapping on the lidar stream offloaded by a client, could use this information to look for potential inter-robot loop closures between its own map and the client's one.

7.2 Future Work

In the localization domain, future work will investigate novel point cloud filtering and point cloud registration strategies to improve accuracy of ego-motion estimation, along

with continuing efforts in decreasing the computational load of LOCUS as it is currently impacting the operation of computationally-constrained platforms.

In the distributed computation domain, future work will test Swarm Manager over large heterogeneous multi-robot systems during real-world field deployments.

Chapter 8

Publications

- *"LOCUS: A Multi-Sensor Lidar-Centric Solution for High-Precision Odometry and 3D Mapping in Real-Time"* Matteo Palieri, Benjamin Morrell, Abhishek Thakur, Kamak Ebadi, Jeremy Nash, Arghya Chatterjee, Christoforos Kanellakis, Luca Carlone, Cataldo Guaragnella, Ali-akbar Agha-mohammadi - IEEE Robotics and Automation Letters, 2020.
- *"LAMP: Large-Scale Autonomous Mapping and Positioning for Exploration of Perceptually-Degraded Subterranean Environments"* Kamak Ebadi, Yun Chang, Matteo Palieri, Alex Stephens, Alex Hatteland, Eric Heiden, Abhishek Thakur, Nobuhiro Funabiki, Benjamin Morrell, Sally Wood, Luca Carlone, Ali-akbar Agha-mohammadi - IEEE International Conference on Robotics and Automation, 2020.
- *"Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion"* Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D. Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, Ali-akbar Agha-mohammadi - IEEE International Conference on Intelligent Robots and Systems, 2020.
- *"DARE-SLAM: Degeneracy-Aware and Resilient Loop Closing in Perceptually-Degraded Environments"* Kamak Ebadi, Matteo Palieri, Sally Wood, Curtis Padgett, Ali-akbar Agha-mohammadi - Journal of Intelligent and Robotic Systems, 2021.

- *"Maximum Correntropy Kalman Filter for Orientation Estimation With Application to LiDAR Inertial Odometry"* Seyed Fakoorian, Matteo Palieri, Angel Santamaria-Navarro, Cataldo Guaragnella, Dan Simon, Ali-akbar Agha-mohammadi - Dynamic Systems and Control Conference, 2020.
- *"Power Management Framework for Optical Infrastructure"* Luigi Mantellini, Marco Mussini, Giorgio Parladori, Domenico Scarpelli, Matteo Palieri, Cataldo Guaragnella, Francesco Nicassio, Francesco Triggiani - IEEE International Conference on Transparent Optical Networks, 2020.
- *"NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge"* Ali Agha, Kyohei Otsu, Benjamin Morrell, David D. Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, Muhammad Fadhil Ginting, Kamak Ebadi, Matthew Anderson, Torkom Pailevanian, Edward Terry, Michael Wolf, Andrea Tagliabue, Tiago Stegun Vaquero, Matteo Palieri, et al. - Journal of Field Robotics, 2021.
- *"Swarm Manager: Distributed Situational Awareness on a Heterogeneous Multi-Robot System"* Matteo Palieri et al. Currently in submission process.

Bibliography

[1] Mica Endsley. 'Situation awareness analysis and measurement, chapter theoretical underpinnings of situation awareness'. In: *A Critical Review* (Jan. 2000), pp. 3–33 (cit. on p. 1).

[2] Lynne E Parker, Daniela Rus and Gaurav S Sukhatme. 'Multiple mobile robot systems'. In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1335–1384 (cit. on p. 1).

[3] T. Arai, E. Pagello and L.E. Parker. 'Guest editorial advances in multirobot systems'. In: *IEEE Transactions on Robotics and Automation* 18.5 (Oct. 2002), pp. 655–661. DOI: 10.1109/tra.2002.806024. URL: https://doi.org/10.1109/tra.2002.806024 (cit. on p. 2).

[4] Robert Bogue. 'Robots for monitoring the environment'. In: *Industrial Robot: An International Journal* (2011) (cit. on p. 2).

[5] Valerio Digani et al. 'Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy'. In: *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE. 2013, pp. 209–215 (cit. on p. 2).

[6] James S Jennings, Greg Whelan and William F Evans. 'Cooperative search and rescue with a team of mobile robots'. In: *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*. IEEE. 1997, pp. 193–200 (cit. on p. 2).

[7] Haris Balta et al. 'Integrated data management for Science & a fleet of search-and-rescue robots'. In: *Journal of Field Robotics* 34.3 (2017), pp. 539–582 (cit. on p. 2).

[8] Junichi Haruyama et al. 'Lunar holes and lava tubes as resources for lunar science and exploration'. In: *Moon*. Springer, 2012, pp. 139–163 (cit. on pp. 2, 3).

[9] Takahiro Sasaki et al. 'Where to map? iterative rover-copter path planning for mars exploration'. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2123–2130 (cit. on p. 2).

[10] M. Bajracharya, M. W. Maimone and D. Helmick. 'Autonomy for Mars Rovers: Past, Present, and Future'. In: *Computer* 41.12 (2008), pp. 44–50. DOI: 10.1109/MC.2008.479 (cit. on p. 2).

[11] Jordan Ford et al. 'Technologies Enabling the Exploration of Lunar Pits'. In: *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)* (2020) (cit. on p. 2).

[12] Keir Groves et al. 'Robotic Exploration of an Unknown Nuclear Environment Using Radiation Informed Autonomous Navigation'. In: *Robotics* 10.2 (2021), p. 78 (cit. on p. 2).

[13] URL: https://www.equipment-news.com/foton-motor-involved-rescue-mission-chiang-rai-thailand/ (cit. on p. 3).

[14] Andrew W Daga et al. 'Lunar and martian lava tube exploration as part of an overall scientific survey'. In: *Annual Meeting of the Lunar Exploration Analysis Group*. Vol. 1515. 2009, p. 15 (cit. on p. 3).

[15] Glen Cushing. 'Candidate Cave Entrances on Mars'. In: *Journal of Cave and Karst Studies, Vol. 74, p. 33-47* 74 (Apr. 2012), pp. 33–47. DOI: 10.4311/2010EX0167R (cit. on p. 3).

[16] Samuel B Kesner et al. 'Mobility and power feasibility of a microbot team system for extraterrestrial cave exploration'. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 4893–4898 (cit. on p. 3).

[17] Timothy Titus et al. 'Science and technology requirements to explore caves in our Solar System'. In: *White Paper* (July 2020). DOI: 10.13140/RG.2.2.22126.43844 (cit. on p. 3).

[18] Vlada Stamenković et al. 'The next frontier for planetary and human exploration'. In: *Nature Astronomy* 3.2 (2019), pp. 116–120 (cit. on p. 3).

[19] T. Touma et al. 'Mars Dogs: Biomimetic Robots for the Exploration of Mars, from its Rugged Surface to its Hidden Caves'. In: *AGU Fall Meeting*. 2020 (cit. on p. 3).

[20] URL: https://www.express.co.uk/news/science/1110234/nasa-announcement-moon-settlement-sea-of-tranquility-lunar-mission (cit. on p. 4).

[21] DARPA. *DARPA Subterranean Challenge*. https://www.subtchallenge.com. 2018 (cit. on p. 7).

[22] URL: https://www.darpa.mil/program/darpa-subterranean-challenge (cit. on p. 7).

[23] Ali Agha et al. 'Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge'. In: *arXiv preprint arXiv:2103.11470* (2021) (cit. on pp. 8–10, 24, 31, 70).

[24] URL: http://wiki.ros.org/melodic (cit. on p. 11).

[25] Khalid Yousif, Alireza Bab-Hadiashar and Reza Hoseinnezhad. 'An overview to visual odometry and visual SLAM: Applications to mobile robotics'. In: *Intelligent Industrial Systems* 1.4 (2015), pp. 289–311 (cit. on p. 14).

[26] C. Cadena et al. 'Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age'. In: *IEEE Transactions on Robotics* 6 (2016), 1309–1332 (cit. on p. 14).

[27] Randall C Smith and Peter Cheeseman. 'On the representation and estimation of spatial uncertainty'. In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68 (cit. on p. 14).

[28] John J Leonard and Hugh F Durrant-Whyte. 'Simultaneous map building and localization for an autonomous mobile robot.' In: *IROS*. Vol. 3. 1991, pp. 1442–1447 (cit. on p. 14).

[29] Feng Lu and Evangelos Milios. 'Globally consistent range scan alignment for environment mapping'. In: *Autonomous robots* 4.4 (1997), pp. 333–349 (cit. on p. 14).

[30] Michael Bloesch et al. 'Robust visual inertial odometry using a direct EKF-based approach'. In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 298–304 (cit. on pp. 14, 16).

[31] Andrew J Davison et al. 'MonoSLAM: Real-time single camera SLAM'. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067 (cit. on p. 14).

[32] Tim Bailey and Hugh Durrant-Whyte. 'Simultaneous localization and mapping (SLAM): Part II'. In: *IEEE robotics & automation magazine* 13.3 (2006), pp. 108–117 (cit. on p. 14).

[33] Michael Calonder. *EKF SLAM vs. FastSLAM–A comparison*. Tech. rep. 2006 (cit. on p. 14).

[34] Zheng Fang et al. 'Robust autonomous flight in constrained and visually degraded shipboard environments'. In: *Journal of Field Robotics* 34.1 (2017), pp. 25–52 (cit. on p. 14).

[35] Michael Montemerlo and Sebastian Thrun. 'Simultaneous localization and mapping with unknown data association using FastSLAM'. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 2. IEEE. 2003, pp. 1985–1991 (cit. on p. 14).

[36] Francisco J Perez-Grau et al. 'An architecture for robust UAV navigation in GPS-denied areas'. In: *Journal of Field Robotics* 35.1 (2018), pp. 121–145 (cit. on p. 14).

[37] Jakob Engel, Vladlen Koltun and Daniel Cremers. 'Direct sparse odometry'. In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625 (cit. on p. 14).

[38] Jakob Engel, Jörg Stückler and Daniel Cremers. 'Large-scale direct SLAM with stereo cameras'. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 1935–1942 (cit. on p. 14).

[39] Stefan Leutenegger et al. 'Keyframe-based visual–inertial odometry using nonlinear optimization'. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334 (cit. on pp. 14, 16).

[40] Michael Milford and Gordon Wyeth. 'Persistent navigation and mapping using a biologically inspired SLAM system'. In: *The International Journal of Robotics Research* 29.9 (2010), pp. 1131–1153 (cit. on p. 14).

[41] Raul Mur-Artal and Juan D Tardós. 'Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras'. In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262 (cit. on p. 14).

[42] Frank R Kschischang, Brendan J Frey and H-A Loeliger. 'Factor graphs and the sum-product algorithm'. In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519 (cit. on p. 14).

[43] Jorge J Moré. 'The Levenberg-Marquardt algorithm: implementation and theory'. In: *Numerical analysis*. Springer, 1978, pp. 105–116 (cit. on p. 15).

[44] Baichuan Huang, Jun Zhao and Jingbin Liu. 'A survey of simultaneous localization and mapping with an envision in 6g wireless networks'. In: *arXiv preprint arXiv:1909.05214* (2019) (cit. on p. 16).

[45] Jorge Fuentes-Pacheco, José Ruiz-Ascencio and Juan Manuel Rendón-Mancha. 'Visual simultaneous localization and mapping: a survey'. In: *Artificial intelligence review* 43.1 (2015), pp. 55–81 (cit. on p. 16).

[46] T. Qin, P. Li and S. Shen. 'Vins-mono: A robust and versatile monocular visual-inertial state estimator'. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020 (cit. on pp. 16, 50).

[47] Thomas Schneider et al. 'maplab: An open framework for research in visual-inertial mapping and localization'. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1418–1425 (cit. on p. 16).

[48] Antoni Rosinol et al. 'Kimera: an open-source library for real-time metric-semantic localization and mapping'. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1689–1696 (cit. on pp. 16, 70).

[49] Davide Scaramuzza and Friedrich Fraundorfer. 'Visual odometry [tutorial]'. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92 (cit. on p. 16).

[50] David Nistér, Oleg Naroditsky and James Bergen. 'Visual odometry'. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. Ieee. 2004, pp. I–I (cit. on p. 16).

[51] Tony Lindeberg. 'Scale invariant feature transform'. In: (2012) (cit. on p. 16).

[52] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. 'Surf: Speeded up robust features'. In: *European conference on computer vision*. Springer. 2006, pp. 404–417 (cit. on p. 16).

[53] Ethan Rublee et al. 'ORB: An efficient alternative to SIFT or SURF'. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571 (cit. on p. 16).

[54] Sunglok Choi, Jaehyun Park and Wonpil Yu. 'Resolving scale ambiguity for monocular visual odometry'. In: *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2013, pp. 604–608 (cit. on p. 16).

[55] Mingyang Li and Anastasios I Mourikis. 'High-precision, consistent EKF-based visual-inertial odometry'. In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711 (cit. on p. 16).

[56] Guoquan Huang, Michael Kaess and John J Leonard. 'Towards consistent visual-inertial navigation'. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 4926–4933 (cit. on p. 16).

[57] Christian Forster et al. 'On-manifold preintegration for real-time visual–inertial odometry'. In: *IEEE Transactions on Robotics* 33.1 (2016), pp. 1–21 (cit. on p. 16).

[58] Raúl Mur-Artal and Juan D Tardós. 'Visual-inertial monocular SLAM with map reuse'. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803 (cit. on p. 16).

[59] Danping Zou, Ping Tan and Wenxian Yu. 'Collaborative visual SLAM for multiple agents: A brief survey'. In: *Virtual Reality & Intelligent Hardware* 1.5 (2019), pp. 461–482 (cit. on p. 16).

[60] Nishant Kejriwal, Swagat Kumar and Tomohiro Shibata. 'High performance loop closure detection using bag of word pairs'. In: *Robotics and Autonomous Systems* 77 (2016), pp. 55–65 (cit. on p. 16).

[61] Relja Arandjelovic et al. 'NetVLAD: CNN architecture for weakly supervised place recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5297–5307 (cit. on p. 16).

[62] Paulo Vinicius Koerich Borges and Stephen Vidas. 'Practical infrared visual odometry'. In: *IEEE Transactions on Intelligent Transportation Systems* 17.8 (2016), pp. 2205–2213 (cit. on p. 17).

[63] T. Dang C. Papachristos S. Khattak F. Mascarich and K. Alexis. 'Robust Thermal-Inertial Localization for Aerial Robots: A Case for Direct Methods'. In: *International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 1061–1068 (cit. on p. 17).

[64] A. Nuchter et al. '6D SLAM with an application in autonomous mine mapping'. In: *Conference on Robotics and Automation, 2004*. Ed. by Ieee International. vol. 2, . IEEE: Proceedings. ICRA'04. 2004, 2004, pp. 1998–2003 (cit. on p. 17).

[65] T. Shan et al. 'LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping'. In: *IEEE/RSJ IROS*. IEEE. 2020 (cit. on pp. 17, 21, 22, 50).

[66] T. Shan and B. Englot. 'LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain'. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4758–4765 (cit. on pp. 17, 18, 21, 22, 50).

[67] J. Zhang and S. Singh. 'LOAM: Lidar Odometry and Mapping in Real-time'. In: *Robotics: Science and Systems, vol. 2*. IEEE. 2014, p. 9 (cit. on pp. 17, 21, 22, 50).

[68] X. Ji et al. 'LLOAM: LiDAR Odometry and Mapping with Loop-closure Detection Based Correction'. In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2475–2480 (cit. on p. 17).

[69] D. Kohler, H. Rapp and D. Andor. 'Real-time loop closure in 2D LIDAR SLAM'. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1271–1278 (cit. on pp. 17, 22, 50).

[70] R. Dubé et al. *SegMap: 3D segment mapping using data-driven descriptors*. preprint. arXiv, 2018. arXiv: 1804.09557 (cit. on p. 17).

[71] Cedric Le Gentil, Teresa Vidal-Calleja and Shoudong Huang. 'IN2LAAMA: Inertial Lidar Localization Autocalibration and Mapping'. In: *IEEE Transactions on Robotics* (2020) (cit. on pp. 17, 21, 22).

[72] Mikaela Angelina Uy and Gim Hee Lee. 'Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition'. In: *Proceedings of the IEEE Confer-*

ence on Computer Vision and Pattern Recognition. 2018, pp. 4470–4479 (cit. on p. 17).

[73] Renaud Dubé et al. 'Segmatch: Segment based loop-closure for 3d point clouds'. In: *arXiv preprint arXiv:1609.07720* (2016) (cit. on p. 17).

[74] Xieyuanli Chen et al. 'OverlapNet: Loop closing for LiDAR-based SLAM'. In: *arXiv preprint arXiv:2105.11344* (2021) (cit. on p. 17).

[75] Kamak Ebadi et al. 'DARE-SLAM: Degeneracy-Aware and Resilient Loop Closing in Perceptually-Degraded Environments'. In: *arXiv preprint arXiv:2102.05117* (2021) (cit. on pp. 17, 18, 70).

[76] Andrea Tagliabue et al. 'LION: Lidar-Inertial observability-aware navigator for Vision-Denied environments'. In: *arXiv preprint arXiv:2102.03443* (2021) (cit. on pp. 17, 22).

[77] S. Singh J. Zhang. 'Laser–visual–inertial odometry and mapping with high robustness and low drift'. In: *Journal of Field Robotics*. 2018, pp. 1242–1264 (cit. on pp. 17, 21, 22, 50).

[78] Yoshua Nava. 'Visual-LiDAR SLAM with loop closure'. PhD thesis. KTH Royal Institute of Technology, 2018 (cit. on p. 17).

[79] José-Luis Blanco-Claraco. 'A Modular Optimization Framework for Localization and Mapping.' In: *Robotics: Science and Systems*. 2019 (cit. on p. 17).

[80] J. M. M. Montiel R. Mur-Artal and J. D. Tardos. 'ORB-SLAM: a versatile and accurate monocular SLAM system'. In: *IEEE transactions on robotics 31, no. 5*. IEEE. 2015, pp. 1147–1163 (cit. on p. 18).

[81] S. Leutenegger et al. 'Keyframe-based visual–inertial odometry using nonlinear optimization'. In: *nternational Journal of Robotics Research, vol. 34, no. 3*. 2015, pp. 314–334 (cit. on p. 18).

[82] M. Hutter M. Bloesch S. Omari and R. Siegwart. 'Robust visual inertial odometry using a direct ekf-based approach'. In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 298–304 (cit. on p. 18).

[83] D. Tardioli et al. 'Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments'. In: *Journal of Field Robotics* 36.6 (2019), pp. 1074–1101 (cit. on p. 18).

[84] Kamak Ebadi et al. 'LAMP: Large-Scale Autonomous Mapping and Positioning for Exploration of Perceptually-Degraded Subterranean Environments'. In: *ICRA*. IEEE. 2020 (cit. on pp. 18, 21, 22, 45, 63, 70).

[85] Paul J Besl and Neil D McKay. 'Method for registration of 3-D shapes'. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606 (cit. on p. 19).

[86] URL: http://www.lix.polytechnique.fr/~maks/Verona_MPAM/TD/TD1/ (cit. on p. 20).

[87] Szymon Rusinkiewicz and Marc Levoy. 'Efficient variants of the ICP algorithm'. In: *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE. 2001, pp. 145–152 (cit. on p. 20).

[88] Yang Chen and Gérard Medioni. 'Object modelling by registration of multiple range images'. In: *Image and vision computing* 10.3 (1992), pp. 145–155 (cit. on p. 20).

[89] Aleksandr Segal, Dirk Haehnel and Sebastian Thrun. 'Generalized-icp.' In: *Robotics: science and systems*. Vol. 2. 4. Seattle, WA. 2009, p. 435 (cit. on pp. 20, 21).

[90] José-Luis Blanco-Claraco. 'A Modular Optimization Framework for Localization and Mapping.' In: *Robotics: Science and Systems*. 2019 (cit. on pp. 21, 22).

[91] M. Bosse and R. Zlot. 'Continuous 3D scan-matching with a spinning 2D laser'. In: *ICRA*. 2009, pp. 4312–4319 (cit. on pp. 21, 22).

[92] 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)*. Vol. 3. IEEE. 2003, pp. 2743–2748 (cit. on pp. 21, 22).

[93] et al. S. Agarwal K. Mierle. 'Ceres solver available Online:' in: *http://ceres-solver.org* (cit. on p. 21).

[94] Giorgio Grisetti et al. 'Least Squares Optimization: from Theory to Practice'. In: *Robotics* 9.3 (2020), p. 51. DOI: 10.3390/robotics9030051 (cit. on p. 21).

[95] W. Shao et al. 'Stereo visual inertial lidar simultaneous localization and mapping'. In: *IROS*. 2019 (cit. on pp. 21, 22, 39, 50).

[96] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. Georgia Institute of Technology, 2012 (cit. on p. 21).

[97] Heng Yang, Jingnan Shi and Luca Carlone. 'Teaser: Fast and certifiable point cloud registration'. In: *arXiv preprint arXiv:2001.07715* (2020) (cit. on p. 21).

[98] H. Ye, Y. Chen and M. Liu. 'Tightly coupled 3d lidar inertial odometry and mapping'. In: *ICRA*. 2019, pp. 3144–3150 (cit. on pp. 22, 39, 50).

[99] Eijiro Takeuchi and Takashi Tsubouchi. 'A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping'. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3068–3073 (cit. on p. 22).

[100] R. Zlot and M. Bosse. 'Three-dimensional mobile mapping of caves.' In: *Journal of Cave & Karst Studies* 76.3 (2014) (cit. on p. 22).

[101] R. Zlot and M. Bosse. 'Efficient large-scale three-dimensional mobile mapping for underground mines'. In: *Journal of Field Robotics* 31.5 (2014), pp. 758–779 (cit. on p. 22).

[102] M. Bosse, R. Zlot and P. Flick. 'Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping'. In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1104–1119 (cit. on p. 22).

[103] URL: https://commercialsecurityinstallers.com/commercial-video-analytics/ (cit. on p. 23).

[104] Christian Szegedy et al. 'Going deeper with convolutions'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (cit. on p. 24).

[105] Wei Liu et al. 'Ssd: Single shot multibox detector'. In: *European conference on computer vision*. Springer. 2016, pp. 21–37 (cit. on p. 24).

[106] Joseph Redmon et al. 'You only look once: Unified, real-time object detection'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788 (cit. on p. 24).

[107] Joseph Redmon and Ali Farhadi. 'YOLO9000: better, faster, stronger'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271 (cit. on p. 24).

[108] Joseph Redmon and Ali Farhadi. 'Yolov3: An incremental improvement'. In: *arXiv preprint arXiv:1804.02767* (2018) (cit. on p. 24).

[109] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao. 'YOLOv4: Optimal Speed and Accuracy of Object Detection'. In: *arXiv preprint arXiv:2004.10934* (2020) (cit. on p. 24).

[110] Wadim Kehl et al. 'Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation'. In: *European conference on computer vision*. Springer. 2016, pp. 205–220 (cit. on p. 24).

[111] Waleed Ali et al. 'Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud'. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0 (cit. on p. 24).

[112] Yin Zhou and Oncel Tuzel. 'Voxelnet: End-to-end learning for point cloud based 3d object detection'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499 (cit. on p. 24).

[113] Martin Simony et al. 'Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds'. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0 (cit. on p. 24).

[114] Changhyun Choi and Henrik I Christensen. '3d textureless object detection and tracking: An edge-based approach'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 3877–3884 (cit. on p. 24).

[115] Zhong-Qiu Zhao et al. 'Object detection with deep learning: A review'. In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232 (cit. on p. 24).

[116] Edward Terry et al. 'Object and Gas Source Detection with Robotic Platforms in Perceptually-Degraded Environments'. In: *RSS Workshop: Robots in the Wild: Challenges in Deploying Robust Autonomy for Robotic Exploration*. 2020 (cit. on p. 24).

[117] Yunyang Xiong et al. 'MobileDets: Searching for Object Detection Architectures for Mobile Accelerators'. In: *arXiv preprint arXiv:2004.14525* (2020) (cit. on p. 24).

[118] Guoqiang Hu, Wee Peng Tay and Yonggang Wen. 'Cloud robotics: architecture, challenges and applications'. In: *IEEE network* 26.3 (2012), pp. 21–28 (cit. on p. 25).

[119] Dominique Hunziker et al. 'Rapyuta: The roboearth cloud engine'. In: *2013 IEEE international conference on robotics and automation*. IEEE. 2013, pp. 438–444 (cit. on p. 25).

[120] Gajamohan Mohanarajah et al. 'Rapyuta: A cloud robotics platform'. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2014), pp. 481–493 (cit. on p. 25).

[121] Kiril Antevski et al. 'Enhancing edge robotics through the use of context information'. In: *Proceedings of the Workshop on Experimentation and Measurements in 5G*. 2018, pp. 7–12 (cit. on p. 25).

[122] Ali Marjovi, Sarvenaz Choobdar and Lino Marques. 'Robotic clusters: Multi-robot systems as computer clusters: A topological map merging demonstration'. In: *Robotics and Autonomous Systems* 60.9 (2012), pp. 1191–1204 (cit. on p. 26).

[123] Brian P Gerkey and Maja J Mataric. 'Sold!: Auction methods for multirobot coordination'. In: *IEEE transactions on robotics and automation* 18.5 (2002), pp. 758–768 (cit. on p. 26).

[124] Yu Zhang and Lynne E Parker. 'Multi-robot task scheduling'. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 2992–2998 (cit. on p. 26).

[125] Paulo AL Rego et al. 'Performing computation offloading on multiple platforms'. In: *Computer Communications* 105 (2017), pp. 1–13 (cit. on p. 27).

[126] Victor Kathan Sarker et al. 'Offloading slam for indoor mobile robots with edge-fog-cloud computing'. In: *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*. IEEE. 2019, pp. 1–6 (cit. on p. 27).

[127] Joshua Vander Hook et al. 'Mars on-site shared analytics information and computing'. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 29. 2019, pp. 707–715 (cit. on pp. 27, 28).

[128] Federico Rossi et al. 'The Pluggable Distributed Resource Allocator (PDRA): a Middleware for Distributed Computing in Mobile Robotic Networks'. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 4337–4344 (cit. on p. 28).

[129] Karthik Kumar et al. 'A Survey of Computation Offloading for Mobile Systems'. In: *Mobile Networks and Applications* 18.1 (Apr. 2012), pp. 129–140. DOI: 10.1007/s11036-012-0368-0. URL: https://doi.org/10.1007/s11036-012-0368-0 (cit. on p. 28).

[130] Luca Ballotta, Luca Schenato and Luca Carlone. 'Computation-communication trade-offs and sensor selection in real-time estimation for processing networks'. In: *IEEE Transactions on Network Science and Engineering* 7.4 (2020), pp. 2952–2965 (cit. on p. 28).

[131] Akhlaqur Rahman et al. 'Energy-efficient optimal task offloading in cloud networked multi-robot systems'. In: *Computer Networks* 160 (2019), pp. 11–32 (cit. on p. 28).

[132] Zicong Hong et al. 'Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments'. In: *IEEE Transactions on Parallel and Distributed Systems* 30.12 (2019), pp. 2759–2774 (cit. on p. 28).

[133] Baichuan Liu et al. 'Joint computation offloading and routing optimization for uav-edge-cloud computing environments'. In: *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2018, pp. 1745–1752 (cit. on p. 28).

[134] Biwei Li et al. 'A high efficient multi-robot simultaneous localization and mapping system using partial computing offloading assisted cloud point registration strategy'. In: *Journal of Parallel and Distributed Computing* 149 (Mar. 2021), pp. 89–102. DOI: 10.1016/j.jpdc.2020.10.012. URL: https://doi.org/10.1016/j.jpdc.2020.10.012 (cit. on p. 28).

[135] Swarnava Dey and Arijit Mukherjee. 'Robotic slam: a review from fog computing and mobile edge computing perspective'. In: *Adjunct Proceedings of the*

13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services. 2016, pp. 153–158 (cit. on p. 29).

[136] Li Qingqing et al. 'Offloading Monocular Visual Odometry with Edge Computing'. In: *Proceedings of the 2019 5th International Conference on Systems, Control and Communications*. ACM, Dec. 2019. DOI: 10.1145/3377458.3377467. URL: https://doi.org/10.1145/3377458.3377467 (cit. on p. 29).

[137] L. Qingqing et al. '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)*. IEEE, Nov. 2019. DOI: 10.23919/icmu48249.2019.9006646. URL: https://doi.org/10.23919/icmu48249.2019.9006646 (cit. on p. 29).

[138] Swarnava Dey and Arijit Mukherjee. 'Robotic SLAM'. In: *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*. ACM, Nov. 2016. DOI: 10.1145/3004010.3004032. URL: https://doi.org/10.1145/3004010.3004032 (cit. on p. 29).

[139] Bruno Duarte Gouveia et al. 'Computation Sharing in Distributed Robotic Systems: A Case Study on SLAM'. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 410–422. DOI: 10.1109/tase.2014.2357216. URL: https://doi.org/10.1109/tase.2014.2357216 (cit. on p. 29).

[140] Jixiang Zhu. *Computation offloading and task scheduling among multi-robot systems*. 2017 (cit. on p. 29).

[141] Kyohei Otsu et al. 'Supervised autonomy for communication-degraded subterranean exploration by a robot team'. In: *2020 IEEE Aerospace Conference*. IEEE. 2020, pp. 1–9 (cit. on p. 30).

[142] Tiago Stegun Vaquero et al. 'Traversability-aware signal coverage planning for communication node deployment in planetary cave exploration'. In: *The International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. 2020 (cit. on p. 31).

[143] Muhammad Fadhil Ginting et al. 'CHORD: Distributed data-sharing via hybrid ROS 1 and 2 for multi-robot exploration of large-scale complex environments'.

In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5064–5071 (cit. on p. 31).

[144] Tiago Vaquero et al. 'Energy-aware data routing for disruption tolerant networks in planetary cave exploration'. In: (2019) (cit. on p. 31).

[145] Bruno Astuto A Nunes et al. 'A survey of software-defined networking: Past, present, and future of programmable networks'. In: *IEEE Communications surveys & tutorials* 16.3 (2014), pp. 1617–1634 (cit. on p. 33).

[146] Nick McKeown et al. 'OpenFlow: enabling innovation in campus networks'. In: *ACM SIGCOMM computer communication review* 38.2 (2008), pp. 69–74 (cit. on p. 33).

[147] Riccardo Trivisonno et al. 'SDN-based 5G mobile networks: architecture, functions, procedures and backward compatibility'. In: *Transactions on Emerging Telecommunications Technologies* 26.1 (2015), pp. 82–92 (cit. on p. 33).

[148] Konstantinos Poularakis et al. 'Bringing SDN to the mobile edge'. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2017, pp. 1–6 (cit. on p. 33).

[149] Liang Zhao et al. 'A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading'. In: *IEEE Transactions on Intelligent Transportation Systems* 22.6 (2020), pp. 3664–3674 (cit. on p. 33).

[150] Matteo Palieri et al. 'Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time'. In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 421–428 (cit. on pp. 37, 65).

[151] Ali Agha. *CoSTAR team website*. 2020. URL: https://costar.jpl.nasa.gov/ (cit. on p. 37).

[152] Jeffrey Scott Vitter. 'Faster methods for random sampling'. In: *Communications of the ACM* 27.7 (1984), pp. 703–718 (cit. on p. 41).

[153] Radu Bogdan Rusu and Steve Cousins. '3d is here: Point cloud library (pcl)'. In: *ICRA*. IEEE. 2011, pp. 1–4 (cit. on pp. 41, 46).

[154] Angel Santamaria-navarro et al. 'Towards Resilient Autonomous Navigation of Drones'. In: *International Symposium on Robotics Research*. 2019 (cit. on pp. 42, 43, 63).

[155] A. Bouman et al. 'Autonomous Spot: Long-range Autonomous Exploration of Extreme Environments with Legged Locomotion'. In: *IROS 2020* () (cit. on pp. 43, 46).

[156] Seyed Fakoorian et al. 'Maximum Correntropy Kalman Filter for Orientation Estimation With Application to LiDAR Inertial Odometry'. In: *Dynamic Systems and Control Conference*. Vol. 84270. American Society of Mechanical Engineers. 2020, V001T05A008 (cit. on p. 45).

[157] Eric Nelson. *Berkley Localization and Mapping*. 2016. URL: https://github.com/erik-nelson/blam (cit. on p. 50).

[158] Shibo Zhao et al. 'Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments'. In: *arXiv preprint arXiv:2104.14938* (2021) (cit. on p. 50).

[159] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017 (cit. on p. 51).

[160] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart. 'Exploring Network Structure, Dynamics, and Function using NetworkX'. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11 –15 (cit. on p. 80).