DECENTRALIZED MULTI-ROBOT SLAM AND AD HOC NETWORK FOR EXPLORATION IN A REMOTE AND ENCLOSED ENVIRONMENT

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

MAY 2022

By

Tetsuya Idota

Dissertation Committee:

Kyungim Baek, Chairperson
Edoardo Biagioni
Kim Binsted
Depeng Li
Monique Chyba

Keywords: multi-robot SLAM, ad hoc network, decentralization, enclosed environment

ABSTRACT

Robotic exploration in an enclosed, unknown, and unstructured environment such as a cave is a challenging problem as the robots cannot directly communicate with any exterior facilities. In such an environment, the robots need to perform Simultaneous Localization and Mapping (SLAM) without any external supports, e.g., GPS, a prior map, wifi access points, and so on. To cope with this situation, the proposed method establishes a decentralized multi-robot system forming an ad hoc network. Each robot is deployed at the entrance of the environment and moves into deeper areas while maintaining the distances to its neighboring robots to locally communicate with each other. In this formation, they perform the decentralized cooperative localization and the distributed submap building.

The decentralized cooperative localization enables the robots to localize themselves with respect to the global reference frame by taking mutual measurements on a relative pose between the robots. Communicating with neighboring robots, each robot updates its estimated location based on sensory data about relative poses with respect to the neighbors and their estimated locations. To deal with the overconfidence problem, which leads to inconsistent estimates due to cyclic updates, the proposed method performs the conservative data exchange. When the robots pass their estimation to each other, they reduce the confidence in their estimation by applying fractional exponents to the probability distributions. As a result, the distributions become "smoother" with less intense peaks. Thereby, the robots can avoid inconsistent amplification of confidence and update their estimation based on each other's information without the knowledge about the entire network topology.

The distributed submap building directs the robots to cooperatively build submaps, each of which represents a small local area in the environment. Each robot builds a series of submaps along its trajectory. When the robot detects that its neighboring robots enter the areas where submaps were built previously and can no longer update the submaps by itself, it passes the submaps to such robots so that the submaps can be used by the receiving robots. Thus, each of the robots does not have to hold all the submaps that it has created from the beginning of the operation, and hence can save the memory space for the mapping process.

Consequently, the robots can deliver consistent results of mapping and localization in a decentralized manner while holding only the submaps associated with their local areas. The proposed methods are implemented in Robot Operating System (ROS) and the protocols for the communication are also designed. For experiments, demonstration, and evaluation, the implemented system is simulated using the Gazebo simulator.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Robotic exploration in an enclosed environment without structured objects such as a cave is one of the challenging problems in autonomous mobile robotics. Being surrounded by walls and ceilings which hinder wireless communication, mobile robots cannot always access external facilities such as GPS, access points to the ideal network, remote control, or any services to support the robot's navigation. This entails necessity of a higher level of autonomy; the robots in such an environment must work independently. In addition, it is crucial to establish in-situ systems to carry out tasks which are usually done by external facilities: localization of the robots, building a map, and so on. In this situation, deployment of multiple robots may be advantageous over single-robot navigation. A multi-robot system can serve the aforementioned services which used to be provided by external facilities, e.g., establishing an ad hoc network to transfer data, working as beacons for localization purposes, and so on. The system is also robust to failures of some robots and efficient at finishing tasks within the given time; deployment of redundant robots covers failures of some malfunctioning robots, and tasks can be achieved faster by being distributed to or shared by multiple robots. However, a potentially complicated cooperation scheme of robots would be required as it is based on local interactions due to the lack of an ideal network. Thus, this area needs more sophisticated system than those applied to office environments, open fields, or any conventional areas with idealized facilities.

Examples of this type of exploration can be found in extreme environments where people cannot easily access. For instance, in the area of search and rescue, a partially collapsed building may be too dangerous for people to get inside to search injured people. A damaged nuclear plant can be a similar case. If there is a leak of radiation at critical levels, no human worker but a mobile robot can get inside to work. Moreover, planetary exploration is another area of application. For example, Mars is estimated to have lava tubes around its volcanoes [2, 3, 4, 5, 6, 7], which are important as a source of scientific data and as a potential shelter for future astronauts from harsh conditions on the surface. But those extraterrestrial caves are too distant and pose a risk if people directly explore them without any prior knowledge about the internal environment. Robotic technologies may shed light on such areas of application, providing viable solutions to the problems that they are confronted with.

The environment under consideration in this study is characterized and assumed as being enclosed, unknown, unstructured, three-dimensional, static, and in a large scale. The environment is enclosed by walls and ceilings blocking access to the external resources or facilities such as GPS. It is also unknown and unstructured; a robot cannot have a prior map of the environment or assume specific structured landscapes such as flat floors. This characteristic leads to a problem of simultaneous localization and mapping (SLAM), in which a robot needs to build a map and con-

currently localize itself in the map. Since the environment is unstructured, a robot needs to build a three-dimensional map rather than floor plans or any other type of two-dimensional map. Unlike streets where many moving objects exist, the environment here is assumed to be static. Finally, the environment may be large. For example, a Martian lava tube may be larger than those on Earth due to the smaller gravitational force on the red planet.

In the situation with properties described above, the use of multiple robots can be advantageous; without using external facilities, they collaborate to localize themselves by exchanging their data while executing their individual mapping tasks stably and quickly. For their collaboration, they establish an ad hoc network. For localization, they exchange their estimated locations and use their neighbors as landmarks to localize themselves. Through these interactions, each robot updates its own location estimate based on the one from the other robot and sensory data about their relative location. Meanwhile, each robot takes care of mapping of a local area by taking sensory data of the surrounding environment. At some point of time, the robot saves the current map and starts to build a new map. By repeating this process while moving, each robot holds a series of segmented maps, or submaps, of the areas along the robot's trajectory. The ad hoc network also serves to transfer their submaps. When a robot enters a previously mapped area, the corresponding submap may be passed to the robot to re-use and update it. While the global map needs to be built at the end of the operation, the robots do not have to return to the original location to gather their mapping data. They can pass their data through their ad hoc network. Thus, they do not have to maintain the energy for a returning way. If the network is formed as a decentralized and meshed network, the system can be robust to malfunctions of some member robots of the network. For decentralized multi-robot exploration in an enclosed environment, several challenging problems arise as described below.

Decentralization of Loop Closure: The multi-robot system working in an enclosed environment raises the problem of decentralization of loop closure. In the case of single-robot exploration, when the robot notices that it visits a previously mapped area by matching submaps, it realigns its submaps and closes a loop of submaps. In the case of multi-robot system, however, a robot in the system cannot always perform this loop closure since each one only has submaps of its local area. They may collaborate to perform an equivalent task by cooperative localization; they localize each other by serving as beacons or artificial landmarks and form a topological graph whose edges represent mutual localization by a pair of robots. If the graph contains a loop, they can effectively update their estimated locations and build submaps at more accurate estimated locations. However, due to the inaccessibility for the robots to access the centralized system, they need to carry out such a task only by interactions with their neighbors in a decentralized manner. Hence, the multi-robot system must require a decentralized method to perform loop closure based on local interactions of the robots.

Overconfidence in Decentralized Cooperative Localization: Overconfidence, or cyclic update, is the problem that a decentralized system gives inconsistent estimates by re-using duplicate information. The mutual interaction for their localization may cause correlations among the robots' estimates. Correlations roughly mean shared information in their estimates, so they need to remove such duplicate information in the received data before using it. Otherwise, the robot would reuse identical information as if it is totally new, leading to overconfidence. If the network does not have a loopy route, a robot may keep track of correlations by monitoring what it passes to the neighbors. However, if there is a loopy route, information may pass through other robots and go back to the robot which sent the information, and it becomes impossible for the robot to recognize the correlated portion in the received information.

Timing of Map Segmentation: Segmentation of a map entails the problem of deciding when to start building a new submap. When each of the multiple robots builds submaps of the local area, it needs criteria to decide the timing to save the current submap and start to build a new one. As the robots keep moving, the timing of starting a new submap affects the size of the resulting submaps; if the robots more frequently switch to a new submap, the submaps will be smaller whereas if they switch less frequently, the submaps become larger. If a submap is too small, a robot will not be able to effectively work for its individual SLAM since it will be difficult to localize itself in such a small submap. On the other hand, if it is too large, the robot and its neighbor may be in the same submap area and it cannot pass the submap to the neighbor since it is still working on the submap. A heuristic solution to this problem was proposed for single-robot exploration [8], but the multi-robot case was not discussed.

Detection of Entry to Areas Mapped by Other Robots: Without an external facility such as the ideal network and centralized control unit, mapping by multiple robots has the problem of how to detect when a robot enters an area that has already been mapped by another robot. Since the robots do not know about the areas being mapped by each other, they need to exchange information to decide if they should transfer their submaps to the neighbors. A robot may frequently ask its neighbors if it enters the area being handled by some of them. However, it would increase the network traffic, causing more power consumption. Thus, the robots need a protocol to tell whether a robot enters other's mapping area while keeping the number of interactions small.

Submap Transfer: A multi-robot system must resolve how a robot uses a submap that was made by another robot, handling uncertainty of their locations. A robot receives a submap from its neighbor once it is detected that the robot enters the corresponding area. However, the robot cannot simply use the submap from its neighbor in the same way as those made by itself. As the

submap is built based on the estimated location of the other robot, the robot receiving the submap needs to take into account the uncertainty of the neighbor's estimation. Otherwise, it would cause inconsistent mapping results. Hence, there should be a scheme to fuse the received submap and the estimated relative poses between the robots so they can import submaps from their neighbors without creating inconsistent mapping.

In this study, we propose methods to generate consistent estimates of robots' locations and submaps for decentralized multi-robot SLAM, addressing the problems described above. We assume that there is a base lander or a central unit holding mobile robots at the entrance of the enclosed environment. Each robot is deployed from the initial location and starts localization and mapping. The robots are assumed to maintain the distances to their neighbors while moving so that they can dynamically form a meshed ad hoc network. The map is represented by a three-dimensional occupancy map and the location is estimated by the particle filter. That is, each particle holds a robot's pose and a submap. The robots estimate their own locations based on their individual SLAM or cooperative localization. The cooperative localization have the robots to exchange their estimation and conduct measurements on their relative locations. After receiving data from the neighbor, the robot updates its estimated location by resampling the particles based on the neighbor's estimated location and sensory data about the mutual location. To avoid over-confidence, the robot deliberately increases uncertainty in its estimate and passes information with the reduced confidence. Each robot starts a new submap when the estimated location tells it has passed a specific distance from the point where the robot starts building the current submap. As it continues, the robot will hold a series of submaps with appropriate sizes for error correction for localizing submaps. While interacting with each other, the robots monitor their neighbors' locations and pass their submaps to those robots that entered the corresponding areas. Once a robot receives a submap from the other robot, it integrates the received submap to its own database, considering the noises caused by the mutual localization.

Combining all, our methods and approaches will perform decentralized multi-robot SLAM in an enclosed environment, addressing the aforementioned challenging problems as follows. For the decentralization of loop closure, the cooperative localization performed by multiple robots can achieve the equivalent effect of loop closure to correct accumulated errors when the robots form a meshed network. This approach does not require a centralized system and it performs in a decentralized manner. For the overconfidence in decentralized cooperative localization, it can be addressed by deliberately reducing confidence in their estimates at each interaction. For the timing of map segmentation, the robots independently decide the timing of map segmentation based on the distance they moved. This allows decentralization of the process, and each of the robots can keep the submap size small enough to transfer the submap to its neighbors and also large enough to perform its individual SLAM. For the detection of entry to areas mapped by other robots, the

robot can address the problem by checking the neighbors' locations. The robot decides if they are entering its previously mapped areas and send the corresponding submaps. For the submap transfer, once receiving a submap from its neighbor, the robot can integrate the submap into its own SLAM process by starting a new segment in which it performs localization and mapping on the transferred submap. Finally, these approaches are tested in robotic simulations.

1.1 Contributions

This dissertation has the following contributions to the field of autonomous mobile robotics:

- A consistent data exchange method for decentralized cooperative localization,
- Development of mathematical models of the data exchange,
- Development of the proposed method using a nonparametric filter,
- Development of a framework for segmenting a map into submaps for multi-robot navigation,
- Development of a framework for transfer and incorporation of submaps between multiple robots, and
- Implementation of the proposed algorithms in a robotic simulation.

1.2 Organization

The rest of this dissertation is organized as follows. Chapter 2 first reviews existing approaches in a few application areas, which give fundamental ideas for the enclosed environment exploration. Then, the environment and hardware characteristics are discussed so that higher-level control can be modeled based on the physical assumptions. After that, it will present an overview of our approach to the multi-robot exploration in an enclosed environment, showing that it is described as a mobile ad hoc network and modeled based on the probability theory. Finally, it will be shown how they can be described as a probabilistic model. Chapter 3 describes the decentralized cooperative localization by a group of robots which establish a meshed ad hoc network to cooperate. In this chapter, the overconfidence problem is introduced and the proposed method is presented to address this problem, followed by two-dimensional simulation results. Chapter 4 presents a version of the proposed method using a nonparametric filter so that it can be applied to occupancy map-based SLAM. It describes how particles are processed to perform the confidence reduction. It also shows two-dimensional simulation results. Chapter 5 describes the process of submap building by each robot and submap transfer to deliver a submap to other robots. It provides details about occupancy map-based SLAM and describe how it is modified to perform segmentation and submap transfer. Chapter 6 focuses on the integration of the decentralized cooperative localization and the distributed submap building. It also discusses how the robots actually send and receive data by

interacting with each other. After protocols for the interactions are introduced, a three-dimensional simulation is presented, including the system descriptions and environment model. Finally, this dissertation is concluded in Chapter 7 with a summary of this study and a discussion of future work.

CHAPTER 2
MODELING THE PROBLEM OF MULTI-ROBOT EXPLORATION

When a robot explores autonomously in an environment where external supports are rarely available, the exploration problem is composed of three parts: Mapping, Localization, and Planning [9, 8]. Especially, mapping and localization are grouped as a crucial problem domain, so called Simultaneous Localization and Mapping (SLAM), because of their inter-dependency: mapping requiring the robot's location and localization requiring the map where the robot is situated. In the probability theory, a system can be modelled as a Dynamic Bayesian Network (DBN), where each node represents a state at a specific time point and each edge represents a conditional probability density function. The SLAM problem is modeled as a DBN and can be solved by the Bayes filter, which estimates the unknown states based on the sensor readings. If we consider multiple robots as a decentralized system, this framework can be applicable to multi-robot exploration as well: the multi-robot SLAM problem, modeled as a DBN for the entire system. If the system is decentralized, the DBN needs to be decomposed into parts, each of which represents a subproblem that a single robot solves.

In this chapter, the problem in multi-robot navigation in an enclosed environment is modeled. After reviewing related work, the environment and robot hardware, which are assumed for the problem, are discussed. It is followed by a discussion of the multi-robot system as an ad hoc network. Then, the assumed multi-robot system is modeled based on the probability theory, and the multi-robot SLAM problem is decomposed into subproblems, while taking into account interdependence of the subproblems.

2.1 Related Work

Existing work for multi-robot systems and hardware designs give an insight to develop a model of the multi-robot exploration. This section reviews existing approaches where robots explore in an unstructured environment in a few aspects such as hardware designs, networking among the robots, modeling by the probability theory, and so on.

2.1.1 Robot Designs for 3D Navigation in an Enclosed Environment

An enclosed and unstructured environment can be found in several areas of application: search and rescue, robotic inspection, and extraterrestrial cavern exploration. These areas have common characteristics of the target environment, and related approaches and technologies to solve their problems can be found.

Search and Rescue: In disastrous situations such as earthquake, infrastructures are damaged and buildings sometimes collapse, trapping people inside. Search and rescue is a crucial application area as robots can help a rescue team by directly entering the dangerous environment to search victims. Due to the damaged structure and scattered objects on the floor, the ground is not necessarily flat or even, and Unmanned Aerial Vehicle (UAV) or Micro Aerial Vehicle (MAV) navigation has been proposed in order to search in an environment where ground-based robots such as rovers cannot easily enter [10, 11, 12, 13, 14, 15, 16, 17].

Due to the GPS-denied environment, many approaches mainly use sensors which do not rely on external supports. A laser scanner or lidar is commonly used to generate an occupancy map. Bachrach et al. proposed aerial robot navigation using a lidar along with an Inertial Measurement Unit (IMU) and a camera [10]. Since their sensors do not rely on external facilities such as GPS, their robot can work in a deeper indoor environment where GPS signals are usually not available. Their approach assumed the environment to be structured to the extent that the robots can generate a three-dimensional map by projecting two-dimensional maps generated by a 2D lidar at the corresponding altitudes. This assumption limits the application of the method only in a structured environment with walls which are intact enough to give information about the floor plan. Kohlbrecher et al. also used a lidar to develop a SLAM system which estimates a six-degree pose of an aerial robot [11]. Similarly, they assumed that the horizontal layout of the indoor environment does not change so significantly when the robot moves vertically. Based on this assumption, the robot builds two-dimensional maps for each floor plan. This simplification lets the system navigate only using an IMU and lidar performing two-dimensional SLAM at multiple stories of a building. Thus, these apporaches indicate that a lidar needs to assume the environment to be structured unless the robot has additional source of sensory data.

In addition to the use of laser scanners, an RGB-D or depth camera provides another way to construct a map of the environment. Bi et al. used an RGB-D or depth camera in development of their aerial robot which visually searches victims [14, 15]. Their navigation system is similar to Kohlbrecher's; the robot uses an IMU and lidar to perform two-dimensional SLAM. However, they also used an RGB-D camera so that they can additionally build a three-dimensional map. The RGB-D camera is also used for SLAM. Perez-Grau et al. employed an RGB-D camera for their UAV navigation for search and rescue, where the UAV is tele-operated but partially self-controlled to safely navigate inside a building [17]. Based on the data provided by the RGB-D camera, the robot performs visual odometry and localizes itself in the generated map. Comparing with a lidar, these approaches using a RGB-D camera can construct a three-dimensional map even in an unstructured environment.

A regular camera is also simply used for localization, but with some assumptions. Tomic et al. used down-looking stereo cameras to localize their aerial robot for urban search and rescue [12]. In their work, they introduced components of the entire mission: detection of a target building,

search for an entrance, entry of the building, and retrieval. They focused on the last two steps since they require the robot to navigate without GPS; the robot must only use a lidar scanner, an IMU, and a camera. In this situation, they proposed localization by tracking features on the ground detected by the cameras. Likewise, Liu and Nejat implemented vision-based localization for their semi-automatic multi-robot navigation in search and rescue [13]. By detecting features in an image captured by the robot's camera, the robot generates 3D point clouds and localizes itself by matching with the previously gained point clouds. The environment is assumed to have abundant landmarks so that the robot can find enough features to match point clouds. Moreover, for the retrieval step, Martinez et al. applied a scene-matching approach for their aerial robot [16]. The robot saves images from the camera while it is flying through the environment. Hence, when it goes into a deeper area of the environment, the robot holds a series of images along its path. Once it decides to retrieve, it compares the current view with the stored image data so as to localize itself. These methods, by using the image data, allow the robot to quickly localize itself and navigate through the entire environment, but they assume the environment contains abundant distinguishable landmarks. The view of each local area needs to be different enough to recognize. While an optical camera may be used to track the robot's velocity by tracking features of the texture of the environment, it may not be applicable to localization in an unstructured environment where a specific landmark is hard to be found such as in a cave.

In terms of communication, they usually assume the robots are wirelessly connected to the external facilities or networks. While external facilities are unavailable, communication between the robots is idealized and access points are assumed to be accessible for the robots at any location. They are generally tele-operated or partially dependent on a human operator as mentioned in some work above [13, 16, 17].

In summary, aerial robots are generally used for search and rescue due to the uneven ground, and they are equipped with laser ranging sensors, an IMU, and cameras so that they can navigate without using GPS. The environment is assumed to have a few characteristics which may not be available in a completely unstructured environment such as a natural cave. As the environment is usually a collapsed building or structures, they assume there are enough landmarks for a camera to detect and distinguish so that the robot can tell where it is located in the environment. In addition, the communication is assumed to be idealized; the robots can communicate with the base station or human operators in the most situations.

Robotic Inspection: Facilities and infrastructures are inspected regularly for their maintenance. However, some of them are too dangerous for human workers to get inside. Robotic probes and systems have been proposed for this purpose [18, 19, 20, 21, 22, 23].

The situation is similar to that of search and rescue: an indoor environment which can be expected to be structured. However, robotic inspection has a different characteristic in terms of

communication such as a lack of access to the ideal network since the inspection areas are not the spaces where people usually work or live. Burri and Nikolic et al. developed a MAV inspection of a thermal power plant boiler [18, 20]. The aerial robot estimates its motion by using its forward-looking stereo camera and an IMU. As the environment is dark, it also uses LEDs to light up the space so that the camera can detect objects and features. Nagatani et al. designed and tested robotic probes for inspection of a damaged nuclear plant [19]. Their robots are ground-based with crawlers for locomotion. They are controlled by a human operator working outside, and they create three-dimensional map using lidar. They tested their robots in the actual nuclear plant in Japan which was destroyed by the earthquake in 2011. To overcome the difficulty of communication due to the radiation, their robots relay data to their neighbors. One of the robots is tethered by a cable which leads to the human operator. This tethered robot transmits signals to other robots wirelessly so that they can perform short-ranged wireless communication. Ozaslan et al. proposed their UAV navigation system for tunnel-like facilities [21, 22]. Addressing the problem of GPS-denied environment, they use an IMU, cameras, and lidars for localization. Given a pre-defined map of the facility to inspect, a human operator provides way points and the robot autonomously follows these specified positions. Faria et al. developed an aerial robot for inspection of marine facilities such as oil well stations [23]. Like Ozaslan's work, the environment is also GPS-denied due to the robot's operations underneath the facility. Their work uses RGB-D cameras to build three-dimensional occupancy map and autonomously chooses unexplored areas as the next destinations.

Like search and rescue, robotic inspection has a similar situation: aerial robots being advantageous for the uneven ground, and a lack of access to external localization facilities such as GPS. On the other hand, there are unique environmental features. It may be lenient in terms of mapping; a prepared map is generally available unlike search and rescue, where damaged buildings may be different from their floor plans. However, since the environment is not necessarily a space where people usually work, access points of the network are less likely available, leading to a more severe communication limitation. As a result, they generally designed a single aerial robot to fly through the environment without communicating with anything and come back to the original location in order to provide the collected data. This type of approach does not require communication during the operation. Nevertheless, it is supposed to be a single-robot exploration, leading to the limited scale of area to explore.

Planetary Exploration: Beyond the Earth, underground and cavern environments will be one of the new frontiers in future space exploration. Through a series of manned and unmanned lunar missions, underground voids of the Moon have been confirmed to exist and they are expected to be large basaltic conduits known as lava tubes created after melted pahoehoe lava oozed-out of the consolidated ground [24, 25, 26, 27, 28, 29, 30]. Similarly, based on the findings by the Mars orbiters such as Mars Odyssey, Mars Express, and Mars Reconnaissance Orbiter, the red planet

is also estimated to have subsurface cavern structures [2, 3, 4, 5, 6, 7, 31]. Martian subsurface void spaces have been drawing attention in terms of two major objectives because of its stable and long-lasting structures [32]. First, caves could be the areas potentially preserving biosignatures. Some of the potential lava tubes could have existed in the ancient time when the Martian surface was said to be covered with liquid water. Even after Mars lost its surface water, the caves could have sheltered surviving micro-organisms for a long time, even up to today. The second significance is of practical usage: extraterrestrial underground structures may be used as potential shelters for future astronauts [25]. The walls of caves protect the internal environment from hazardous features on the surface. Besides, the smaller gravity force makes the Martian caves larger than those on the Earth. This natural shelter could be used as a base for the future manned mission. With a particular condition, a cave could have water ice inside [6], which could be precious resources for activities of the astronauts. Since it is risky to send human beings into unknown subsurface areas from the beginning, it is beneficial to send robotic probes to confirm the stability and safety in the environment. In fact, for exploration in the subsurface world of Mars, robotic missions have been proposed in the last decades [33, 34, 35, 36, 37, 38, 39, 40, 41].

The specific characteristics of this exotic underground environment affected the designs of robot hardware and systems. Unlike the previous application areas, Martian lava tubes are extremely remote and unknown: there are several minutes of communication delay and a spacecraft cannot observe the environment as it is enclosed and isolated by the ground surface. In addition, the landscape is totally unstructured and dark; it is hard to assume that there will be abundant landmarks enough to identify individual locations by a camera. Dubowsky et al. proposed a mission conducted by a very large group of "Microbots," tiny jumping spherical robots [33, 35, 36]. A group of the robots are launched from the base lander, and they are supposed to move one kilometer by hopping to the cave entrance and explore inside for 500 meters. They also designed detailed hardware and tested fuel consumption. The redundancy of the large robot group stabilizes the entire system. Malfunctioning or broken robots can be replaced with other healthy members, and the group continues working without interruption. Lange and Seeni introduced a carrier rover in which those microbots are stored [37]. As the rover reached a cave entrance, the robots are released and deployed into the inside. In their discussion, they characterized Mars cave exploration as follows: unknown structure, smaller gravity, harsh temperature, needs of internal communication, and navigation in the dark. For these conditions, they emphasized the advantage of deploying microbots carried by a main rover to minimize the risk of mission failure. There is also an approach of a heterogeneous robot group, in which robots have different hardware designs or functionalities. Husain et al. developed a heterogeneous robot team, composed of 2D mapping robots, a 3D modeling robot, and a sampling robot [38]. The comprising robots are all ground-based. The mapping and modeling robots are wheeled, and the sampling robot is a crawler. In their demonstration, a pair of 2D mapping robots generates a floor plan of the interior structure and a

3D modeling robot creates detailed models of the environment. Based on the acquired information, a sampling robot utilizes its drilling instrument to sample the ground material. Although their approach focused on activities in a cave, the terrain was assumed to be lenient enough for the ground-based robots to move.

Cavern or subsurface exploration of Mars is the next step for the future missions of planetary exploration. It has been analysed and discussed from different perspectives. Whittaker analyzed Martian subsurface exploration based on steps of the entire mission denoted as "vantage points:" orbital, flyover, surface, pit descent, and subsurface [40, 42]. The orbital step is carried out by a spacecraft releasing a lander from an orbit of the planet. The flyover step is taken by the released lander. Once the lander steps on the ground, it releases a rover or any ground-based robot. This robot takes care of the surface step by heading for the entrance of the cave. Then, a descending robot is responsible for the pit access/descent, which is the process to enter the internal structure while solving the specific navigational problems. Cave Hopper is quite a simple approach where the robot directly jumps into a pit hole [40]. As it was designed for caves of the Moon, whose gravitational force is small enough to jump, it may not be applicable to the Martian surface. Finally, the subsurface step is operated to explore the unknown cavern environments. Whittaker addressed several potential problems in the subsurface exploration: rough terrain, shortage of power supply, and limitation of communication. Fink et al. proposed "Tier-Scalable Reconnaissance," which is composed of multiple levels of exploration including ground, aerial and underground, and multiple agents controlled by their own [41]. In this approach, a base rover launches multiple robotic probes, and they perform their work on their own decision. The conventional type of robotic exploration using a single, remotely controlled, huge rover could not always work for exploration in risky areas such as lava tubes and caves; a minor malfunction of a local part could affect the entire system. A closed space prevents a rover from communicating with a remote pilot. The redundant and flexible approach could alleviate the situation by covering potential system failures.

In addition to frameworks of robotic exploration in a Martian cave, several hardware designs of aerial robots have been proposed. Arizona State University developed ball-like flying robots named Pit-bot to collaboratively explore Lunar and Martian lava tubes [34]. A group of the ball robots are launched from a base rover and explore in a cave. Each robot is equipped with thermal rockets or thrusters. The main thruster is equipped on the bottom, and the altitude control thrusters are near the top. As a group of the robots, they map the environment in a decentralized way [43]. Each individual robot appears as a ball-shaped marker in an image. They detect each other by their stereo camera and measure distances based on triangulation. The hopping mode with optimized fuel consumption has been materialized, and it is planned to develop the hovering mode in order to construct a 3D panoramic image. Meanwhile, Astrobotic Technology announced that they planned to develop a propulsive flying robot for exploration in Martian lava tubes [44]. The robot is to hop by spring struts and fly by thrusters. For scientific exploration, the robot will use compressed

atmospheric gas as propellant in order to avoid contamination of the environment by exhausted gas. The single-robot control system was designed based on information theory [45, 46]. They also announced that they developed and tested their flyer in a glacier cave in Iceland [47, 48]. The design is purely intended for cavern exploration; it uses only ranging sensors such as 3D lidar scanner for navigation without a camera. Moreover, the Asteroid and Lava Tube In Situ Resource Utilization (ISRU) Prospecting Free Flyer Project proposed their development of a small flyer which will be used for resource utilization in unstructured environments [49]. Although their primary target location is asteroids, lava tubes were also addressed as one of the possible destinations. The robot has four thrusters using cold gas propulsion. They mentioned the usability of multiple small flyers for their maneuverability, accessibility to the target location, and redundancy. Unlike those propulsive flying robots, the Jet Propulsion Laboratory (JPL) has been developing a rotary-winged robot which will fly in the Martian atmosphere [50]. It is a coaxial helicopter as a flying probe to Mars. They tested their rotor-craft design in a pressure chamber to simulate its performance in the Martian environment, and it was finally deployed as Ingenuity helicopter with Perseverance rover, which landed on Mars in 2021 [51]. This project demonstrated the possibility to use a propeller-based aerial robot in the Martian atmosphere at some altitude.

In short, planetary exploration has less assumptions than the previous application areas. In addition to lacking GPS and access points to the ideal network, the environment is completely unstructured and unknown. Moreover, due to the nature of the remote exploration, the operation is required to be done in limited time. To address these limitations, multi-robot exploration is proposed so that the robots can establish their ad hoc network and cooperatively process their tasks. Flying robots for a thinner atmosphere are also being designed and proposed so that they can explore in an three-dimensional space of a planetary cave.

Commonly, the environment is assumed to have uneven terrains and it is recognized the usability of aerial robots for the 3D navigation. A robot is generally equipped with laser ranging sensors such as lidar, cameras, and an IMU. RGB-D cameras are also used to get point clouds representing the terrain structure. Among the three application areas, search and rescue seems to assume the most lenient environment where the network access is available. Robotic inspection assumes a lack of access points of the ideal network and focuses on single-robot exploration, which does not necessarily rely on the ideal network. Planetary exploration has the most severe situation where the environment is unknown, unstructured, and possibly large. Addressing a lack of the ideal network, high risks of failure, and large environments, it is suggested that multi-robot exploration is advantageous.

2.1.2 Multi-Robot System and a Network

As indicated in the aforementioned application areas, it is beneficial to use multiple robots in terms of ability to build an ad hoc network: robustness by redundancy, and scalability of the group size. Due to the lack of the ideal network, the multi-robot system needs to build their own network to carry data. To build a multi-robot system communicating through their own network, there are two major approaches: separate systems where robots interact with network nodes spread out in the environment, and an integrated system of a group of robots which also perform as network nodes.

Multiple Robots Interacting with Network Nodes: Separation of the two components — mobile robots and network nodes — can simplify the navigation task into a graph search problem; mobile robots navigate in a roadmap of stationary nodes directing them.

Batalin and Sukhatme proposed an approach to deploy markers into the environment to aid the navigation of a mobile robot [52]. They introduced an algorithm for the coverage task, where a mobile robot deploys and uses markers as sign posts in an environment. The markers keep track on the directions where the robot explored in the past, and suggest the robot the desired direction so that the robot can visit less explored areas. They applied a probabilistic navigation approach to the sensor network which integrates the coverage task and the sensor network [53]. In the approach, the robot does not perform mapping or localization. Instead, the network executes tasks to direct the robot to the destination. Each network node has a probability that the robot is currently in the proximity of the node so that the network provides a discrete probability distribution for the robot's location. Mapping is performed by deploying additional nodes and expanding the network in the environment. As they extended the proposed method, Batalin and Sukhatme addressed two problems: the coverage/navigation task and the deployment of sensor network [54]. Unlike their previous work, the markers communicate with each other through two types of communication; one is a long-range communication to pass data within the network and the other is a short-range communication between the robots. Finally, they provided a theoretical analysis of their coverage and sensor deployment algorithm, referring to their algorithm as the Least Recently Visited (LRV) algorithm [55]. The environment is modeled as a graph of sensor nodes that the robot has deployed. The LRV algorithm was proved to be complete on a finite graph. It was modeled as a search algorithm to traverse the graph to cover the network based on the given information from the visited nodes.

Gasparri et al. introduced two different algorithms for the coverage problem of multiple mobile robots in sensor networks [56]. The problem is formulated as finding a path in a graph of sensor nodes which can dynamically change and multiple robots are traversing the nodes so that they cover the entire network. Both of the proposed algorithms are distributed so multiple robots can cooperate without a centralized control. The first algorithm temporarily constructs paths for

the robots and adjust them. The second one constructs an approximated Hamiltonian path and partitions it so that each robot is assigned to the divided path. Thus, sensor nodes aid mobile robots to navigate by providing a graph-based environment model.

Capitán Fernández et al. modeled and developed the problem of cooperation of sensory network and mobile robots [57]. They addressed the preliminary problems to integrate the network and the group of robots, such as localization of robots and sensors. The system is modeled as complementary coordination between the sensor network and multiple robots; the robots provide mobility to sensors by carrying and deploying them, and the sensors give the robots a functionality to pass data across the system. In their work, the localization of nodes in the sensor network and the robots are implemented using Received Signal Strength Indication (RSSI), which represents the signal strength to estimate distances from other nodes. Range-only SLAM (RO-SLAM) is based only on radio measurements of distances between nodes. Their system lets the sensor nodes process data so the computational costs can be shared by both the sensor network and the robots. Their approach is intended for a large-scale system of sensor networks and UAVs where the network is composed of clusters of sensor nodes. Each cluster has a cluster head, a special node which can communicate with the UAVs. The role of a cluster head is transferred among all nodes so that a specific node does not significantly consume its energy. Additionally, the authors mentioned that the Bayesian techniques used for their sensor fusion can be decentralized, and used the covariance intersection (CI) algorithm to solve the problem of double-counting information. Nevertheless, the CI-based method needs to assume that a probability distribution always follows a normal distribution.

Sauter et al. developed a method for aerial vehicles to increase the accuracy of localization with respect to the targets spread in the environment [58]. The robots and targets individually hold RF emitters and the robots receive the signals from the emitters. Based on the combination of Time Difference of Arrival (TDOA) and RSSI, the robots acquire sensory measurements on the relative distances to the other robots and to the targets. The robots deliberately move to areas where they can increase the accuracy of the sensory data with respect to the targets and also that they can cover as a large area as possible. They tested their approach in a simulation in which fixed-winged robots fly in a space with the diameter of about 100 meters. In their simulation testing, they reported the error in the localization with respect to the targets was in a few centimeters and the errors in the localization with respect to other robots was in tens of centimeters, indicating that the localization using static objects produces better accuracy than that using moving objects.

Separation into multi-robot and networking systems may simplify a navigation problem into a path finding problem in a network graph. However, there is an assumption that network nodes are deployed over the environment. If they are not initially deployed, the robots require to spread them while they are moving. In that case, it requires a scheme for the robots to install network nodes into the environment.

Multiple Robots Working as Network Nodes: If the robots also work as network nodes, they can build a network just by entering into the environment. As each robot plays the role of a network node, the system can be considered as an ad hoc network whose nodes can dynamically change their locations, thereby changing the network topology.

For example, robots can be viewed as an ad hoc network when they individually perform SLAM but also locally share some data with neighboring robots. Chen et al. presented cooperative graph SLAM [59]. In their work, each robot performs graph SLAM, finding features in the environment and estimating the locations of the features and its trajectory. The trajectory is represented with a series of way points through which the robot has passed. The robots can share a part of way points so that they can improve their graph SLAM. As they use graph SLAM, the robots need to always estimate the entire trajectories, which could grow as they keep exploring the environment. Unlike graph SLAM, online SLAM does not have this issue as it estimates only the current location. Furthermore, if the robots use online SLAM, they will only need to pass their estimate of the current location, instead of that on the entire trajectory.

Cunningham and Dellaaert proposed the distributed feature-based SLAM by a group of UAVs which communicate with each other, performing online SLAM [60]. It is based on Distributed Data Fusion (DDF). While the robots perform SLAM locally, they also share their temporary results with neighbors to improve each member's performance. Their idea is based on the decentralized sensor network developed by Nettleton [61], Makarenko and Durrant-Whyte [62], and Makarenko et al. [63]. Nettleton developed distributed estimation algorithms running in a sensor network composed of UAVs and ground-base stations [61]. In the approach, every state is estimated by an information filter. While each platform or UAV maintains its own pose estimate, individuals in the network communicate with each other to build and share a global feature-based map. To identify the commonly shared information among the individuals to eliminate over-confidence, the network is formed in a tree structure, where they can easily store and track the common information for each channel. While the proposed method is supposed to work in a tree-structured network so that a node sees the same data only once, the author also mentioned a network in arbitrary topology and addressed the problem of duplicated information. In that case, there is a possibility that the same information appears again through another channel, leading to overconfidence. Several possible measures were presented: tagging each item of sensory data, spanning the network into a tree dynamically, and use of Covariance Intersection (CI) update to generate more conservative estimates.

Makarenko et al. introduced the Active Sensor Network (ASN) as a generalized framework for distributed information gathering [63]. The system combines information from multiple components distributed over the environment, generates actions of the components to maximize the information gain, and also reconfigures the network topology. Two decentralization principles are addressed: no central control and no knowledge about the global network topology. In addition,

the decentralized approach accommodates three abilities: scalability, robustness, and modularity. The design concept focuses on services rather than components. Services include localization, sensing, data fusion, communication with the network, and so on. Components are implemented to provide these services. A physical entity such as a robot and a human can have more than one component. In their work, Makarenko and Durrant-Whyte presented the Bayesian decentralized data fusion (BDDF) [62]. The system estimates the locations of landmarks or spacial configurations in the environment. An estimate is represented by a probability distribution over the state space, and the distribution is inferred from sensory data based on the Bayesian theorem. Nodes in the network can exchange their data to refine their estimation. A node extracts new information from the distribution of the other node by subtracting the common information that they share. This framework could cause overconfidence if there is cyclic networks in the system. So the nodes must maintain the acyclic topology of the network by restricting it to a tree structure.

In summary, multiple robots navigating in an enclosed environment need to build an ad hoc network to communicate with each other. There are two major approaches to establish an ad hoc network: interacting with separately deployed network nodes, and performing as moving network nodes. If the robots interact with network nodes separately deployed in the environment, the navigation problem is simplified as the path finding in the network graph. The robots can also communicate with each other through the network. However, as the environment is unstructured, the robots need to deploy network nodes while moving in the environment. On the other hand, if the robots can form a network by directly communicating with each other, they do not need to deploy additional network nodes, while the network topology dynamically changes as the robots are acting as network nodes that move in the environment.

In both approaches, the problem of cyclic updates, or overconfidence, needs to be addressed in a decentralized system. This problem appears when a pair of data are fused without considering shared information, or correlations, in them. If the robots strictly form a tree network, which never contains a cyclic route, they can track correlations in the information which a robot receives from its neighbors. However, this topology restriction requires control over the entire system so that the network guarantees to have no cyclic route and it becomes hard to decentralize the system. Covariance Intersection (CI) applies a specific method to combine information from multiple sources and it can fuse data without knowledge of correlation, allowing the system to be fully decentralized, although it requires additional assumptions such as parameterized representation of the probability distribution, which limits the application area.

2.1.3 Exploration Model Based on Probability and Information Theories

Robotic exploration is composed of three interdependent problems: localization, mapping, and planning [9]. Planning is the problem of deciding an action to gain the maximized information

from the environment, and it is crucially affected by localization and mapping [64]. While this dissertation focuses more on localization and mapping, it is worth discussing how the planning problem has been addressed in the past as it can give an insight for tackling the localization and mapping problems.

Candidate plans are sampled and evaluated using specific criteria on the current results of localization and mapping. The frontier-based approach evaluates unoccupied areas adjacent to unknown regions as frontiers, and performs path planning to reach the closest frontier as the next destination [65]. It has also been applied to multi-robot exploration [66, 67]. This approach is simple but it does not take into account uncertainty in the estimates. The decision theory generalizes this technique; instead of calculating the path length to each frontier, it evaluates a utility value of each destination based on particular objectives [9, 68]. The information theory provides a framework to define the utility of a frontier or vantage point so that a robot efficiently minimizes uncertainty or information entropy in estimation about the environment. Expected information gain, or entropy reduction, tells how much information a robot will gain when it takes a measurement at a frontier [69, 70, 71]. As the entropy is calculated by a probability distribution, the information theory is related to the probability theory. For example, Stachniss et al. proposed a method to evaluate candidate actions by calculating the expected entropy reduction in the robot's estimation given by the Rao-Blackwellized Particle Filter (RBPF) [68].

Thus, the information and probability theories shed light on the robotic exploration problem by handling uncertainty in the estimates. To make a decision based on the entropy reduction in the estimates of the robot's location and maps, localization and mapping problems need to be modeled in a probabilistic framework.

2.2 Characteristics and Assumptions of Environment and Hardware

This dissertation focuses on an inference mechanism cooperatively operated by multiple robots. As a high level of the system design and methods are specifically considered, low level of characteristics, such as physical restrictions existing in the environment and the hardware, are assumed to be idealized.

In this section, the assumptions and idealization about the environment and the robot's hardware are described and discussed.

2.2.1 The Environment

The environment is defined as follows. It is mostly based on a cavern environment but it is applicable to other environments if the following conditions meet. First, the environment is an enclosed space. It is surrounded by walls and ceilings which block communication and an access to external facilities

and supports such as remote control and GPS signals. Second, the environment is unstructured. It may have uneven floors with obstacles. The walls and ceilings are also not flat or even. That is, unlike an office-like environment, two-dimensional mapping is not applicable since a floor plan at every altitude is not identical. Because of the lack of specific corners with flat walls and ceilings, it is assumed to have no specific landmarks for localization or any navigation process. On the other hand, textures of the surface of the environment — floors, walls, and ceilings — are assumed to have features used for visual odometry. Lastly, the environment is static. The structure of the environment does not change over time.

The characteristics mentioned above are summarized and enumerated as follows.

- Remote and Enclosed
  - no external support
- Unstructured
  - no assumption about the structure
  - uneven floor
  - no landmark
  - texture may be detectable
  - map must be three-dimensional
- Static
  - map does not change

2.2.2 The Hardware and Physical Characteristics of the Robot

The physical characteristics of the drone in the simulation are defined and idealized in terms of battery lifetime, propellant, sensing, and communication.

Battery Life: The battery lifetime depends on how much propulsion a drone needs to fly. On Mars, the condition may not necessarily be severe. The thinner Martian atmosphere may require more propulsion, but the smaller gravitational force may less. Also, it may be coped by minimizing power consumption, e.g., reducing payloads and planning optimized trajectories and communication. Or the lifetime may be extended by bucket-relaying battery packs while the base lander is recharging batteries by solar panels. Here, to focus on development of the sound inference system, the battery lifetime is assumed to be long enough to conduct the entire operation, and the items mentioned above are left as open questions.

Propellant: For the same reason as the battery lifetime is idealized, the propellant is also assumed to be unlimited. Chemical rockets or any type of propulsion whose propellant can contaminate the environment may not be preferable for scientific exploration. In fact, propellers and compressed atmospheric gas are proposed for propulsion of a drone flying in the Martian atmosphere [50, 44]. These types of propulsion requires atmospheric gas and electricity but not additional propellant. Thus, propellant can be idealized as the battery lifetime is unlimited.

Sensing: The drone's sensing in the simulation is defined based on the following sensors: ranging sensors, a depth camera, an IMU, and odometry. The ranging sensors are used for obstacle avoidance and autopilot. Based on the ranging data to obstacles, the drone decides the direction to fly. The depth camera is used for SLAM. It provides point clouds representing obstacles in front of the robot. Based on this data, the robot builds an occupancy map and localizes itself with respect to the map. The IMU is used to acquire the robot's orientation. Drifting noises are assumed to be small enough for pose estimation. The odometry tells the velocity of the robot. In an actual environment, it can be implemented by visual odometry, using a camera. In the simulation, it is simulated by a generic odometry module with artificial noises.

- Ranging Sensors: Control
- Depth Camera: Mapping
- IMU: Orientation
- Odometry (Visual): Velocity

Communication/Interactions of Robots: Finally, communication and interactions between the drones are defined as follows. Each robot has an ability to take a measurement on its relative pose with respect to the base lander and other robots. The measurement data is composed of ranging and direction to the target with which the robot is interacting. It can be implemented by estimation based on image data of the target or using Received Signal Strength Indication (RSSI) during communication. In the simulation, the robots receive relative pose with artificial noises. When they communicate with each other, they pass their estimated locations and occupancy maps. The map is formatted in the octree structure, which makes the data size to be at most a few hundreds of kilo bytes. With this compression, it is assumed that the robots exchange their data in real time.

2.3 Multiple Robots as a Mobile Ad Hoc Network

In the related work previously reviewed, several characteristics of robotic exploration in an enclosed environment have been mentioned. Due to the uneven terrain, aerial robots such as UAVs and

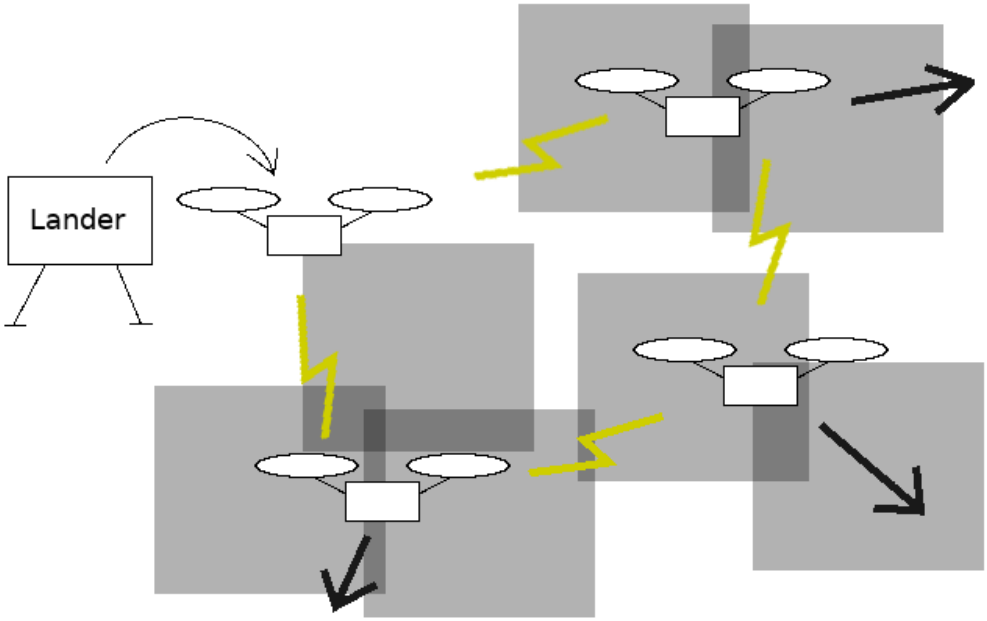

Figure 2.1: The lander deploying robots which go forward to deeper areas of the environment, expanding their ad hoc network and maintaining their local submaps (depicted as gray squares).

MAVs have been drawing attention. In a large environment, multi-robot systems have an advantage as they can be deployed over the environment and each robot can execute its task in its local area. However, an enclosed and unstructured environment does not allow communication with external facilities. The multiple robots will not be able to access the ideal network to share information directly. Rather, they will need to establish their own facility to communicate by themselves, namely an ad hoc network. Although the use of a separate network, which the robots interact with, can simplify the navigation problem to a path finding problem in the network graph, it could impose an additional task on the robots and require a hardware design to deploy network nodes while navigating through the environment. Instead, they can form an ad hoc network by playing a role as mobile network nodes and communicating with each other. Based on these considerations, in this dissertation, we consider a multi-robot system where each robot performs individual SLAM and communicates with its neighbors to establish a mobile ad hoc network for cooperative localization. To cope with uncertainty in the estimation, the system must be described as a probabilistic model.

Figure 2.1 shows an abstract idea of the multi-robot exploration. At the entrance area of an environment, the base lander deploys each robot into the environment. The deployed robots form a swarm in which they are wirelessly communicating. They proceed into deeper areas while maintaining distances from their neighbors. When a robot is too close to its neighbors, it tries to move away from them. Meanwhile, if it is going to be too far away to communicate with its neighbors, it tries to stay so others can catch up with it to avoid disconnection. As the lander keeps deploying robots, previously deployed ones are “pushed away” from the lander. In this way, as the

network extends into deeper areas, each member makes maps based on locally gained information: its own sensory data and the information provided by other members.

Generally, mapping is precise in a small local area since distortion of the robot's trajectory can be small enough. On the other hand, noises or errors are accumulated as the robot moves for a longer distance, and the accuracy can significantly degenerate. This is significant when a robot takes a long trip through the environment and goes back to the initial location. Due to the accumulated error, the robot may think it is located at a different place even if it is actually at the original location. Since the mapping process depends on the robot's estimated locations, the resulting map will be significantly distorted. To handle this problem, it is effective to build a set of submaps, which the global map is composed of, so that the submaps can be re-aligned later to correct the distortion of the global map [72]. Loop closure is a typical approach to reduce this type of errors. In the single-robot SLAM, when the robot notices that the current local map shares a part of the region which has been previously mapped, the local map's frame is moved so it matches with the other.

In this dissertation, loop-closure is performed in a different way for the multi-robot SLAM. Instead of closing a loop of submaps, the robots close a chain of interactions. The robots perform localization by using the ad hoc network where each pair of robots exchange information about their global locations. When these interactions form a loop, they effectively reduce uncertainty in their localization estimates. Submaps are built based on their estimated locations which are updated by the cooperative localization, leading to more accurate mapping results. While the conventional loop closure requires to find common features in submaps to re-align, this approach does not need such a process. Submap building requires some criteria such as timing of segmentation and monitoring the neighbor's location. While they are moving, each robot builds submaps of the local area where it is situated. At some point, it saves the current submap and starts with a new submap so that it can pass submaps to the neighbors without stopping mapping. Through the frequent interactions with the neighbors, the robot is supposed to know the recent locations of the neighbors. Once it notices that another robot is entering one of its submap areas, it passes the submap to the robot.

Thus, this approach contains two components: Decentralized Cooperative Localization and Distributed Submap Building. The following subsections describe each item.

2.3.1 Decentralized Cooperative Localization

The robots establish an ad hoc network where they locally communicate with each other. Based on the relative pose or location between them, they update their estimated locations.

Figure 2.2 depicts the idea of the localization in an ad hoc network. Each robot serves as a beacon, providing neighbors with its estimated location. While communicating with each other, a robot also calculates the distance to the other robot based on the signal strength. If the sensory data indicates the robot is actually located further away from the other than the robot was thinking, the

estimated location is moved away from the other robot. If the data indicates the robot is located rather closer, the estimated location is moved toward to the other robot.

The network topology is meshed. The robots can communicate with any others once they are within a communication range and perform loop-closure to improve the localization. It is also decentralized; they do not have a centralized entity to organize or manage the network traffic in a whole system. This approach allows each robot behave and communicate locally without taking care of the larger areas. In this type of network, however, there is a problem when they copy their information. Due to the mesh topology, the network contains cyclic routes. This leads to multiple paths through which information is passed. Eventually, a robot may receive identical information which it originally generated. If it takes place repeatedly, the information is "amplified," so is the error in it. This leads to overconfidence; a robot estimates a wrong state with overly high confidence. For example, in the case of localization, a robot would think it is located within a 10-cm radius of a goal but it was actually 20 meters away from there. When it communicates with others, it sends this information with high confidence, leading to the same problem at other robots. Keeping history of updates on the estimates may avoid the problem, but it would require a huge amount of memory space to keep such history data and additional computation. To deal with it,

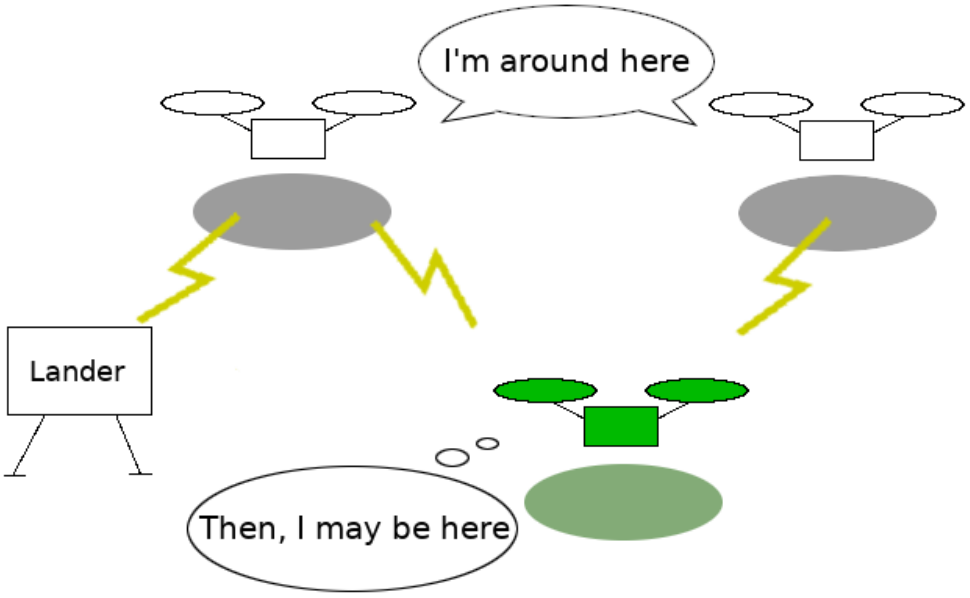

Figure 2.2: An abstract diagram of cooperative localization. Some robots like the top-left robot are close enough to communicate with the lander and perform global localization and update the estimated locations. Additionally, the robots are communicating with each other, taking measurement on relative poses, and exchanging the estimation about their locations. Based on the mutual measurements and the exchanged estimation, even if they are far away from the lander, they can update their estimated locations.

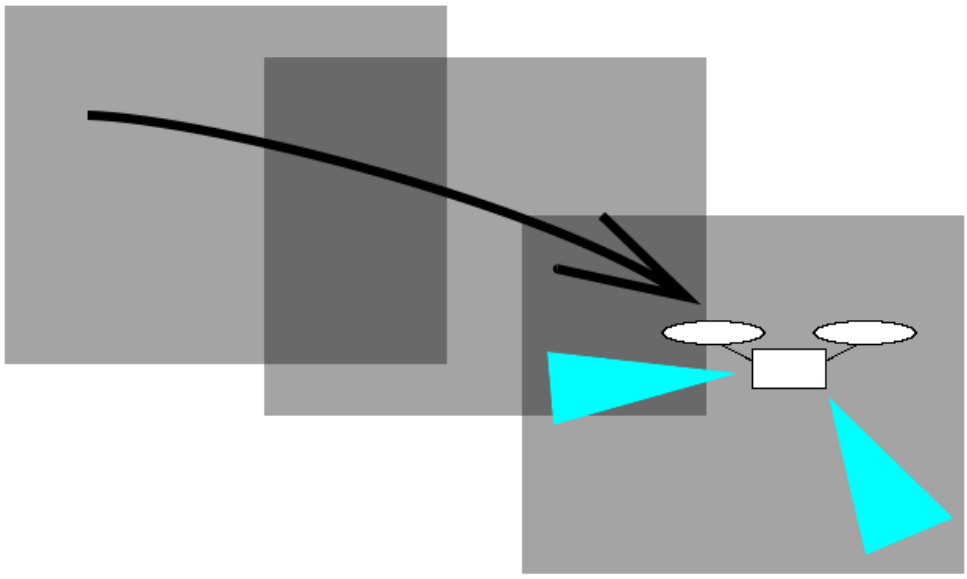

Figure 2.3: An abstract diagram of submap building by a single robot. While moving through the environment, the robot builds submaps (gray squares) based on sensory data (cyan triangles) such as ranging data. It performs localization and mapping with respect to the local reference frame, which is to be separately updated with respect to the global frame.

the proposed approach instead "reduces" the confidence when a robot exchanges its information with others. They still receive information but with lower confidence. If the confidence is reduced properly at each communication, the estimates can keep its consistency.

2.3.2 Distributed Submap Building

Each of the robots performs its individual SLAM by taking sensory data about the surrounding environment and takes care of submaps about its local area. Figure 2.3 shows an abstract view of this process. By taking sensory data from the environment, it builds a submap. At a specific point of time, it starts to build a new submap. As the figure indicates, the submaps may overlap with each other. The robot also performs localization with the submap that it is currently building. This process performs conventional SLAM problem on each submap: estimating the robot's location and maps about the environment. Thus, a robot may use an existing SLAM approach for this process. For example, one of the possible approaches is the use of the particle filter, which stochastically samples robot's poses and maps.

Submaps can be built by a multi-robot system by distributing the tasks over the robots. While a robot generates a series of submaps through the environment, the robot uses and updates only the latest submap for its SLAM process. The robot would just keep all the submaps on its memory if the system had only one robot. On the other hand, a multi-robot system lets the robot pass

these submaps to its neighbors who are entering the corresponding areas so it can save its memory space. Moreover, the receiving robot can also save computation for the mapping as it does not have to build a map from scratch.

2.4 Probabilistic Models

In the previous section, the components of the proposed multi-robot navigation were described with the challenging issues being addressed. Generally, mapping and localization are problems to estimate specific states, which can be terrains of the environment, the location of the robot, and so on. Since estimation entails uncertainty, the system needs to be described as a probabilistic model. In addition, mapping and localization in an unknown environment is like a chicken-and-egg problem. When a robot builds a map of the environment, it needs to know where it is located. On the other hand, to learn where it is located, it needs to have a map. This inter-dependency is called the simultaneous localization and mapping (SLAM) problem.

In this section, after the probabilistic model of the single-robot SLAM is described, an extension to the multi-robot SLAM is discussed.

2.4.1 Single-Robot SLAM

The probability theory sheds light on the SLAM problem where there exists significant uncertainty. Localization and mapping is a process of estimating a state with the robot's location and the environment by using known information such as control inputs and sensory data. The estimation of the state S given known data D is expressed as a probability distribution, $p(S|D)$, and it can be solved by the Bayesian inference.

$$p(S|D) = \frac{p(D|S)p(S)}{p(D)} = \frac{p(D|S)p(S)}{\int p(D|S)p(S)dS} = \eta p(D|S)p(S) \tag{2.1}$$

where $p(S)$ is a prior probability about the state S and $p(D|S)$ is a probability model about the observed data D given a specific state. In the case of SLAM, it is a probability about sensory data given a specific set of the robot's location and maps. The second term is derived by the Bayes' rule, and it leads to a normalized multiple of the sensory model's distribution and state prior.

The Bayesian inference tackles the SLAM problem while it entails a large number of parameters. Sensory data $z_{0:t}$ and control inputs $u_{0:t}$ are known parameters, and the robot's trajectory or sequence of locations $x_{0:t}$ and the map m are a state to be estimated.

$$p(S|D) = p(x_{0:t}, m|z_{0:t}, u_{0:t}) \tag{2.2}$$

A dynamic Bayesian Network (DBN) graphically represents the Bayesian inference performed in SLAM. Figure 2.4 shows the DBN of the state transitions in the single robot exploration. The arrows

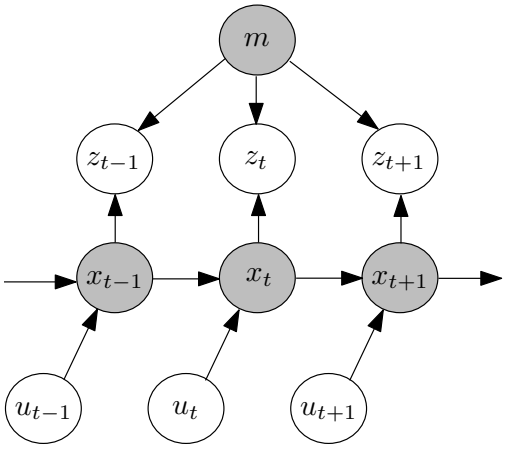

Figure 2.4: Dynamic Bayesian Network of the single robot navigation. The arrows represent cause-and-effect relations. Gray nodes represent the map m and the robot's trajectory x_t, which are unknown. White nodes represent control inputs u_t and sensory data z_t, that are directly observed.

represent cause-and-effect relations between nodes, which are modeled as conditional probability distributions. The robot's location x_t and the map m are not visible while its control inputs u_t and the sensory data z_t are known or observable. These hidden values are estimated or inferred in a form of posterior conditioned by the visible values through motion and sensor models: $p(x_t|x_{t-1}, u_t)$ and $p(z_t|x_t, m)$.

Specifically, the posterior can be factorized into the estimation on the two items: localization and mapping [64].

$$\begin{aligned} p(x_{0:t}, m|z_{0:t}, u_{0:t}) &= \frac{p(x_{0:t}, m, z_{0:t}, u_{0:t})}{p(z_{0:t}, u_{0:t})} = \frac{p(m|u_{0:t}, x_{0:t}, z_{0:t})p(u_{0:t}, x_{0:t}, z_{0:t})}{p(z_{0:t}, u_{0:t})} \\ &= \frac{p(m|x_{0:t}, z_{0:t})p(u_{0:t}, x_{0:t}, z_{0:t})}{p(z_{0:t}, u_{0:t})} \\ &= p(m|x_{0:t}, z_{0:t})p(x_{0:t}|u_{0:t}, z_{0:t}) \end{aligned} \tag{2.3}$$

Each component is inferred by applying motion and sensor models to the previous states as follows,

$$\begin{aligned} p(x_{0:t}|u_{0:t}, z_{0:t}) &= p(x_t|x_{0:t-1}, u_{0:t}, z_{0:t})p(x_{0:t-1}|u_{0:t}, z_{0:t}) \\ &= p(x_t|x_{t-1}, u_t)p(x_{0:t-1}|u_{0:t-1}, z_{0:t-1}) \end{aligned} \tag{2.4}$$

$$p(m|x_{0:t}, z_{0:t}) = \eta p(z_t|x_t, m)p(m|x_{0:t-1}, z_{0:t-1}) \tag{2.5}$$

where η is a normalizing factor.

From Equations (2.3), (2.4), and (2.5), the estimation can be expressed by the motion and

sensor models.

$$\begin{aligned} p(x_{0:t}, m|z_{0:t}, u_{0:t}) &= \eta p(x_t|x_{t-1}, u_t)p(z_t|x_t, m)p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1})p(m|x_{0:t-1}, z_{0:t-1}) \\ &= \eta p(x_t|x_{t-1}, u_t)p(z_t|x_t, m)p(x_{0:t-1}, m|z_{0:t-1}, u_{0:t-1}) \end{aligned} \tag{2.6}$$

Thus, the state can be estimated by applying the motion model $p(x_t|x_{t-1}, u_t)$ and the sensor model $p(z_t|x_t, m)$ to the previous estimate $p(x_{0:t-1}, m|z_{0:t-1}, u_{0:t-1})$ incrementally at each time step.

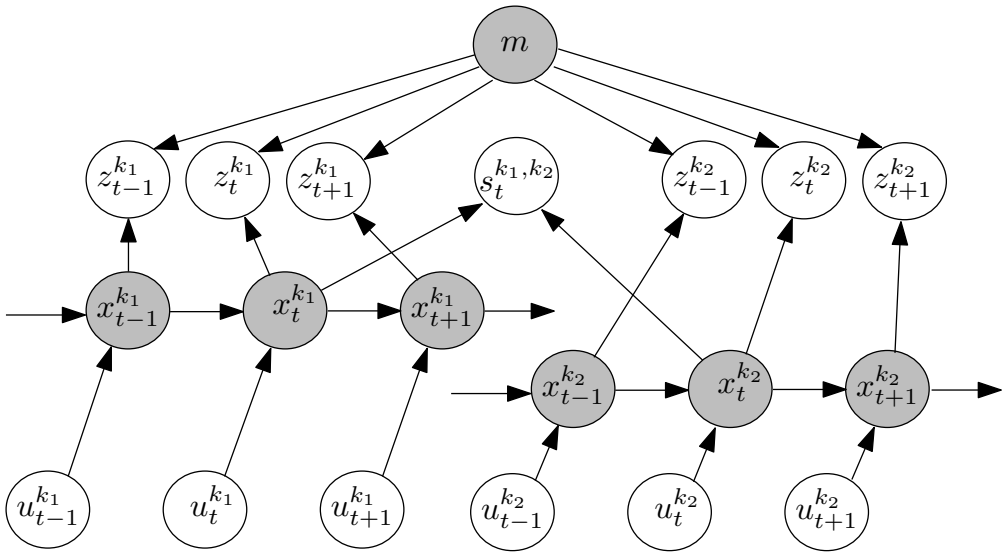

Figure 2.5: Dynamic Bayesian Network of multi-robot navigation in the marching formation. States of two robots (the k_1-th and k_2-th) and their mutual measurement ($s_t^{k_1,k_2}$) are shown.

2.4.2 Multi-Robot SLAM

The DBN of the single-robot SLAM can be extended for the multi-robot system as shown in Figure 2.5. Each robot's location is denoted as $x_t^k, 1 \leq k \leq K$ where K is the number of robots. For simplicity, the figure displays only states of two robots $x_t^{k_1}$ and $x_t^{k_2}$. In addition to the sensory data from the environment z_t^k taken by the k-th robot, they also take a relative measurement $s_t^{i,j}$ between the i-th and j-th robots, and its sensor model is assumed to have a commutative characteristic.

$$p(s_t^{i,j}|x_t^i, x_t^j) = p(s_t^{j,i}|x_t^i, x_t^j) \tag{2.7}$$

Let $\mathbf{S}_t$ be a set of relative measurements taken at time t.

$$\mathbf{S}_t = \{s_t^{i,j} : \forall(i,j) \in \mathbf{L}_t\} \tag{2.8}$$

where $\mathbf{L}_t$ is a set of pairs of indexes of robots which are linked and performing mutual measurements.

Then, the posterior on all robot's locations and the map can be calculated based on the motion and sensor models as follows. First, it can be factorized into those on mapping and localization.

$$p(x_{0:t}^{1:K}, m | z_{0:t}^{1:K}, u_{0:t}^{1:K}, \mathbf{S}_{0:t}) = p(m | x_{0:t}^{1:K}, z_{0:t}^{1:K}) p(x_{0:t}^{1:K} | z_{0:t}^{1:K}, u_{0:t}^{1:K}, \mathbf{S}_{0:t}) \tag{2.9}$$

The map is estimated by applying the sensor model $p(z_t^k | x_t^k, m)$ to the previous state.

$$\begin{aligned} p(m | x_{0:t}^{1:K}, z_{0:t}^{1:K}) &= \eta p(z_t^{1:K} | x_t^{1:K}, m) p(m | x_{0:t-1}^{1:K}, z_{0:t-1}^{1:K}) \\ &= \eta \left\{ \prod_{k=1}^{K} p(z_t^k | x_t^k, m) \right\} p(m | x_{0:t-1}^{1:K}, z_{0:t-1}^{1:K}) \end{aligned} \tag{2.10}$$

where η is a normalization factor. This recursive form is extended into the form of multiplication of all the sensor models and the priors.

$$\begin{aligned} p(m | x_{0:t}^{1:K}, z_{0:t}^{1:K}) &= \eta \left\{ \prod_{k=1}^{K} p(z_t^k | x_t^k, m) \right\} p(m | x_{0:t-1}^{1:K}, z_{0:t-1}^{1:K}) \\ &= \eta' \prod_{\tau=1}^{t} \left\{ \prod_{k=1}^{K} p(z_\tau^k | x_\tau^k, m) \right\} p(m | x_0^{1:K}, z_0^{1:K}) \\ &= \eta'' \prod_{\tau=1}^{t} \left\{ \prod_{k=1}^{K} p(z_\tau^k | x_\tau^k, m) \right\} p(z_0^{1:K} | x_0^{1:K}, m) p(x_0^{1:K}, m) \\ &= \eta'' \prod_{\tau=1}^{t} \left\{ \prod_{k=1}^{K} p(z_\tau^k | x_\tau^k, m) \right\} \left\{ \prod_{k=1}^{K} p(z_0^k | x_0^k, m) \right\} \prod_{k=1}^{K} p(x_0^k, m) \\ &= \eta'' \prod_{\tau=0}^{t} \left\{ \prod_{k=1}^{K} p(z_\tau^k | x_\tau^k, m) \right\} \prod_{k=1}^{K} p(x_0^k, m) \end{aligned} \tag{2.11}$$

where η' and η'' are normalizing factors. This can be transformed as a product of the mapping results from all the robots.

$$\begin{aligned} p(m | x_{0:t}^{1:K}, z_{0:t}^{1:K}) &= \eta'' \prod_{\tau=0}^{t} \left\{ \prod_{k=1}^{K} p(z_\tau^k | x_\tau^k, m) \right\} \prod_{k=1}^{K} p(x_0^k, m) \\ &= \eta'' \prod_{k=1}^{K} \left\{ \prod_{\tau=0}^{t} p(z_\tau^k | x_\tau^k, m) \right\} p(x_0^k, m) \\ &= \eta''' \prod_{k=1}^{K} p(z_t^k | x_t^k, m) p(m | x_{0:t-1}^k, z_{0:t-1}^k) \\ &= \eta''' \prod_{k=1}^{K} p(m | x_{0:t}^k, z_{0:t}^k) \end{aligned} \tag{2.12}$$

where η''' is a normalizing factor. This shows that the global map can be constructed from the maps locally built by individual robots.

The locations are estimated by their motion models and the sensor models.

$$
\begin{aligned}
&p(x_{0:t}^{1:K}|z_{0:t}^{1:K}, u_{0:t}^{1:K}, \mathbf{S}_{0:t}) \\
&= \eta p(\mathbf{S}_t|x_t^{1:K}) p(z_t^{1:K}|x_t^{1:K}) p(x_t^{1:K}|x_{0:t-1}^{1:K}, u_t^{1:K}) p(x_{0:t-1}^{1:K}|z_{0:t-1}^{1:K}, u_{0:t-1}^{1:K}, \mathbf{S}_{0:t-1}) \\
&= \eta \prod_{k=1}^{K} \left\{ p(\mathbf{S}_t^k|x_t^{1:K}) p(z_t^k|x_t^k) p(x_t^k|x_{t-1}^k, u_t^k) \right\} p(x_{0:t-1}^{1:K}|z_{0:t-1}^{1:K}, u_{0:t-1}^{1:K}, \mathbf{S}_{0:t-1}) \\
&= \eta \prod_{(i,j)\in\mathbf{L}_t} p(s_t^{i,j}|x_t^i, x_t^j) \prod_{k=1}^{K} \left\{ p(z_t^k|x_t^k) p(x_t^k|x_{t-1}^k, u_t^k) \right\} p(x_{0:t-1}^{1:K}|z_{0:t-1}^{1:K}, u_{0:t-1}^{1:K}, \mathbf{S}_{0:t-1})
\end{aligned}
\tag{2.13}
$$

where $\mathbf{S}_t^k = \{s_t^{k,j} \in \mathbf{S}_t : (k,j) \in \mathbf{L}_t, k < j\}$. Because it is a product of probabilities, this shows that each robot can independently performs localization for itself and each pair of robots perform mutual localization.

Thus, the model indicates that the multi-robot SLAM can be decomposed into decentralized cooperative localization and distributed submap building, which are individually performed by each robot. In the following chapters, each component of the problem is discussed in detail, and solutions are proposed.

CHAPTER 3
OVERCONFIDENCE PROBLEM AND CONSERVATIVE DATA EXCHANGE

In an unknown environment, it is crucial for the robots to estimate where they are located. Due to the difficulty to access external facilities such as GPS, the robots in an enclosed environment must get information from their surrounding areas to localize themselves. However, the environment does not necessarily have enough landmarks to identify local areas.

Cooperative localization by multiple robots may provide a solution to the exploration in such

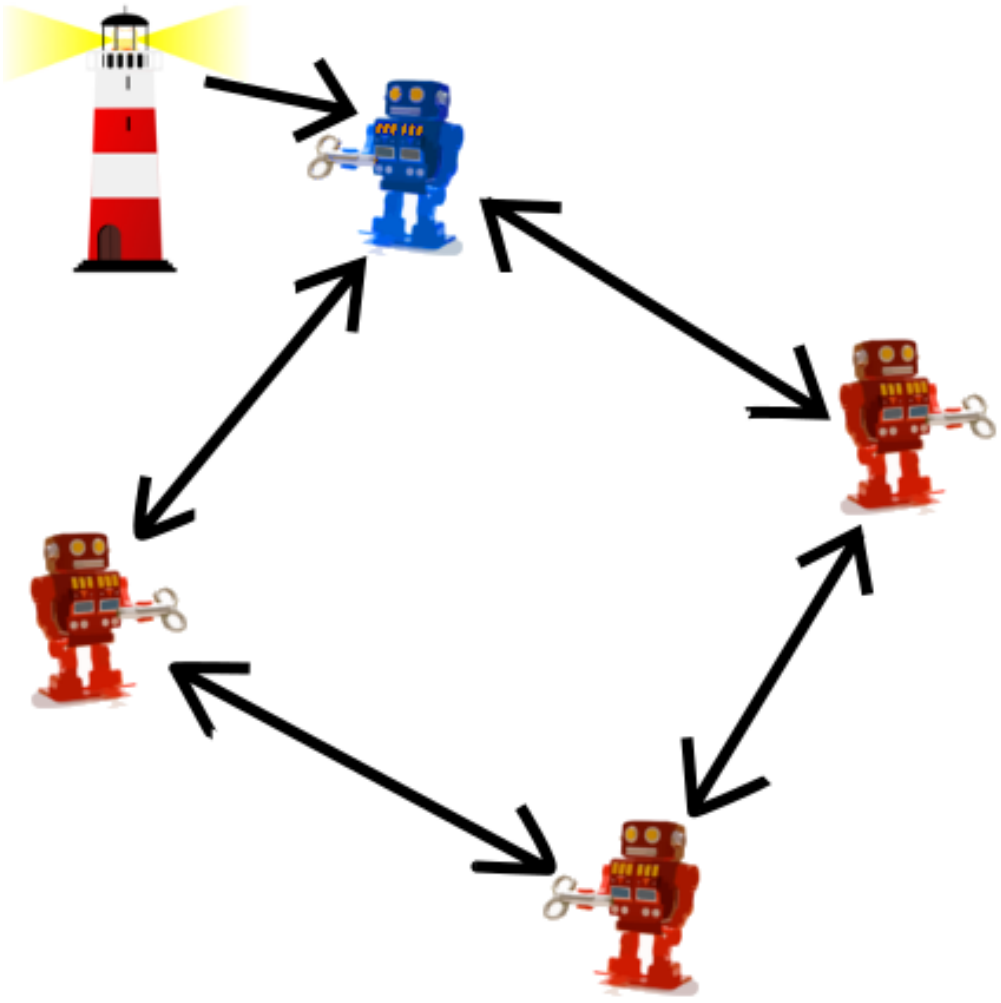

Figure 3.1: An illustration of cooperative localization. A limited number of robots (the blue robot in this case) can access landmarks while all robots (including the blue) can exchange information through local communication to update their estimations about current locations.

an enclosed and featureless environment since each of the robots can act as a landmark. Figure 3.1 illustrates an abstract concept of cooperative localization. A small group of robots may access a facility – e.g. a base lander for planetary exploration or a control unit for search and rescue – which provides global localization. In the figure, the blue robot has access to the lighthouse, which represents the base station, and the robot can be localized with respect to the base station. On the other hand, every robot can communicate with others while taking measurements on distances to them.

If the robots are situated close to each other, they may use vision-based localization and assume the localization is accurate enough [43]. However, if the robots are deployed in a huge area and the distance measurement system has significant noises, they need to handle uncertainty in the estimation. This is often managed in a probabilistic way; an estimate is described as a probability distribution over a state space and the distribution is updated using the data acquired from the environment. That is, the robots update their estimates based on the sensory data and the estimated locations of their neighbors. When the interactions of the robots form a loop, they can gain a similar benefit to loop-closure, addressing one of the challenging issue.

For these interactions, the robots form a decentralized ad hoc network. Decentralization of a network makes the system to be robust to anomalies that sporadically take place; the entire system is expected to remain operational even when some robots in the network fail their tasks. A decentralized system is more significantly advantageous in an unknown environment than a centralized system, which could critically fail if specific components are not functional.

However, when they estimate their locations in a probabilistic way as described above, there is another challenging issue; a decentralized network entails the problem of overconfidence. If the network has a cyclic route, the same information can be passed through the same point. In that case, the robots may reuse the identical information to update their estimates, which leads to inconsistent estimates. It can be avoided by keeping history of received data or restricting the network topology to an acyclic graph such as a tree. But the former requires excessively huge memory spaces and the latter cannot perform the loop closure.

As a solution to the challenging issues, i.e., multi-robot loop-closure and overconfidence, we propose a conservative data exchange scheme for decentralized cooperative localization performed by a meshed ad hoc network composed of mobile robots. In the proposed approach, the robots exchange their estimates with neighbors while taking measurements of distance to them. Each robot updates its estimate based on the measurement data and the estimate from the neighboring robot. They form a meshed network to close a loop of interactions. To handle overconfidence, the confidence of the robots' estimates is reduced before they are exchanged. Specifically, fractional exponents are applied: ω for the probability distribution about its estimate to keep, and $1 - \omega$ for the one to pass to the other robots, where $0 \leq \omega \leq 1$. This operation guarantees that no identical information is reused.

After the review of related work, this chapter describes a multi-robot system as an ad hoc network and discusses related problems such as overconfidence. Then, the proposed method is described and discussed with respect to an ad hoc network and confidence reduction. Finally, the simulation results are presented.

3.1 Related Work

Interactions of robots are based on wireless communication. Wireless communication has a finite range and may be blocked by obstacles in an enclosed environment. In terms of this situation, the technologies for mobile devices to exchange messages, e.g., cell phones, may give an insight. The AllNet network [73] is an ad hoc network of mobile devices designed to provide interpersonal messaging without an centralized facility. The system applies an epidemic routing; each device forwards data to the neighbors in its communication range so that all the devices in the network can receive the data at the end. Each of the mobile devices also works as a *data mule*: a data node which physically moves to a different location before forwarding data. Within an ad hoc network, AllNet typically delivers each message multiple times, each time to a different device that may be able to deliver it to the intended destination. Each device may receive an individual message more than once, but only needs to forward at most one copy to every other device. This is a kind of broadcast. Broadcast has the advantage of simplicity, and of working well even when the mobility of nodes in the network is high. A 3D simulation was conducted to demonstrate multiple flying robots passing a packet to all the robot members in the swarm [74]. In the simulation, the robots continuously fly in an ellipse-shaped route while passing packets. The simulation results showed that a deliberately changing network topology could improve the network routing. This indicates that a network of mobile robots can follow a similar routing strategy. However, as the network composed of robots is dealing with a different type of data, i.e., probability distributions, the robots need specific methods to exchange their data, instead of copying the message.

In addition to passing data, the robots may also take an measurement on a relative position with respect to the neighbor. By giving relative positions, a robot uses its neighboring robots as landmarks and vice versa: relative or mutual localization. This type of localization can be found in the field of swarming robots [75, 76, 77]. They usually use a vision sensor such as a camera to get ranging and bearing information of their neighbors: how far and in which directions they are located. Based on the information, the robots maintain their relative positions so that the entire group can move in specified formation. In the field of Internet of Things (IoT), radio waves can also be used for measurement on the relative pose to track the locations of wireless network devices. Based on the ranging information given by the sub-centimeter sized devices, they can perform 3D localization [78]. However, in these fields, the robots or devices focus on localization with respect to a specific device such as an individual robot and the access point of Wi-Fi, and they do not localize themselves with respect to the global reference frame.

Using both an ad hoc network and measurements of relative positions, the robots may localize themselves in the environment. This is a basic idea of cooperative localization; multiple robots exchange their data and measurements to estimate their locations in the global reference frame. In terms of communication, there are problems for updating estimation based on the received data from other robots: out-of-sequence measurements (OOSM) and cyclic update [79]. OOSM is a problem in the system where the network nodes estimate the same targets, i.e., all members' locations, and some node receives old sensory data after it has already updated its estimation. Cyclic update takes place when identical information is passed back to the robot which originally sent it out, and the robot uses it to update its own estimation.

The OOSM can be found in the system which is not distributed but decentralized; all the robots share the same tasks but execute independently. In a decentralized system, a robot may receive old information after it updates its estimates. To address this issue, robots may pass raw sensory data and perform localization of the other robot from scratch [80] or "retrodict" states to the time point when the OOSM data was gained [81]. Another approach is to detect checkpoint when the robot can update its estimates without OOSM [82]. As they identify each set of sensory data, they also inherently avoid the cyclic update problem. However, all the approaches mentioned above, need additional memory spaces to keep sensory data.

If each robot estimates separate parameters, e.g., its own location, and fuses sensory data to its estimation, which is passed to the other members in the network, the OOSM problem can be avoided. For example, in a distributed sensor network, each network node uses the estimated locations of others and updates its own estimated location based on the received information. As they repeat exchanging data, their estimation becomes correlated. If they naively update their estimation, they would reuse identical information, leading to cyclic update. To deal with this, each node employs a mechanism to eliminate correlated information such as the channel filter [63, 83]. The channel filter monitors the information which the node has already passed to the other nodes, and it subtracts this information from the incoming data. This approach works consistently in an acyclic network. If the network has a cyclic route, however, it is infeasible to generate consistent estimates since correlations are caused by another route which is unknown to the node. Cyclic update causes overconfidence in the estimation about a state, e.g., nodes' locations. When overconfidence takes place, the system becomes too "confident" about a particular state and its estimation becomes inconsistent with the actual state.

Thus, the OOSM can be simply avoided by passing the updated estimation instead of raw sensory data. However, the cyclic update becomes critical, appearing as the overconfidence problem in such a scenario. Some sensory network approaches addressed the issue in an acyclic network, but it will not work for multi-robot ad hoc network, whose network topology is not necessarily acyclic.

In such a system, where multiple robots individually exchange their estimation in a topologically changing network, the overconfidence problem has been generally addressed by maintaining

correlated information in the framework of a Kalman filter. For example, Roumeliotis and Bekey introduced a distributed localization method of multiple robots by estimating locations of the entire group of robots by the decomposed Kalman filters [84]. In their work, the matrix of the Kalman filter for the entire group of robots is decomposed into subsets of matrices each of which represents individual robot's localization and the correlations of pairs of robots. This eventually delivers the same results as a centralized Kalman filter for the large original covariance matrix. The overconfidence was addressed by calculating the matrices about correlations between robots. While the calculation can be distributed over the robots, they are supposed to have the ideal network to share information. Luft et al. improved the distributed Kalman filter so that the robots can work without the ideal network [85, 86]. In this approach, the matrix for all the robots is similarly decomposed and distributed to the robots, but instead of simply holding sub-matrices about the correlations, they hold decomposed matrices from those. This modified form allows the robots to update their estimation only by pairwise communication which does not require an access to the ideal network. While it effectively performs distributed Kalman filters, however, the robots need to hold additional data and update them. Each robot needs to update all the matrices about correlations every time it updates its estimation even when the process is just individual localization without any interaction with other robots. The size of this kind of data and the required computation grows as the size of the group of robots increases.

Instead of decomposing the centralized matrix, Bahr et al. proposed the Interleaved Update (IU), which generate a set of Kalman filters for each robot based on combinations of the neighbors' information [87]. Each robot estimates its own location by the Kalman filter, and passes its estimate to as many neighbors as possible by broadcasting in its communication range. Once another robot receives its neighbor's estimate in addition to mutual measurement data, it duplicates its Kalman filter and applies the received data to the copied filter. As a result, it gains double filters, representing the original estimate and the newly created one based on the original and the neighbor's filters. When it further receives data and creates a new filter, there will be a set of multiple filters based on all the combination of information sources. A filter is selected from the set so that the original and the received filters do not share common sources of data on which they are based. Since identical data is never reused in update steps on their filters, this approach successfully maintains correlated data, and hence avoids overconfidence on their locations. Nevertheless, each robot needs to keep matrices based on all possible combinations of data from the neighbors. Thus, the decomposition of the Kalman filter allows the system to distribute the computation, but it requires to hold a set of matrices representing the robot's location and correlation, which is dependent on the size of the robot group.

Unlike a Kalman filter, Nerurkar et al. introduced a Maximum A Posteriori (MAP) approach for cooperative localization [88]. Instead of generating a probability distribution about the current location, their method gives the most probable path from the original location to the current

location based on the acquired sensory data. The approach is formulated as a matrix operation for smoothing the robots' paths up to the current time point, based on sensory data from mutual measurement of each pair of robots. This calculation task is distributed to individual robots; each member performs a part of the calculation. It was demonstrated that the proposed method outperformed a method using Kalman filters. However, it was also noted that the method requires synchronous communication, which is not usually possible in a decentralized navigation scenario.

Another approach to address the overconfidence problem is the Covariance Intersection (CI), which is a variant of the Kalman filter. CI generates consistent estimates when the correlation between nodes is unknown [89, 90]. Let a pair of nodes have means and covariance matrices (a_1, A_1) and (a_2, A_2) as their estimates. If their estimates are independent of each other, the Kalman filter generates a fused estimate (μ, Σ) by the following equation.

$$\Sigma = \left(A_1^{-1} + A_2^{-1}\right)^{-1} \tag{3.1}$$

$$\mu = \Sigma\left(a_1 A_1^{-1} + a_2 A_2^{-1}\right) \tag{3.2}$$

If the network contains a cyclic route and the nodes are repeatedly interacting, it entails overconfidence as their estimates contain correlations. The CI method modifies the aforementioned equations as follows:

$$\Sigma = \left(\omega A_1^{-1} + (1-\omega) A_2^{-1}\right)^{-1} \tag{3.3}$$

$$\mu = \Sigma\left(\omega a_1 A_1^{-1} + (1-\omega) a_2 A_2^{-1}\right) \tag{3.4}$$

where $0 \leq \omega \leq 1$. Each node calculates an optimal ω to minimize $|\Sigma|$. There exists a closed form solution to calculate this value [91]. CI was also applied to the SLAM problem where a robot estimates locations of itself and also landmarks in the environment [92]. As a robot navigates through an environment with multiple landmarks, the robot needs to estimate locations of the landmarks as well as its own location. The CI algorithm provides consistent estimates when correlations between landmarks are unknown, whereas a standard Kalman filter would fail in that situation. Similarly, the CI algorithm was also applied to the cooperative localization performed by multiple robots [93, 94, 95] where each pair of robots take measurements of relative poses and exchange data with each other to update their estimates by the CI algorithm. While CI provides consistent results, the CI approaches assume all robots always hold consistent estimates. This assumption makes them to be vulnerable to one-point failure. When one of the robots accidentally generates a wrong estimate with high confidence, this inconsistent information will eliminate consistent yet less confident estimates.

In summary, a group of robots can cooperatively localize themselves by establishing an ad hoc network where they conduct measurements of relative positions. By exchanging their estimation instead of raw sensory data, they can avoid the OOSM problem, but they need to address the

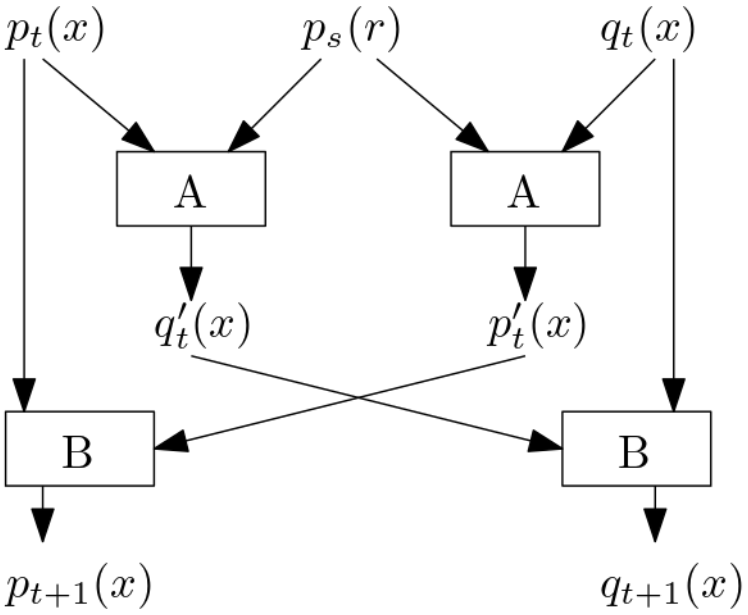

Figure 3.2: Naive data exchange: robots generate new estimations $p'_t(x)$ and $q'_t(x)$ based on the sensory data $p_s(r)$ and the other robot's estimates (process A). Then, they update their estimates by multiplying the output of process A and normalizing the results (process B).

overconfidence caused by cyclic updates. To deal with overconfidence, it is crucial to maintain an uncertainty in their estimation. The methods using Kalman filters generally decompose and distribute the covariance matrix, representing uncertainties, among the robots and maintain the correlation information when they update their estimation. This means they need an extra memory space for the correlation information and more computation, which grow in the size of the group of robots. On the other hand, the methods using Covariance Intersection only hold covariance matrices for each robot's location without any matrix about correlations between the robots, and they update their estimation in a conservative manner so that the system can avoid being too confident. As both approaches are parametric filtering, they assume the probability distributions follow a normal distribution.

3.2 Overconfidence Problem in an Ad Hoc Network

For cooperative localization based on a decentralized multi-robot system performing an ad hoc network, overconfidence is one of the critical problems as the robots would end up with inconsistent estimation. The problem is typically significant in an enclosed environment, where the robots reply on their interactions without any external positioning facilities such as GPS. As reviewed in the related work, the overconfidence takes place when they re-use identical information and inappropriately reduce an uncertainty in their estimation, leading to overconfident and inconsistent

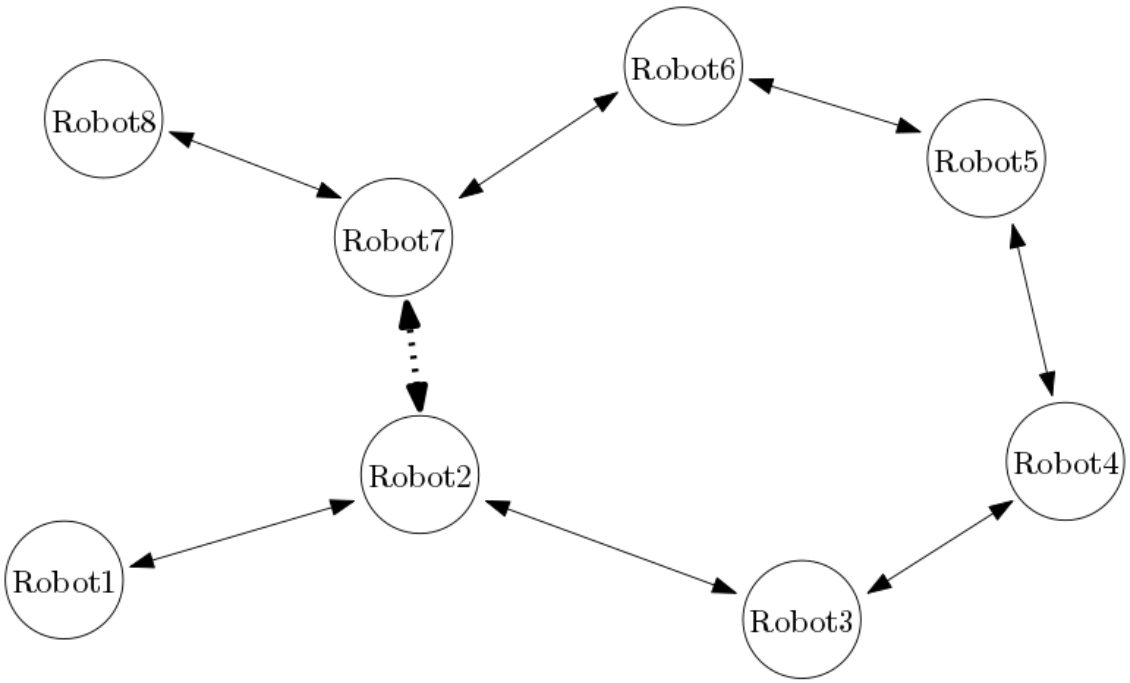

Figure 3.3: Loop closure by multiple robots. Arrows represent mutual measurements between robots. When Robot2 and Robot7 find each other, they can perform loop closure.

estimation. In this section, it is described in detail how the problem appears in an ad hoc network.

To perform cooperative localization, the robots form an ad hoc network, working as network nodes which transfer their estimation about each individual location. Each robot takes a measurement to the other with which it is communicating to exchange their estimated locations. Based on the measured distance and the other robot's location, the robot can update its own estimated location. Assume that there are two robots estimating their own locations at time t described as probability distributions, $p_t(x)$ and $q_t(x)$, respectively. When they communicate, they take a measurement about the distance between them. The sensor may be modeled as a probability distribution $p_s(r)$ of measurement data r. Figure 3.2 shows a flow of exchanging data and updating location estimates. Each box represents a process to generate a new distribution based on the given inputs. In boxes labelled as A, new estimates are generated based on the other robot's estimate and the sensory data. This process is implemented by convolution.

$$p'_t(x) = \int q_t(x')p_s(f(x - x'))dx' \tag{3.5}$$

$$q'_t(x) = \int p_t(x')p_s(f(x - x'))dx' \tag{3.6}$$

The function f is a transformation from the state space to the sensory data space.

Subsequently, distributions $p'_t(x)$ and $q'_t(x)$ are used to update each robot's estimate in boxes

labelled as B. The updated estimate $p_{t+1}(x)$ is calculated by multiplying the robot's estimate $p_t(x)$ with $p'_t(x)$, and normalizing the result. $q_{t+1}(x)$ is updated in the same way.

$$p_{t+1}(x) = \frac{p_t(x)p'_t(x)}{\int p_t(x')p'_t(x')dx'} = \eta_{p_{t+1}} p_t(x)p'_t(x) \tag{3.7}$$

$$q_{t+1}(x) = \frac{q_t(x)q'_t(x)}{\int q_t(x')q'_t(x')dx'} = \eta_{q_{t+1}} q_t(x)q'_t(x) \tag{3.8}$$

where $\eta_{p_{t+1}}$ and $\eta_{q_{t+1}}$ are normalization factors.

This mutual measurement can lead to loop closure in a meshed ad hoc network. Figure 3.3 shows mutual measurement among eight robots. After Robot2 and Robot7 update their estimated location, their updated locations affect other's estimation as they continue the process. Eventually, their locations are corrected to meet their mutual measurements, leading to loop closure. Thus, they can close a loop without finding features in the environment.

As reviewed in the related work, this strategy introduces a cyclic update. Once a cyclic route forms for loop closure, the identical information may pass through the same route multiple times. In Figure 3.3, the estimated location of Robot2 affects the estimation by Robot3. Robot3 affects Robot4; Robot4 affects Robot5, and so on. Finally, Robot7 will give to Robot2 the information affected by Robot2. That is, the robot will update its estimation by its own estimate delivered in the past. This cyclic update amplifies the error or noises contained in their data, leading to inconsistent estimation, namely overconfidence.

3.3 Conservative Data Exchange

To address the overconfidence problem, the proposed method does not copy or amplify information which already exists in the entire system. Instead, when a pair of robots exchange their information, it "divides" the confidence in the probability distributions by applying fractional exponents and passes the reduced information. As a result, each robot generates distributions with lower amplitudes and avoids overconfidence. Unlike decomposing Kalman filters, it does not need to track information about correlation. While it is similar to CI-based methods, it updates the estimation in a different way. It also does not need to assume the probability distribution to follow a normal distribution, making a nonparametric filter applicable. This section describes the proposed method in detail and shows how the robots deal with the problem.

3.3.1 Confidence Reduction

The proposed method addresses the overconfidence problem by reducing confidence in the estimation when the robots exchange their information. In terms of the probability theory and the information theory, this confidence reduction can be materialized by applying a fractional exponent

to a probability distribution. For example, when a robot applies square root to the probability distribution followed by normalization, the resulted distribution looks smoothed, intuitively indicating reduced confidence. This can be proved by representing the uncertainty as information entropy.

In the information theory, information entropy is used to quantify how much uncertainty a probability distribution contains [96]. In a continuous space $x \in \mathbb{R}$, the entropy of a probability density function $p(x)$ is calculated by differential entropy.

$$h(p) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx \tag{3.9}$$

It indicates "how smoothly" the probability is distributed on the support set, which is the coordinates of the environment here. If probability is distributed over a large area, the entropy increases. If it is in a smaller area, the entropy decreases. Thus, information entropy provides a gauge of uncertainty the robot holds about its estimation. This mathematical model gives an insight to quantify how much the confidence changes by a specific operation. Let a new distribution $p'(x)$ be generated by the power of $1/n$ of the function where n is an integer greater than 1.

$$p'(x) = \frac{(p(x))^{\frac{1}{n}}}{\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx} = \eta (p(x))^{\frac{1}{n}} \tag{3.10}$$

where η is a normalization factor. Then, the entropy of this function is

$$h(p') = -\int_{-\infty}^{\infty} p'(x) \log p'(x) dx \tag{3.11}$$

If the resulting distribution $p'(x)$ gains more uncertainty, the entropy must grow ($h(p') > h(p)$).

To calculate the difference of entropy values, there are two important quantities: cross entropy and relative entropy. Cross entropy is entropy of a distribution based on another.

$$h(p', p) = -\int_{-\infty}^{\infty} p'(x) \log p(x) dx \tag{3.12}$$

Relative entropy, also known as Kullback-Leibler divergence, is expressed as follows.

$$D_{KL}(p'||p) = \int_{-\infty}^{\infty} p'(x) \log \frac{p'(x)}{p(x)} dx \geq 0 \tag{3.13}$$

with equality if $p(x) = p'(x)$. And, it is known to be non-commutative.

$$D_{KL}(p'||p) \neq D_{KL}(p||p') \tag{3.14}$$

Also, cross entropy and relative entropy have the following relationship.

$$h(p', p) = h(p') + D_{KL}(p'||p) \tag{3.15}$$

These expressions give the following equation.

$$\begin{aligned} h(p') &= -\int_{-\infty}^{\infty} p'(x) \log p'(x) dx \\ &= -\int_{-\infty}^{\infty} p'(x) \log \eta(p(x))^{\frac{1}{n}} dx \\ &= -\log \eta - \int_{-\infty}^{\infty} p'(x) \log (p(x))^{\frac{1}{n}} dx \\ &= \log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) - \frac{1}{n} \int_{-\infty}^{\infty} p'(x) \log p(x) dx \\ &= \log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) + \frac{1}{n} h(p', p) \\ &= \log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) + \frac{1}{n} (h(p') + D_{KL}(p'||p)) \end{aligned} \tag{3.16}$$

This leads to the following equation.

$$\log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) = \frac{n-1}{n} h(p') - \frac{1}{n} D_{KL}(p'||p) \tag{3.17}$$

The left-hand side of Equation (3.17) can also be expressed by $D_{KL}(p||p')$.

$$\begin{aligned} D_{KL}(p||p') &= \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{p'(x)} dx \\ &= -\log \eta + \frac{n-1}{n} \int_{-\infty}^{\infty} p(x) \log p(x) dx \\ &= \log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) - \frac{n-1}{n} h(p) \\ \log \left(\int_{-\infty}^{\infty} (p(x))^{\frac{1}{n}} dx \right) &= \frac{n-1}{n} h(p) + D_{KL}(p||p') \end{aligned} \tag{3.18}$$

Since Equations (3.17) and (3.18) hold the same value,

$$\begin{aligned} \frac{n-1}{n} h(p') - \frac{1}{n} D_{KL}(p'||p) &= \frac{n-1}{n} h(p) + D_{KL}(p||p') \\ h(p') &= h(p) + \frac{n}{n-1} D_{KL}(p||p') + \frac{1}{n-1} D_{KL}(p'||p) \geq h(p) \end{aligned} \tag{3.19}$$

with equality when $p(x) = p'(x)$. In the case of applying square root, i.e., $n = 2$, the entropy of the

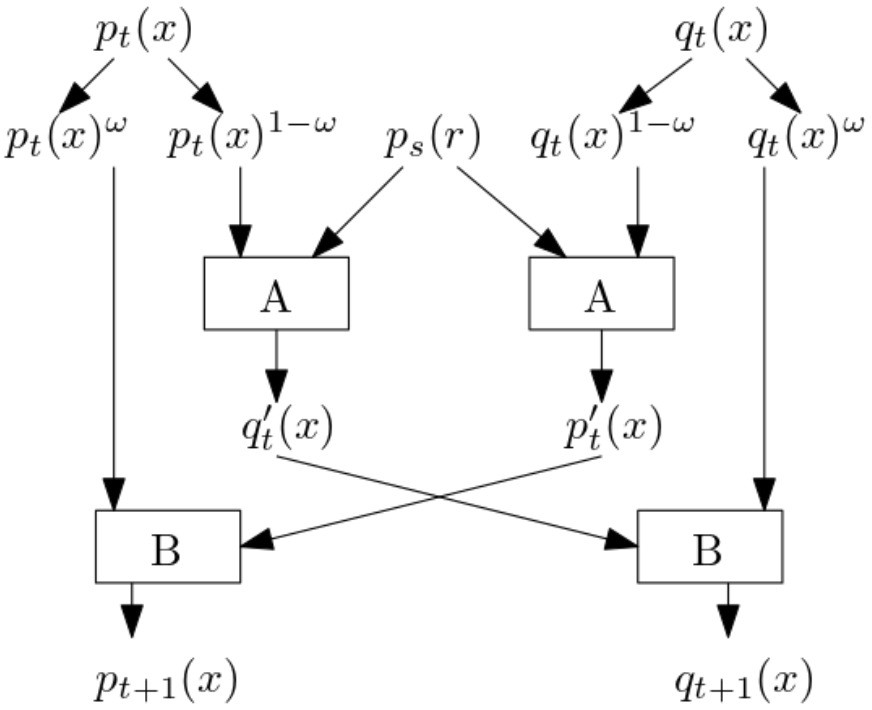

Figure 3.4: Conservative data exchange: each robot first applies a fractional exponent ω and generates a pair of distributions. It keeps one of them and passes the other to the neighboring robot.

resulting distribution $h(p')$ is simplified as follows.

$$h(p') = h(p) + 2D_{KL}(p||p') + D_{KL}(p'||p) \geq h(p) \tag{3.20}$$

Thus, the entropy of a probability density function which is taking the power of $1/n$ of the original function always increases unless it generates an identical distribution.

3.3.2 Exchange Procedure

After reducing the confidence, a pair of robots exchange their data, i.e., their estimation with reduced confidence. Figure 3.4 shows a workflow of the proposed method. At the beginning, a fractional exponent ω, where $0 \leq \omega \leq 1$, is applied to the robots' estimates. Then, the first robot keeps $(p_t(x))^{\omega}$ and passes $(p_t(x))^{1-\omega}$ to the second robot. On the other hand, the second robot keeps $(q_t(x))^{\omega}$ and passes $(q_t(x))^{1-\omega}$ to the first robot. The rest of the process is the same as the naive method. Thus, the resulting distributions are expressed as follows.

$$\begin{aligned} p_{t+1}(x) &= \eta_{p_{t+1}} \left(p_t(x)\right)^{\omega} p'_t(x) \\ &= \eta_{p_{t+1}} \left(p_t(x)\right)^{\omega} \int \left(q_t(x')\right)^{1-\omega} p_s(f(x-x'))dx' \end{aligned} \tag{3.21}$$

$$
\begin{aligned}
q_{t+1}(x) &= \eta_{q_{t+1}} \left(q_t(x)\right)^{\omega} q'_t(x) \\
&= \eta_{q_{t+1}} \left(q_t(x)\right)^{\omega} \int \left(p_t(x')\right)^{1-\omega} p_s(f(x-x'))dx'
\end{aligned}
\tag{3.22}
$$

where $\eta_{p_{t+1}}$ and $\eta_{q_{t+1}}$ are normalization factors. This process does not amplify the information in the original estimates.

If the location estimate follows a normal distribution, the process becomes similar to the Kalman filter and the CI algorithm. To show that, let the distributions of the robots' estimates and sensory data be represented as

$$
\begin{aligned}
p_t(x) &\sim \mathcal{N}(\mu_{p,t}, \Sigma_{p,t}) \\
q_t(x) &\sim \mathcal{N}(\mu_{q,t}, \Sigma_{q,t}) \\
p_s(r) &\sim \mathcal{N}(\mu_r, Q).
\end{aligned}
\tag{3.23}
$$

Then, the intermediate estimates $p'_t(x)$ and $q'_t(x)$ are expressed as follows.

$$
\begin{aligned}
p'_t(x) &\sim \mathcal{N}\left(\mu_{q,t} - Hr, \frac{1}{1-\omega}\Sigma_{q,t} + HQH^T\right) \\
&= \mathcal{N}\left(\hat{\mu}_{p,t}, \frac{1}{1-\omega}\left(\Sigma_{q,t} + Q'\right)\right) \\
&= \mathcal{N}\left(\hat{\mu}_{p,t}, \frac{1}{1-\omega}\hat{\Sigma}_{p,t}\right)
\end{aligned}
\tag{3.24}
$$

$$
\begin{aligned}
q'_t(x) &\sim \mathcal{N}\left(\mu_{p,t} + Hr, \frac{1}{1-\omega}\Sigma_{p,t} + HQH^T\right) \\
&= \mathcal{N}\left(\hat{\mu}_{q,t}, \frac{1}{1-\omega}\left(\Sigma_{p,t} + Q'\right)\right) \\
&= \mathcal{N}\left(\hat{\mu}_{q,t}, \frac{1}{1-\omega}\hat{\Sigma}_{q,t}\right)
\end{aligned}
\tag{3.25}
$$

where H is a transformation matrix from the sensory space to the state space, $\hat{\mu}_{p,t} = \mu_{q,t} - Hr$, $\hat{\mu}_{q,t} = \mu_{p,t} + Hr$, and $Q' = (1-\omega)HQH^T$.

The means and covariance matrices are updated as follows.

$$\Sigma_{p,t+1} = \left(\omega\Sigma_{p,t}^{-1} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\right)^{-1} \tag{3.26}$$

$$\mu_{p,t+1} = \Sigma_{p,t+1}\left(\omega\Sigma_{p,t}^{-1}\mu_{p,t} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\hat{\mu}_{p,t}\right) \tag{3.27}$$

$$\Sigma_{q,t+1} = \left(\omega\Sigma_{q,t}^{-1} + (1-\omega)\hat{\Sigma}_{q,t}^{-1}\right)^{-1} \tag{3.28}$$

$$\mu_{q,t+1} = \Sigma_{q,t+1}\left(\omega\Sigma_{q,t}^{-1}\mu_{q,t} + (1-\omega)\hat{\Sigma}_{q,t}^{-1}\hat{\mu}_{q,t}\right) \tag{3.29}$$

This update step calculates an inverse matrix six times. It can be converted into a more efficient

form, which is used for SLAM problems.

$$
\begin{aligned}
\Sigma_{p,t+1} &= \left(\omega\Sigma_{p,t}^{-1} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\right)^{-1} \\
&= \hat{\Sigma}_{p,t}\hat{\Sigma}_{p,t}^{-1}\left(\omega\Sigma_{p,t}^{-1} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\right)^{-1}\Sigma_{p,t}^{-1}\Sigma_{p,t} \\
&= \hat{\Sigma}_{p,t}\left(\Sigma_{p,t}\left(\omega\Sigma_{p,t}^{-1} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\right)\hat{\Sigma}_{p,t}\right)^{-1}\Sigma_{p,t} \\
&= \hat{\Sigma}_{p,t}\left(\omega\hat{\Sigma}_{p,t} + (1-\omega)\Sigma_{p,t}\right)^{-1}\Sigma_{p,t} \\
&= \frac{\hat{\Sigma}_{p,t}}{1-\omega}\left(\frac{\hat{\Sigma}_{p,t}}{1-\omega} + \frac{\Sigma_{p,t}}{\omega}\right)^{-1}\frac{\Sigma_{p,t}}{\omega} \\
&= \frac{\hat{\Sigma}_{p,t}}{1-\omega}\left(\frac{\Sigma_{p,t}}{\omega} + \frac{\Sigma_{q,t}}{1-\omega} + HQH^T\right)^{-1}\frac{\Sigma_{p,t}}{\omega}
\end{aligned}
\tag{3.30}
$$

Now that the first and last matrices in the right-hand side of Equation (3.30) can be switched as shown below.

$$
\begin{aligned}
\Sigma_{p,t+1} &= \Sigma_{p,t}\Sigma_{p,t}^{-1}\left(\omega\Sigma_{p,t}^{-1} + (1-\omega)\hat{\Sigma}_{p,t}^{-1}\right)^{-1}\hat{\Sigma}_{p,t}^{-1}\hat{\Sigma}_{p,t} \\
&= \Sigma_{p,t}\left(\omega\hat{\Sigma}_{p,t} + (1-\omega)\Sigma_{p,t}\right)^{-1}\hat{\Sigma}_{p,t} \\
&= \frac{\Sigma_{p,t}}{\omega}\left(\frac{\Sigma_{p,t}}{\omega} + \frac{\Sigma_{q,t}}{1-\omega} + HQH^T\right)^{-1}\frac{\hat{\Sigma}_{p,t}}{1-\omega}
\end{aligned}
\tag{3.31}
$$

The middle part of the last equation can be written as follows.

$$
\begin{aligned}
&\left(\frac{\Sigma_{p,t}}{\omega} + \frac{\Sigma_{q,t}}{1-\omega} + HQH^T\right)^{-1} \\
&= H^{*T}\left(\frac{H^*\Sigma_{p,t}H^{*T}}{\omega} + \frac{H^*\Sigma_{q,t}H^{*T}}{1-\omega} + Q\right)^{-1}H^* \\
&= H^{*T}S_{p,t}^{-1}H^*
\end{aligned}
\tag{3.32}
$$

where H^* is a pseudo-inverse of H such that $H^* = (H^TH)^{-1}H^T$ and $S_{p,t} = \frac{H^*\Sigma_{p,t}H^{*T}}{\omega} + \frac{H^*\Sigma_{q,t}H^{*T}}{1-\omega} + Q$.

Finally, the updated covariance matrix is expressed as a simplified form.

$$\begin{aligned}
\Sigma_{p,t+1} &= \frac{\hat{\Sigma}_{p,t}}{1-\omega} H^{*T} S_{p,t}^{-1} H^* \frac{\Sigma_{p,t}}{\omega} \\
&= \left(\frac{\Sigma_{p,t}}{\omega} + \frac{\hat{\Sigma}_{p,t}}{1-\omega} - \frac{\Sigma_{p,t}}{\omega} \right) H^{*T} S_{p,t}^{-1} H^* \frac{\Sigma_{p,t}}{\omega} \\
&= \left(\left(H^{*T} S_{p,t}^{-1} H^* \right)^{-1} - \frac{\Sigma_{p,t}}{\omega} \right) H^{*T} S_{p,t}^{-1} H^* \frac{\Sigma_{p,t}}{\omega} \\
&= \left(I - \frac{\Sigma_{p,t}}{\omega} H^{*T} S_{p,t}^{-1} H^* \right) \frac{\Sigma_{p,t}}{\omega} \\
&= \left(I - K_{p,t} H^* \right) \frac{\Sigma_{p,t}}{\omega}
\end{aligned} \tag{3.33}$$

where $K_{p,t}$ is a Kalman gain such that $K_{p,t} = \frac{1}{\omega} \Sigma_{p,t} H^{*T} S_{p,t}^{-1}$. Based on Equations (3.31), (3.32), and (3.33), the mean is also updated by the Kalman gain.

$$\begin{aligned}
\mu_{p,t+1} &= \Sigma_{p,t+1} \left(\omega \Sigma_{p,t}^{-1} \mu_{p,t} + (1-\omega) \hat{\Sigma}_{p,t}^{-1} \hat{\mu}_{p,t} \right) \\
&= \omega \Sigma_{p,t+1} \Sigma_{p,t}^{-1} \mu_{p,t} + (1-\omega) \Sigma_{p,t+1} \hat{\Sigma}_{p,t}^{-1} \hat{\mu}_{p,t} \\
&= \omega \left(I - K_{p,t} H^* \right) \frac{\Sigma_{p,t}}{\omega} \Sigma_{p,t}^{-1} \mu_{p,t} + (1-\omega) \frac{\Sigma_{p,t}}{\omega} H^{*T} S_{p,t}^{-1} H^* \frac{\hat{\Sigma}_{p,t}}{1-\omega} \hat{\Sigma}_{p,t}^{-1} \hat{\mu}_{p,t} \\
&= \mu_{p,t} - K_{p,t} H^* \mu_{p,t} + \frac{\Sigma_{p,t}}{\omega} H^{*T} S_{p,t}^{-1} H^* \hat{\mu}_{p,t} \\
&= \mu_{p,t} - K_{p,t} H^* \mu_{p,t} + K_{p,t} H^* \hat{\mu}_{p,t} \\
&= \mu_{p,t} + K_{p,t} H^* \left(\hat{\mu}_{p,t} - \mu_{p,t} \right) \\
&= \mu_{p,t} + K_{p,t} H^* \left(\mu_{q,t} - Hr - \mu_{p,t} \right) \\
&= \mu_{p,t} + K_{p,t} \left(-r - H^* \left(\mu_{p,t} - \mu_{q,t} \right) \right)
\end{aligned} \tag{3.34}$$

Similarly, the other robot's estimate is expressed using its Kalman gain $K_{q,t}$.

$$\Sigma_{q,t+1} = \left(I - K_{q,t} H^* \right) \frac{\Sigma_{q,t}}{\omega} \tag{3.35}$$

$$\mu_{q,t+1} = \mu_{q,t} + K_{q,t} \left(r - H^* \left(\mu_{q,t} - \mu_{p,t} \right) \right) \tag{3.36}$$

where $S_{q,t} = \frac{H^* \Sigma_{q,t} H^{*T}}{\omega} + \frac{H^* \Sigma_{p,t} H^{*T}}{1-\omega} + Q$ and $K_{q,t} = \frac{1}{\omega} \Sigma_{q,t} H^{*T} S_{q,t}^{-1}$.

Thus, the confidence reduction can be performed with matrix operations when the probability distributions are assumed to follow a normal distribution. Although the proposed method is similar to the Covariance Intersection as both of them actually apply fractional exponents to the probability distributions, the use of CI could amplify the original information sources. Figure 3.5 shows a workflow of the CI-based method. While the proposed method applies a fractional exponent to "divide" p_t and q_t, the CI-based method uses fractional exponents to weigh information to generate

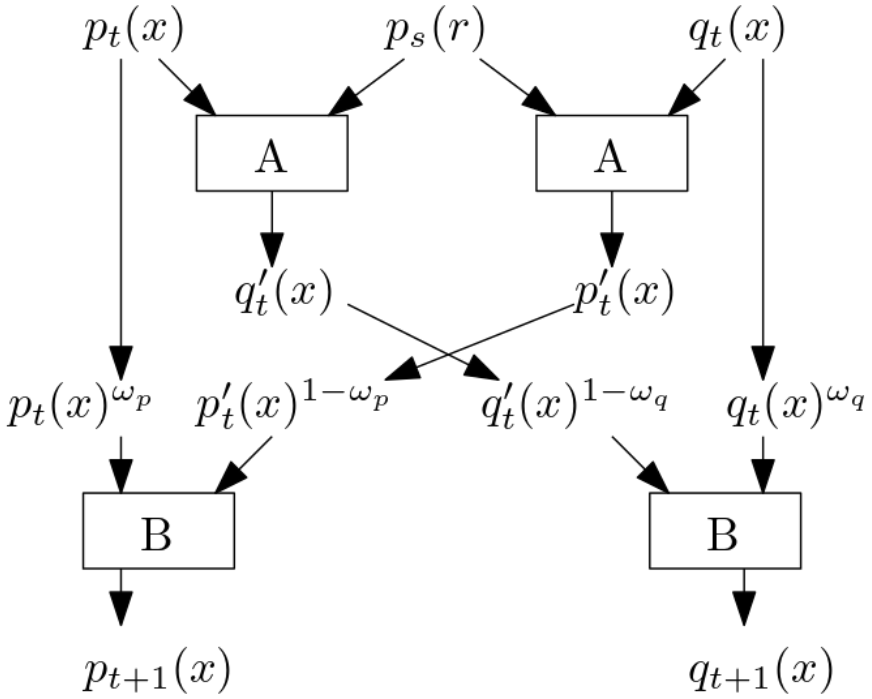

Figure 3.5: CI-based data exchange (for comparison with the proposed method): fractional exponents ω_p and ω_q are separately determined to optimize an arbitrary objective function, which generally represents uncertainty in the resulting distributions p_{t+1} and q_{t+1}, e.g., determinant of the covariance matrix.

p_{t+1} and q_{t+1}. Since ω_p and ω_q are determined separately (generally to minimize the uncertainty in p_{t+1} and q_{t+1}), this process could amplify identical information. For example, if the first robot's location is estimated with much higher confidence compared to that of the second robot, ω_p will be close to 1 and ω_q will be to 0. Then, both robots will estimate their locations based on the same information from the first robot.

3.4 Modeling the Overconfidence Problem

The proposed method reduces confidence in the estimation and addresses the overconfidence problem in cooperative localization, where the robots exchange their estimation in a decentralized manner. To mathematically model and show how it works, here, the problem is formulated as a simplified case.

3.4.1 Simplified Problem Model

For simplicity, a pair of robots are considered. Figure 3.6 shows components in this model. They are estimating a particular state x. Their estimation is described as probability distributions; Robot1's

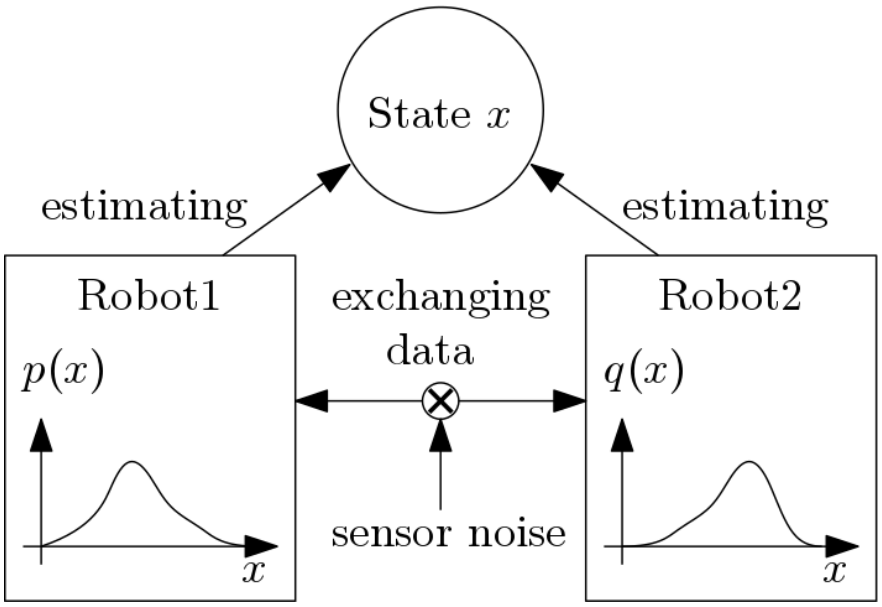

Figure 3.6: A simplified model of two robots exchanging data. Both robots estimate the state x. Their estimation is described as probability distributions. Robot1's estimate is $p(x)$ and Robot2's is $q(x)$. They do not have direct access to information about the state but they exchange and update their estimation. The exchanged data may possibly contain noises.

estimation is described as $p(x)$ and Robot2's is as $q(x)$. Initially, they have prior information: $p_0(x)$ and $q_0(x)$. They do not have any direct access to information about x. Instead, they exchange and update their probability distributions. When a robot passes its distribution to the other robot, some noise may be added. Finally, they update their estimation based on the distributions passed from the others with some noise. At time t, their estimation is denoted as $p_t(x)$ and $q_t(x)$.

This model is a simplified case of mutual localization in one dimensional space. The robots do not have a volume and they are represented as points in the 1D space. The two robots are actually located at the same location, but the robots do not know it, and each one updates its estimation based on mutual localization with the other robot. This model can be easily extended to a more general case in which they are located at different locations by introducing translation between two states that the robots are estimated respectively.

3.4.2 Methods to Analyze

Based on the simplified problem model, the proposed method, Conservative Data Exchange, is analyzed and reviewed while being compared with other methods. Here, two other methods are analyzed to show how the proposed method outperforms them. The first one is a simple method which naively fuses probability distributions without considering correlations between the robots, as shown in Figure 3.2. The other method employs Covariance Intersection (CI), as shown in

Figure 3.5. Hence, there are three methods being modeled and analyzed.

1. The Naive Method
2. The CI-based Method
3. The Conservative Data Exchange (proposed method)

Each method is analyzed by delivering converging probability distributions, which represent the robots' estimation, when infinitely repeating their interactions. By comparing with the naive method, it is analyzed how the proposed method can avoid the overconfidence. It is also shown that the proposed method gives a geometric mean of the priors unlike the CI-based method, which selects a prior with lower entropy as the resulting distribution.

3.5 Noiseless Cases

If the sensor does not have any noise, the robots can send their probability distributions without any distortion. In this section, the resulting distributions after infinitely exchanging data without noises are evaluated in the cases of the naive method, the CI-based, and the conservative data exchange.

3.5.1 Naive Method

Since the sensor is assumed to provide noiseless data, the sensory data about the relative location is always zero as the robots are located at the same location. Then, the naive method calculates the current estimates based on the previous estimates as follows.

$$p_t(x) = \frac{p_{t-1}(x)q_{t-1}(x)}{\int p_{t-1}(x')q_{t-1}(x')dx'} = \eta p_{t-1}(x)q_{t-1}(x) \tag{3.37}$$

$$q_t(x) = \frac{p_{t-1}(x)q_{t-1}(x)}{\int p_{t-1}(x')q_{t-1}(x')dx'} = \eta p_{t-1}(x)q_{t-1}(x) \tag{3.38}$$

where η is a normalization factor. Since it is just multiplying the probabilities followed by normalization, the uncertainty in their estimation is reduced. If they repeat this process multiple times, they keep reducing the uncertainty although they do not gain any new information from outside. Their uncertainty will eventually reach zero while their estimated state is inconsistent, leading to overconfidence. This phenomenon can be described as follows.

Consider functions $f_t(x)$ and $g_t(x)$ which are proportional to the robots' probability distributions $p_t(x)$ and $q_t(x)$, satisfying the following relations.

$$p_t(x) = \frac{f_t(x)}{\int f_t(x')dx'} \tag{3.39}$$

$$q_t(x) = \frac{g_t(x)}{\int g_t(x')dx'} \tag{3.40}$$

Let's say these functions $f_t(x)$ and $g_t(x)$ are exchanged and updated in the same way as $p_t(x)$ and $q_t(x)$ are, but without normalization. When this process is infinitely repeated, they will converge to a specific function form. Since $p_t(x)$ and $q_t(x)$ can be calculated by normalizing $f_t(x)$ and $g_t(x)$ respectively, if there exist $f_t(x)$ and $g_t(x)$ converging to an impulse function, $p_t(x)$ and $q_t(x)$ will also converge to an impulse function, leading to zero uncertainty. To find such $f_t(x)$ and $g_t(x)$, $f_0(x)$ and $g_0(x)$ are defined as follows.

$$f_0(x) = \sqrt{\alpha}p_0(x) \mid \exists\alpha, x \in \mathbf{R} : \max\left(\alpha p_0(x)q_0(x)\right) > 1 \tag{3.41}$$

$$g_0(x) = \sqrt{\alpha}q_0(x) \mid \exists\alpha, x \in \mathbf{R} : \max\left(\alpha p_0(x)q_0(x)\right) > 1 \tag{3.42}$$

Assuming the robots exchange their estimation at each time step, the functions are defined as multiplication of their previous values, and they can be described as the multiplication of the priors. At $t \geq 1$, $f_t(x)$ and $g_t(x)$ can be the same form expressed with the priors.

$$f_t(x) = g_t(x) = f_{t-1}(x)g_{t-1}(x) = \left(\alpha p_0(x)q_0(x)\right)^{2^{t-1}} = \left(f_0(x)g_0(x)\right)^{2^{t-1}} \tag{3.43}$$

Let x_{max} be the value of x maximizing $\alpha p_0(x)q_0(x)$. Then, $\lim_{t\to\infty} f_t(x_{max})$ diverges to infinity, implying that $\lim_{t\to\infty} f_t(x)$ will be an impulse function whose impulse is at x_{max}. This can be proved by comparing the values of $f_t(x)$ at x_{max} and $x \neq x_{max}$. Let $D_t(x)$ be the difference between $f_t(x_{max})$ and $f_t(x)$ where $x \neq x_{max}$.

$$\begin{aligned} D_t(x) &= f_t(x_{max}) - f_t(x) \\ &= \left(f_0(x_{max})g_0(x_{max})\right)^{2^{t-1}} - \left(f_0(x)g_0(x)\right)^{2^{t-1}} \\ &= \left(f_0(x_{max})g_0(x_{max}) - f_0(x)g_0(x)\right) \sum_{i=0}^{2^{t-1}-1} \left\{ \left(f_0(x_{max})g_0(x_{max})\right)^{2^{t-1}-1-i} \left(f_0(x)g_0(x)\right)^{i} \right\} \end{aligned} \tag{3.44}$$

Since $\lim_{t\to\infty} \sum_{i=0}^{2^{t-1}-1} \left\{ \left(f_0(x_{max})g_0(x_{max})\right)^{2^{t-1}-1-i} \left(f_0(x)g_0(x)\right)^{i} \right\} \to \infty$,

$$\forall x \in \mathbf{R}, x \neq x_{max} : \lim_{t\to\infty} D_t(x) \to \infty \tag{3.45}$$

Thus, $f_t(x)$ will become an impulse function whose peak is at x_{max},

$$\lim_{t\to\infty} f_t(x) \to \delta(x - x_{max}) \tag{3.46}$$

where $\delta(x)$ is the delta function holding this property:

$$\forall c \in \mathbf{R} : \int \delta(x-c)f(x)dx = f(c) \tag{3.47}$$

This convergence to an impulse function affects the differential entropy of the probability distribution, which quantitatively represents the uncertainty in the estimate. The differential entropy of $p_t(x)$ and $q_t(x)$ are expressed as follows.

$$h(p_t) = -\int p_t(x)\log(p_t(x))dx \tag{3.48}$$

$$h(q_t) = -\int q_t(x)\log(q_t(x))dx \tag{3.49}$$

Based on Equation (3.39), $h(p_t)$ can be expressed using $f_t(x)$ as follows.

$$\begin{aligned}
h(p_t) &= -\int p_t(x)\log(p_t(x))dx \\
&= -\int p_t(x)\log\frac{f_t(x)}{\int f_t(x')dx'}dx \\
&= -\int p_t(x)\log f_t(x)dx - \int p_t(x)\log\frac{1}{\int f_t(x')dx'}dx \\
&= -\frac{1}{\int f_t(x')dx'}\int f_t(x)\log f_t(x)dx - \log\frac{1}{\int f_t(x')dx'}\int p_t(x)dx \\
&= -\frac{1}{\int f_t(x')dx'}\int f_t(x)\log f_t(x)dx - \log\frac{1}{\int f_t(x')dx'}
\end{aligned} \tag{3.50}$$

When $f_t(x)$ converges to an impulse function, the entropy is led to negative infinity.

$$\begin{aligned}
\lim_{t\to\infty} h(p_t) &\to -\frac{1}{\int \delta(x'-x_{max})dx'}\int \delta(x-x_{max})\log\delta(x-x_{max})dx - \log\frac{1}{\int \delta(x'-x_{max})dx'} \\
&= -\int \delta(x-x_{max})\log\delta(x-x_{max})dx - \log 1 \\
&= -\log\left(\delta(0)\right) = -\log(\infty) \\
&= -\infty
\end{aligned} \tag{3.51}$$

Similarly, based on Equation (3.40), $h(q_t)$ can be expressed using $g_t(x)$ and led to the same result.

$$\lim_{t\to\infty} h(q_t) \to -\infty \tag{3.52}$$

Thus, when the interaction is repeated infinitely, the differential entropy of the probability distribution diverges to negative infinity, resulting in zero of uncertainty in the distribution. This means

that the naive method causes infinite confidence in the estimation, i.e., overconfidence.

3.5.2 CI-based

The CI-based method applies fractional exponents to the probability distributions before multiplication and normalization.

$$p_t(x) = \frac{p_{t-1}^{\omega_p}(x) q_{t-1}^{1-\omega_p}(x)}{\int p_{t-1}^{\omega_p}(x') q_{t-1}^{1-\omega_p}(x') dx'} = \eta_{p_t} p_{t-1}^{\omega_p}(x) q_{t-1}^{1-\omega_p}(x) \tag{3.53}$$

$$q_t(x) = \frac{p_{t-1}^{1-\omega_q}(x) q_{t-1}^{\omega_q}(x)}{\int p_{t-1}^{1-\omega_q}(x') q_{t-1}^{\omega_q}(x') dx'} = \eta_{q_t} p_{t-1}^{1-\omega_q}(x) q_{t-1}^{\omega_q}(x) \tag{3.54}$$

where η_{p_t} and η_{q_t} are normalization factors.

The fractional exponents ω_p and ω_q are optimized so that the resulting probability distribution has the smallest uncertainty. Since the entropy of the resulting distribution is minimized as well when the uncertainty is minimized, ω_p and ω_q are selected based on the following equations.

$$\omega_p = \underset{\omega}{\operatorname{argmin}}\, h\left(p_t\right) \tag{3.55}$$

$$\omega_q = \underset{\omega}{\operatorname{argmin}}\, h\left(q_t\right) \tag{3.56}$$

In the simplified problem, with one-dimensional space, it is simply to choose the distribution with lower entropy [91]. For example, the fractional exponent is set to 1 if the entropy of the robot's own distribution is smaller than that of the other robot's distribution. Otherwise, it is set to 0.

$$\omega_p = \begin{cases} 1 & \text{if } h(p_{t-1}) \leq h(q_{t-1}) \\ 0 & \text{otherwise} \end{cases} \tag{3.57}$$

$$\omega_q = \begin{cases} 1 & \text{if } h(p_{t-1}) \geq h(q_{t-1}) \\ 0 & \text{otherwise} \end{cases} \tag{3.58}$$

Therefore, the resulting distributions will be either one of the original distributions.

$$p_t(x) = \begin{cases} p_{t-1}(x) & \text{if } h(p_{t-1}) \leq h(q_{t-1}) \\ q_{t-1}(x) & \text{otherwise} \end{cases} \tag{3.59}$$

$$q_t(x) = \begin{cases} q_{t-1}(x) & \text{if } h(p_{t-1}) \geq h(q_{t-1}) \\ p_{t-1}(x) & \text{otherwise} \end{cases} \tag{3.60}$$

As the robots repeat the interaction, the resulting distributions are determined based on the initial

distributions.

$$p_t(x) = \begin{cases} p_0(x) & \text{if } h(p_0) \leq h(q_0) \\ q_0(x) & \text{otherwise} \end{cases} \tag{3.61}$$

$$q_t(x) = \begin{cases} q_0(x) & \text{if } h(p_0) \geq h(q_0) \\ p_0(x) & \text{otherwise} \end{cases} \tag{3.62}$$

Thus, the CI-based method can be viewed as selecting a distribution with the lowest uncertainty. Since the entropy of the resulting distribution never gets less than that of the original distributions, the estimation will not be overconfident. This method works when all the robots are assumed to always give consistent estimation. If there is a one-point failure in the system and one of the robots gives inconsistent estimation with higher confidence, this estimation will be copied to other robots.

3.5.3 Conservative Data Exchange

Like the CI-based method, the conservative data exchange applies a fractional exponent before exchanging the probability distributions, but the fractional exponent is set to a constant.

$$p_t(x) = \frac{p_{t-1}^{\omega}(x) q_{t-1}^{1-\omega}(x)}{\int p_{t-1}^{\omega}(x') q_{t-1}^{1-\omega}(x') dx'} = \eta_{p_t} p_{t-1}^{\omega}(x) q_{t-1}^{1-\omega}(x) \tag{3.63}$$

$$q_t(x) = \frac{p_{t-1}^{1-\omega}(x) q_{t-1}^{\omega}(x)}{\int p_{t-1}^{1-\omega}(x') q_{t-1}^{\omega}(x') dx'} = \eta_{q_t} p_{t-1}^{1-\omega}(x) q_{t-1}^{\omega}(x) \tag{3.64}$$

where η_{p_t} and η_{q_t} are normalization factors.

As this method is expected to avoid overconfidence, $p_t(x)$ and $q_t(x)$ will not converge to an impulse function when repeating the interactions infinitely. To see if the convergence to an impulse function can be avoided, let's consider the functions $f_t(x)$ and $g_t(x)$, defined in Equations (3.39) and (3.40). Then, let $f_0(x)$ and $g_0(x)$ be the prior distributions.

$$f_0(x) = p_0(x) \tag{3.65}$$

$$g_0(x) = q_0(x) \tag{3.66}$$

As a constant fractional exponent is applied, they are updated as follows.

$$f_t(x) = f_{t-1}^{\omega}(x) g_{t-1}^{1-\omega}(x) \tag{3.67}$$

$$g_t(x) = f_{t-1}^{1-\omega}(x) g_{t-1}^{\omega}(x) \tag{3.68}$$

Based on these definitions, $f_t(x)$ and $g_t(x)$ are expected to be expressed by the prior distributions

$p_0(x)$ and $q_0(x)$ and some real numbers a_t and b_t.

$$f_t(x) = p_0^{a_t}(x) q_0^{b_t}(x) \tag{3.69}$$

$$g_t(x) = p_0^{b_t}(x) q_0^{a_t}(x) \tag{3.70}$$

By Equations (3.65) through (3.68), a_t and b_t can be expressed as recursion.

$$a_0 = 1 \tag{3.71}$$

$$b_0 = 0 \tag{3.72}$$

$$a_{t+1} = \omega a_t + (1-\omega) b_t \tag{3.73}$$

$$b_{t+1} = (1-\omega) a_t + \omega b_t \tag{3.74}$$

The closed form expressions of a_t and b_t are acquired by the following steps. First, the triple-term recursion is formulated.

$$\begin{aligned} a_{t+2} &= \omega a_{t+1} + (1-\omega) b_{t+1} \\ &= \omega a_{t+1} + (1-\omega)\left((1-\omega) a_t + \omega b_t\right) \\ &= \omega a_{t+1} + (1-\omega)^2 a_t + \omega(1-\omega)\left(\frac{a_{t+1}}{1-\omega} - \frac{\omega a_t}{1-\omega}\right) \\ &= 2\omega a_{t+1} + (1-2\omega) a_t \end{aligned} \tag{3.75}$$

This formula leads to the following characteristic equation and its solutions.

$$\begin{aligned} x^2 - 2\omega x - 1 + 2\omega &= 0 \\ (x-1)(x+1-2\omega) &= 0 \end{aligned} \tag{3.76}$$

By the solutions, Equation (3.75) becomes two geometric recursions.

$$\begin{aligned} a_{t+2} + (1-2\omega) a_{t+1} &= a_{t+1} + (1-2\omega) a_t \\ a_{t+2} - a_{t+1} &= (2\omega - 1)(a_{t+1} - a_t) \end{aligned} \tag{3.77}$$

And the closed forms are expressed as follows.

$$\begin{aligned} a_{t+1} + (1-2\omega) a_t &= a_1 + (1-2\omega) a_0 \\ &= 1 - \omega \\ a_{t+1} - a_t &= (2\omega - 1)^t (a_1 - a_0) \\ &= (2\omega - 1)^t (\omega - 1) \end{aligned} \tag{3.78}$$

From the equations above, the closed forms of a_t and b_t can be acquired.

$$\begin{aligned} 2(1-\omega)a_t &= (1+(2\omega-1)^t)(1-\omega) \\ a_t &= \frac{1}{2} + \frac{(2\omega-1)^t}{2} \end{aligned} \tag{3.79}$$

$$b_t = \frac{1}{2} - \frac{(2\omega-1)^t}{2} \tag{3.80}$$

Since $0 < \omega < 1$, when the robots repeats interactions infinitely, a_t and b_t converge to a constant.

$$\begin{aligned} \lim_{t\to\infty} a_t &\to \frac{1}{2} \\ \lim_{t\to\infty} b_t &\to \frac{1}{2} \end{aligned} \tag{3.81}$$

This concludes that f_t converges to multiplication of squared roots of the priors.

$$\lim_{t\to\infty} f_t(x) = \lim_{t\to\infty} p_0^{a_t}(x) q_0^{b_t}(x) \to p_0^{\frac{1}{2}}(x) q_0^{\frac{1}{2}}(x) = \sqrt{p_0(x)q_0(x)} \tag{3.82}$$

Therefore, the entropy of p_t will converge to a constant.

$$\lim_{t\to\infty} h(p_t) = \lim_{t\to\infty} h\left(\frac{f_t(x)}{\int f_t(x')dx'}\right) \to h\left(\frac{\sqrt{p_0(x)q_0(x)}}{\int \sqrt{p_0(x')q_0(x')}dx'}\right) = C \tag{3.83}$$

This means that the conservative data exchange can avoid the divergence to the negative infinity and overconfidence.

What if there are more than two robots? Let's say there are K robots. Each robot interacts with others at random. Like Equation (3.39), let's consider $f_{i,t}(x)$ based on robot i's probability distribution $p_{i,t}(x)$.

$$p_{i,t}(x) = \frac{f_{i,t}(x)}{\int f_{i,t}(x')dx'} \tag{3.84}$$

And, let robot i hold its prior distribution $p_{i,0}(x)$ and a function $f_{i,0}(x)$, which is defined as follows.

$$f_{i,0}(x) = p_{i,0}(x) \tag{3.85}$$

Then, the function $f_{i,t}(x)$ is updated as follows.

$$f_{i,t}(x) = f_{i,t-1}^{\omega}(x) f_{j,t-1}^{1-\omega}(x) \tag{3.86}$$

Note that $f_{j,t-1}(x)$ is the function of the other robot j with which the robot i interacts at time t.

As they repeat the interactions, $f_{i,t}(x)$ will be a product of the priors with fractional exponent $a_{k,t}$.

$$\begin{aligned} f_{i,t}(x) &= p_{1,0}^{a_{1,t}}(x) p_{2,0}^{a_{2,t}}(x) p_{3,0}^{a_{3,t}}(x) \ ... \ p_{K,0}^{a_{K,t}}(x) \\ &= \prod_{k=1}^{K} p_{k,0}^{a_{k,t}}(x) \end{aligned} \tag{3.87}$$

From Equation (3.85), the exponents for the robot i at time $t = 0$ is defined as follows.

$$\begin{aligned} a_{i,0} &= 1 \\ \forall j \neq i : a_{j,0} &= 0 \end{aligned} \tag{3.88}$$

As Equation (3.86) shows the exponents are just updated when the robots exchange their data, the exponent at time t is expressed as recursion.

$$a_{i,t} = \omega a_{i,t-1} + (1 - \omega) a_{j,t-1} \tag{3.89}$$

When the robots infinitely repeat the interactions, the exponents converge to specific values. As the robots randomly choose one another to interact at each time step, we can consider the expected values of the exponents. Let's say the robot i interacts with the other robot j. The expected value of $a_{i,t}$ can be expressed as follows.

$$\mathbf{E}\left[a_{i,t}\right] = \omega \mathbf{E}\left[a_{i,t-1}\right] + (1 - \omega) \mathbf{E}\left[a_{j,t-1}\right] \tag{3.90}$$

Similarly, the robot j interacts with another robot k, and the expected value of $a_{j,t}$ is expressed below. As the robot k may be the robot i ($k = i$) or one of the rest of the $K - 2$ robots ($k \neq i \neq j$), $\mathbf{E}\left[a_{k,t-1}\right]$ can be extended to the average value of the exponents of the robot i and the $K - 2$ other robots.

$$\begin{aligned} \mathbf{E}\left[a_{j,t}\right] &= \omega \mathbf{E}\left[a_{j,t-1}\right] + (1 - \omega) \mathbf{E}\left[a_{k,t-1}\right] \\ &= \omega \mathbf{E}\left[a_{j,t-1}\right] + (1 - \omega) \left(\frac{\mathbf{E}\left[a_{i,t-1}\right] + (K - 2) \mathbf{E}\left[a_{j,t-1}\right]}{K - 1} \right) \\ &= \frac{1 - \omega}{K - 1} \mathbf{E}\left[a_{i,t-1}\right] + \frac{K - 2 + \omega}{K - 1} \mathbf{E}\left[a_{j,t-1}\right] \end{aligned} \tag{3.91}$$

For notational simplicity, let us denote them by a_t and b_t.

$$\begin{aligned} a_t &= \mathbf{E}\left[a_{i,t}\right] \\ b_t &= \mathbf{E}\left[a_{j,t}\right] \end{aligned} \tag{3.92}$$

Then, from Equation (3.88),

$$\begin{aligned} a_0 &= \mathbf{E}\left[a_{i,0}\right] = 1 \\ b_0 &= \mathbf{E}\left[a_{j,0}\right] = 0 \end{aligned} \tag{3.93}$$

Equations (3.90) and (3.91) can then be expressed by a_t and b_t,

$$\begin{aligned} a_t &= \omega a_{t-1} + (1-\omega) b_{t-1} \\ b_t &= \frac{1-\omega}{K-1} a_{t-1} + \frac{K-2+\omega}{K-1} b_{t-1} \end{aligned} \tag{3.94}$$

In the same as the case of the two robots, the recursive formulas lead to the following triple-term recursive formula.

$$\begin{aligned} a_{t+2} &= \omega a_{t+1} + (1-\omega) b_{t+1} \\ &= \omega a_{t+1} + (1-\omega)\left(\frac{1-\omega}{K-1} a_t + \frac{K-2+\omega}{K-1} b_t\right) \\ &= \frac{K\omega + K - 2}{K-1} a_{t+1} + \frac{1-K\omega}{K-1} a_t \end{aligned} \tag{3.95}$$

and the following geometric recursive formulas.

$$\begin{aligned} a_{t+2} - a_{t+1} &= \frac{K\omega - 1}{K-1}(a_{t+1} - a_t) \\ a_{t+2} - \frac{K\omega - 1}{K-1} a_{t+1} &= a_{t+1} - \frac{K\omega - 1}{K-1} a_t \end{aligned} \tag{3.96}$$

Then,

$$\begin{aligned} a_{t+1} - a_t &= \left(\frac{K\omega - 1}{K-1}\right)^t (a_1 - a_0) \\ &= \left(\frac{K\omega - 1}{K-1}\right)^t (\omega - 1) \\ a_{t+1} - \frac{K\omega - 1}{K-1} a_t &= a_1 - \frac{K\omega - 1}{K-1} a_0 \\ &= \omega - \frac{K\omega - 1}{K-1} \\ &= \frac{1-\omega}{K-1} \end{aligned} \tag{3.97}$$

Finally, the closed form expression of a_t is acquired.

$$a_t = \frac{1}{K} + \frac{K-1}{K}\left(\frac{K\omega - 1}{K-1}\right)^t \tag{3.98}$$

Since $0 < \omega < 1$, when they repeat interactions infinitely, a_t, i.e., the expected value of $a_{i,t}$, converges to a constant based on the number of robots K.

$$\lim_{t\to\infty} a_t = \lim_{t\to\infty} \mathbf{E}\left[a_{i,t}\right] \to \frac{1}{K} \tag{3.99}$$

Then, from Equation (3.87), $f_{i,t}$ also converges to a product of the priors at time $t = 0$, i.e., $p_{k,0}$, to the power of $1/K$.

$$\lim_{t\to\infty} f_{i,t} \to \prod_{k=1}^{K} p_{k,0}^{\frac{1}{K}}(x) = \left(\prod_{k=1}^{K} p_{k,0}(x)\right)^{\frac{1}{K}} \tag{3.100}$$

This means that $p_{i,t}$ converges to a function proportional to a K-th root of the product of the K priors. As this does not lead to an impulse function, the entropy of $p_{i,t}$ converges to a constant.

$$\lim_{t\to\infty} h(p_{i,t}) = \lim_{t\to\infty} h\left(\frac{f_{i,t}(x)}{\int f_{i,t}(x')dx'}\right) \to h\left(\frac{\prod_{k=1}^{K} p_{k,0}^{\frac{1}{K}}(x)}{\int \prod_{k'=1}^{K} p_{k',0}^{\frac{1}{K}}(x')dx'}\right) = C \tag{3.101}$$

Thus, the entropy of the resulting distribution never gets negative infinity, avoiding overconfidence. Unlike the CI-based method, the proposed method does not prioritise a distribution with small entropy. This indicates that it may make the multi-robot system stable even if there is a malfunctioning robot in the group that gives erroneous estimation with higher confidence.

3.6 Cases with Sensor Noises

Previously, the cases without noises have been modeled and presented to show that overconfidence takes place when they exchange their data without confidence reduction. Here, the cases with considering the sensor noises will be discussed and presented.

Like the previous section, the two robots calculate their individual estimates $p_t(x)$ and $q_t(x)$ by exchanging their previous estimates $p_{t-1}(x)$ and $q_{t-1}(x)$. With noises, on the other hand, the sensory data contains noises, d_{t-1}. This makes the robot to translate or shift the distribution of the other robot $q_{t-1}(x)$ by d_{t-1}, leading to $q_{t-1}(x - d_{t-1})$. Before the multiplication of its previous estimate and the one from the other robot, the robot needs to take the noises into consideration, and it needs to convert the distribution $q_{t-1}(x-d_{t-1})$ based on the sensor model $r(x)$. Let's call this converted distribution $p'_{t-1}(x)$, estimation for the robot based on the other robot's estimation and the sensory data. Similarly, the other robot converts $p_{t-1}(x-d_{t-1})$ into $q'_{t-1}(x)$. These distributions reflecting the noises can be expressed by convolution by the sensor model.

$$
\begin{aligned}
p'_{t-1}(x) &= \int_{-\infty}^{\infty} q_{t-1}(x - d_{t-1} - x')r(x')dx' \\
&= \int_{-\infty}^{\infty} q_{t-1}(x - x')r(x' - d_{t-1})dx' \\
q'_{t-1}(x) &= \int_{-\infty}^{\infty} p_{t-1}(x - d_{t-1} - x')r(x')dx' \\
&= \int_{-\infty}^{\infty} p_{t-1}(x - x')r(x' - d_{t-1})dx'
\end{aligned}
\tag{3.102}
$$

The sensor model $r(x)$ is usually expressed as a normal distribution.

$$
r(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.103}
$$

3.6.1 Naive Method

The naive method simply combines the generated distribution with the robot's own distribution by multiplication and normalization.

$$
\begin{aligned}
p_t(x) &= \frac{p_{t-1}(x)p'_{t-1}(x)}{\int p_{t-1}(x')p'_{t-1}(x')dx'} = \eta_{p_t} p_{t-1}(x)p'_{t-1}(x) \\
q_t(x) &= \frac{q_{t-1}(x)q'_{t-1}(x)}{\int q_{t-1}(x')q'_{t-1}(x')dx'} = \eta_{q_t} q_{t-1}(x)q'_{t-1}(x)
\end{aligned}
\tag{3.104}
$$

where η_{p_t} and η_{q_t} are normalization factors.

The robot's interactions alternately repeat the convolution in Equation (3.102) and the multiplication and normalization in Equation (3.104). Based on the equations, $p_t(x)$ and $q_t(x)$ can be expressed by using older distributions, $p_{t-1}(x)$, $q_{t-1}(x)$, $p_{t-2}(x)$, $q_{t-2}(x)$, and so on. For example, $p_t(x)$ can be extended into a form composed of $p_{t-2}(x)$ and $q_{t-2}(x)$ as follows.

$$
\begin{aligned}
p_t(x) =& \eta_{p_t} p_{t-1}(x) p'_{t-1}(x) \\
=& \eta_{p_t} \eta_{p_{t-1}} p_{t-2}(x) p'_{t-2}(x) p'_{t-1}(x) \\
=& \eta_{p_t} \eta_{p_{t-1}} p_{t-2}(x) \\
& \int_{-\infty}^{\infty} q_{t-2}(x - x'') r(x'' - d_{t-2}) dx'' \int_{-\infty}^{\infty} q_{t-1}(x - x') r(x' - d_{t-1}) dx' \\
=& \eta_{p_t} \eta_{p_{t-1}} p_{t-2}(x) \\
& \int_{-\infty}^{\infty} q_{t-2}(x - x'') r(x'' - d_{t-2}) dx'' \int_{-\infty}^{\infty} \eta_{q_{t-1}} q_{t-2}(x - x') q'_{t-2}(x - x') r(x' - d_{t-1}) dx' \\
=& \eta_{p_t} \eta_{p_{t-1}} p_{t-2}(x) \\
& \int_{-\infty}^{\infty} q_{t-2}(x - x'') r(x'' - d_{t-2}) dx'' \\
& \int_{-\infty}^{\infty} \eta_{q_{t-1}} q_{t-2}(x - x') \int_{-\infty}^{\infty} p_{t-2}(x - x' - x''') r(x''' - d_{t-2}) dx''' r(x' - d_{t-1}) dx'
\end{aligned}
\tag{3.105}
$$

This equation can be reformed so that the multiplications of probability distributions can be first done followed by the convolution with the sensor models.

$$
\begin{aligned}
p_t(x) =& \eta_{p_t} \eta_{p_{t-1}} \eta_{q_{t-1}} \\
& \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_{t-2}(x) p_{t-2}(x - x' - x''') q_{t-2}(x - x') q_{t-2}(x - x'') \\
& r(x' - d_{t-1}) r(x'' - d_{t-2}) r(x''' - d_{t-2}) dx' dx'' dx'''
\end{aligned}
\tag{3.106}
$$

Likewise, $q_t(x)$ can be expressed by a multiplications of $q_{t-2}(x)$ and $p_{t-2}(x)$ followed by the convolution with the sensor models.

$$
\begin{aligned}
q_t(x) =& \eta_{q_t} \eta_{q_{t-1}} \eta_{p_{t-1}} \\
& \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} q_{t-2}(x) q_{t-2}(x - x' - x''') p_{t-2}(x - x') p_{t-2}(x - x'') \\
& r(x' - d_{t-1}) r(x'' - d_{t-2}) r(x''' - d_{t-2}) dx' dx'' dx'''
\end{aligned}
\tag{3.107}
$$

These equations show that the older probability distributions $p_{t-2}(x)$ and $q_{t-2}(x)$ appear as a factor more than once in the updated probability distributions $p_t(x)$ and $q_t(x)$. If $p_{t-2}(x)$ and $q_{t-2}(x)$ are extended as well and the extension is repeated, $p_t(x)$ and $q_t(x)$ can be eventually expressed by

using multiple factors of the original distributions at $t = 0$, i.e., $p_0(x)$ and $q_0(x)$.

$$\begin{aligned} p_t(x) =& \eta_{p_t}\eta_{p_{t-1}}\eta_{q_{t-1}}\cdots \\ & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots p_0(x)p_0(x-x'-x''')\cdots q_0(x-x')q_0(x-x'')\cdots \\ & r(x'-d_{t-1})r(x''-d_{t-2})r(x'''-d_{t-2})\cdots dx'dx''dx'''\cdots \\ q_t(x) =& \eta_{q_t}\eta_{q_{t-1}}\eta_{p_{t-1}}\cdots \\ & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots q_0(x)q_0(x-x'-x''')\cdots p_0(x-x')p_0(x-x'')\cdots \\ & r(x'-d_{t-1})r(x''-d_{t-2})r(x'''-d_{t-2})\cdots dx'dx''dx'''\cdots \end{aligned} \tag{3.108}$$

These multiple factors of the original probability function is similar to the noiseless case, which has led to an impulse function when t is infinitely large. But here, the convolution part must be investigated so that the naive method with sensor noises also leads to an impulse function when t is infinitely large.

Because of the convolution, there are infinitely many combinations each of which is a multiplication of functions that are shifted (or translated). But, since a probability function is always non-negative, the product of the differently shifted functions cannot be greater than that of the identically shifted functions. To prove that, consider a non-negative function $f(x)$ whose maximum is at $x_{\max}$.

$$f(x_{\max}) > f(x), \forall x \neq x_{\max} \tag{3.109}$$

Then, consider a multiplication of the functions each of which is shifted by different values, $a, b \in \mathbf{R}$, i.e., $f(x-a)f(x-b)$, whose maximum is at $x'_{\max}$.

$$f(x'_{\max}-a)f(x'_{\max}-b) > f(x-a)f(x-b), \forall x \neq x'_{\max} \tag{3.110}$$

When a and b are the same value $c \in \mathbf{R}$, its maximum is equivalent to that of $f(x)f(x)$.

$$\begin{aligned} \max\left(f(x-c)f(x-c)\right) &= \max\left(f(x-c)\right)\max\left(f(x-c)\right) \\ &= \max\left(f(x)\right)\max\left(f(x)\right) \\ &= f(x_{\max})f(x_{\max}) \end{aligned} \tag{3.111}$$

Then,

$$\begin{aligned}\max\left(f(x-a)f(x-b)\right) &= f(x'_{\max}-a)f(x'_{max}-b)\\ &< f(x_{\max})f(x_{\max})\\ &= \max\left(f(x-c)f(x-c)\right)\end{aligned} \tag{3.112}$$

Thus, the maximum of a multiplication of differently shifted functions $f(x-a)f(x-b)$ is always less than the maximum of that of identically shifted functions $f(x-c)f(x-c)$.

If it is repeated n times,

$$\max\left((f(x-c))^n\right) > \max\left(\prod_{i=1}^{n} f(x-c_i)\right) \tag{3.113}$$

where there exist some c_i and c_j that are not equal, i.e., $\exists i,j \in \mathbf{N} \mid 1 \le i < j \le n, c_i \neq c_j$. This inequality gets more significant as n increases. Consider the ratio of the left hand side and the right hand side.

$$\frac{\max\left((f(x-c))^n\right)}{\max\left(\prod_{i=1}^{n} f(x-c_i)\right)} \tag{3.114}$$

$$\begin{aligned}\frac{\max\left((f(x-c))^n\right)}{\max\left(\prod_{i=1}^{n} f(x-c_i)\right)} &= \frac{\max\left((f(x-c))^n\right)}{\max\left(f(x-c_n)\prod_{i}^{n-1} f(x-c_i)\right)}\\ &> \frac{\max\left((f(x-c))^n\right)}{\max\left(f(x-c_n)\right)\max\left(\prod_{i}^{n-1} f(x-c_i)\right)}\\ &= \frac{\max\left(f(x-c)\right)}{\max\left(f(x-c_n)\right)}\frac{\max\left((f(x-c))^{n-1}\right)}{\max\left(\prod_{i}^{n-1} f(x-c_i)\right)}\\ &= \frac{\max\left((f(x-c))^{n-1}\right)}{\max\left(\prod_{i}^{n-1} f(x-c_i)\right)}\end{aligned} \tag{3.115}$$

Thus, this ratio holds the recursive inequality and it increases as the multiplication is repeated. This indicates that the product of identically shifted functions will be more significant than that of those with smaller maximum values.

Since a probability density function is always non-negative, it holds this characteristic above. As the robots interact with each other multiple times, the identically shifted functions will become more significant. Hence, the probability distribution will get close to the multiplication of the original probability distributions which are identically shifted. Like the noiseless case, infinite

products of a function with normalization leads to an impulse function at the maximum of the original function. Let x at the maximum of the function be $x_{\max}$.

$$p_0(x-a)q_0(x-b)p_0(x-a)q_0(x-b)p_0(x-a)q_0(x-b)\cdots \to \delta(x-x_{\max}) \tag{3.116}$$

Then, $p_t(x)$ will be expressed as follows.

$$\begin{aligned}\lim_{t\to\infty} p_t(x) \to & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\delta(x-x_{\max})r(x'-d_{t-1})r(x''-d_{t-2})r(x'''-d_{t-2})\cdots dx'dx''dx'''\cdots \\ =&\delta(x-x_{\max})\int_{-\infty}^{\infty} r(x'-d_{t-1})dx'\int_{-\infty}^{\infty} r(x''-d_{t-2})dx''\int_{-\infty}^{\infty} r(x'''-d_{t-2})dx'''\cdots \\ =&\delta(x-x_{\max})\end{aligned} \tag{3.117}$$

Therefore, $p_t(x)$ converges to an impulse function. This leads to the same entropy in the case without the sensory noises, which is negative infinity. Thus, the naive method will cause overconfidence in the case of including sensory noises.

3.6.2 CI-based

The CI-based method applies fractional exponents to the robot's own probability distribution and the convoluted distribution which represents the one from the other.

$$\begin{aligned} p_t(x) &= \frac{p_{t-1}^{\omega_p} p_{t-1}'^{1-\omega_p}(x)}{\int p_{t-1}^{\omega_p}(x')p_{t-1}'^{1-\omega_p}(x')dx'} = \eta_{pt} p_{t-1}^{\omega_p}(x)p_{t-1}'^{1-\omega_p}(x) \\ q_t(x) &= \frac{q_{t-1}^{\omega_q}(x) q_{t-1}'^{1-\omega_q}(x)}{\int q_{t-1}^{\omega_q}(x')q_{t-1}'^{1-\omega_q}(x')dx'} = \eta_{qt} q_{t-1}^{\omega_q}(x)q_{t-1}'^{1-\omega_q}(x) \end{aligned} \tag{3.118}$$

where η_{pt} and η_{qt} are normalization factors. As defined, the fractional exponents are optimized so that the resulting distribution will have the smallest entropy. This results in selecting either the original distribution or the convoluted distribution from the other, depending on their entropy values. That is, the robot keeps the original distribution if it has a smaller entropy. Otherwise, the original distribution will be replaced with the one from the other robot.

$$p_t(x) = \begin{cases} p_{t-1}(x) & \text{if } h(p_{t-1}) \le h(p'_{t-1}) \\ p'_{t-1}(x) & \text{otherwise} \end{cases} \tag{3.119}$$

$$q_t(x) = \begin{cases} q_{t-1}(x) & \text{if } h(q'_{t-1}) \ge h(q_{t-1}) \\ q'_{t-1}(x) & \text{otherwise} \end{cases} \tag{3.120}$$

Since convolution does not decrease the entropy, the replacement will not take place twice.

For example, if the entropy of the original distribution $p_0(x)$ is small enough to replace the other robot's distribution $q_0(x)$, the first robot will keep its own $p_0(x)$ and the second robot will replace its distribution $q_0(x)$ with the convoluted distribution given from the first $q'_0(x)$. After this exchange, they will no longer replace their distributions with the one from each other. Thus, the resulting distributions are determined based on the initial distributions.

$$p_t(x) = \begin{cases} p_0(x) & \text{if } h(p_0) \leq h(p'_0) \\ p'_0(x) & \text{otherwise} \end{cases} \tag{3.121}$$

$$q_t(x) = \begin{cases} q_0(x) & \text{if } h(q'_0) \geq h(q_0) \\ q'_0(x) & \text{otherwise} \end{cases} \tag{3.122}$$

As the entropy will not diverge to negative infinity, the overconfidence can be avoided, but in a similar way as the noiseless case, it assumes every robot to always give consistent estimation.

3.6.3 Conservative Data Exchange

Like the CI-based method, the conservative data exchange applies fractional exponents but in a different way. It applies a constant exponent ω and $1-\omega$ to the probability distributions, and the robot will have $p^\omega(x)$ and $p^{1-\omega}(x)$, and the other robot will have $q^\omega(x)$ and $q^{1-\omega}(x)$. To interact with each other, they keep the first distributions and exchange their second ones; the robot keeps $p^\omega(x)$ and sends $p^{1-\omega}(x)$ to the other robot, and the other robot keeps $q^\omega(x)$ and sends $q^{1-\omega}$.

They generate their updated estimates $p_t(x)$ and $q_t(x)$ as follows.

$$\begin{aligned} p_t(x) &= \frac{p^\omega_{t-1}(x)p'_{t-1}(x)}{\int p^\omega_{t-1}(x')p'_{t-1}(x')dx'} = \eta_{p_t} p^\omega_{t-1}(x)p'_{t-1}(x) \\ q_t(x) &= \frac{q^\omega_{t-1}(x)q'_{t-1}(x)}{\int q^\omega_{t-1}(x')q'_{t-1}(x')dx'} = \eta_{q_t} q^\omega_{t-1}(x)q'_{t-1}(x) \end{aligned} \tag{3.123}$$

$p'_{t-1}(x)$ and $q'_{t-1}(x)$ are calculated as follows.

$$\begin{aligned} p'_{t-1}(x) &= \int_{-\infty}^{\infty} q^{1-\omega}_{t-1}(x - d_{t-1} - x')r(x')dx' \\ &= \int_{-\infty}^{\infty} q^{1-\omega}_{t-1}(x - x')r(x' - d_{t-1})dx' \\ q'_{t-1}(x) &= \int_{-\infty}^{\infty} p^{1-\omega}_{t-1}(x - d_{t-1} - x')r(x')dx' \\ &= \int_{-\infty}^{\infty} p^{1-\omega}_{t-1}(x - x')r(x' - d_{t-1})dx' \end{aligned} \tag{3.124}$$

Based on the definitions above, $p_t(x)$ is extended as follows.

$$
\begin{aligned}
p_t(x) =& \eta_{p_t} p_{t-1}^{\omega}(x) p'_{t-1}(x) \\
=& \eta_{p_t} \eta_{p_{t-1}}^{\omega} p_{t-2}^{\omega^2}(x) p'^{\omega}_{t-2}(x) p'_{t-1}(x) \\
=& \eta_{p_t} \eta_{p_{t-1}}^{\omega} p_{t-2}^{\omega^2}(x) \\
& \left(\int_{-\infty}^{\infty} q_{t-2}^{1-\omega}(x - x'') r(x'' - d_{t-2}) dx'' \right)^{\omega} \int_{-\infty}^{\infty} q_{t-1}^{1-\omega}(x - x') r(x' - d_{t-1}) dx' \\
=& \eta_{p_t} \eta_{p_{t-1}}^{\omega} p_{t-2}^{\omega^2}(x) \\
& \left(\int_{-\infty}^{\infty} q_{t-2}^{1-\omega}(x - x'') r(x'' - d_{t-2}) dx'' \right)^{\omega} \\
& \int_{-\infty}^{\infty} \eta_{q_{t-1}}^{1-\omega} q_{t-2}^{\omega(1-\omega)}(x - x') q'^{1-\omega}_{t-2}(x - x') r(x' - d_{t-1}) dx' \\
=& \eta_{p_t} \eta_{p_{t-1}}^{\omega} p_{t-2}^{\omega^2}(x) \\
& \left(\int_{-\infty}^{\infty} q_{t-2}^{1-\omega}(x - x'') r(x'' - d_{t-2}) dx'' \right)^{\omega} \\
& \int_{-\infty}^{\infty} \eta_{q_{t-1}}^{1-\omega} q_{t-2}^{\omega(1-\omega)}(x - x') \left(\int_{-\infty}^{\infty} p_{t-2}^{1-\omega}(x - x' - x''') r(x''' - d_{t-2}) dx''' \right)^{1-\omega} r(x' - d_{t-1}) dx' \\
=& \eta_{p_t} \eta_{p_{t-1}}^{\omega} \eta_{q_{t-1}}^{1-\omega} \\
& \int_{-\infty}^{\infty} p_{t-2}^{\omega^2}(x) \left(\int_{-\infty}^{\infty} p_{t-2}^{1-\omega}(x - x' - x''') r(x''' - d_{t-2}) dx''' \right)^{1-\omega} \\
& q_{t-2}^{\omega(1-\omega)}(x - x') \left(\int_{-\infty}^{\infty} q_{t-2}^{1-\omega}(x - x'') r(x'' - d_{t-2}) dx'' \right)^{\omega} r(x' - d_{t-1}) dx' \\
=& \eta_{p_{t-1}} \eta_{p_{t-1}}^{\omega} \eta_{q_{t-1}}^{1-\omega} \\
& \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} \left(p_{t-2}^{\omega^2}(x) p_{t-2}^{(1-\omega)^2}(x - x' - x''') \right)^{\frac{1}{1-\omega}} r(x''' - d_{t-2}) dx''' \right)^{1-\omega} \\
& \left(\int_{-\infty}^{\infty} \left(q_{t-2}^{\omega(1-\omega)}(x - x') q_{t-2}^{\omega(1-\omega)}(x - x'') \right)^{\frac{1}{\omega}} r(x'' - d_{t-2}) dx'' \right)^{\omega} r(x' - d_{t-1}) dx'
\end{aligned}
\tag{3.125}
$$

If overconfidence took place when the robots repeat interactions, the probability distribution would become an impulse function, whose maximum value is infinity. Consider the maximum of $p_t(x)$. It cannot be greater than the maximum of the case when all sensor noise (e.g., d_t) are 0 and the

sensor model is noiseless, i.e., $r(x) = \delta(x)$, where $\delta(x)$ is an impulse function.

$$
\begin{aligned}
\max\left(p_t(x)\right) \leq & \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega} \\
& \max\left(\int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty}\left(p_{t-2}^{\omega^2}(x)p_{t-2}^{(1-\omega)^2}(x-x'-x''')\right)^{\frac{1}{1-\omega}}\delta(x''')dx'''\right)^{1-\omega}\right. \\
& \left.\left(\int_{-\infty}^{\infty}\left(q_{t-2}^{\omega(1-\omega)}(x-x')q_{t-2}^{\omega(1-\omega)}(x-x'')\right)^{\frac{1}{\omega}}\delta(x'')dx''\right)^{\omega}\delta(x')dx'\right) \\
= & \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega} \\
& \max\left(\int_{-\infty}^{\infty}\left(\left(p_{t-2}^{\omega^2}(x)p_{t-2}^{(1-\omega)^2}(x-x')\right)^{\frac{1}{1-\omega}}\right)^{1-\omega}\right. \\
& \left.\left(\left(q_{t-2}^{\omega(1-\omega)}(x-x')q_{t-2}^{\omega(1-\omega)}(x)\right)^{\frac{1}{\omega}}\right)^{\omega}\delta(x')dx'\right) \\
= & \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega} \\
& \max\left(\int_{-\infty}^{\infty}p_{t-2}^{\omega^2}(x)p_{t-2}^{(1-\omega)^2}(x-x')q_{t-2}^{\omega(1-\omega)}(x-x')q_{t-2}^{\omega(1-\omega)}(x)\delta(x')dx'\right) \\
= & \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega}\max\left(p_{t-2}^{\omega^2}(x)p_{t-2}^{(1-\omega)^2}(x)q_{t-2}^{\omega(1-\omega)}(x)q_{t-2}^{\omega(1-\omega)}(x)\right) \\
= & \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega}\max\left(p_{t-2}^{\omega^2+(1-\omega)^2}(x)q_{t-2}^{2\omega(1-\omega)}(x)\right)
\end{aligned}
\tag{3.126}
$$

Likewise,

$$
\max\left(q_t(x)\right) \leq \eta_{q_t}\eta_{q_{t-1}}^{\omega}\eta_{p_{t-1}}^{1-\omega}\max\left(q_{t-2}^{\omega^2+(1-\omega)^2}(x)p_{t-2}^{2\omega(1-\omega)}(x)\right) \tag{3.127}
$$

The inner part of *max* appears in the noiseless case of the conservative data exchange when Equations (3.63) and (3.64) are extended as follows.

$$
\begin{aligned}
p_t(x) &= \eta_{p_t}p_{t-1}^{\omega}(x)q_{t-1}^{1-\omega}(x) \\
&= \eta_{p_t}\left(\eta_{p_{t-1}}p_{t-2}^{\omega}(x)q_{t-2}^{1-\omega}(x)\right)^{\omega}\left(\eta_{q_{t-1}}q_{t-2}^{\omega}(x)p_{t-2}^{1-\omega}(x)\right)^{1-\omega} \\
&= \eta_{p_t}\eta_{p_{t-1}}^{\omega}\eta_{q_{t-1}}^{1-\omega}p_{t-2}^{\omega^2+(1-\omega)^2}(x)q_{t-2}^{2\omega(1-\omega)}(x)
\end{aligned}
\tag{3.128}
$$

$$
\begin{aligned}
q_t(x) &= \eta_{q_t}p_{t-1}^{1-\omega}(x)q_{t-1}^{\omega}(x) \\
&= \eta_{q_t}\left(\eta_{p_{t-1}}p_{t-2}^{\omega}(x)q_{t-2}^{1-\omega}(x)\right)^{1-\omega}\left(\eta_{q_{t-1}}q_{t-2}^{\omega}(x)p_{t-2}^{1-\omega}(x)\right)^{\omega} \\
&= \eta_{q_t}\eta_{p_{t-1}}^{1-\omega}\eta_{q_{t-1}}^{\omega}q_{t-2}^{\omega^2+(1-\omega)^2}(x)p_{t-2}^{2\omega(1-\omega)}(x)
\end{aligned}
\tag{3.129}
$$

where η_{p_t}, $\eta_{p_{t-1}}$, and $\eta_{q_{t-1}}$ are normalization factors. Thus, their upper bounds are the noiseless case of the conservative data exchange, whose maximum is always finite. Therefore, $p_t(x)$ and $q_t(x)$ do not converge to an impulse function and they can avoid overconfidence. In addition, as each information spreads over the system, the method may be stable even when there is a malfunctioning

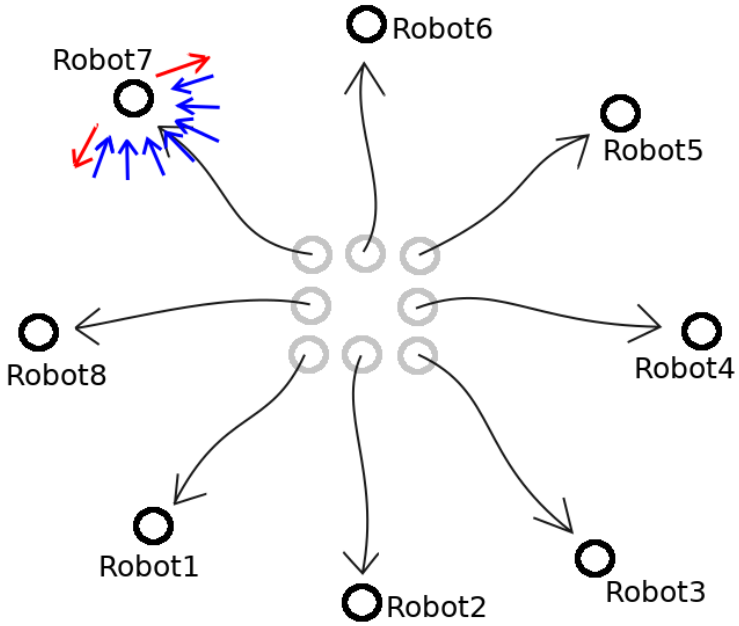

Figure 3.7: Motions of robots to simulate. Each circle represents a robot. As indicated by the gray circles, the eight robots are initially deployed in a square formation at the beginning, and moved by forces determined by their positions. For example, the red and blue arrows are forces being applied to Robot7 (the top-left robot). The red arrows pull the robot toward its neighbors (attractive forces). The blue arrows pushes it away from the others (repulsive forces). Eventually, the robots move away from each other while keeping themselves in a circular formation.

robot giving an inconsistent estimation.

3.7 Simulation

The proposed method has been simulated and demonstrated in a two-dimensional space $\mathbf{x} = \{x_1, x_2\}$. The performance was compared with the naive method and the CI-based method, and the resulting work was presented in Ubiquitous Robots 2020 [97]. The source code is available at Github [98]. We also included the centralized version of localization in this section, which provides an upper bound for the accuracy by estimating all the locations of the robots in a centralized manner.

In this section, the simulation setups are first described, including the details about the centralized version. Then, the simulation results are presented.

3.7.1 Simulation Setups

In the simulation, the probability distributions are assumed to follow a normal distribution. Also, a robot's state is represented by the two-dimensional location only without orientation. Figure 3.7 shows how robots move in the simulation. The gray circles represent the robots' initial locations, and the black circles are those after the simulation starts. They are first deployed in a square formation. For each robot, forces are calculated based on their relative locations. An attractive force is applied so the robot is pulled toward its neighboring robot. A repulsive force is also calculated with respect to all the other robots. By simulating these forces, the robots keep a circular formation while they move. They keep moving until the balance of the forces reaches an equilibrium. Noises are also added to the forces so the trajectories acquire some randomness. The simulation is conducted for 100 seconds and an equilibrium is reached in the first 50 seconds. After the 50 seconds, the simulation parameters such as the attractive and repulsive forces are changed so the robots move further away until they reach another equilibrium. That is, they reach an equilibrium twice during the simulation. If their estimates entail overconfidence, they converge to inconsistent estimates when they reach the first equilibrium. After that, since they continue to estimate their locations with significant errors until the second equilibrium, this motions should make the effects of overconfidence appear more significantly.

For localization, the robots take measurement about distances between them. The ad hoc network that they establish is assumed to be fully connected, therefore a robot can communicate with every other robots. Note that this fully connected network contains abundant cyclic routes. When a pair of robots communicate, they measure the distance between them and exchange their estimates. In this simulation, the first deployed robot of the group also has access to landmarks and performs global localization so that it can provide other robots with information regarding the environment.

The proposed method was compared with three other approaches. The first one is a naive method which does not apply the fractional exponents; each robot just copies its estimate without any amplitude reduction. This is equivalent to the extended Kalman filter (EKF) assuming the estimated locations do not have correlations. The second one is a Covariance Intersection-based method, in which each robot copies its own estimate and also performs covariance intersection to update it. For this method, the ω is determined so that the trace of the generated covariance matrix is minimized, whereas it is selected empirically and set to 0.95 for the proposed method. The last one is the centralized version. This version performs EKF on a large covariance matrix about the entire system: each robot's covariance and also correlations between them. Since it has an access to the correlation information, the resulting estimation provides an upper-bound of accuracy for the other methods, which do not have knowledge about the correlations between the robots.

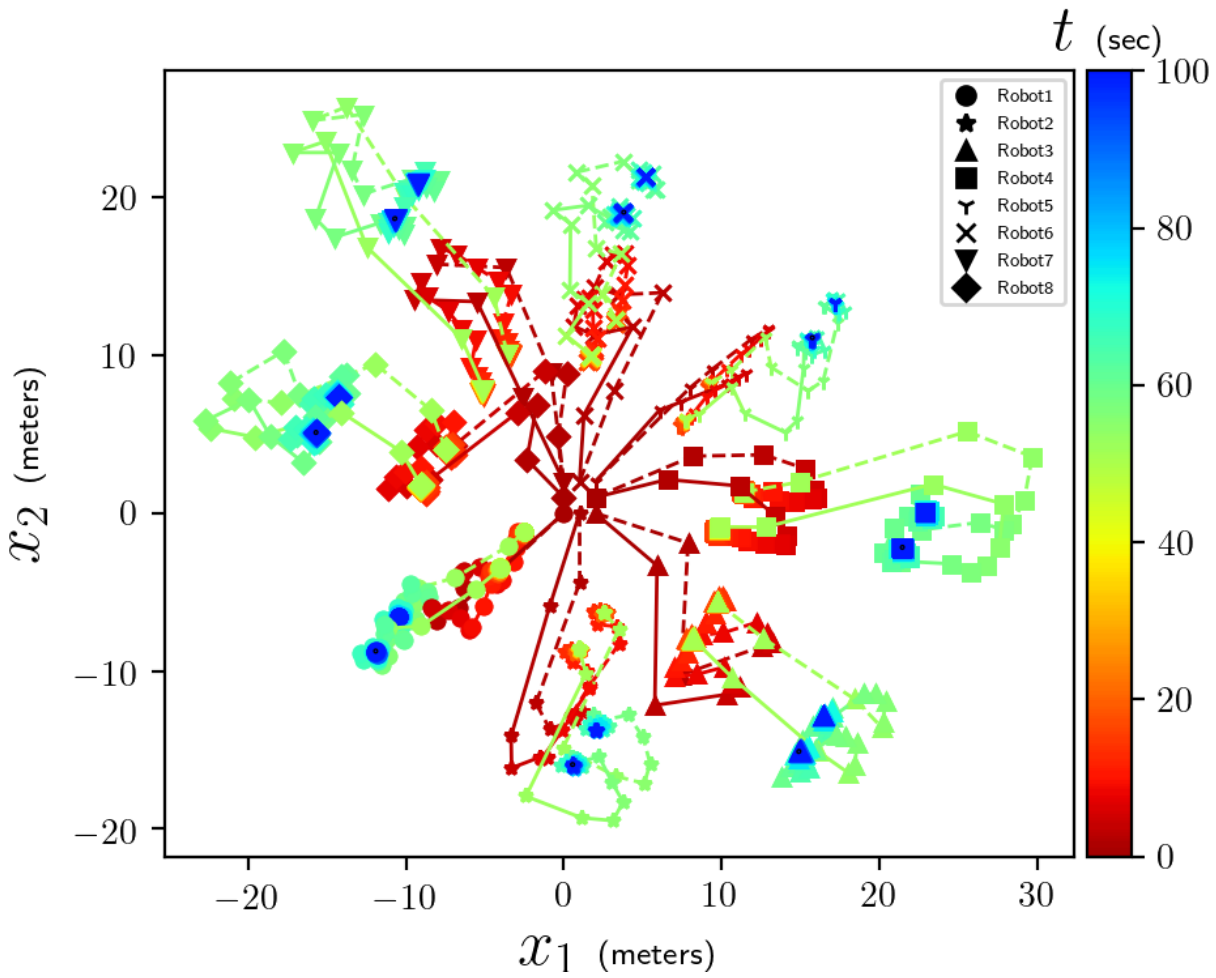

Figure 3.8: Simulated trajectories of robots and estimations provided by the naive method in the two-dimensional space $\mathbf{x} = \{x_1, x_2\}$. The dashed lines are actual trajectories. The solid lines are trajectories estimated by the robots. The colors represent simulation time (brown is the oldest and blue is the latest.)

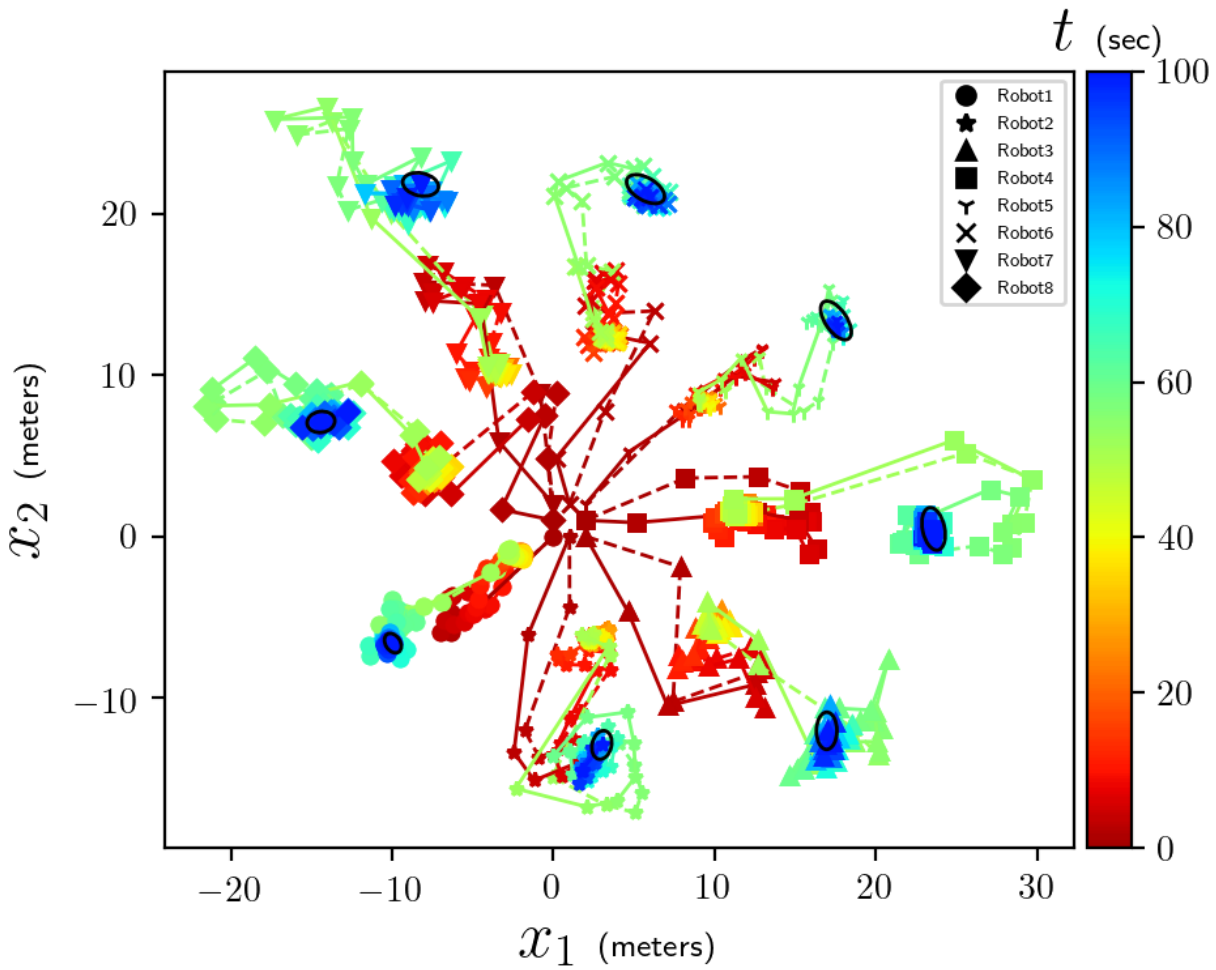

Figure 3.9: Simulated trajectories of robots and estimations provided by the CI-based method in the two-dimensional space $\mathbf{x} = \{x_1, x_2\}$. The dashed lines are actual trajectories. The solid lines are trajectories estimated by the robots. The colors represent simulation time (brown is the oldest and blue is the latest.) The ellipses represent the covariance of their estimates at the last time step.

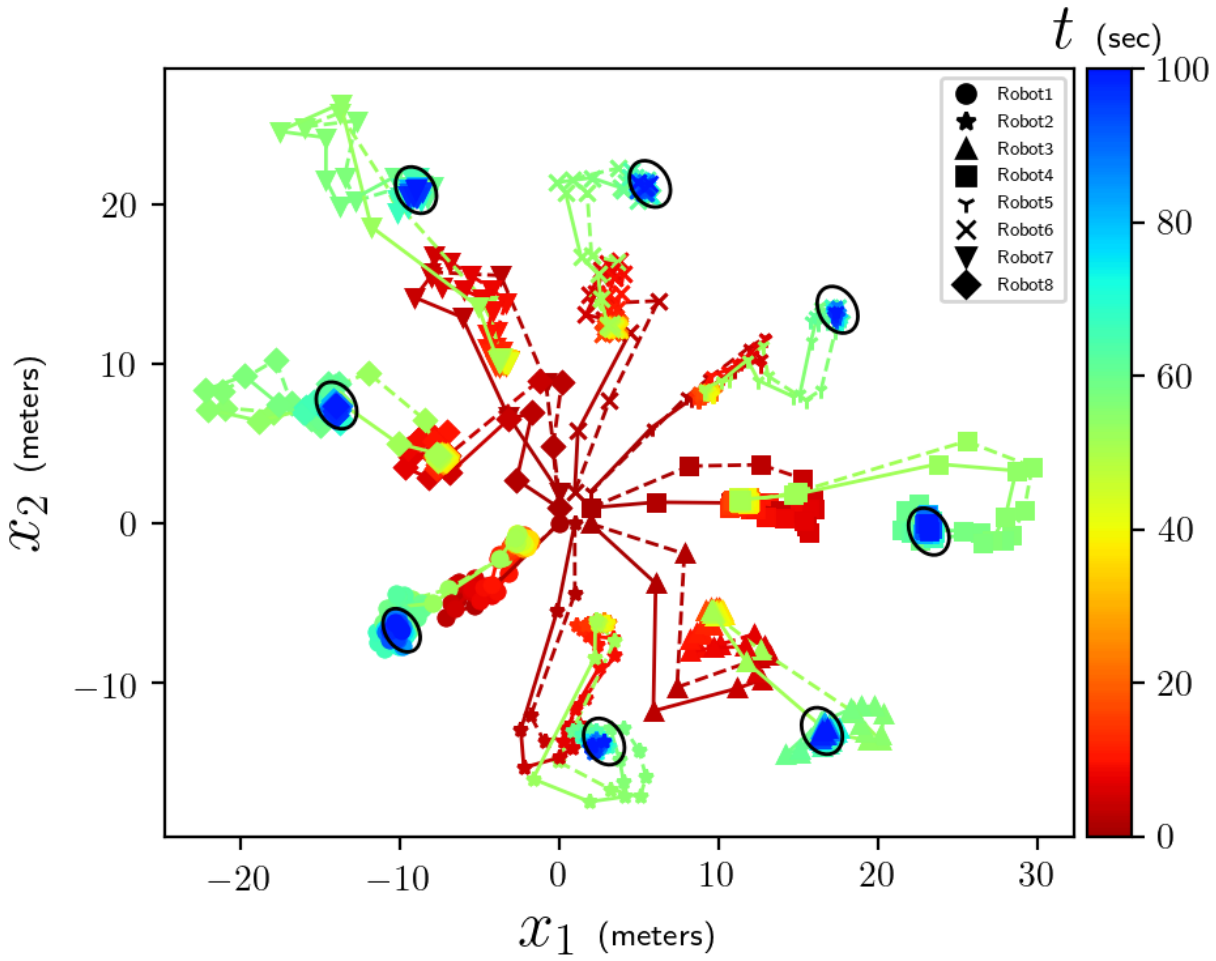

Figure 3.10: Simulated trajectories of robots and estimations provided by the proposed method in the two-dimensional space $\mathbf{x} = \{x_1, x_2\}$. The dashed lines are actual trajectories. The solid lines are trajectories estimated by the robots. The colors represent simulation time (brown is the oldest and blue is the latest.) The ellipses represent the covariance of their estimates at the last time step.

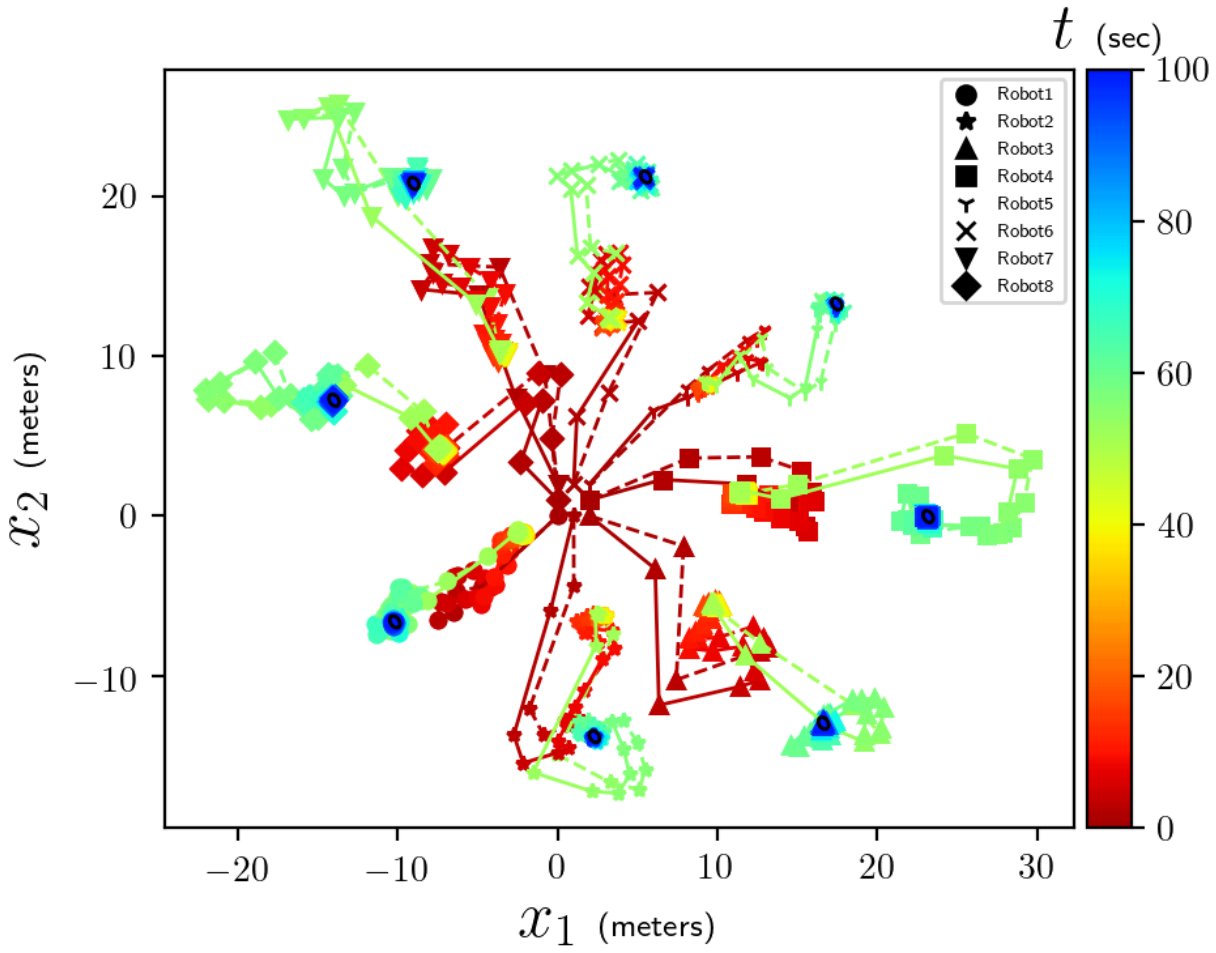

Figure 3.11: Simulated trajectories of robots and estimations provided by the centralized method in the two-dimensional space $\mathbf{x} = \{x_1, x_2\}$. The dashed lines are actual trajectories. The solid lines are trajectories estimated in a centralized manner. The colors represent simulation time (brown is the oldest and blue is the latest.)

Naive	CI-based	Proposed	Centralized
2.86	0.86	0.52	0.46

Table 3.1: Average errors in the estimation about the robots' location (x, y). They are calculated by accumulating errors and dividing them by the total time steps and the number of robots.

3.7.2 Simulation Results

Figures 3.8, 3.9, 3.10, and 3.11 show the simulated trajectories and those estimated provided by each method for comparison. The dashed lines are actual trajectories, and the solid lines are estimations. While randomized noises are applied to the robots' motion so they wriggle while moving, the same random seed for the noises was used so the robots follow the same routes in all cases. In the results, the naive method shows significant discrepancies from the true trajectories while estimations by the CI-based and the proposed methods trace the true trajectories more closely like that of the centralization case.

As a quantitative measure for comparison, an average error was calculated as follows.

$$E = \frac{1}{TN} \sum_{t=0}^{T} \sum_{k=1}^{N} \left\| \mathbf{x}_{k,t} - \boldsymbol{\mu}_{k,t} \right\| \tag{3.130}$$

where $\mathbf{x}_{k,t}$ and $\boldsymbol{\mu}_{k,t}$ are the actual and estimated locations of k-th robot at time t in the two-dimensional space, respectively. The average error of each method is shown in Table 3.1.

As visually observed in the figures, the naive method produces a significantly larger error than the other methods. It is also clear that the proposed method performs better than the CI-based method giving the smallest average error.

Figures 3.12, 3.13, 3.14, and 3.15 show the change of estimation error through the simulation for each robot. The sudden increases of errors right after 0 and 50 seconds are because the robots started moving at high velocities to reach an equilibrium. As we see in the figure, the errors of naive method converge to a specific value, which indicates overconfidence. The CI-based and proposed methods generates lower errors, which are closer to those by the centralized version. Although the CI-based methods shows significantly lower errors compared to the naive method, it is outperformed by the proposed method.

We have also compared the covariances estimated by each method throughout the simulation. For this comparison, a square root of the determinant of the covariance matrix was calculated. Figure 3.16 shows how this value is changing through the simulation for the 5th robot. For the naive method, it is almost always close to zero. Together with the fact that the error of naive method converges to a specific level, it clearly indicates overconfidence. The centralized version gives lower values that can be used as lower bounds of uncertainty as it has knowledge of correlations between robots while others do not. For both CI-based and the proposed methods this value is

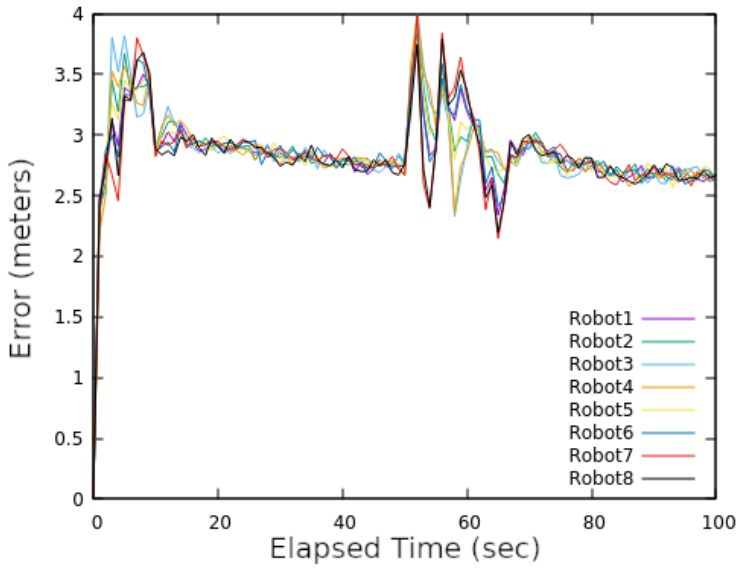

Figure 3.12: Localization errors of the naive method: err1∼8 represent errors of estimated locations of Robot1∼8, respectively. The horizontal axis is elapsed time and the vertical axis is the error of the robot's estimation from the actual location.

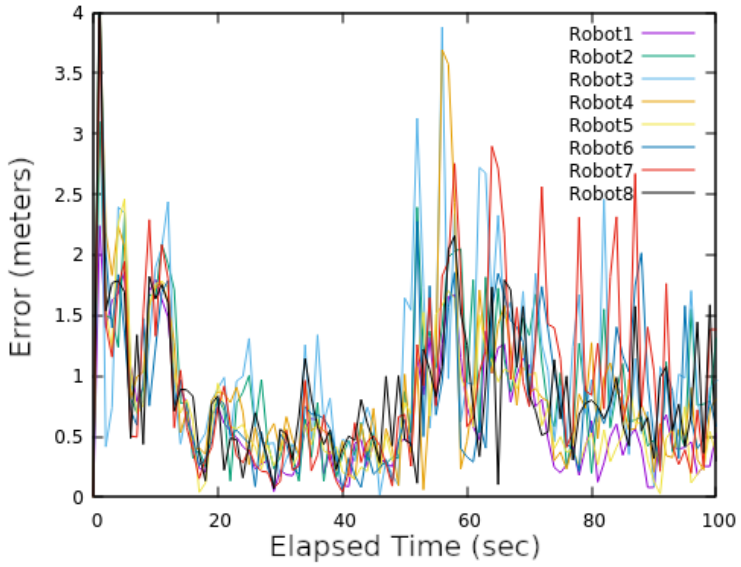

Figure 3.13: Localization errors of the CI-based method: err1∼8 represent errors of estimated locations of Robot1∼8, respectively. The horizontal axis is elapsed time and the vertical axis is the error of the robot's estimation from the actual location.

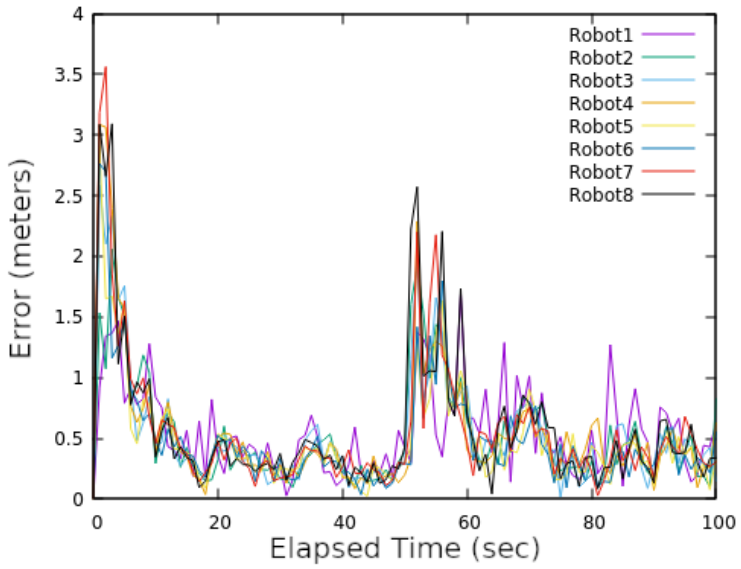

Figure 3.14: Localization errors of the proposed method: err1∼8 represent errors of estimated locations of Robot1∼8, respectively. The horizontal axis is elapsed time and the vertical axis is the error of the robot's estimation from the actual location.

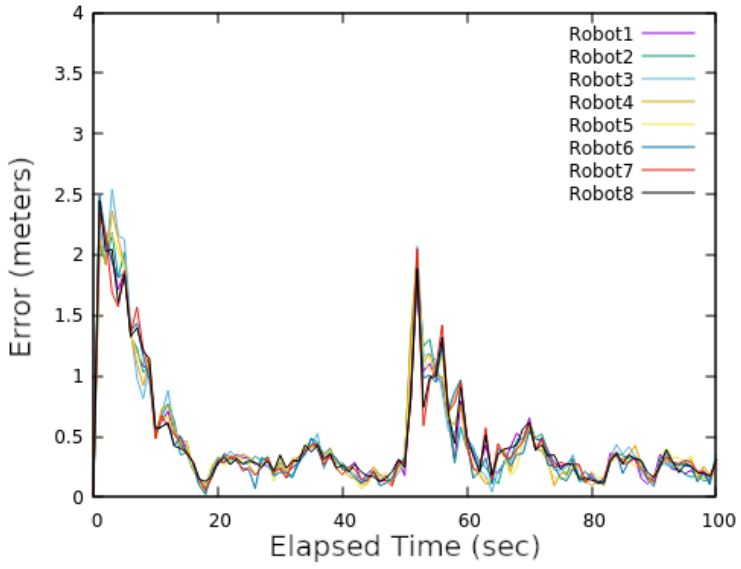

Figure 3.15: Localization errors of the centralized version: err1∼8 represent errors of estimated locations of Robot1∼8, respectively. The horizontal axis is elapsed time and the vertical axis is the error of the robot's estimation from the actual location.

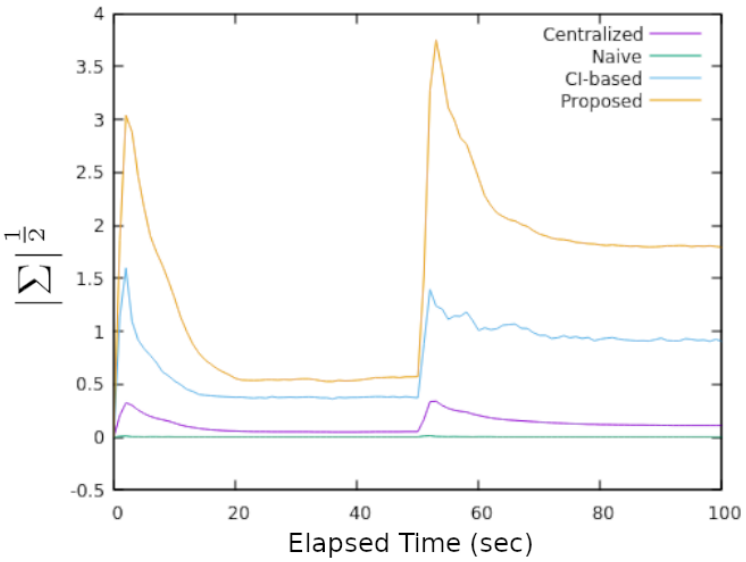

Figure 3.16: $|\Sigma|^{\frac{1}{2}}$ of Robot5. The square root of determinant of the covariance matrix (vertical axis) is displayed during the simulation time from 0 to 100 seconds (horizontal axis) to roughly show how large the covariance matrix is. The values of the naive method are so small that the graph is hardly visible, indicating the distribution has a sharp peak. The centralized version gives a lower bound of uncertainty as it has knowledge about correlations between robots. The CI-based method generates the middle line and the proposed method generates the larger values.

larger throughout the simulation and sharply increases when the robot starts to move to the next equilibrium. Then it is gradually reduced to a specific level. This phenomenon appears to be more significant for the proposed method than the CI-based one, indicating uncertainty is handled properly.

Thus, the simulation results indicate that the proposed method outperforms the other methods without knowledge about correlation between robots.

CHAPTER 4
CONSERVATIVE DATA EXCHANGE BY NON-PARAMETRIC FILTER

The previous chapter provided the general probabilistic models of exchanging information among multiple robots. In the simulation, the probability distributions were supposed to follow a normal distribution, with a mean and a covariance matrix. Like this parametric method, a nonparametric method such as sampling particles based on a given distribution can also be used for the proposed method. For 3D SLAM with occupancy mapping, the robot needs to estimate the occupancy parameter for each grid in the map. If the map size is n by n by n grids, the total number of parameters for the map would be n^3. If a parametric method such as Kalman filter was applied to this situation, it would need a tremendously large matrix which is infeasible to process. On the other hand, a nonparameteric method just samples possible maps based on the acquired information, and it does not need to have such a huge matrix. Thus, the nonparametic method is especially usable due to its complexity of the mapping.

However, it is challenging to perform the conservative data exchange by a particle filter. A particle filter uses a set of sampled particles to represent a probability distribution, but when a distribution is modified by a fractional exponent, the particles no longer represent the distribution. It will need additional information and process so that the particles can represent the modified distribution. After reviewing the basics of particle filter, this chapter introduces a method to

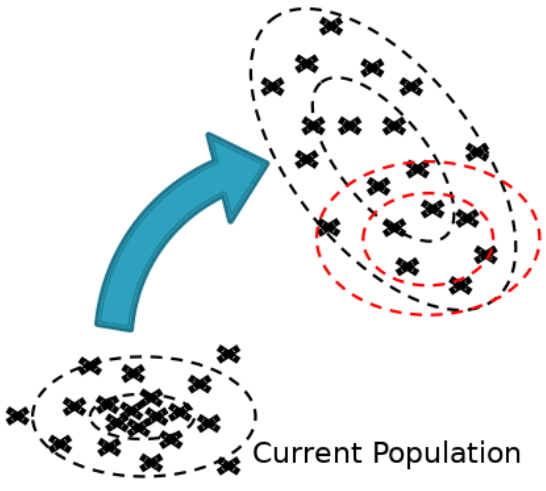

Figure 4.1: Particle filter expresses a probability distribution by density of particles. A motion model predicts the next state by moving the current population, leading to a lower density. A sensor model resamples particles whose states do not contradict with the gained sensory data, leading to the sharper density (red).

represent the modified distribution by using the particles with resampling weights. In addition, the nonparameteric version of the proposed method is demonstrated in a simulation, comparing with the naive method.

4.1 Review of Particle Filter for Occupancy Map-Based SLAM

A particle filter is a nonparametric Bayes filter, which statistically takes samples from a state space [64, 9]. Unlike a Kalman filter or any other parametric filters, the probability distribution is represented by a population of sampled particles each of which represents a specific state.

Figure 4.1 depicts particles representing a probability distribution to estimate. The particles are processed through the three steps to update the probability distribution: prediction, evaluation, and resampling.

The prediction step samples a possible state or location by the motion model, which is a probabilistic model of the robot's motion based on control inputs u and the previous state $x_{t-1}^{[i]}$.

$$\hat{x}_t^{[i]} \sim p(x|u, x_{t-1}^{[i]}) \tag{4.1}$$

Then, in the evaluation step, the particles are evaluated based on acquired sensory data. If a particle explains well according to the data, a higher weight is given to the particle. Otherwise, the particle will get a lower weight. Let's say $f(x)$ is a target distribution, which the new population is supposed to represent, and $g(x)$ is a proposal distribution, which is represented by the current population. A resampling weight $w_t^{[i]}$ for particle i is calculated as follows.

$$w_t^{[i]} = \frac{f(\hat{x}_t^{[i]})}{g(\hat{x}_t^{[i]})} \tag{4.2}$$

In the resampling step, particles are sampled from the evaluated population so as to form a new population. A particle is chosen with a probability that is proportional to its weight $w_t^{[i]}$. If the values of the target and proposal distributions are equal at a particle $\hat{x}_t^{[i]}$, the weight $w_t^{[i]}$ will be 1. If the value of the target distribution is higher than that of the proposal distribution, the weight will be higher so that the particle would be chosen at a higher chance. Otherwise, the weight will be lower so that the particle would be less likely to be selected. Therefore, those particles with higher weights get a higher chance to survive, leading to the resulting population representing the target distribution.

4.2 Confidence Reduction for Particles

As an additional part of the process of the particle filter, conservative data exchange reduces the confidence in the estimation by applying additional sampling weights so that the resulting particles can represent a probability distribution with lower confidence. Let's say a robot has a set of particles

X_1 representing $p(x)$, and let the other robot's distribution be $q(x)$ and the particles be X_2. Then, each iteration taken by the first robot is composed of the following steps.

1. Calculate weights W_1 and W_1' to represent $p^{\omega}(x)$ and $p^{1-\omega}(x)$ respectively (confidence reduction)
2. Data exchange: Send the particles and weights $[X_1, W_1']$ representing $p^{1-\omega}(x)$ to the other robot & receive the particles and weights $[X_2, W_2']$ representing $q^{1-\omega}(x)$ from the other robot
3. Calculate weights W_1'' based on $[X_2, W_2']$ and sensory data on relative pose of a pair of robots
4. Calculate weights W_1''' by multiplying W_1 and W_1''
5. Resample the particles X_1 based on the combined weights W_1'''

Step 4 simply multiplies the two types of weights for each particle and step 5 follows the same process of the conventional particle filter. On the other hand, steps 1 through 3 require special processes. The rest of this section will mathematically describe the first three steps.

4.2.1 Weights for Confidence Reduction

For the first step: confidence reduction, a weight for each particle is calculated based on the concept of particle filter. Weights are calculated so that the target distribution appears as ηp^{ω}.

$$w(x) = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta p^{\omega}(x)}{p(x)} = \eta p^{\omega-1}(x) \tag{4.3}$$

where x is a state the particle represents. Note that the normalization factor η can be omitted for actual calculation since it does not affect the resampling process.

The weights for $\eta p^{1-\omega}$ is calculated in the same way.

$$w'(x) = \frac{\eta p^{1-\omega}(x)}{p(x)} = \eta \frac{1}{p^{\omega}(x)} \tag{4.4}$$

Since the distribution is represented by particles, $p(x)$ needs to be estimated by the kernel density estimation (KDE). By KDE, the value of p at x is calculated as

$$p(x) = \eta \sum_{i \in N} k(x - x_i) = \eta \sum_{i \in N} e^{-\frac{||x - x_i||^2}{2\sigma^2}} \tag{4.5}$$

where η is a normalization factor (and this can also be omitted for resampling). The kernel window size σ affects estimation of the distribution and it is manually set at the moment.

4.2.2 Data Exchange

At step 2, the particles and the weights are sent as $p^{1-\omega}$ to the other robot. This approach can skip an additional resampling step which generates particles representing $p^{1-\omega}$. Let's say X_1 is samples representing p about the robot and X_2 is samples representing q about the other robot. The first robot calculates weights W_1 and W_1' so that $[X_1, W_1]$ represents p_1^{ω} and $[X_1, W_1']$ represents $p^{1-\omega}$. Similarly, the second robot calculates W_2 about q^{ω} and W_2' about $q^{1-\omega}$. Then, the robots exchange their particles and weights representing $p^{1-\omega}$ and $q^{1-\omega}$. Finally, they will have these items:

- The first robot
  - $[X_1, W_1]$ (representing p^{ω})
  - $[X_2, W_2']$ (representing $q^{1-\omega}$)
- The second robot
  - $[X_2, W_2]$ (representing q^{ω})
  - $[X_1, W_1']$ (representing $p^{1-\omega}$)

4.2.3 Weights Based on Sensory Data

Each robot needs to evaluate its own particles based on the other's particle and the sensory data of relative pose. For example, the first robot calculates weights W_1'' for particles in X_1 based on $[X_2, W_2']$ and sensory data on relative pose so that $[X_1, W_1'']$ can represent the distribution about the first robot estimated by $q^{1-\omega}(x)$ and the sensor model. For each particle $x_1^{[i]}$ in X_1, a weight $w_1''^{[i]}$ in W_1'' is calculated as follows. First, a particle $x_2^{[j]}$ is randomly selected from X_2 at a probability proportional to W_2'. Then, sensory data is traced with respect to the selected particle, resulting in a particle $x_1'^{[j]}$ about the first robot sampled by the second robot's information and sensory data. This particle is compared with each particle $x_1^{[i]}$ in X_1, and a value is calculated by a normal distribution representing the sensor model; if $x_1'^{[j]}$ and $x_1^{[i]}$ are close to each other, a higher value is given, and the value will be otherwise smaller. This process is repeated for specified times. The weight $w_1''^{[i]}$ is calculated by summing the values generated for $x_1^{[i]}$ in this comparison process.

$$w_1''^{[i]} = \sum_j e^{-\frac{\left(x_1^{[i]} - x_1'^{[j]}\right)^2}{2\sigma_{sensor}^2}} \tag{4.6}$$

This process is also applied to the second robot. Finally, this set of weights is used for the rest of the steps. Step 4 calculates the weight $w'''^{[i]}$ in W''' for the resampling in step 5 by multiplying the weights for p^{ω} and those calculated on step 3.

$$w'''^{[i]} = w^{[i]} w''^{[i]} \tag{4.7}$$

4.3 Sub-Sampling for Lossy Data Compression

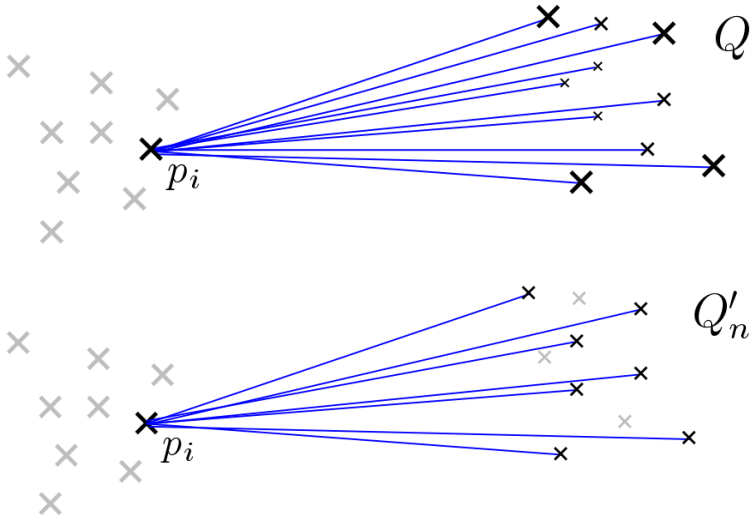

Figure 4.2: Evaluation of particle p_i in a robot's estimate based on the other robot's estimate by using the true population Q (Top) and the sampled population Q'_n (Bottom). A particle in Q has an importance weight (represented by the size of the symbol). Q'_n is a set of n particles sampled from Q with replacement by weighted sampling.

Cooperative localization requires a pair of robots to exchange their particles with weights representing their estimates whose uncertainty is increased by a fractional exponent. Then, the robot evaluates each particle against those given by the other robot. If the communication bandwidth is limited, the size of the particles to pass between robots is problematic. Also evaluation of a large number of particles may be computationally expensive. Here, the possibility to reduce the number of particles is discussed by approximating the original estimate from the other robot.

When the robots exchange data, each particle has a weight for weighted sampling to represent the probability distribution with a fractional exponent. Let a particle of the robot for evaluation be p_i and the particles given from the other robot be $q_j \in Q$, where Q is the other robot's population, as shown in the top image of Figure 4.2. Each particle q_j has a weight u_j for weighted sampling so that, when a particle q is randomly selected from Q, the probability for q to be q_j is expressed as follows.

$$P(q = q_j) = \frac{u_j}{\sum_k u_k} \tag{4.8}$$

The weight for the particle p_i is calculated by the sensor model $eval(*, *, *)$ given sensory measurement r on the relative pose. If all the particles from Q are used instead of weighted sampling, the value is expressed by the summation of the evaluation weighted by u_j.

$$w_i = \eta \sum_j u_j eval(p_i, r, q_j) \tag{4.9}$$

where η is a normalization factor. As it is based on the sum of the evaluation values, this weight can be represented by the expected value of *eval* about Q.

$$\begin{aligned} w_i &= \eta \sum_j u_j eval(p_i, r, q_j) \\ &= \eta \left(\sum_k u_k \right) \sum_j \frac{u_j}{\sum_k u_k} eval(p_i, r, q_j) \\ &= \eta \left(\sum_k u_k \right) \sum_j P(q = q_j) eval(p_i, r, q_j) \\ &= \eta' E[eval(p_i, r, q)] \end{aligned} \tag{4.10}$$

Note the *eval* function is nonlinear, indicating the following inequality.

$$E[eval(p_i, r, q)] \neq eval(p_i, r, E[q]) \tag{4.11}$$

Hence, particle p_i cannot be simply evaluated by the expected state from Q.

Next, consider evaluation of p_i against the particles which are randomly sampled from Q. Let q' be a particle selected from Q by weighted sampling with replacement. The probability for particle q_j to be selected is proportional to its weight u_j.

$$P(q' = q_j) = \frac{u_j}{\sum_k u_k} \tag{4.12}$$

Let Q'_n be a set of the n particles sampled in such a way so that Q'_n approximates Q, as shown in the bottom image of Figure 4.2. The expected value of *eval* by the samples of Q'_n can be calculated by the accumulated *eval* values divided by n

$$E[eval(p_i, r, q')] = \frac{\sum_{j=1}^{n} eval(p_i, r, q'_j)}{|Q'_n|} = \frac{\sum_{j=1}^{n} eval(p_i, r, q'_j)}{n} \tag{4.13}$$

where $q'_j \in Q'_n$. When this value is close enough to the expected value of *eval* based on Q, the sampled particles Q'_n approximates Q well. Consider the following value to represent the difference between the expected values.

$$d_n = E\left[\left(E[eval(p_i, r, q')] - E[eval(p_i, r, q)] \right)^2 \right] \tag{4.14}$$

Q'_n generates a close expected value of *eval* to the expected value of *eval* on Q when the value d_n

approaches 0. This value d_n is extended as follows.

$$
\begin{aligned}
d_n =& E\left[\left(E[eval(p_i, r, q')] - E[eval(p_i, r, q)]\right)^2\right] \\
=& E\left[E[eval(p_i, r, q')]^2 - 2E[eval(p_i, r, q')]E[eval(p_i, r, q)] + E[eval(p_i, r, q)]^2\right] \\
=& E\left[\left(\frac{\sum_j^n eval(p_i, r, q'_j)}{n}\right)^2 - 2\frac{\sum_j^n eval(p_i, r, q'_j)}{n}E[eval(p_i, r, q)] + E[eval(p_i, r, q)]^2\right] \\
=& E\left[\left(\frac{\sum_j^n eval(p_i, r, q'_j)}{n}\right)^2\right] - 2E\left[\frac{\sum_j^n eval(p_i, r, q'_j)}{n}\right]E[eval(p_i, r, q)] + E[eval(p_i, r, q)]^2 \\
=& \frac{\sum_j^n E\left[eval(p_i, r, q'_j)^2\right] + 2\sum_{k<l} E\left[eval(p_i, r, q'_k)\right] E\left[eval(p_i, r, q'_l)\right]}{n^2} \\
& - 2\frac{\sum_j^n E\left[eval(p_i, r, q'_j)\right]}{n}E[eval(p_i, r, q)] + E[eval(p_i, r, q)]^2
\end{aligned}
\tag{4.15}
$$

As q' is selected in the same way as q is from Q, the terms with q' can be expressed by using q in Q and n.

$$
\begin{aligned}
d_n =& \frac{nE\left[eval(p_i, r, q)^2\right] + (n^2 - n)E\left[eval(p_i, r, q)\right]^2}{n^2} \\
& - 2\frac{nE\left[eval(p_i, r, q)\right]}{n}E[eval(p_i, r, q)] + E[eval(p_i, r, q)]^2 \\
=& \frac{E\left[eval(p_i, r, q)^2\right] + (n-1)E\left[eval(p_i, r, q)\right]^2}{n} - E[eval(p_i, r, q)]^2 \\
=& \frac{E\left[eval(p_i, r, q)^2\right] - E[eval(p_i, r, q)]^2}{n}
\end{aligned}
\tag{4.16}
$$

The numerator is the variance of $eval(p_i, r, q)$, which is independent on the random sampling for Q'_n. This shows that d_n continuously decreases as n grows. In other words, d_n is reduced every time a particle is sampled from Q to comprise Q'_n. If the random sampling from Q is repeated infinitely, it eventually converges to 0.

$$
\lim_{n\to\infty} d_n \to 0 \tag{4.17}
$$

n is determined so that d_n can be small enough to approximate Q, depending on the application area. When n is set to a smaller number than $|Q|$, the random sampling eventually performs lossy compression on the data of Q because this can reduce the data size to pass to the other robot. To find the optimal n, the variance of $eval(p_i, r, q)$ would need to be calculated. However, this would also require to pass and use all particles of Q, which contradicts the data compression purpose. Thus, it is infeasible to optimize n without sending all the particles. Therefore, here, n is empirically

set to a constant value instead of dynamically choosing the optimal n every time the robot sends the data.

4.4 Simulation

The two-dimensional simulation of the conservative data exchange was presented in the previous chapter, by using a parametric filter. All the probability distributions were assumed to follow a normal distribution and all the methods were performed by using Extended Kalman Filter (EKF).

Here, it is simulated and demonstrated by using a nonparametric filter, i.e., a particle filter. Additionally, the state of a robot in the simulation is extended from two parameters (x and y) to three parameters (x, y, and the orientation).

4.4.1 Simulation Setups

Like the simulation in the previous chapter, eight robots are initially deployed in a square near the origin, and their motions are determined based on their relative locations and they move away from each other while maintaining the distance to their neighboring robots. Unlike the previous simulation, each robot has an additional parameter to estimate: orientation. The motion control is applied to the velocity on the forward direction and rotational rate. Based on these settings, the robots separately perform localization to estimate their 2D locations and orientations. Each pair of robots take measurement on their relative poses and update their estimated locations. Additionally, the first robot can also perform global localization by taking a measurement on its location with respect to the global reference frame.

In this section, the naive and proposed methods are demonstrated and compared. While the proposed method applies the confidence reduction to the probability distribution, the naive method simply skips this step so that overconfidence is reproduced.

The CI-based method and the centralized method are not considered here due to the difficulty of implementation using a particle filter. The CI-based method basically assumes that the probability distribution follows a normal distribution, which is not necessarily true when the distribution is represented by particles. Especially, when one of the parameters to estimate is about the rotation of the robot and the relative pose measurement is provided in a polar coordinate system, the probability distribution represented by the particles will no longer follow a normal distribution. Even if the CI-based method directly processes particles without calculating covariance matrices, there is a computational issue. With the given sets of particles, it would need to optimize the fractional exponent to minimize the entropy of the resulting probability distribution. As the probability distributions are represented by particles, this optimization process would not be able to be done analytically.

The centralized method has another reason which makes it hard to use a particle filter. Since

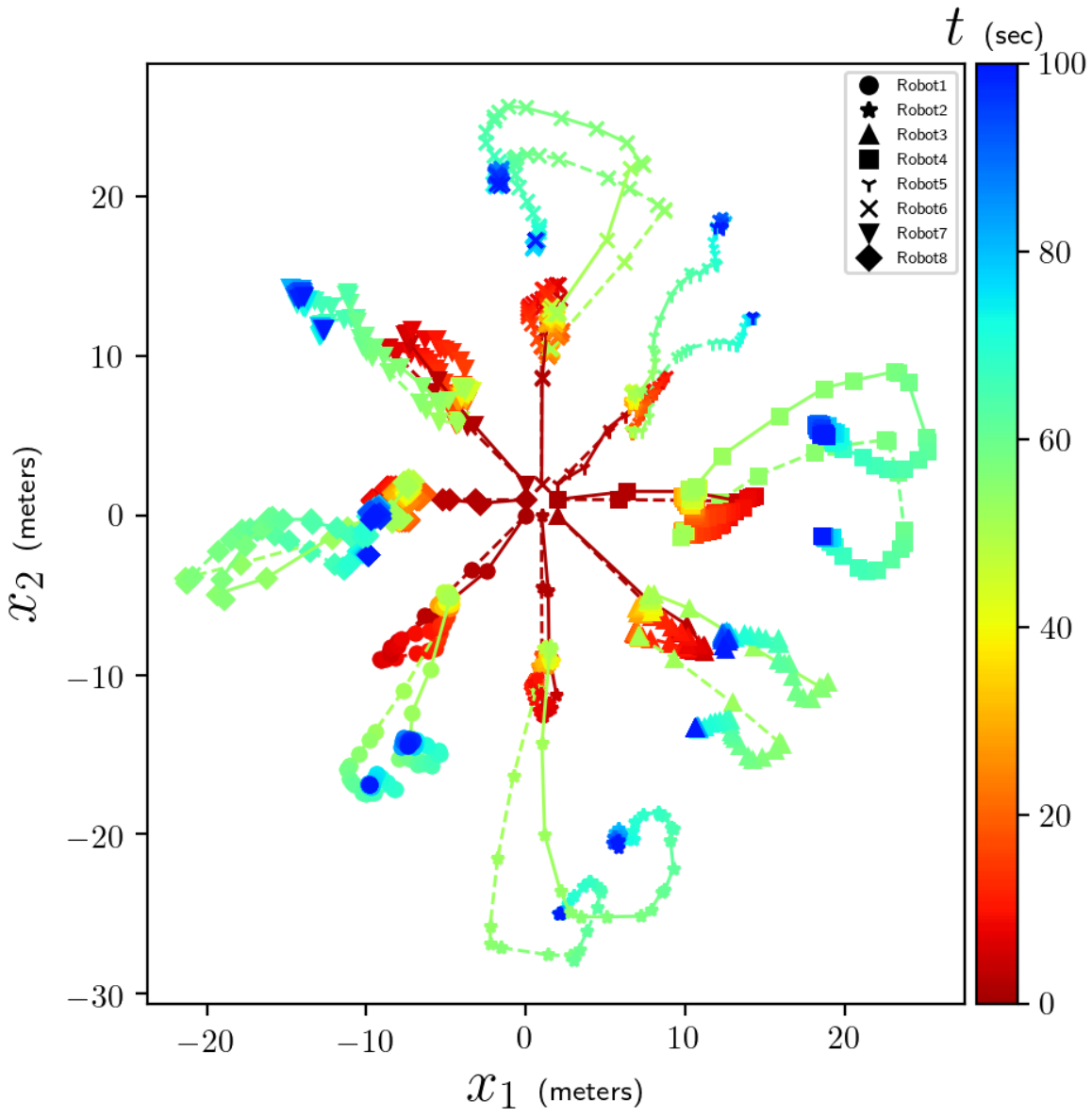

Figure 4.3: Trajectories of the eight robots estimated by the naive method. The dashed lines are the ground truth trajectories, and the thick lines are their estimation. The colors indicate the simulation time ranging from 0 second to 100 seconds.

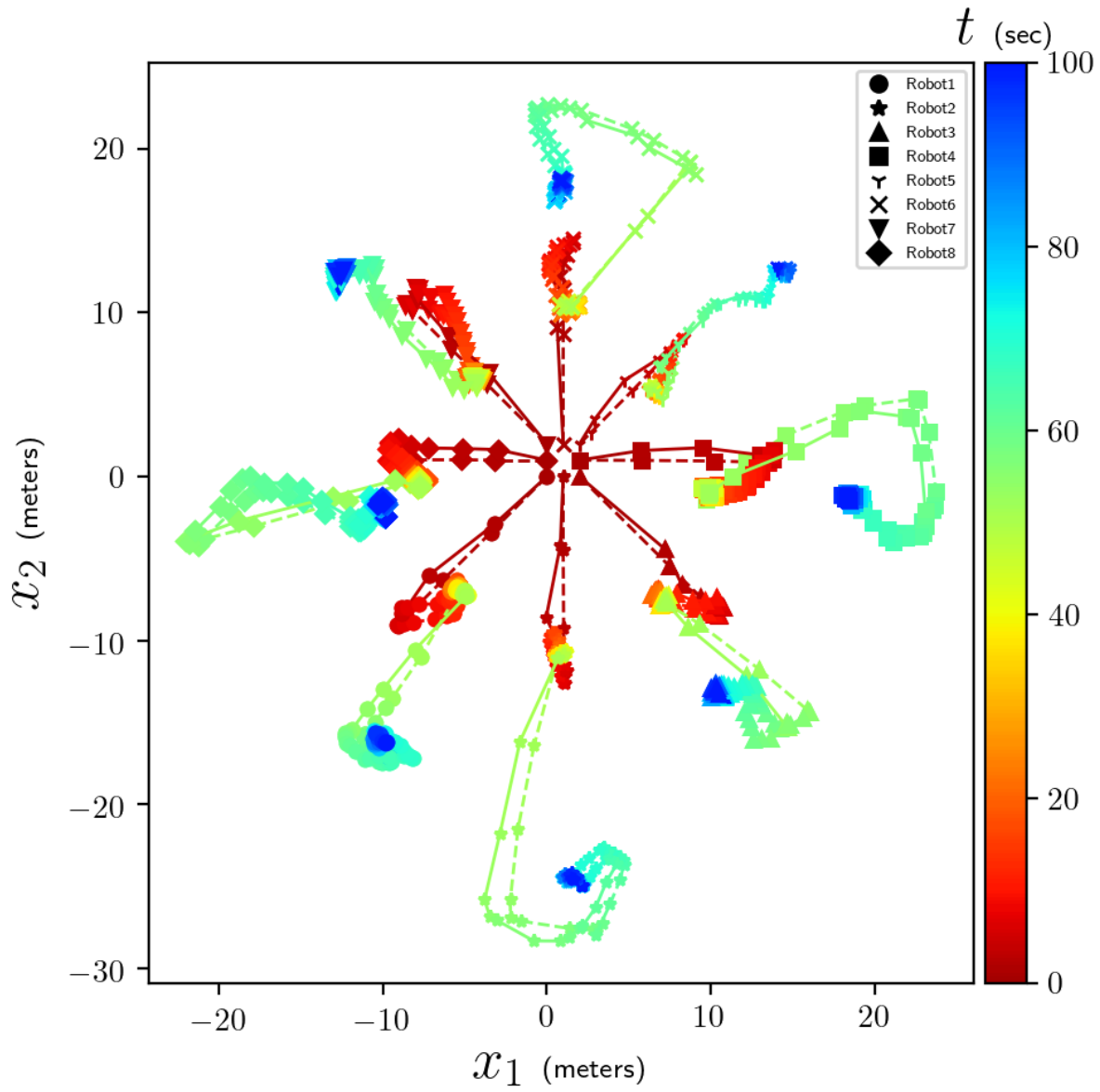

Figure 4.4: Trajectories of the eight robots estimated by the proposed method. The dashed lines are the ground truth trajectories, and the thick lines are their estimation. The colors indicate the simulation time ranging from 0 second to 100 seconds.

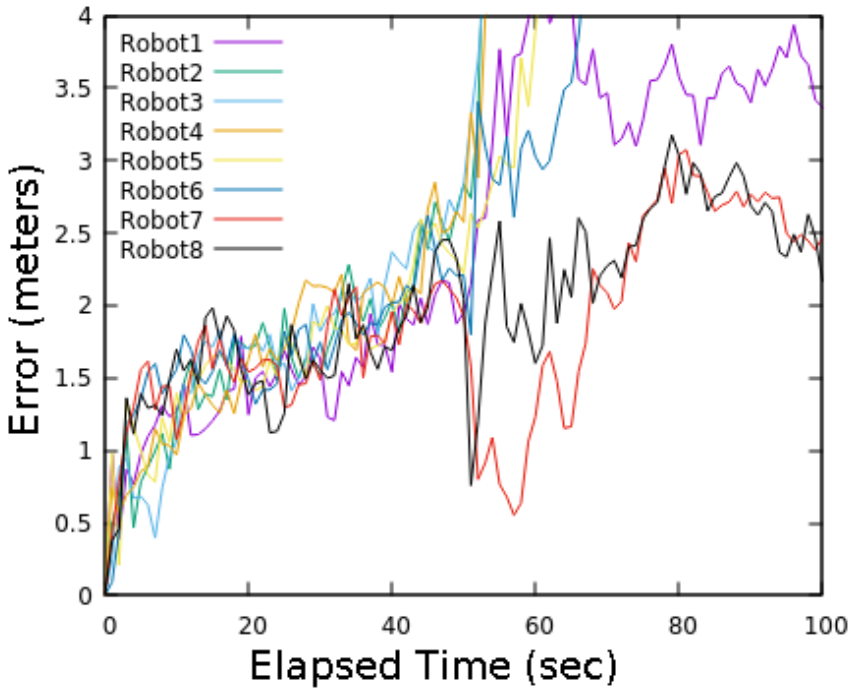

Figure 4.5: Errors in the estimated locations of the eight robots by the naive method.

it considers the correlations between the robots, it samples the state of the entire system. That is, each particle would have the locations of all the robots. This requires a large number of particles to represent the distribution in such a high-dimensional space. When each robot holds N particles for the particle filter individually estimating the robot's state, the number of particles required for the centralized version would be N to the power of M where M is the number of the robots in the system. Obviously, this number would grow exponentially as the number of robots increases.

4.4.2 Simulation Results

Figures 4.3 and 4.4 show the robot's trajectories based on the naive and proposed methods, respectively. The color of the line indicates the simulation time; the red lines are older than the blue ones. The thick lines are their estimated trajectories and the dashed lines are the ground truth. The robots reached the first equilibrium at $t = 50$, and stayed in a circle with a radius of 10 meters. Then, as the motion setting was updated, they moved further away. While the naive method generates estimated trajectories far away from the ground truth, the proposed method generates more accurate trajectories.

The errors in the estimation of 2D location were calculated and compared. Figures 4.5 and 4.6 show Euclidean distances between the ground truth and the estimated locations, i.e., the errors in the estimation given by the naive and proposed methods, respectively. The naive method generated

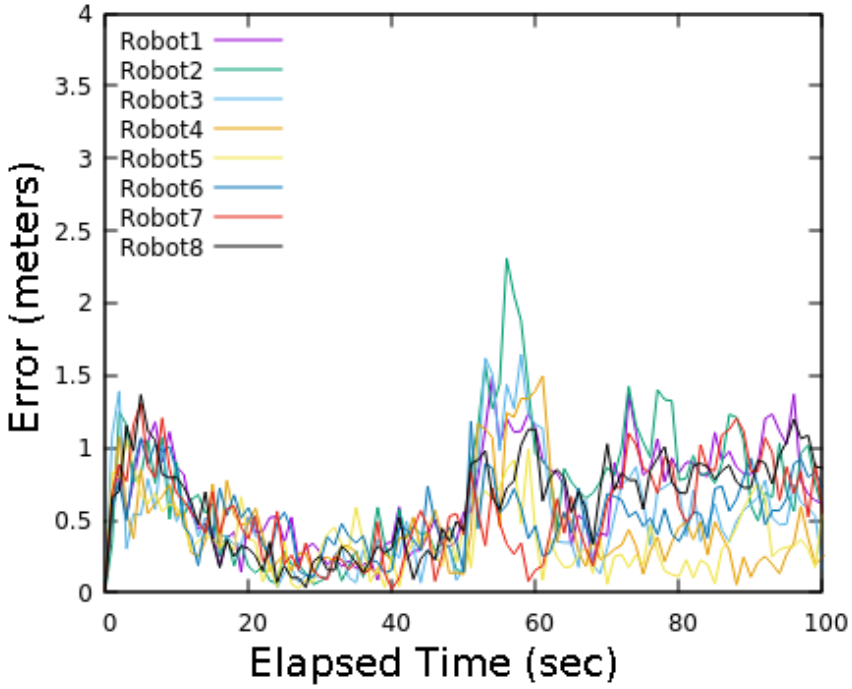

Figure 4.6: Errors in the estimated locations of the eight robots by the proposed method.

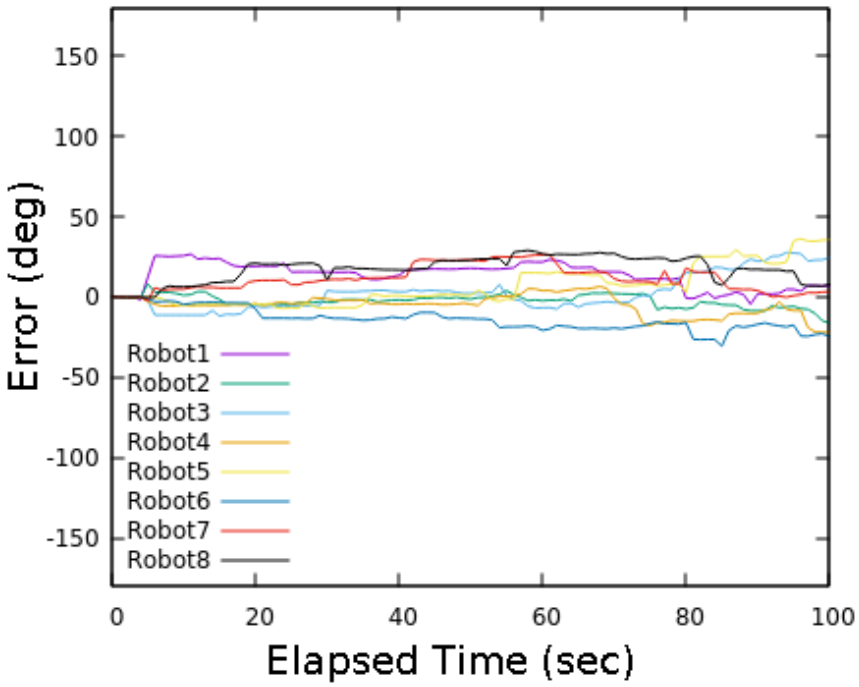

Figure 4.7: Errors in the estimated orientations of the eight robots by the naive method.

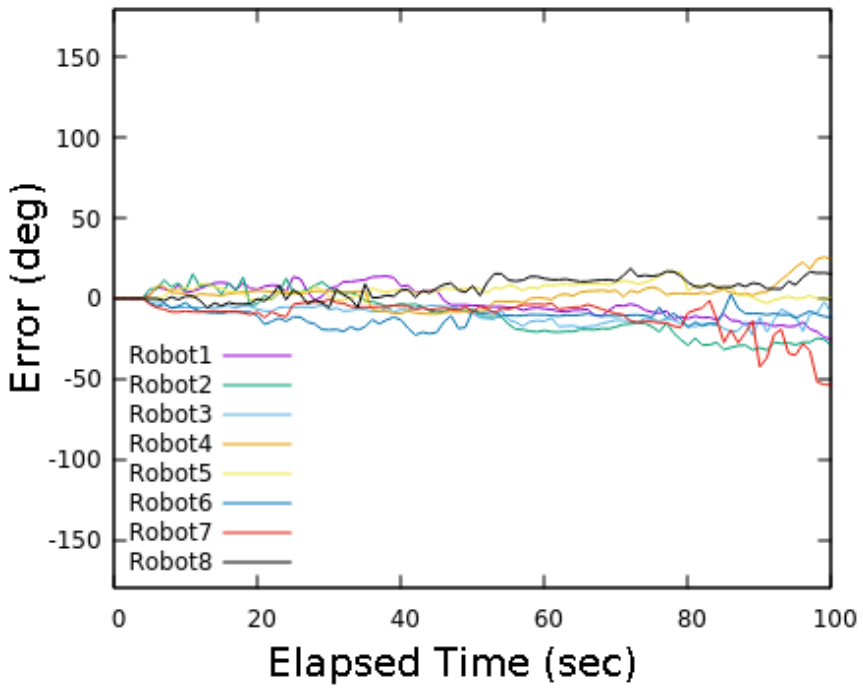

Figure 4.8: Errors in the estimated orientations of the eight robots by the proposed method.

	Location Error (meters)			Orientation Error (deg)		
	Overall	First	Second	Overall	First	Second
Naive	3.01	1.64	4.38	10.61	8.13	13.10
Proposed	0.56	0.42	0.70	8.99	5.91	12.07

Table 4.1: Average errors in the estimation about the robots' location (x, y) and orientation over the entire time (Overall), the first half (First), and the second half (Second) of the simulation. They are calculated by accumulating errors caused by the particles and dividing it by the number of particles, the total time steps, and the number of robots.

larger errors and these errors tended to increase, indicating it is causing overconfidence. The errors are increased significantly during the second half of the simulation as they moved at a higher speed. On the other hand, the proposed method generated smaller errors. The errors increased when they started at the beginning and decreased until the simulation time reached 50 seconds. During the second half of the simulation, the errors did not drastically increase like the naive method.

Additionally, the errors in the orientation estimation were also compared. Figures 4.7 and 4.8 show the errors in their estimation for orientation in degrees. The naive method generated significant amounts of errors in the earlier time and these errors stay large throughout the simulation. The proposed method gave slightly better results. They were kept relatively low in the first 50 seconds while they drastically increased during the second half of the simulation. As the motion noise

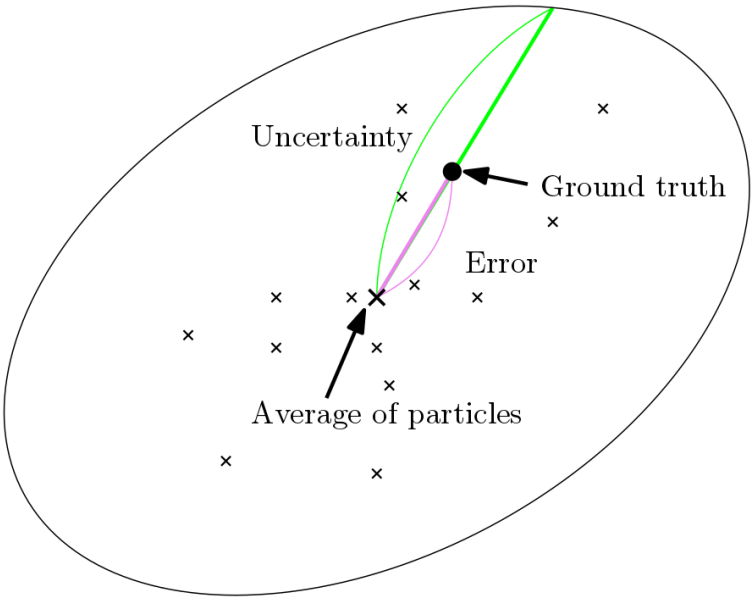

Figure 4.9: Error and uncertainty of the estimation by particles. The error is represented by the difference between the ground truth location and the average of the particle locations. The uncertainty is represented by the length of the line that starts from the average location, passes the ground truth, and ends at the edge of the ellipse representing the 95% confidence of the covariance.

to be added in the orientation is set to be proportional to the robot's velocity in the simulation, the increase of the errors in the second half indicates that particle depletion might have happened and there may be room for adjustment of parameters to improve the performance such as the number of particles, the highest speed the robots can take, and so on.

Table 4.1 shows average errors in the estimated locations and orientations over the entire simulation time, the first half and the second half of the simulation. The average error caused by the proposed method is significantly smaller compared to the naive method.

The uncertainty of the location estimation was also evaluated. First, the covariance of the particle locations is calculated. Figure 4.9 depicts the ellipse that represents the 95% confidence of the covariance, in addition to the average of the particle locations and the ground truth location. From the average location to the edge of the ellipse, there is a line passing through the ground truth location. The length of this line is calculated and used as a metric of the uncertainty. When the uncertainty is larger than the error, the ground truth falls inside the ellipse, indicating that the estimation is plausible. Otherwise, the ground truth will fall outside of the ellipse, which indicates that the estimation is not plausible. Figures 4.10 and 4.11 show the results for each robot. The naive method generated smaller uncertainty than the proposed method did while the errors were actually growing, indicating its estimation is inconsistent. On the other hand, the proposed method gave significantly better results. At the end of the first half of simulation, the ground truth locations fell within their covariance ellipses with 95 % confidence. After that, the robots increased their

speeds and the errors sometimes exceeded the ellipses possibly because their higher motion speeds affected the estimation quality.

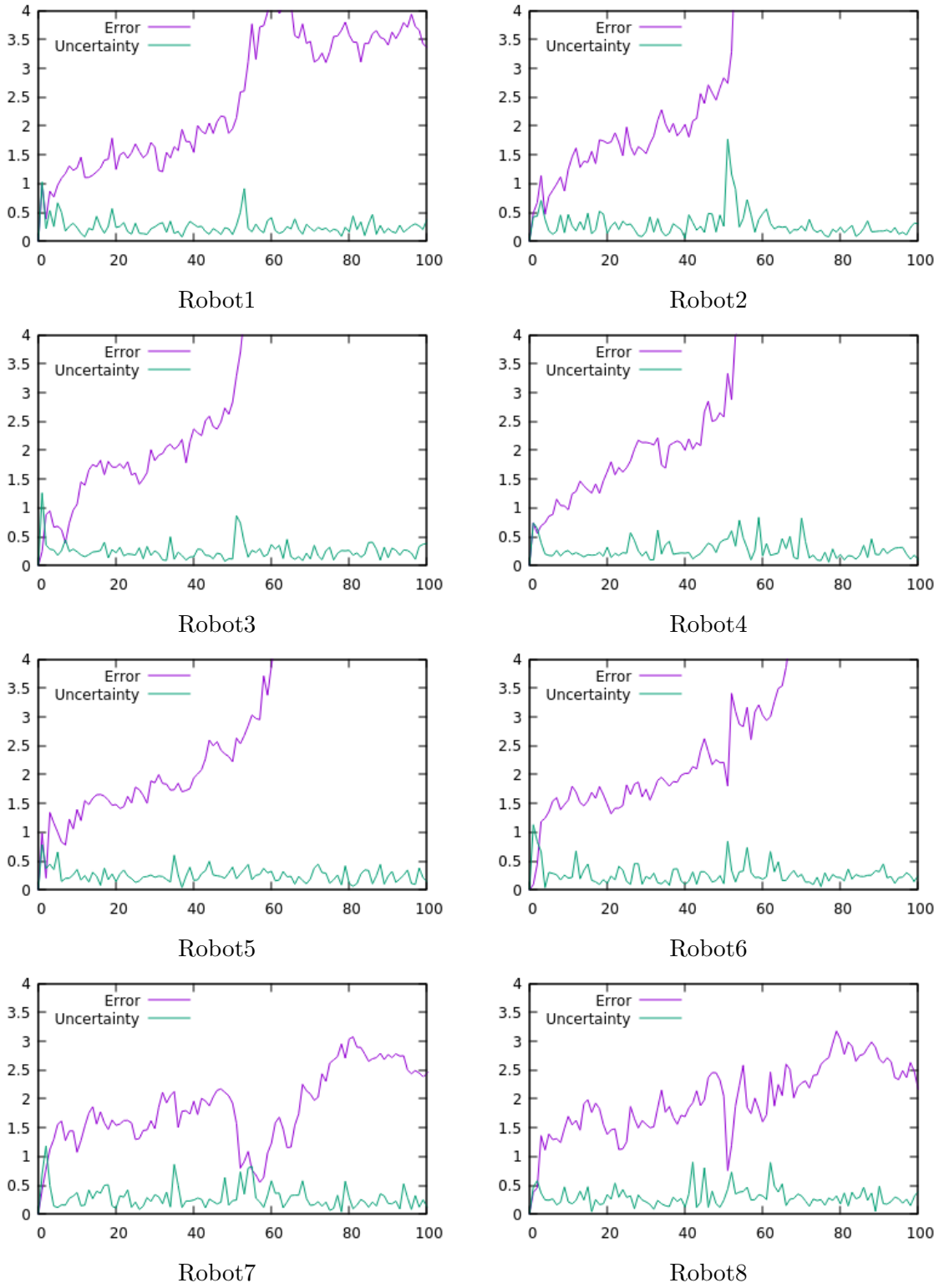

Figure 4.10: Error and uncertainty in the estimated locations by the naive method. The vertical axis is the error and uncertainty in meters. The horizontal axis is the elapsed time in seconds.

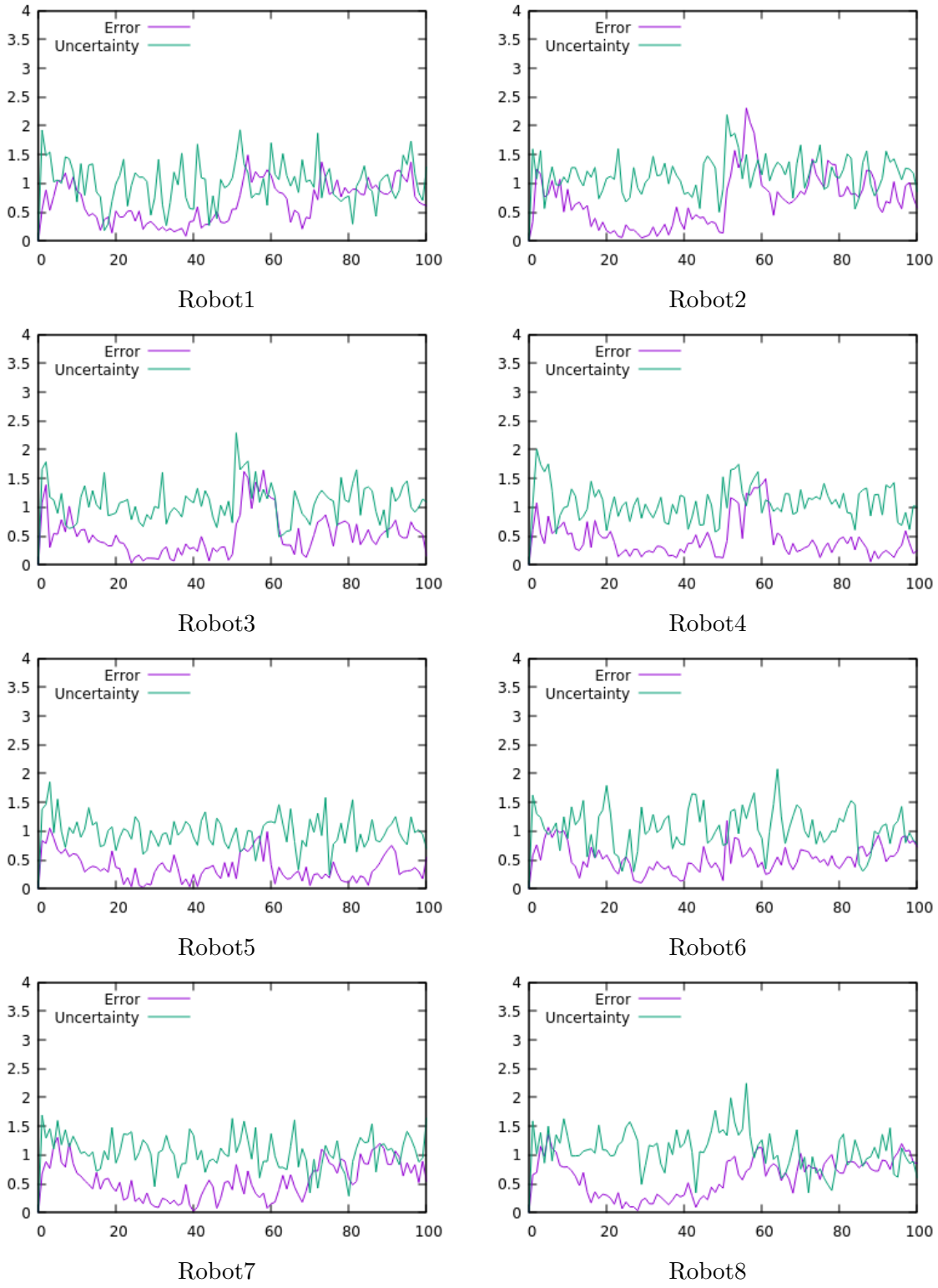

Figure 4.11: Error and uncertainty in the estimated locations by the proposed method. The vertical axis is the error and uncertainty in meters. The horizontal axis is the elapsed time in seconds.

CHAPTER 5
DISTRIBUTED SUBMAP BUILDING

The mapping task is decomposed into submap building tasks that are distributed among the robots. Each robot performs SLAM in the local area where it is situated. This task can be executed locally and done by conventional approaches.

In a multi-robot system, the robots can transfer their submaps to their neighbors who are entering the corresponding areas. If the neighboring robot uses the received submap, it will not need to build a submap of the corresponding area from scratch and the whole system can efficiently build the map of the environment. In addition, as the robots can hold fewer submaps, they can save the memory space for the mapping tasks.

After reviewing the related work, this chapter will describe the mapping process performed by each robot and will discuss the submap transfer between robots. The presented techniques and methods set the stage for implementation in the robotic systems and simulations in the next chapter.

5.1 Related Work

The mapping task is to express the environment based on the measurements acquired from the robot's sensors, generally using either feature-based or location-based mapping formats [64]. The feature-based map is a list of estimated locations of objects or landmarks which the robot detects and identifies in the environment. As this mapping format assumes that the robot can easily find features in the structured environment like indoor rooms, it is hardly applicable to an unstructured environment such as a natural cave where adequate features are not necessarily found. On the other hand, the location-based map is a list of estimated states of specific spaces. The occupancy grid-based map is a typical example. The map represents grid-partitioned spaces and estimates the occupancy in each space. Since it does not rely on the existence of identifiable features in the environment, the occupancy grid-based map is appropriate for the unstructured environments.

To express the unstructured environment where the floor, walls, and ceiling are uneven, the map needs to be three-dimensional. The three-dimensional occupancy grid-based map is composed of voxels or cubic cells each of which holds a probability of the corresponding space to be occupied by obstacles. However, due to the curse of dimensionality, the 3D map representation tends to take up significantly large memory spaces, leading to necessity of compression techniques. Octree is one of the compression techniques for 3D occupancy grid-based maps [99]. The map is formatted in a data structure called "octree," a tree where each node can hold eight child nodes. The root node represents a voxel covering the entire space. The node can be expanded and hold eight child nodes which represent eight equally divided voxels. This expansion repeats until enough resolution is gained. Then, each leaf node holds a state indicating the voxel is either occupied, empty, or

unknown. If all the eight leaf nodes hold the same state, they are merged and their parent node becomes a leaf node. This operation reduces the number of leaf nodes, compressing the data.

These techniques of the 3D mapping allows a robot to perform SLAM in the unstructured environment. Fairfield introduced underwater SLAM techniques using the 3D occupancy grid-based mapping as a part of the Rao-Blackwellized Particle Filter (RBPF) [8]. The RBPF is a particle filter that estimates the robot's location and the map by sampling particles. Each particle represents a specific location of the robot and a map built based on the location. By updating the particles and resampling them based on the sensory data, the RBPF provides an estimation of the robot's location and the map.

Fairfield et al. also addressed the SLAM problem in a large environment by introducing the segSLAM technique [72]. Because of the uncertainty caused by sensory noises and the motion model, the localization inherently contains uncertainty or errors. As the robot navigates for a long period of time, it suffers from the accumulated errors due to the locality of sensory data, which means the sensors only provides information about a local area. Since the map is built based on the estimated location, this accumulated errors negatively affect the mapping process. The segSLAM tackles the problem by segmenting the SLAM process. Instead of building a single map, it builds a series of submaps. The segSLAM maintains database of segmented submaps of the environment aside from the population of particles each of which represents a robot's pose and the map that is currently being built. At a specific timing, the robot saves the current submap to the database and starts to build a new submap. As repeating this process, the robot generates a series of submaps. The segSLAM performs loop-closure on the submaps. When the robot detects that it is entering an area which has previously been mapped, it retrieves the corresponding submap from the database. As each submap is associated with their previous and next submaps based on their relative poses, the robot can close a "loop" of submaps by re-aligning the submaps. For example, the incremented Smoothing and Mapping (iSAM) expresses the relations between the submaps as metric constraints and adjusts alignments of submaps based on the constraints so the chain of the submaps can be smoothed to form a loop [100, 101]. Since the submaps are relocated at more accurate positions, the loop-closure of submaps corrects the accumulated errors in the SLAM process.

In the loop-closure process, there is a problem about timing to segment the SLAM process. The timing of segmentation affects the submap size and consequently the SLAM quality due to the loop closure of the submaps. If the segmentation is less frequent, the submaps tend to be large. As the submap gets larger, it will contain more accumulated errors in the mapping results, causing distortion of the submap. The contained errors make the loop-closure ineffective as they cannot be corrected by re-aligning the submaps. On the other hand, if the segmentation is rather frequent, the submaps tend to be small. As the robot builds more submaps for the same space, it needs to process more submaps for loop closure, requiring more computational power. Moreover, it would not be able to localize itself based on such a small submap. Thus, the robot needs to find an

optimal timing to start to build a new submap so as to perform the loop-closure effectively. As a part of the work for the segSLAM, Fairfield proposed two methods: motion error segmentation and predictive score segmentation [8]. The first method decides to segment the SLAM process when the uncertainty in the robot's estimation exceeds a specific threshold. The second method calculates a score of the current submap's ability to predict future sensory data. When the score drops, it decides to start building a new submap. While the second method outperforms the first in a structured environment, they mentioned that there was not significant difference in an unstructured space.

Hence, their SLAM techniques allow the robot to perform localization and mapping without relying on any external supports or resources. However, their techniques and methods are designed for the underwater and ground-based robot navigation. When the robot starts over mapping for the segmentation, it is supposed to stay on a specific location so that it can initially build a part of the map. This assumption may not be applicable to other types of robots such as an aerial vehicle. In addition, the robot needs to visit some locations more than once. For example, the loop-closure of submaps requires the robot to visit the location where the robot has already visited. To deliver the mapping results, the robot also needs to come back to the location where it was initially deployed because the ideal network or communication is not available in the environment.

Deployment of multiple robots are beneficial to address the issues raised by the single-robot SLAM. If they cooperatively localize themselves, they can improve the mapping accuracy without using the loop-closure of submaps since the submaps can be built based on more accurate estimated locations. They can also transfer the submaps by establishing an ad hoc network. In addition, the robots can also coordinate to determine their paths so that their mapping covers the whole environment. Thus, each single robot does not need to go to the locations where it has already visited, and the mapping operation can be done in a shorter time compared to the single-robot mapping. In this area, a variety of approaches have been proposed during the last few decades.

Singh and Fujimura introduced cooperative mapping by heterogeneous mobile robots which are different in size [102]. Once a member finds an unexplored space which it cannot access due to its size, it calls other members for help by broadcasting. Zlot applied a market control architecture to asynchronous multi-robot collaboration [103]. They perform auction for mapping task assignments and pass local maps without synchronization. Howard et al. introduced an incremental deployment of multiple robots to deploy the robots all over the environment [104]. To maximize their coverage area of mapping, the robots scatter in the environment by interacting each other. If a robot finds another robot blocking its path, it changes its destination to the location of the blocking robot, which takes over the original destination. As a result, it "pushes" another robot toward its destination. In these approaches, although the robots cooperatively build maps about the environment, they are assumed to know their correct locations and have an access to the ideal network so that they can communicate with any other robots.

Williams et al. presented a mapping method in SLAM for multiple mobile robots [105]. Each robot generates a feature-based map in its local reference frame. At a specific interval of time, local maps are integrated into a global map by data association. Likewise, Rodriguez-Losada et al. presented a local map fusion method which updates a global feature map with a local map generated by an individual robot [106]. A local map is a list of estimated locations of individual segments representing a wall or an obstacle's surface. Each robot has its own local map and it updates the global map by transforming the reference frame of the local map. However, as they use feature-based maps, their methods are supposed to work in structured environments to detect sufficient features or landmarks.

Instead of the feature-based mapping, Thrun presented a mapping method by using a metric map as a part of the SLAM problem, and extended the probabilistic SLAM to the multi-robot scenario [107]. To perform multi-robot mapping as a probability theoretic method, each robot performed a particle filter to estimate its own location and a 2D occupancy grid-based map about its local area. For cooperation, they matched the locally built maps to generate a global map. In addition to the 2D map, 3D occupancy grid-based maps can also be efficiently merged. Jessup et al. introduced an efficient merge algorithm for 3D occupancy grid-based maps [108, 109]. The 3D map is formatted in the octree data structure to efficiently access and update the map data. Then, the maps are transformed to the same reference frame, and finally merged into the global map. However, the merging process is supposed to be performed off-line in a centralized manner. This prohibits the robots from cooperating with each other in the middle of the operation.

Contreras et al. introduced a method of decentralized 3D mapping based on octree-based point clouds [110]. Each of the vehicles individually builds its local point cloud map from lidar data. Once a pair of robots find that their local maps share a common space, they exchange their map data and merge the local maps by re-registering point-clouds into another octree data frame. However, because their system is intended to be implemented to automobiles, their localization purely depends on GPS.

Unlike merging the metric maps, a topological map loosely connects the maps and allows the robots to update the mapping data even after combining the local maps. Howard introduced a manifold map of metric maps [111]. The graph-based map is allowed to overlap with another map. Once a pair of robots find each other, they take mutual observation to resolve the overlapping parts of their map, leading to a merged map. This approach of using a topological map also appears in other work. Pfingsthorn et al. introduced a hybrid map integrating graph-based and occupancy grid-based mapping [112]. In the same concept, Chang et al. designed a Multi-Robot SLAM (MR-SLAM) algorithm for two-dimensional navigation [113]. Similarly, Dube et al. presented a mapping method using a combination of a pose graph and sets of point clouds each of which is associated with the corresponding node in the pose graph [114]. Generally, in their work, the topological information of the graph-based map allows the robots to perform loop-closure and merge local

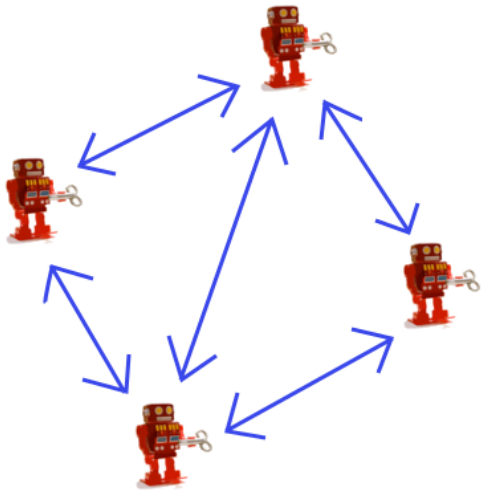

Figure 5.1: Cooperative localization: Multiple robots taking measurements on their relative poses with respect to their neighbors, forming loops of interactions.

maps to improve their occupancy grid-based maps. However, these methods are assumed to use the ideal network to share the data or perform the loop-closure task off-line.

Throughout the existing work reviewed above, the SLAM problem in an enclosed environment was addressed by both the single-robot and the multi-robot approaches, but with some restrictions, leaving room for improvement. The single-robot methods perform the loop-closure of submaps to improve their mapping performance, yet requiring the robot to re-visit the same location for the process. Multi-robot SLAM techniques address the mapping task as multiple tasks which each of the robots individually handle. Some of the approaches perform a similar process of the loop-closure of submaps by using a topological map. However, since the topological map needs to be processed in a centralized way, the loop-closure of submaps by multiple robots is performed off-line or with the ideal network. The other approaches also assume that the robots can access the ideal network or centralized resources. This assumption prohibits the robots from performing SLAM in a decentralized way.

5.2 Approach: Mapping Based on Cooperative Localization

In this study, we present the distributed submap building. The proposed mapping process is performed along with the decentralized cooperative localization, which was presented in the previous chapters. Mapping based on the cooperative localization corrects the errors in a different way from the loop-closure of submaps. Figure 5.1 shows an abstract view of cooperative localization, in which multiple robots take measurements on their relative pose with respect to each other. They

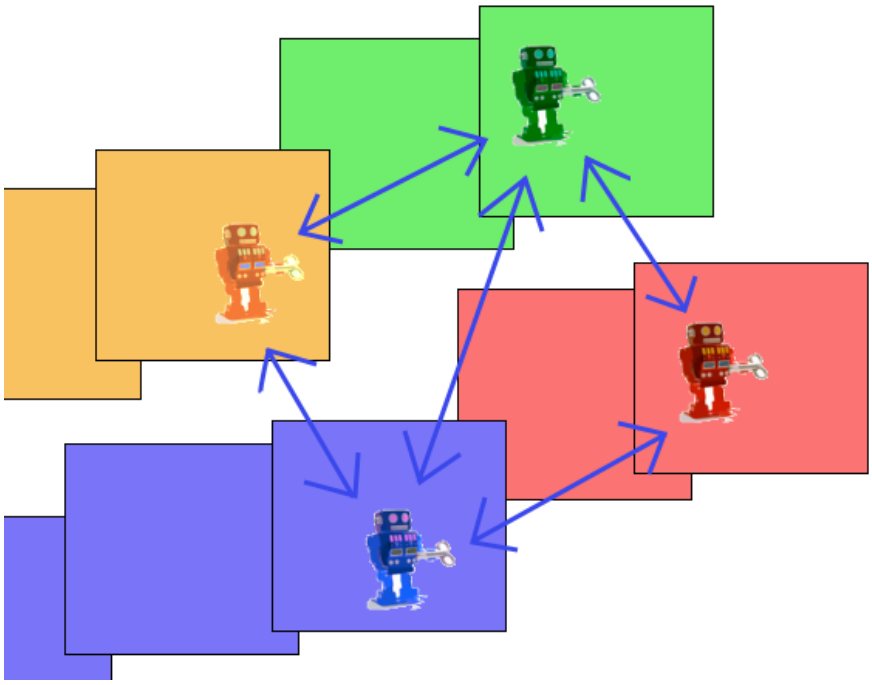

Figure 5.2: Each robot builds a sequence of submaps (depicted as squares in the corresponding color) based on the estimated locations given by the cooperative localization.

can form a "loop" of interactions. If they pass their own estimated locations, they may perform a similar process of the loop-closure without finding a landmark in the environment or closing a loop of submaps. Thus, based on the cooperative localization, the robots can build more accurate maps.

As the robots establish an ad hoc network for the cooperative localization, they also use the network to transfer their mapping data between the robots. Each robot builds submaps of regions where it has visited. It locally builds a submap and localizes itself by its SLAM process. Figure 5.2 depicts segmented SLAM performed by multiple robots. While a single robot can process one submap at once, multiple robots can take care of multiple submaps concurrently in the way that each robot processes the area where it is located. When a neighboring robot is entering the area where the corresponding submap is available, the submap is transferred to the entering neighbor from the robot which originally built it. Then, the neighboring robot starts using the transferred submap for its SLAM process.

The proposed method provides advantages for multi-robot mapping. For example, the method can be performed in a decentralized manner. The existing methods of multi-robot SLAM needed to perform the loop-closure of submaps in a centralized manner since the computation needs all the mapping data to close a loop of relations of the submaps. On the other hand, the proposed approach performs the loop-closure on the robots' locations instead of those of submaps, and it can

be performed without using any centralized resources. Also, because the segmentation of SLAM generates submaps rather than a single global map, this allows a robot to pass some submaps to the other robot. As each robot holds partial maps, this improves the memory efficiency as they can reduce the memory space to store the mapping data. In addition, the robots can cooperatively perform mapping with their neighbors; newly deployed robots receive already built submaps from the others. Finally, as the ad hoc network can deliver the submaps from the robots to the base station, the robots do not need to go back to the initial location. Hence, the robots can explore into deep areas of the environment without considering the return trips.

In the rest of this chapter, the details of the proposed approach are provided. First, the mapping process which each individual robot performs is described. Then, the map segmentation and the submap transfer are presented by addressing challenging problems in the multi-robot system without an ideal network.

5.3 Individual SLAM Process

While performing the cooperative localization, each individual robot also performs its SLAM process, and the submap is updated in this procedure. This local process is based on the existing SLAM methods with different types of sensors. Equation (2.6) shows the mathematical model of the Bayes filter for SLAM problem. This model is implemented in a variety of filtering methods. When a map is occupancy grid-based, a nonparametric filter such as Rao-Blackwellized Particle Filter (RBPF) is generally used [64].

RBPF is a nonparametric filter that estimates the robot's location and builds a map by representing and updating a probability distribution with particles. Like the original particle filter shown in Figure 4.1, the probability distribution is expressed by a population of particles which are sampled based on the motion and sensor models. An area with dense particles indicates a higher likelihood.

In RBPF, each particle holds a specific sampled state: a robot's pose x and a map m that is based on a specific trajectory. Let's say there are n particles at time t. Then, the population of particles will be n pairs of a pose and a map.

$$\left\{\{x_t^{[1]}, m_t^{[1]}\}, \{x_t^{[2]}, m_t^{[2]}\}, ..., \{x_t^{[n]}, m_t^{[n]}\}\right\} \tag{5.1}$$

As described in Section 4.1, the particles are processed in the following steps: prediction, evaluation, and resampling. RBPF also adds another step: map update. In the rest of this section, each step is described in details.

5.3.1 Prediction

Based on Equation (4.1), the prediction step changes the pose x of each particle based on the motion model and control inputs u. The control inputs are given by odometry or estimated distance the robot has moved since the last iteration. The particles are moved by this estimated distance with some additional noise given by the motion model.

Odometry data is given by a variety of sensors. Generally, an IMU is used for any kind of robots. In addition to an IMU, other types of sensors can be used for its hardware characteristics. For example, encoders attached at the wheels are generally used for ground-based robots, telling how many times the wheels have rotated. Doppler Velocity Log (DVL) is used for underwater robots, providing approximation about the robot's velocity [8]. The aerial navigation, however, is not able to use these types of sensors. Obviously, encoders on the wheels cannot be used. DVL is also not applicable as it depends on dense medium such as water; air or gaseous medium may not allow it to provide accurate velocity data. Let alone, the thin Martian atmosphere. Instead of these sensors, visual odometry is generally used for aerial robots; velocities are estimated by tracking the view of the terrain texture [115, 116, 117]. As a camera generates infrared images, it is also applicable to a dark space. Lidar odometry is also a possible approach: motion estimation by comparing lidar scans at different time points [118, 119, 120]. While some of them detect features in the lidar scans and perform a similar process like visual odometry [120], the others use lidar scans directly by comparing them as point clouds by Incremental Closest Point (ICP) [118, 119].

As an infrared depth camera can generate both image data and point clouds, both visual and lidar odometry methods are applicable to an aerial robot equipped with an infrared depth camera. Thus, in this work, the robots are assumed to use an IMU and an infrared depth camera to generate odometry data.

Finally, the population gains the changed pose x in each particle.

$$\left\{\{\hat{x}_{t+1}^{[1]}, m_t^{[1]}\}, \{\hat{x}_{t+1}^{[2]}, m_t^{[2]}\}, ..., \{\hat{x}_{t+1}^{[n]}, m_t^{[n]}\}\right\} \tag{5.2}$$

Due to the noises in the odometry data, the resulting population of particles tend to scatter over a larger area, indicating larger uncertainty. This error is corrected in the next steps.

5.3.2 Evaluation

This step evaluates each particle with the gained sensory data. If the sensory data matches with the map well, the corresponding particle gains a higher weight value for resampling.

As an infrared depth camera is assumed to be used in this work, the sensory data is given as point cloud: a list of coordinates of detected points of obstacles with respect to the camera. The scan matching technique is used for matching point clouds with an occupancy grid-based map [11], but it is only applicable to a 2D map. Another approach is to trace the point cloud data in a

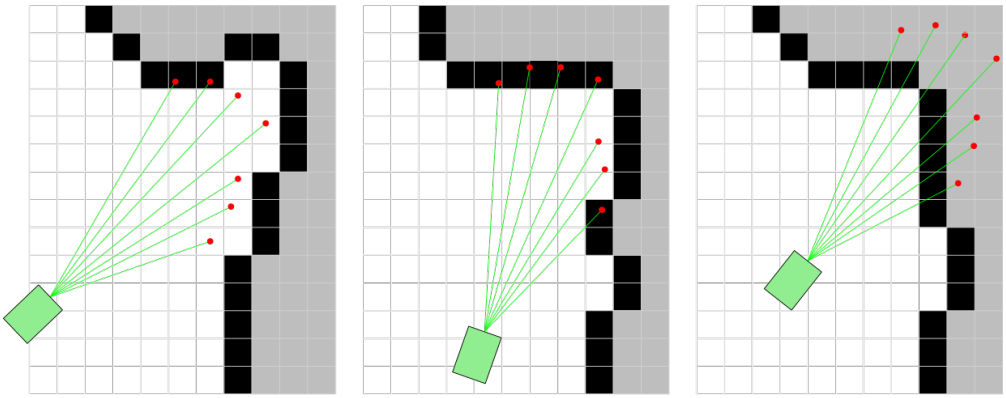

Figure 5.3: Weighing particles. Each particle holds a specific state of the robot's location and the map. Given sensory data, a set of rays are traced in the state of each particle. The particles shown in the left and right figures cause contradiction with the sensory data and get lower weights; the data lands on free space in the particle's map (left) and it runs through the obstacles in the particle's map (right). The particle shown in the middle figure, on the other hand, allows the sensory data to land on more occupied cells, leading to a higher weight.

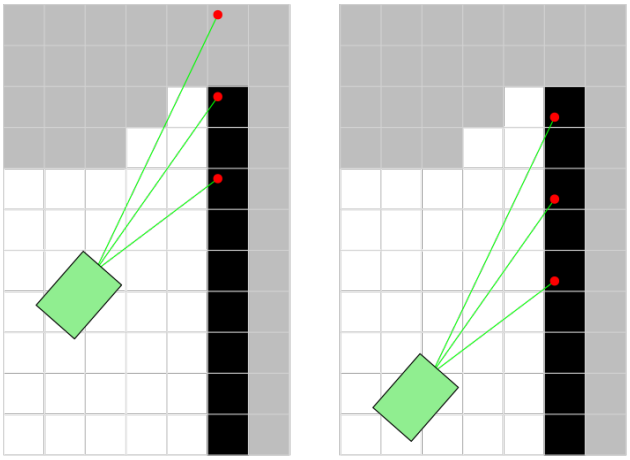

Figure 5.4: Two cases of ray tracing. Point cloud data is placed in the map with two different positions. The occupied, empty, and unknown areas are colored in black, white, and gray, respectively. In the left case, two points hit the occupied areas while the other one falls in the unknown area. In the right case, all the three points fall in the occupied areas.

map. Figure 5.3 shows how the point cloud data is traced in an occupancy grid-based map in each particle. The points in the data are placed in the map of the particle, and the particle is evaluated based on how the data matches with the map.

An intuitive way to evaluate a particle may be just to count the numbers of the points landing on the occupied areas and those landing on wrong areas. If a particle gets more points falling in the occupied area, it will get a higher weight value. However, this intuitive way can cause an issue when the map contains unknown areas. Figure 5.4 shows matching of point cloud data with two

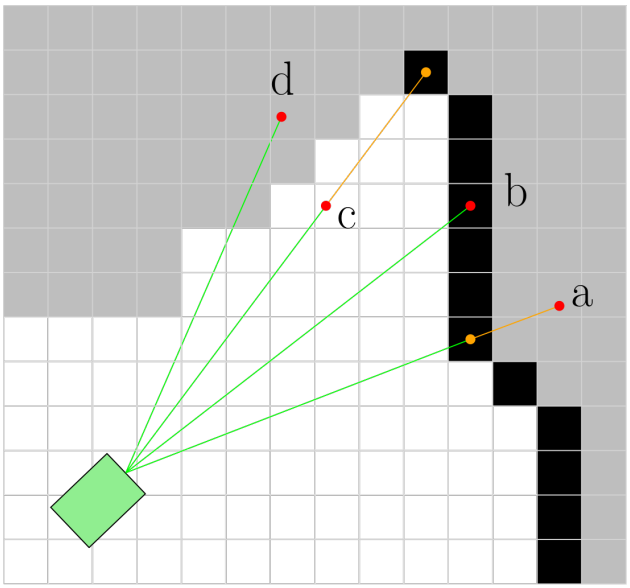

Figure 5.5: Ray tracing of point cloud data in an occupancy grid-based map. The rectangle at the bottom represents the sensor giving the point cloud data. Among the four points, only point "b" falls in the occupied area. Point "a" goes beyond the occupied area, point "c" falls in the empty area, and point "d" falls in the unknown area. The red points are the actual point cloud data. The orange points are simulated points traced in the map based on the sensor's pose. The lengths of the orange line segments are the distances between the actual data points (red) and the simulated points (orange).

particles. The right case has more points falling in the occupied areas so it would gain a higher weight value if the particles were evaluated based on counting the number of points on occupied areas. But, since the robot is building the map concurrently and the map contains unknown areas, some detected points may fall in such an unknown area, and the left case could be also reasonable. Hence, it is desirable to evaluate both cases with similar weights.

In this work, the point cloud data is ray-traced in the map. The ray-tracing process simulates sensor rays based on the sensor's pose and the map, and each point from the actual sensory data is checked to see how close it is to the simulated point. For example, in Figure 5.5, four points in the point cloud data are placed in the map. The ray-tracing process draws lines from the sensor's location toward the points to simulate the rays of the sensor. When it hits the occupied area, it drops the simulated point at the center of the occupied area hit by the ray (orange points in the figure). Point "d" is discarded in this process as it is in the unknown area and a simulated point cannot be generated.

Then, the sensor model of each point is defined based on the distance between the actual and

simulated points. Let z be a point in the data which gained a simulated point $\hat{z}$ by the ray-tracing.

$$p(z|x,m) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}|z-\hat{z}|^2} \tag{5.3}$$

where σ is the standard deviation determined based on the characteristics of the sensor beam.

Finally, the weight value $w_{t+1}^{[i]}$ is calculated based on the average probability of the sensor model. To get the average of a set of probabilities, the geometric mean is calculated [121].

$$w_{t+1}^{[i]} = \eta \prod_{z \in Z} \left(p(z|\hat{x}_{t+1}^{[i]}, m_t^{[i]}) \right)^{\frac{1}{|Z|}} \tag{5.4}$$

where η is a normalization factor and Z is a set of points which have the corresponding simulated points. With this calculation, the weight value is reduced only by erroneous points such as points "a" and "c" in Figure 5.5, but not by a point falling in the unknown areas such as point "d".

5.3.3 Resampling

After calculating the weight values, the particles are resampled based on their weights. It randomly chooses a particle in the population at a probability which is proportional to the weight. If a particle was evaluated with a higher weight, it will be resampled at a higher chance.

$$\{x_{t+1}^{[i]}, m_t^{[i]}\} = \{\hat{x}_{t+1}^{[j]}, m_t^{[j]}\} \sim \frac{w_{t+1}^{[j]}}{\sum_{k=1}^{n} w_{t+1}^{[k]}} \tag{5.5}$$

Then, the new population will have the resampled particles.

$$\left\{ \{x_{t+1}^{[1]}, m_t^{[1]}\}, \{x_{t+1}^{[2]}, m_t^{[2]}\}, ..., \{x_{t+1}^{[n]}, m_t^{[n]}\} \right\} \tag{5.6}$$

5.3.4 Map Update

The map is updated by applying the point cloud data. Generally, the ray-casting is used to integrate point cloud data into an occupancy grid-based map [99]. In this study, we follow the generic approach for this part. As shown in Figure 5.6, points from the point cloud data are placed in the map based on the pose. The areas where the points fall get higher probabilities of occupancy. If the areas are between the points and the sensor, they will gain lower probabilities. These generated probabilities are integrated to the existing probabilities of the corresponding areas in the map.

Finally, the map of each particle is updated and the population looks as follows.

$$\left\{ \{x_{t+1}^{[1]}, m_{t+1}^{[1]}\}, \{x_{t+1}^{[2]}, m_{t+1}^{[2]}\}, ..., \{x_{t+1}^{[n]}, m_{t+1}^{[n]}\} \right\} \tag{5.7}$$

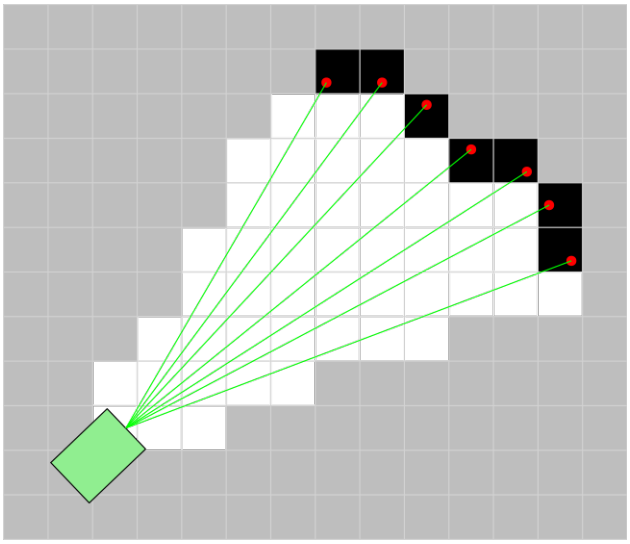

Figure 5.6: Map update by point cloud data (red dots). Each point draws a line. The grids on the line gets lower probabilities (white cells) while those containing the detected points get high values (black cells). Other grids (gray cells) remain unchanged.

Through these steps, particles with states agreeing with the acquired sensory data can survive highly likely, providing reasonable estimates on the robot's pose and maps.

5.4 Map Segmentation

Segmentation of SLAM is a technique to divide the SLAM process into smaller tasks. In the case of single-robot navigation, the robot divides the entire process into multiple SLAM processes each of which builds a submap in the corresponding local area. Multi-robot navigation can apply a similar approach in which each robot builds a series of submaps by segmenting the individual SLAM process, and it also takes advantage of cooperation of the robots. In this section, the issue of segmentation for multi-robot SLAM is discussed, followed by the discussion of the timing of segmentation. Then, the details of the segmentation process are presented.

5.4.1 Timing of Segmentation

The timing of segmentation is crucial as it determines the size of each submap. If the robot frequently segments its SLAM process, it would generate many smaller submaps. Otherwise, it would generate fewer and larger submaps.

As our approach closes a loop of interactions of the robots by the cooperative localization, the optimization techniques used for the loop-closure of submaps are not necessary. However, the proposed approach entails another issue associated with the multi-robot SLAM. Figure 5.7 shows two cases of different sizes of submaps. In the figure at the top, the robot segments its SLAM

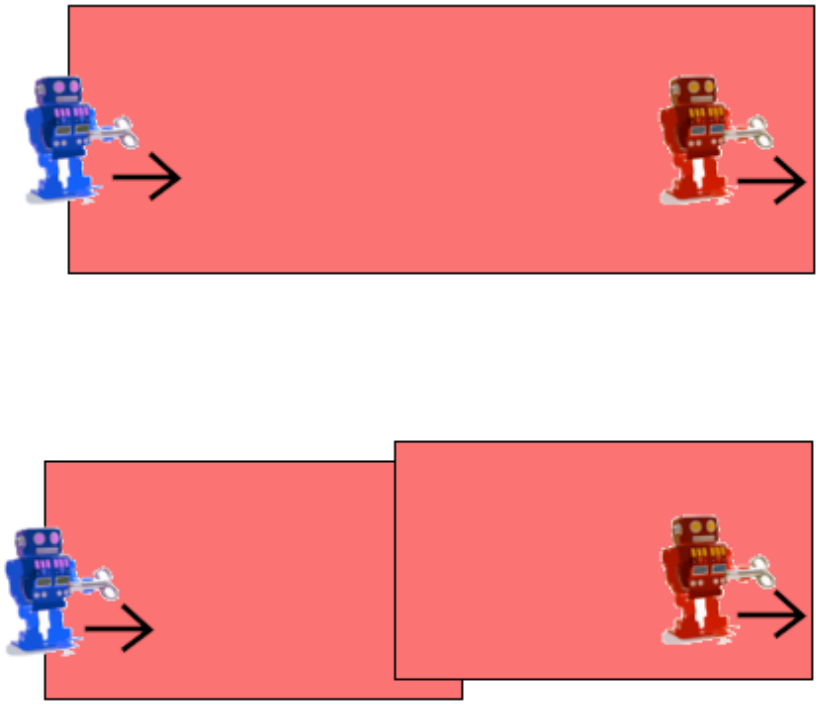

Figure 5.7: Two cases with different sizes of submaps. The top image is a case with large submaps and the bottom is a case with small submaps. In both cases, the red and blue robots are moving toward the right side. The red rectangles are submaps built by the red robot. The blue robot is entering the red robot's submap area. While the case at the bottom allows the red robot to pass the submap into which the blue robot is entering, the case at the top cannot allow it as both robots are in the same submap.

process less frequently and generates a submap with a long interval. When the blue robot, which is moving behind the red robot, enters the submap area, the red robot cannot pass the submap to the blue robot since the submap is still being built. Thus, less frequent segmentation leads to submaps with larger intervals and it prohibits the robots from performing the submap transfer. On the other hand, in the figure at the bottom, the red robot more frequently segments its SLAM process and generates submaps with shorter intervals. As the blue robot enters the submap that is not currently being built, the red robot can pass the submap to the blue robot. Nevertheless, there would be a bottleneck of the computation if the robots segmented their SLAM processes excessively frequently. Therefore, the robots need to balance the frequency of the segmentation.

In this work, we assume that the robots move through the enclosed environment such as a natural cave. While moving into deeper areas, the robots do not rapidly change their directions. When the robot keeps building a single submap, the submap grows in the direction to which the robot is moving. As the robot does not go backward, the location where the robot started to build the submap and the robot's current location will correspond to each end of the submap. Based

on this assumption, the robots determine to segment their SLAM processes as follows: When the robot starts to build a submap, it saves its estimated location. Then, while building the submap, the robot checks the length of the line from the current estimated location to the saved location. If it exceeds a specific threshold, the robot segments the SLAM process and starts to build a new submap. Thus, the submap size is confined by the threshold. Since the robots keep some distance from each other to form a meshed network, if the threshold is set to be less than this distance, the robots will always have submaps to pass to the neighbor.

5.4.2 Segmentation Process

For the segmentation of SLAM, the robots save their SLAM results and start over new SLAM processes. When a sampling-based SLAM such as a particle filter is used, this means that each robot saves the current particles and starts over the SLAM process with new particles. As we use RBPF, the particle filter holds a set of particles each of which is a set of a specific location x of the robot and a submap m built based on it.

$$\left\{\{x^{[1]}, m^{[1]}\}, \{x^{[2]}, m^{[2]}\}, ..., \{x^{[n]}, m^{[n]}\}\right\} \tag{5.8}$$

Once the robot decides to start a new segment, it deals with the particles as follows.

1. Save the current particles (the copied particles represent the previous segment)
   $segment_{i-1} = \left\{\{x_{i-1}^{[1]}, m_{i-1}^{[1]}\}, \{x_{i-1}^{[2]}, m_{i-1}^{[2]}\}, ..., \{x_{i-1}^{[n]}, m_{i-1}^{[n]}\}\right\}$
2. Clear the submaps in the current particles
   $segment_i = \left\{\{x_i^{[1]}, \emptyset\}, \{x_i^{[2]}, \emptyset\}, ..., \{x_i^{[n]}, \emptyset\}\right\}$
3. Each particle builds a submap from scratch based on its location, resulting in
   $segment_i = \left\{\{x_i^{[1]}, m_i^{[1]}\}, \{x_i^{[2]}, m_i^{[2]}\}, ..., \{x_i^{[n]}, m_i^{[n]}\}\right\}$

Right after starting a new segment at Step 2, localization cannot be performed due to the empty submaps. Practically, for several seconds, each particle refers to its "ancestor" in the previous segment. Then, each particle is evaluated with the existing submap in its ancestor particle for localization. At Step 3, as the submaps in the current particles are built large enough, the particles refer to their current submaps for localization. Unlike the existing segmentation method, which requires the robot to stay to build a new submap, this approach allows the robots to keep moving even when they are segmenting their SLAM processes. Thus, the SLAM process is segmented by saving the particles as a segment and starting over the new SLAM process.

5.5 Submap Transfer

As the robots hold a series of submaps by the segmentation, they can pass the submaps to the other robots entering in the corresponding areas. This section explains how the robots interact with each other so that they can pass their submaps and integrate the received submaps into their own SLAM processes.

5.5.1 Entry Detection

As each robot performs the cooperative localization, the robot knows the estimated locations of the neighboring robots. This information is used for the entry detection in the submap transfer process.

A submap holds information about the ranges of X, Y, and Z axes, where the submap exists with respect to the global reference frame. The robot checks if the location of the neighbor falls in these ranges. If the coordinates of the neighbor's location fall in these ranges, the robot determines that the neighbor is entering the area that is covered by the corresponding submap. While a segment has multiple submaps, we assume that the ranges of a submap are not significantly different from those of the others in the same segment, and one of the submaps is selected for this entry check for each segment.

Let us assume that the robot i is communicating with its neighboring robot j as shown in

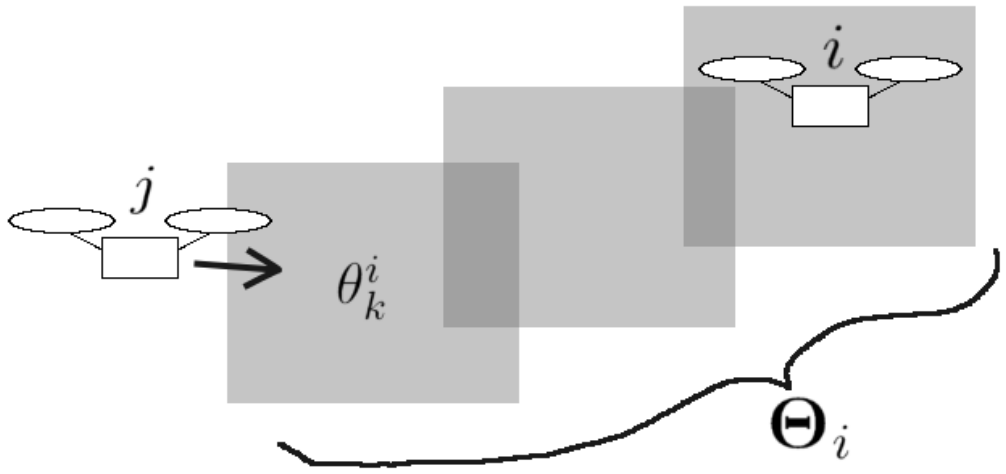

Figure 5.8: Entry detection of a neighboring robot. The robot i holds a set of segments Θ_i, each of which holds particles with sbumaps. Each gray square is one of the submaps in the segment. If the robot j is entering the ranges of the submap in the segment θ_k^i, the robot passes the submap data to the robot j.

Figure 5.8. The robot i holds a list of segments Θ_i. As they perform the cooperative localization, the robot i knows the estimated location of the robot j. If the robot i finds that the robot j is entering the area associated with the submap in the segment θ_k^i, it passes the corresponding submap data to the neighbor.

5.5.2 Transfer Process

Since the SLAM process is based on a particle filter, a segmented SLAM process holds a set of particles each of which holds a submap. In terms of the communication bandwidth, it may be infeasible to send all the particles to the other robot. This problem is addressed by sampling a subset of the particles and transferring the submaps of the selected particles. The sampling is done in a similar way described in Section 4.3. After a robot receives submaps from a neighbor, it integrates them into its SLAM process as follows.

Suppose that the robot has the following particles currently,

$$segment_i = \left\{ \{x_i^{[1]}, m_i^{[1]}\}, \{x_i^{[2]}, m_i^{[2]}\}, ..., \{x_i^{[n]}, m_i^{[n]}\} \right\}$$

and it is receiving a set of submaps M from the other robot. Let a map in the set be $m_{\text{new}}^j \in M$. The receiving robot saves the current segment, and then starts a new SLAM process by creating a new segment with the received submaps. For each particle in the new segment, one of the received set of the submaps is selected randomly. Hence, the k-th particle's map is set as

$$m_{i+1}^{[k]} = m_{\text{new}}^j \sim \frac{1}{|M|}.$$

Finally, the robot has the new segment.

$$segment_{i+1} = \left\{ \{x_{i+1}^{[1]}, m_{i+1}^{[1]}\}, \{x_{i+1}^{[2]}, m_{i+1}^{[2]}\}, ..., \{x_{i+1}^{[n]}, m_{i+1}^{[n]}\} \right\}$$

Thus, the robot saves the existing submaps and switches to a new process with the received submap data. After this integration process, the robot continues its SLAM process.

CHAPTER 6
SYSTEM DEVELOPMENT BASED ON ROS AND SIMULATION USING GAZEBO

To demonstrate and analyze the decentralized cooperative localization and the distributed submap building, they are implemented in a robotic system. In this study, we use Robot Operating System (ROS): the middleware to run robotics programs [122]. Each robotics program, e.g., control, SLAM, etc, runs as a ROS node and communicate with other ROS nodes in the framework provided by ROS. Through the ROS system, the robotics programs gain data from the sensors and provide control inputs to the robot's hardware. In the actual situation, each robot runs its own ROS nodes individually and needs to communicate with each other. Hence, communication protocols are also designed.

The developed ROS nodes are demonstrated in a simulation using a 3D robotics simulator. Here, we use the Gazebo simulator, which is compatible with the ROS system [123]. In addition to the physical characteristics of the robot hardware and the environment, the inter-robot communication is also simulated by developing a plugin; the communication may be blocked by obstacles in the environment. Finally, the proposed methods are evaluated by comparing the simulation results quantitatively.

In this chapter, the communication protocols are first discussed and presented. Then, the system components of the ROS and the Gazebo are described. It is followed by the designs of the simulation models. After that, the simulation setups are described and presented with the evaluation metrics. Finally, the simulation results are provided to evaluate the proposed method.

6.1 Protocols

The robots, running separate processes individually, need protocols to synchronize and communicate with their neighbors so that they can exchange data of mapping and localization. The decentralized cooperative localization requires communication for its data exchange, where a pair of robots exchange their estimation about their locations. The distributed submap building requires communication for its submap transfer, where the robot passes its submap to its neighbor. This section describes how the robots synchronously communicate with each other for these interactions.

6.1.1 Decentralized Cooperative Localization: Data Exchange

The interactions for the cooperative localization are performed in a decentralized manner and each robot needs to individually work on several steps: detection of neighbors, initiation of interaction, exchange of their data, and update on its estimation by using the acquired data from the interaction. The previous chapters about the cooperative localization focused on the last step, where

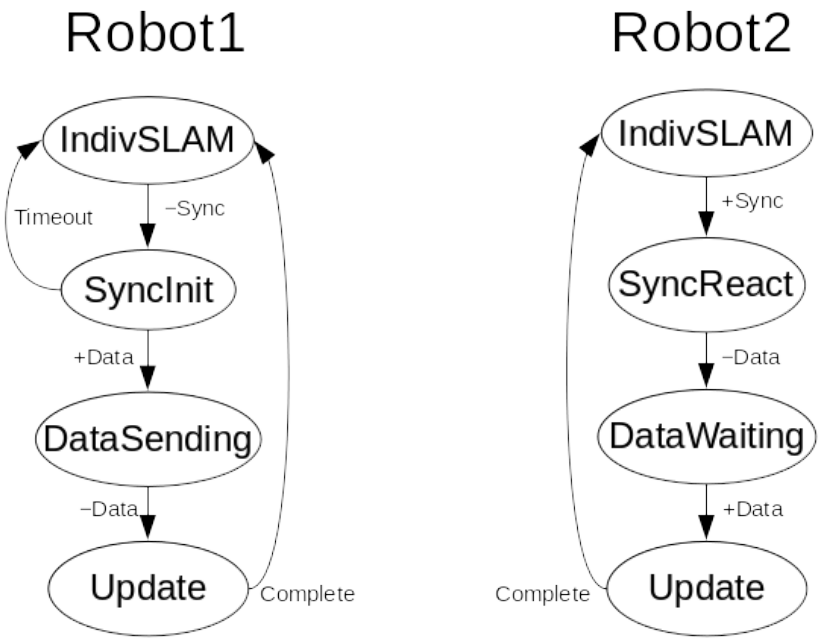

Figure 6.1: State transitions of a pair of robots exchanging data. Robot1 is a robot initiating the interaction and Robot2 is a robot responding to the request. Their states are set to "IndivSLAM" while they are individually performing SLAM. They switch to next states when the specified conditions are met. "Sync" represents a synchronization packet, and "Data" represents a data packet with the robot's estimation. The signs "−" and "+" represent transmission and reception of the packet, respectively.

the conservative data exchange is performed, assuming the robots access their data without the preliminary steps. In an actual system, the robots require the preliminary steps because each robot is working independently and needs to establish a pairwise communication to exchange data. This section describes these preliminary steps which allow the robots to facilitate the communication for the cooperative localization.

Detection of Neighbors: While performing its individual SLAM, a robot maintains a list of its neighbors which is used to choose a robot to interact for the cooperative localization. To update the neighbor list, a robot periodically broadcasts a beacon packet, which can be transmitted in its communication range. Once another robot receives the beacon packet, it replies back. As receiving this replying packet, the originating robot updates its neighbor list.

Initiation of Interaction: Periodically, each robot attempts to initiate an interaction with another robot in its communication range. From the neighbor list, the robot randomly selects one of

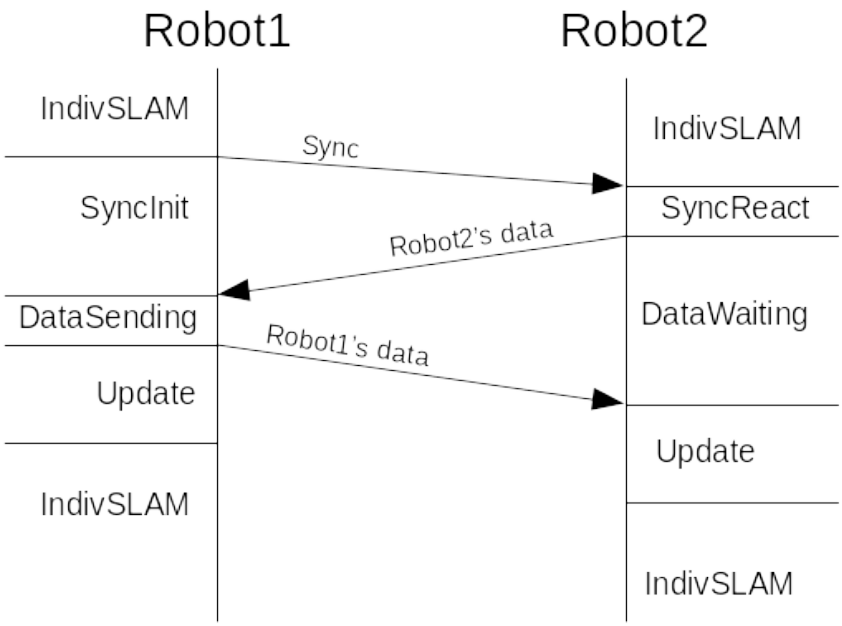

Figure 6.2: Protocol diagram along the states of the robots. Robot1 is a robot initiating the interaction and Robot2 is a robot responding to the request. The vertical lines represent time, which passes from the top to the bottom. The horizontal lines represent state transition. The arrows represent transmissions of packets.

them to interact. Then, it sends a synchronization packet to the selected robot. This synchronization packet is used as a request to interact. If the robot does not receive a reply within a specified time period, it aborts the attempt due to timeout.

Exchange of Data: Once a robot receives a synchronization packet, it sends its data to the robot who sent the packet as a response. When this data packet is received, the other robot's data is sent back. Finally, both robots get each other's data with relative pose information which is stored in the data packet.

This process is designed and controlled based on a finite state machine. The robots change their internal states so that they can locally synchronize with each other to exchange data. Figure 6.1 shows how a pair of robots change their internal states while exchanging the data. Transmission of a synchronization packet is specified "−Sync" and reception is "+Sync". "−Data" and "+Data" are transmission and reception of a data packet, respectively. They are initially in "IndivSLAM" as they are individually performing its SLAM process. Once Robot1 initiates the interaction, it sends a synchronization packet to Robot2 (−Sync), and Robot1's state changes to "SyncInit." In state "SyncInit," Robot1 can go back to "IndivSLAM" if it determines that Robot2 no longer replies possibly because of a collision of requests (Timeout). When Robot2 receives the synchronization

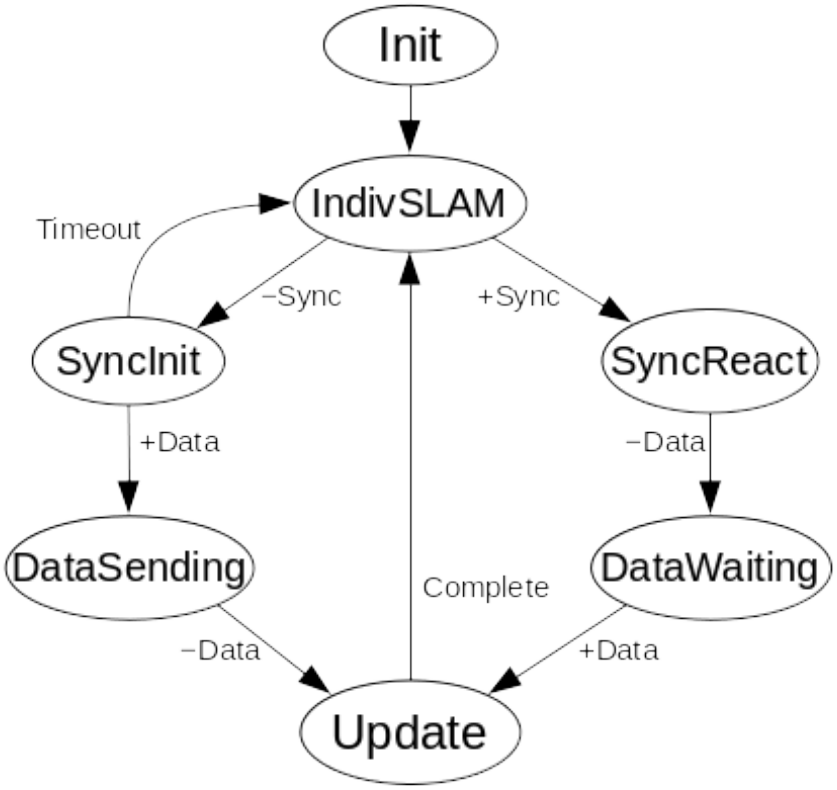

Figure 6.3: The entire state machine performed by an individual robot. "Init" is the state in which the robot initializes its settings and immediately enters into "IndivSLAM" once it starts its individual SLAM. If it is initiating an interaction, the state follows the route starting with "SyncInit." If it is responding to the other's request for an interaction, it follows the route starting with "SyncReact."

packet (+Sync), it enters into the state "SyncReact" in which it sends its data to Robot1 (−Data), and changes its state to "DataWaiting." Once Robot1 receives Robot2's data (+Data), it enters into "DataSending," immediately sends its data to Robot2 (−Data), and enters into the "Update" state. Accordingly, Robot2 receives Robot1's data (+Data) and moves to the state "Update." In the "Update" state, the robots update their estimation by using the data from each other. Once the update is done (Complete), the state is set back to "IndivSLAM."

Figure 6.2 depicts communication between the two robots and the timings of transmission and reception along their states. They send data interchangeably and each transmission affects the other robot's state transition. A synchronization packet plays an important role to avoid collisions of requests from multiple robots and make sure that data packets are sent only when a pair of robots establish their interactions. Since the system is decentralized, there is a chance that a robot receives

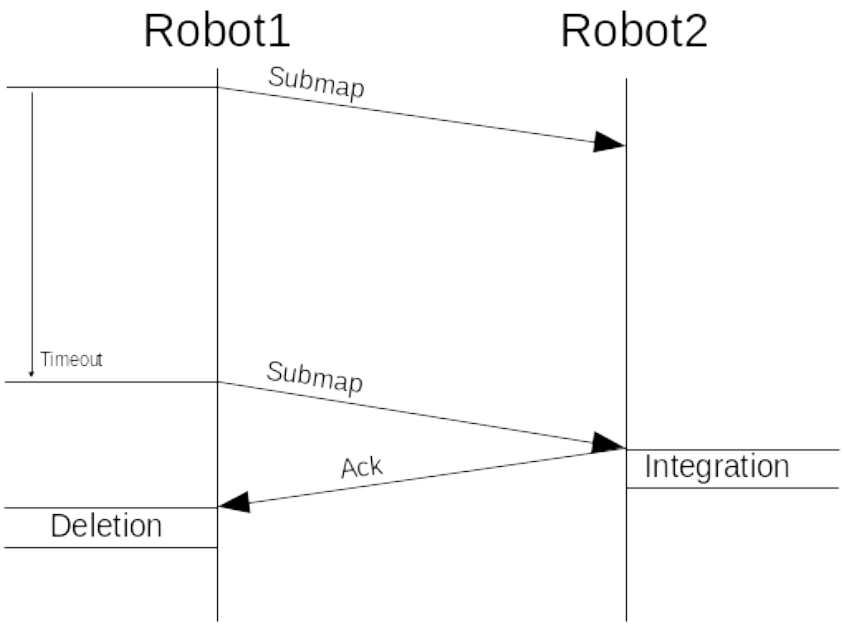

Figure 6.4: Protocol diagram of the submap transfer. Robot1 is sending its submap to Robot2. If it does not receive back a confirmation packet, it re-send the submap. Once Robot2 receives the submap, it sends back a confirmation packet (Ack) to Robot1 and integrates the submap into its SLAM process. After receiving the confirmation, Robot1 deletes the corresponding submap.

multiple packets from more than one neighbors. Robot1 does not know Robot2 is in "IndivSLAM" when it sends a synchronization packet. Robot2 may be in another state to interact with other robots and in such a case, it will not be able to respond to Robot1. Once a certain length of time passes, Robot1 will abort their requests without sending their data packets and attempt another request later. Since the data size of a synchronization packet is supposed to be significantly smaller than that of a data packet, the entire size of wasted data is expected to be small. Thus, a pair of robots synchronously exchange their data without a centralized control.

Each robot is designed to perform both roles; it can initiate an interaction with its neighbor and also react to the other's request. Figure 6.3 shows the entire state machine. "Init" is the initialization state in which the robot initializes its settings at the beginning of its operation. It enters "IndivSLAM" once it starts its individual SLAM process. Then, for interactions with other robots, it moves on either "SyncInit" or "SyncReact," depending on whether they are sending a request or reacting to the other's request.

6.1.2 Distributed Submap Building: Submap Transfer

The interactions for the submap transfer is simpler than the interactions for the data exchange. The reason is that the robots do not need to interrupt their individual SLAM process or wait

for the other robot's action to finish each step. To depict it, the sender and receiver's tasks are individually described as follows. Here, the robot that sends its submap is named as "Sender" and the one receiving the submap is as "Receiver."

Sender: Once it detects that its neighbor is entering an area covered by its submap, the sender robot simply passes the submap to the neighbor. The sent submap is then marked as a map to delete. As the submap is marked, it is not used for the entry detection so that it will not be sent multiple times in a short time.

Receiver: Once the submap is received, the receiver robot integrates the submap into its SLAM process. Once the robot finished the integration of the submap, it sends confirmation to the sender.

Sender: When the sender robot receives the confirmation from the receiver, it deletes the marked submap and the whole process is complete. If the confirmation is not received after a predetermined time, the sender robot unmarks the submap, assuming the receiver could not receive the submap, and it will start over the submap transfer process.

Figure 6.4 depicts the interaction described above. Robot1 is the sender and Robot2 is the receiver. It shows that Robot1 sent the submap to Robot2 but it did not receive the confirma-

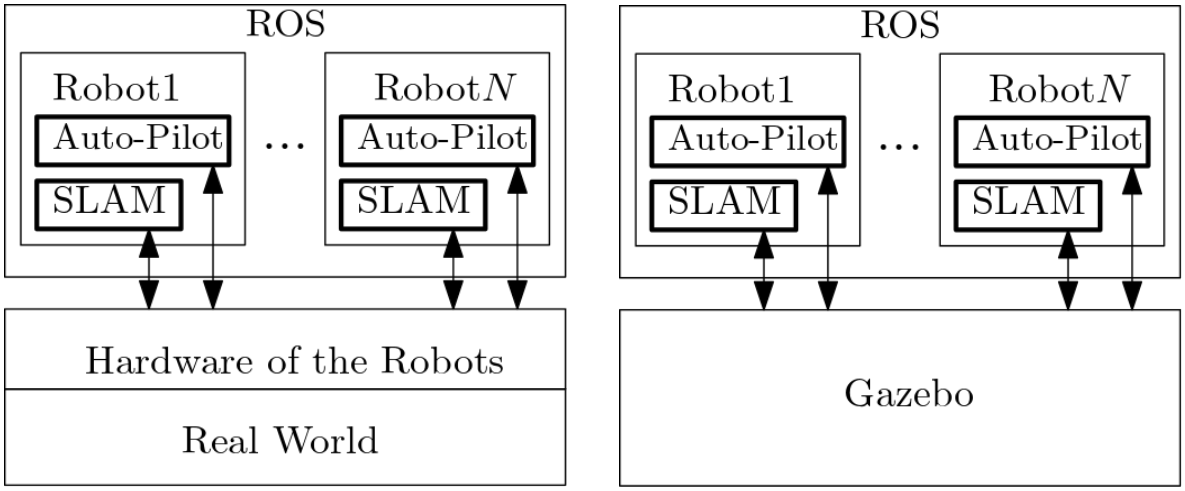

Figure 6.5: Overview of the ROS system in the cases of the real world (left) and the simulation (right). The rectangles with thick outlines represent ROS nodes: the programs running in the framework of ROS. While each robot holds several ROS nodes, here, only the two major ROS nodes "Auto-Pilot" and "SLAM" are shown for simplicity. In the real world (left), the ROS nodes interact with the actual hardware of the robots which work in the environment. On the other hand, in the case of simulation (right), the hardware and the environment are simulated by the Gazebo simulator, which interacts with the ROS nodes.

tion from Robot2 within the specified time period. After the timeout, Robot1 attempts to send its submap again. When Robot2 receives the submap, it sends the confirmation to Robot1 and integrates the submap into its SLAM process. Receiving the confirmation from Robot2, Robot1 then deletes the corresponding submap. These tasks of "Integration" and "Deletion" are done in the state of "IndivSLAM." Unlike those for the cooperative localization, the interactions for the submap transfer do not require to interrupt their SLAM processes to wait for the action from other robots. Therefore, when it enters in the state of "IndivSLAM," each robot asynchronously checks the received submap and processes it.

6.2 System Components

Figure 6.5 shows an overview of the ROS system in the cases of the real world (left) and the simulation (right). In both cases, the programs run as ROS nodes running in the framework of ROS. A set of ROS nodes consist of a system for an individual robot. For example, the "SLAM" node processes tasks for both the decentralized cooperative localization and the distributed submap building. The "Auto-Pilot" node performs as a reactive agent, which decides the robot's direction to move, reacting to the gained ranging sensory data. These ROS nodes do not directly communicate with those for another robot.

In the real world, the ROS nodes interact with the hardware of each corresponding robot through the ROS system. The hardware provides sensory data to the ROS nodes and also accepts control inputs from them. The communication between the robots is also performed by the ROS nodes interacting with the hardware. In this study, we conduct simulations by using the Gazebo simulator. In the Gazebo simulator, the kinematics and dynamics of the environment and the hardware of the robots are simulated. The simulator interacts with the ROS nodes by providing

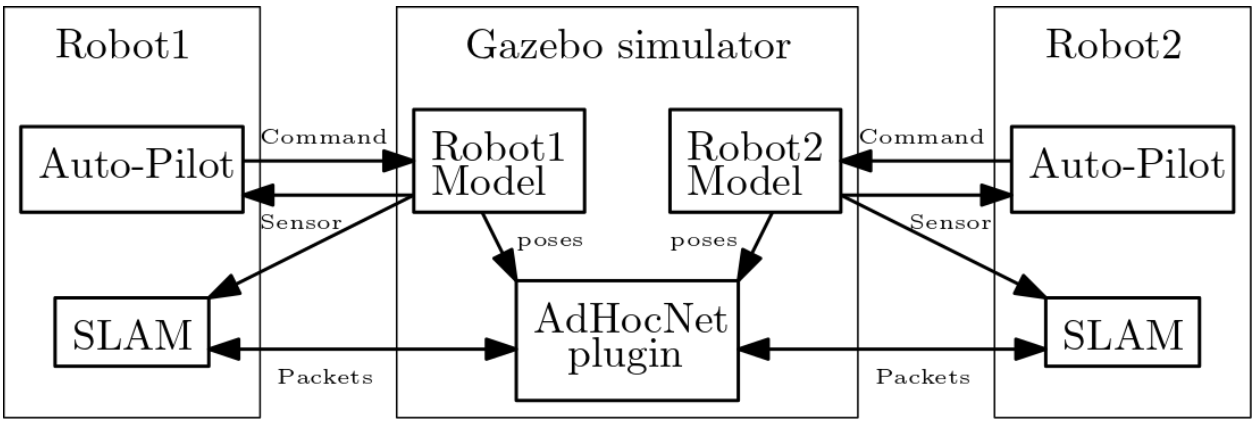

Figure 6.6: Components in the simulation with two robots communicating with each other. Gazebo simulator updates the robot's hardware models and runs the "AdHocNet" plugin which handles local communication. Arrows represent data passed between the components.

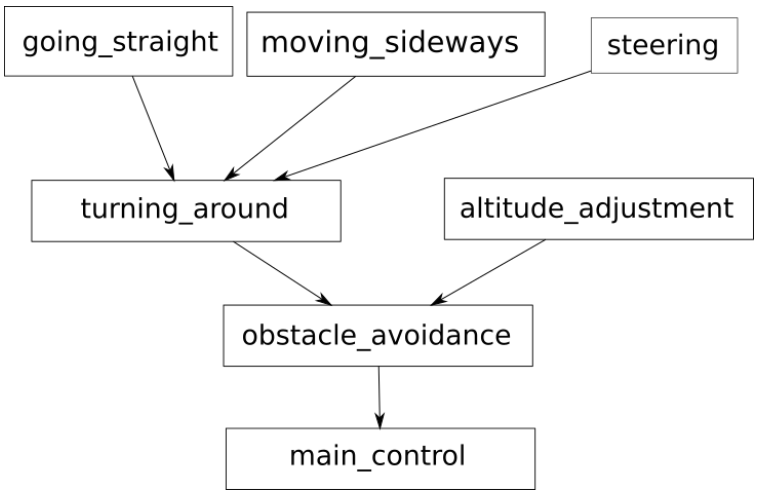

Figure 6.7: Reactive agent control. The control node is composed of sub-parts, each of which generates basic behaviors reacting to the gained sensory data. The arrows show the flow of control data. Receiving control data from the upper parts, each part overrides the received data or integrates them with its own control data and gives the output to the lower parts. The "main_control" part (at the bottom) finally generates the integrated control output for the robot.

data from the simulated hardware to the ROS nodes and also accepting data from the ROS nodes. The simulator allows interactions between the robots only when they are in their communication range and there is no obstacle hindering their communication. In this structure of simulation, the robots interact with each other as if each robot's ROS nodes run on a separate machine.

Therefore, the simulator mediates all the inter-robot communication and interactions. Figure 6.6 shows details about the components of the simulator and ROS nodes for two robots interacting with each other. The Gazebo simulator contains the robot hardware models; the physics and dynamics of each roobot's hardware is simulated by these models. In addition, as inter-robot communication is performed in this study, we designed a plugin "AdHocNet" to simulate the communication. By checking the robot hardware models and the environment structure, it allows data to be passed if the robots are in the communication range and there is no obstacle between them. The plugin also allows the robots to perform global localization when they are close to the origin, assuming they can interact with the base station or lander while they are close to it. The "Auto-Pilot" node takes the sensory data from the simulator and gives commands to move the robot model. The "SLAM" node takes sensory data for the individual SLAM. It also exchanges packets with the "AdHocNet" plugin and establish an ad hoc network for the decentralized cooperative localization and the distributed submap building.

As this study focuses more on the SLAM process and inter-robot communication for those tasks, the robot's control is designed to be simple; the robots are supposed to follow the wall while keeping the distance to their neighbors. For this purpose, the "Auto-Pilot" node is designed based on a

simple reactive agent. Figure 6.7 is a diagram of the reactive agent running in the "Auto-Pilot" node to control the robot to follow the wall. It is composed of several parts, each of which generates basic behaviors. Each part follows the following steps.

- Receive sensory data from the sensors and control data from the upper nodes.
- Generate its control data, reacting to the sensory data.
- Decide whether to override the received control data or integrate them with its control data.
- Release the resulting control data to the lower node.

Finally, a desired behavior emerges from the integrated control data. To deploy multiple robots while they maintain distances between them, some components takes data about distances to the neighboring robots. The "going_straight" determines the forward velocity based on the distance to the neighbor behind the robot. If the neighbor is approaching close to the robot, it moves

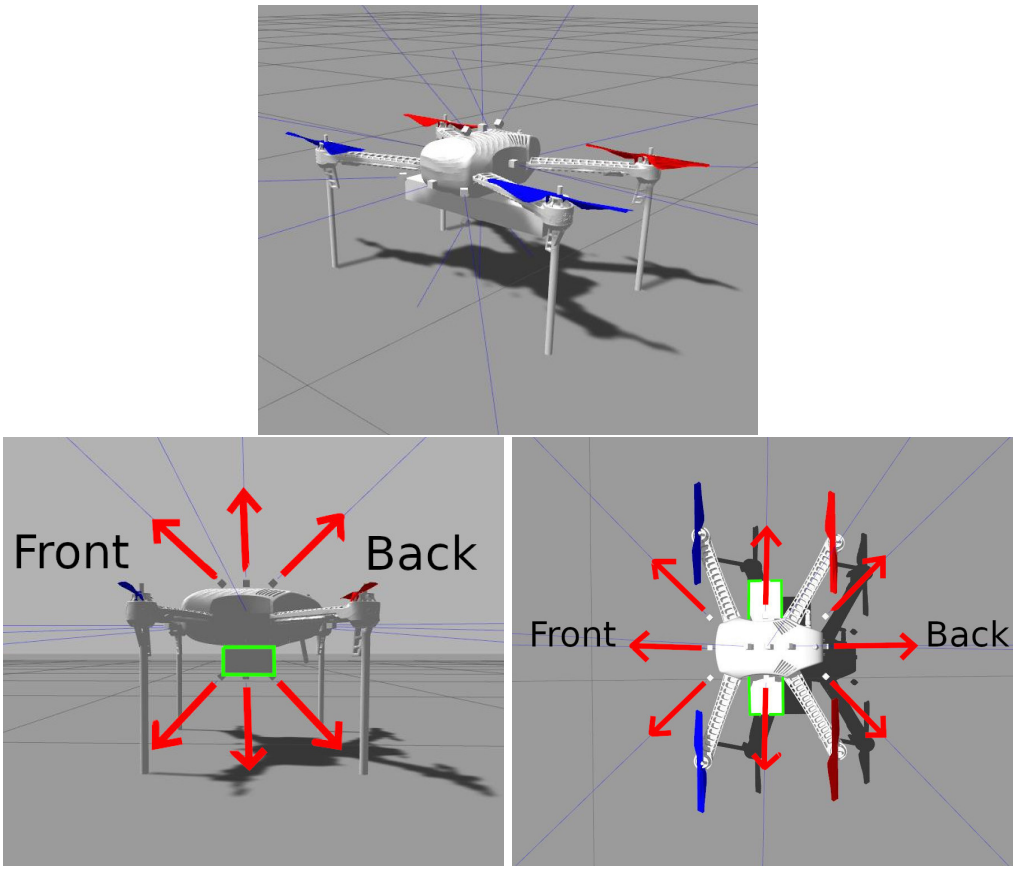

Figure 6.8: Drone model with sensors. The bottom left and bottom right pictures are views from the side and the top, respectively. The red arrows represent the beams of the ranging sensors. The rectangle box highlighted by green lines represents the depth camera.

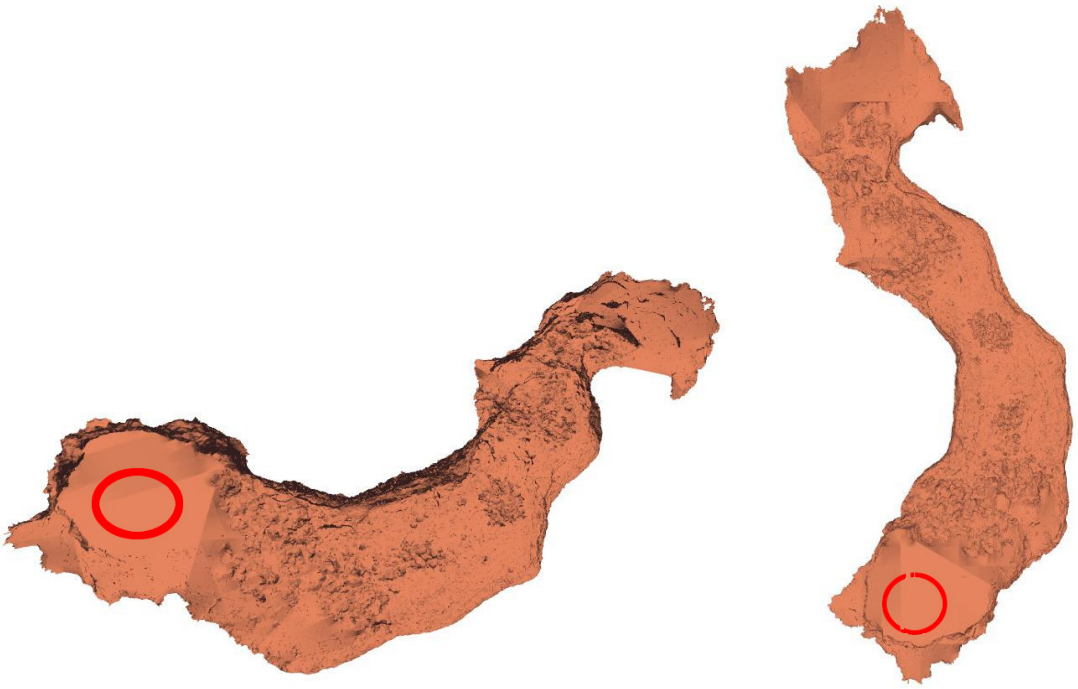

Figure 6.9: The meshed model of the simulation environment based on the dataset of the Indian Tunnel in Idaho [1]. The left image is the diagonal view of the model. The right image is the top view. The red circle represents the entry area, where the robots are deployed.

forward. Otherwise, it generates zero velocity so that the robot stays there. The "steering" and "moving_sideways" also take the neighbor's distance into consideration so that the robot can keep the distance from it. By keeping the distances from other robots and also from the obstacles, i.e., the walls, they eventually form a meshed formation to establish an ad hoc network where the robots can communicate with each other within a given communication range.

6.3 Robot and Environment Models

Figure 6.8 shows the robot model. The model is based on the Micro Aerial Vehicle (MAV) model provided in the package of rotorS [124]. For the model, a depth camera is mounted at the bottom of the body. The depth camera looks forward and generates point cloud data of the terrain in front of the robot, and the data is used for the SLAM process. It also has ranging sensors: three on the top pointing above, three on the bottom pointing below, and eight around the robot's body pointing horizontally. Each of them tells the distance to a detected obstacle, and the data is used for the control.

Figure 6.9 shows the simulation environment model where the robot models fly. The model is based on the dataset of the Indian Tunnel, located in Idaho [1]. As the point clouds in the dataset

is too dense to generate a meshed model, the data was down-sampled by the Poisson sampling code [125] and meshed up by Blender, the 3D modeling software, to generate this simulation model. The space is roughly 80 meters long and 20 meters wide. The ground is uneven and there are sporadic boulders. Data for both ends of the tunnel are missing and substituted with artificial surface.

6.4 Simulation Setups

This section describes how the simulation is conducted and how the simulation results are evaluated. First, the configuration of the simulation is described. The deployment of the robots and the resulting formation are explained, followed by the testing cases to be conducted in the simulation. Then, the evaluation metrics are defined to compare the testing cases.

6.4.1 Configuration

In Figure 6.9, the red circle represents the entry area. The base lander is assumed to be in this area, and the robots are deployed one by one. The timing of the deployment is determined based on the distance from the base lander to the previously deployed robot. Once the distance to the previously deployed robot from the lander exceeds a threshold, a new robot is deployed. This threshold is manually set based on the preliminary runs of the simulation. The control of the robot is configured to have the robots keep the distances to their neighbors about 10 to 15 meters so that they can form a meshed network in the simulation environment. Based on this interval, the threshold is set to 10.5 meters so that the robots can be seamlessly deployed. Every other robot is deployed facing in a different direction so that the robots can scatter in the space and eventually form a meshed network. Based on the characteristics of ZigBee, one of the low-power wireless communication technologies [126], the communication range is set to 30 meters for the communication between the robots and the base lander, while the communication is blocked when there is an obstacle such as a wall between them. While they are flying in a meshed formation, they perform the individual SLAM and the cooperative localization every one second and every 0.4 seconds, respectively. Once 10 robots are deployed and scattered in the environment, the simulation is terminated.

In this simulation, we integrate and demonstrate the proposed methods focusing on the extent to which the proposed methods handle the overconfidence and improve the mapping performance, compared to the naive method. The CI-based method and the centralized method are not considered for comparison. As the RBPF is a particle filter, the probability distributions are represented by particles. Hence, the CI-based and the centralized methods cannot be implemented due to the same reasons for the simulation of the cooperative localization by a particle filter described in Section 4.4.1. Therefore, the simulation is conducted in four cases: the naive method and the proposed method, with or without the distributed submap building for each method.

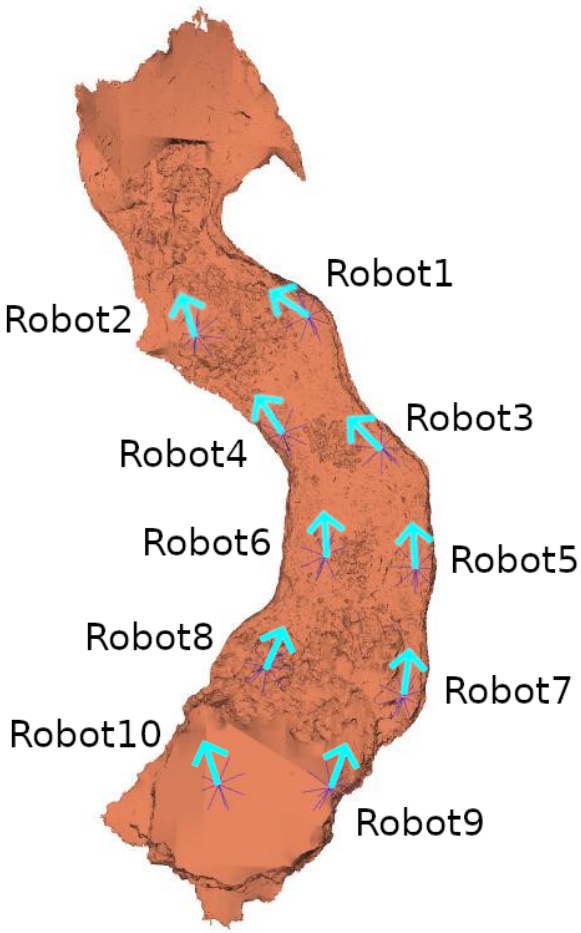

Figure 6.10: Deployment for the multi-robot navigation. The cyan arrows represent the robots moving in the environment. The multi-robot navigation deploys each of the ten robots one by one, and the robots move into the deeper area while keeping the distance from each other.

In all the cases, ten robots are deployed and form a meshed network as shown in Figure 6.10. In the first two cases, the robots perform the cooperative localization and they do not perform the distributed submap building. Instead, each robot builds a single map. On the other hand, in the last two cases, the robots perform the distributed submap building: segmenting their SLAM processes and transferring their submaps to their neighbors. The cases with and without the distributed submap building are evaluated with the naive method and the proposed method. Therefore, in the first and third cases, the robots perform the cooperative localization with the naive method, in which they update their estimate without reducing the confidence. On the other hand, in the second and fourth cases, the robots perform it with the proposed method, in which they deliberately reduce the confidence in their estimate.

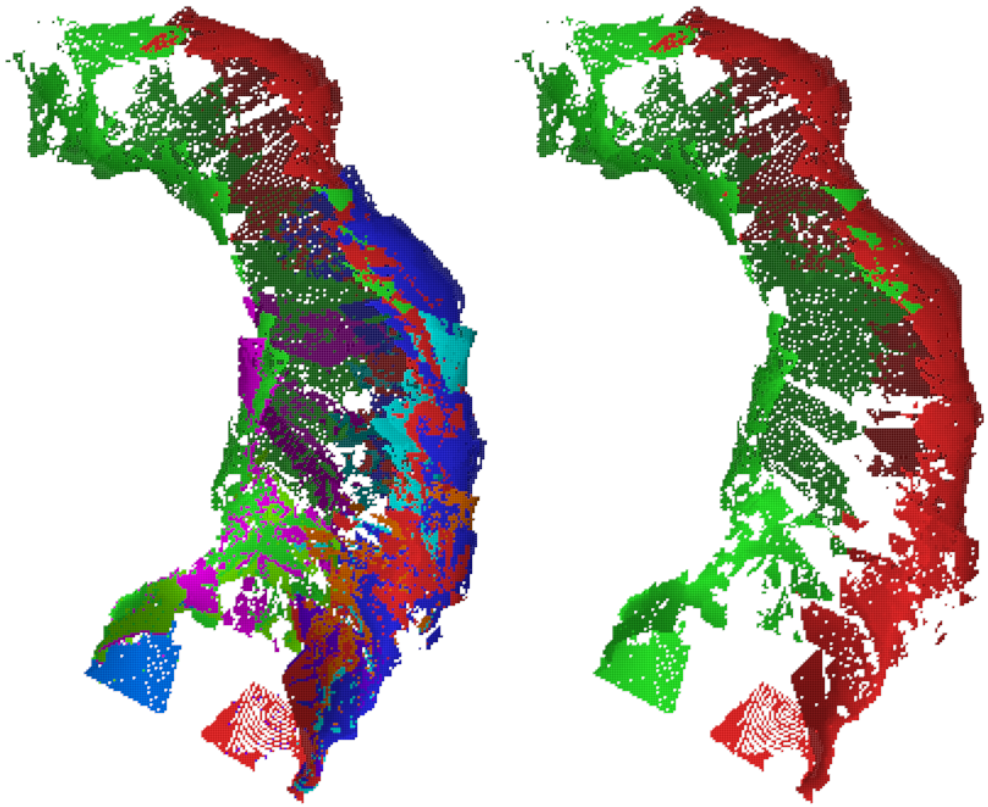

Figure 6.11: Generation of the entire map by the proposed method without the distributed submap building. In the left image, the maps from all the robots are rendered. Let's say the robots named as Robot1 through 10 in the order of deployment. The submaps are colored in red for Robot1, in green for Robot2, in blue for Robot3, in yellow for Robot4, in cyan for Robot5, in magenta for Robot6, in brown for Robot7, in light green for Robot8, in purple for Robot9, and in sky blue for Robot10. (Note some submaps are not visible as they are overlapped by the other submaps entirely.) For evaluation, the maps from Robot1 (Red) and Robot2 (Green) are selected to generate the entire map as shown in the right image.

6.4.2 Evaluation Metrics

The simulation results are evaluated based on the resulting maps. In the case without the distributed submap building, the maps from the first two robots (Robot1 and Robot2) are used to generate the map for evaluation since these two robots provide the largest maps covering the submaps of the other robots. As an example, Figure 6.11 shows the generation of the map for the proposed method. In the case with the distributed submap building, the maps from all the robots are used to generate the map to evaluate as each robot holds submaps about its local area. For example, Figure 6.12 shows the resulting map for the proposed method. These generated maps

Figure 6.12: Generation of the entire map by the proposed method with the distributed submap building. Let's say the robots named as Robot1 through 10 in the order of deployment. The submaps are colored in red for Robot1, in green for Robot2, in blue for Robot3, in yellow for Robot4, in cyan for Robot5, in magenta for Robot6, in brown for Robot7, in light green for Robot8, in purple for Robot9, and in sky blue for Robot10. As each robot holds submaps about its local area, an entire map is generated by combining the submaps from all the robots.

are compared against the ground truth map shown in Figure 6.13. It was generated through a simulation in which a robot was manually controlled to fly through the environment and performed mapping based on the ground truth locations.

These resulting maps then need to be converted into a specific format. The maps are originally occupancy maps but, for comparison, they are converted into point clouds. As shown in Figure 6.14, occupied cells in an occupancy map are converted into points so that the resulting points can represent the occupied area. This conversion is processed in a C++ program [127], and the resulting point clouds are generated by the Point Clouds Library (PCL) [128]. The ground truth map is also converted into point clouds through the same process. A set of point clouds from each of the resulting maps are evaluated against the ground truth point clouds by performing the could-

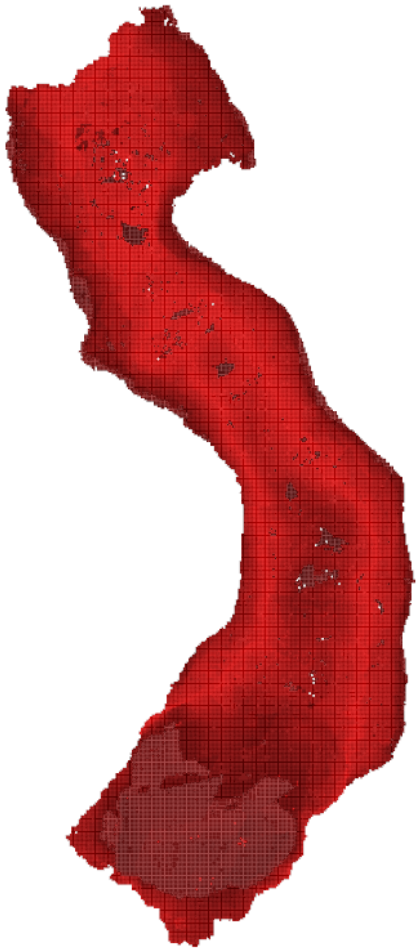

Figure 6.13: The ground truth map generated by mapping based on the ground truth trajectory of the robot which was manually controlled.

to-cloud comparison using the open source software CloudCompare [129]. Figure 6.15 shows the cloud-to-cloud comparison of two sets of point clouds. The red points are from the map to evaluate, and the blue points are from the ground truth map. From each point of the map to evaluate, the distance to the closest point in the ground truth data is calculated as the error of the point. Finally, the average error is calculated as the metric of evaluation.

6.5 Simulation Results

Figures 6.16 and 6.17 show the results of cloud-to-cloud comparison of the point clouds from each simulation case against the ground truth data. The entry area of the environment is represented at the bottom and the deep area is at the top. The color represents the amount of error calculated by the comparison. The value increases as the color changes from blue toward red. When it exceeds

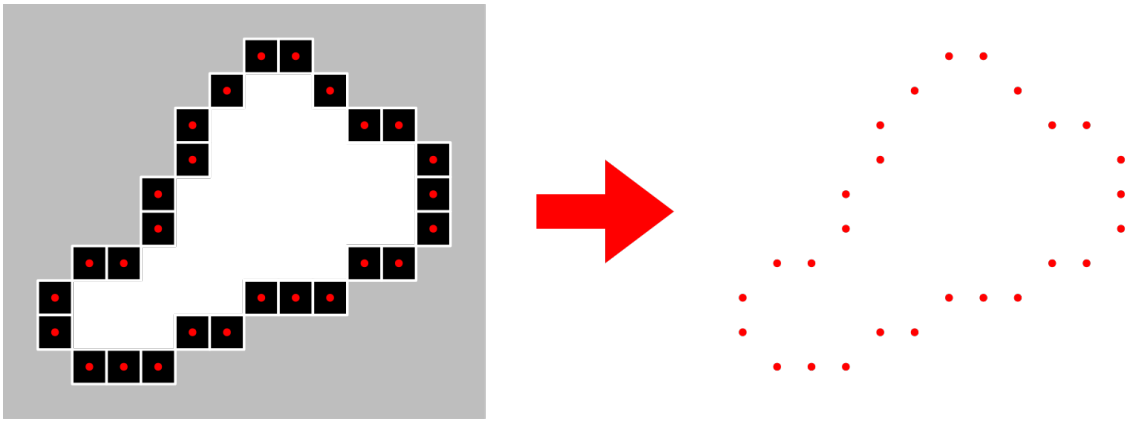

Figure 6.14: Conversion of occupancy map data to point clouds. On the left side, occupancy map is depicted with open cells (white), occupied cells (black), and unknown cells (gray). The coordinates of the occupied cells (red dots) are extracted as point clouds, depicted on the right side.

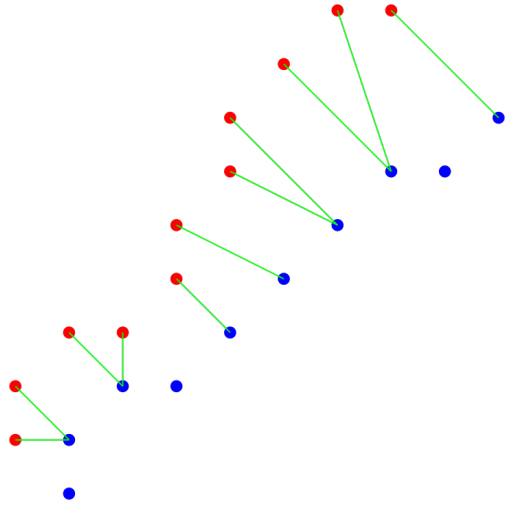

Figure 6.15: Comparison of the target point clouds to evaluate (red dots) against the reference point clouds from the ground truth map (blue dots). Each point in the target point clouds is checked in terms of the distance to the closest point in the reference point clouds (green line).

two meters, it is colored in pure red.

The results of the cases without the distributed submap building are shown in Figure 6.16. The left and right images show the result from the naive method and the proposed method, respectively. As the naive method generated a larger green and red areas than the proposed method did, the results indicate that the proposed method performs the mapping process more accurately. Similarly, the proposed method with the distributed submap building (right image in Figure 6.17) generates more accurate map than the naive method (left image in Figure 6.17).

Table 6.1 shows the average distance value for each case. In the case without the distributed

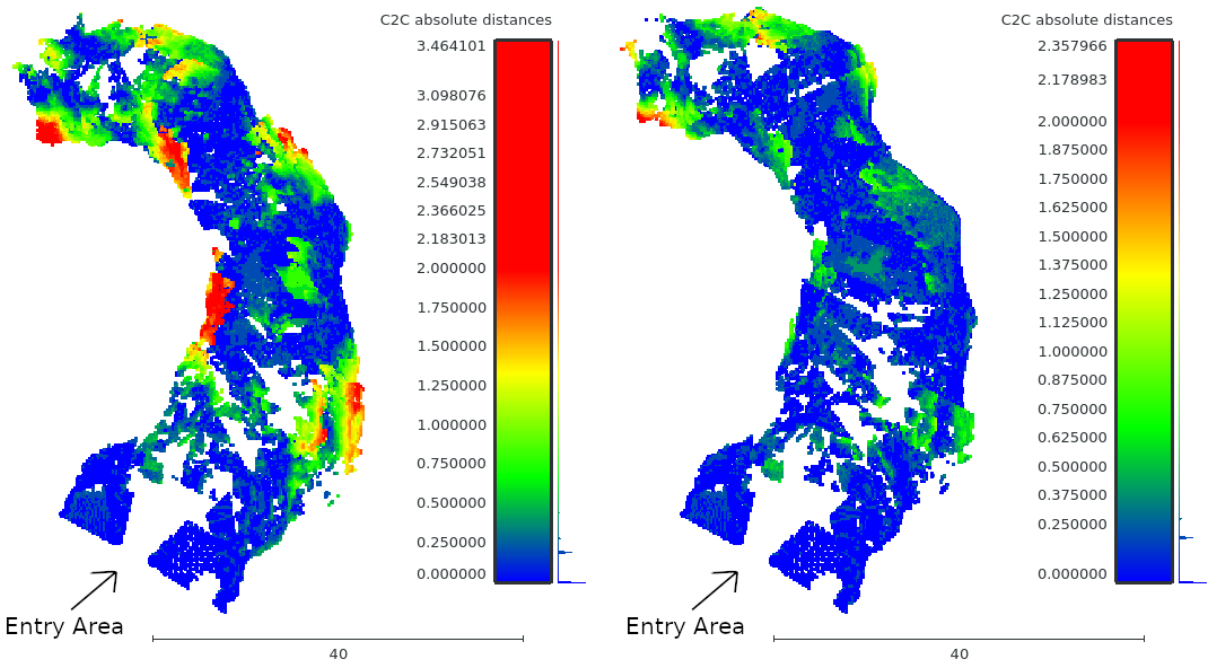

Figure 6.16: Cloud-to-cloud comparison for the case without the distributed submap building, using the naive method (left) and the proposed method (right). The entry area of the environment is at the bottom and the robot moved from the bottom toward the top. The color of points represents the distances ranging from 0 meter (blue) to 2+ meters (red).

submap building, the proposed method resulted in significantly smaller average error than the naive method, indicating that the proposed method can handle the overconfidence problem and improve the mapping performance. Similarly, the proposed method outperforms the naive method in the case with the distributed submap building as well. It is also notable that, for each method, the accuracy does not change significantly whether or not the distributed mapping is applied; the differences of the errors between the cases are just a few centimeters or less. This indicates that the distributed submap building does not negatively affect the mapping performance.

In addition, the distributed submap building was evaluated by comparing the data size of the

Case		Average Error (meters)
w/o Distributed Submap Building	Naive	0.388181
	Proposed	0.223311
w/ Distributed Submap Building	Naive	0.363706
	Proposed	0.225828

Table 6.1: Average errors in the point clouds generated in the simulation cases, measured by the distance between the generated points and the ground truth points.

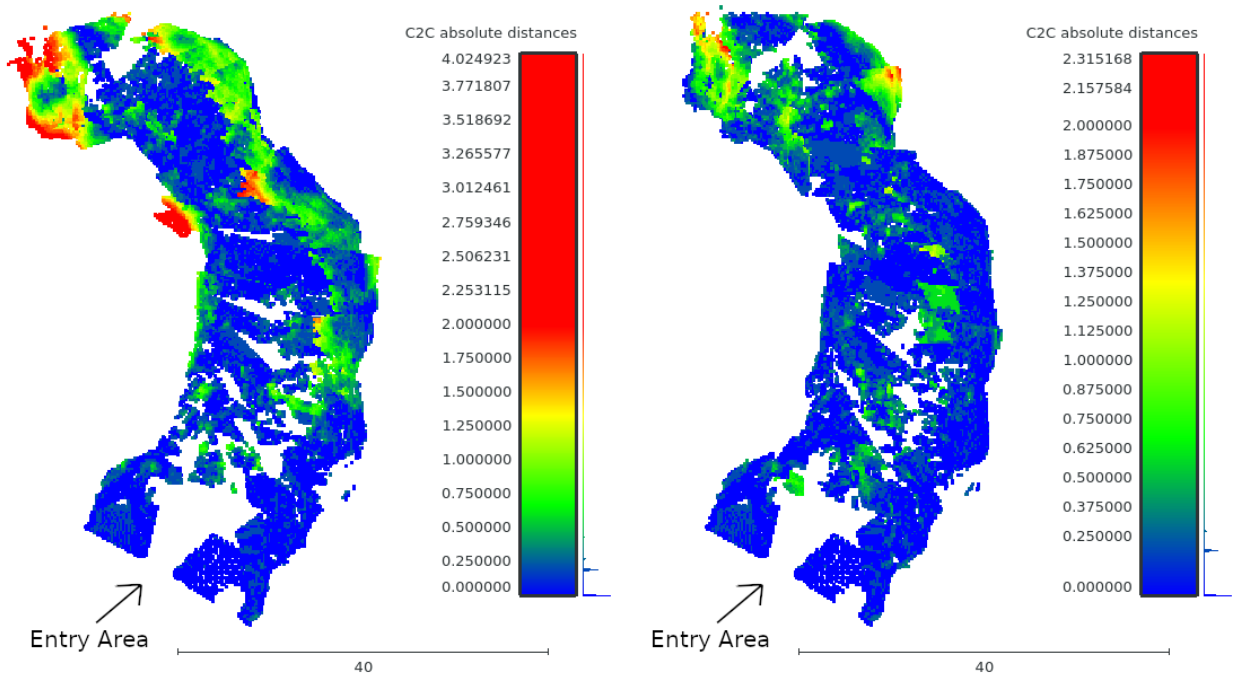

Figure 6.17: Cloud-to-cloud comparison for the case with the distributed submap building, using the naive method (left) and the proposed method (right). The entry area of the environment is at the bottom and the robot moved from the bottom toward the top. The color of points represents the distances ranging from 0 meter (blue) to 2+ meters (red).

generated maps. The data size is calculated by the utility function provided by the octomap library [99], which uses the number of nodes in the octree data structure as follows.

$$\text{Data Size of a Map} = (\#\text{ of Internal Nodes} \times X) + (\#\text{ of Leaf Nodes} \times Y) \tag{6.1}$$

where X is the data size of an internal node and Y is the data size of a leaf node. In the cases with the distributed submap building, the robots may hold more than one submap. Hence, the data size is calculated as the summation of the data size of the submaps held by the robot.

Figure 6.18 shows the data size of the maps generated by each robot. In the case without the

Case		Average (MiB)	Maximum (MiB)
w/o Distributed Submap Building	Naive	1.736	3.44
	Proposed	1.743	3.44
w/ Distributed Submap Building	Naive	0.847	1.51
	Proposed	0.851	1.31

Table 6.2: The average and maximum data sizes of the mapping data in MiB.

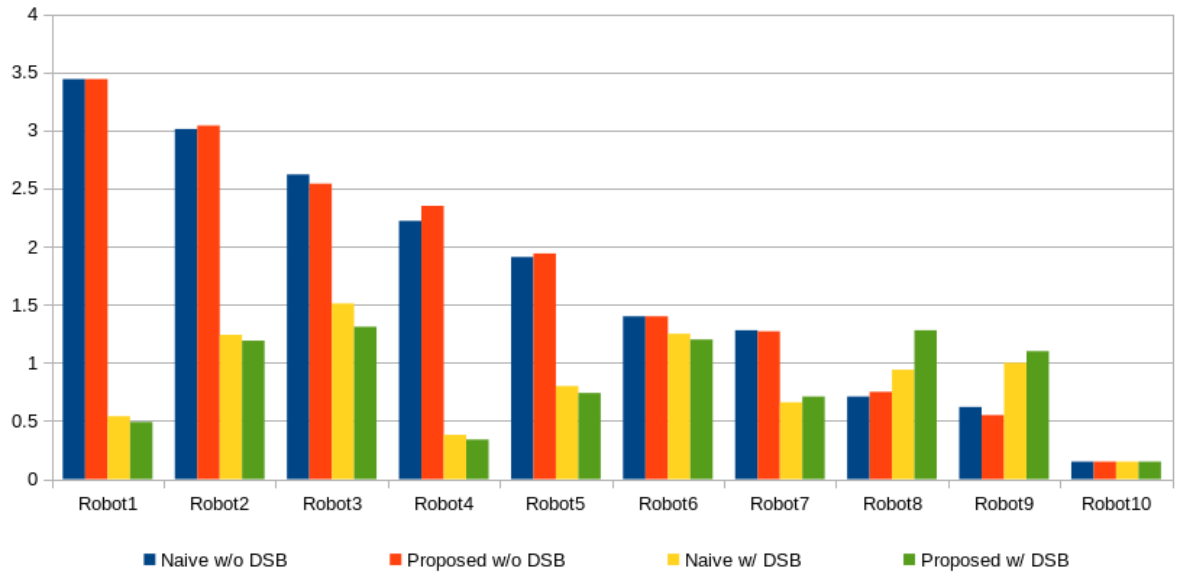

Figure 6.18: Comparison of data size in MiB of the generated maps by the naive and proposed methods in the cases with and without the distributed submap building (DSB). As a robot may hold more than one submap in the case with the distributed submap building, it is calculated as the summation of the data size of the submaps that the robot holds.

distributed submap building, the robots that were deployed earlier hold larger sized data than those newly deployed since the data size increases as the robots explore more areas of the environment. On the other hand, when the map data is distributed as submaps, the data size becomes smaller. Some robots have larger data size than others since they hold more submaps. Table 6.2 shows the average and the maximum data size held by the robots in each case. Without the distributed submap building, the size of the mapping data tends to be larger. For example, the size of the data that Robot1 holds exceeded 3 MiB. Hence, the average data size becomes larger. On the other hand, when the robots performed the distributed submap building, the average data size is about 0.85 MiB and the maximum size is about 1.3 to 1.5 MiB. Obviously, each robot only needs to hold a few submaps so the memory space can be saved. Thus, the distributed submap building can significantly improve the memory efficiency.

In this simulation, the communication bandwidth was assumed to be large enough for the robots to perform the distributed submap building. In the real world, the distributed submap building may cause a trade-off with communication as the robots need to transmit the mapping data, which is relatively larger than the localization data for the cooperative localization. When the submaps were losslessly compressed and saved into files, the file size of a single submap was about 10 KiB. Even if the robot transmits the submaps of all the particles without subsampling, the total data to send to the neighbor will be several hundreds KiB. Thus, if the bandwidth of communication

covers the required data size to transmit, the distributed submap building is expected to work in practice.

CHAPTER 7
CONCLUSIONS

This dissertation work addresses the problems arising in robotic exploration of a remote, unstructured, and enclosed environment such as a Martian lava tube. In this type of environment, multi-robot SLAM with occupancy grid-based mapping is suitable as the robots can cooperatively explore a large area of the environment that rarely has landmarks to detect. Nonetheless, due to the lack of external supports such as GPS and access points to the ideal network, each robot can only sense and access its local area. This locality of sensing and communication for each robot prevents the existing techniques from working in this type of environment.

Hence, this work presents a set of multi-robot SLAM techniques: mainly, the decentralized cooperative localization and the distributed submap building. These techniques enable the robots to cooperatively localize themselves and build occupancy gird-based maps about the environment without relying on external or centralized resources.

7.1 Contributions

The presented work enables the multi-robot SLAM in the aforementioned challenging situation, resulting in the following contributions.

Decentralization of Cooperative Localization: The decentralization of cooperative localization allows the robots to estimate their locations in the enclosed environment without relying on any external supports. Each robot only needs to interact with its neighbors as this approach is based on the local interactions: measurements on the relative poses and exchanging the estimates of their locations.

Conservative Data Exchange: The conservative data exchange enables the robots to avoid overconfidence in their estimation while performing decentralized cooperative localization. The proposed method enables each robot to handle its estimation without knowing the correlations between the robots' estimation in the entire system.

By conducting 2D simulations and developing mathematical models of the data exchange, we showed that the proposed method avoids overconfidence and outperforms the naive and CI-based methods in the localization performance.

Integration with Mapping: The non-parametric version of the decentralized cooperative localization enables the robots to perform occupancy grid-based mapping with the decentralized cooperative localization. This technique allows the proposed method to perform as a part of SLAM in an unstructured environment with few landmarks to detect.

Distribution of Submap Building Tasks: The methods for the distributed submap building, which are segmentation of a map into submaps and the submap transfer, enable each robot to perform mapping in the local area where the robot is situated. By these methods, the robots independently produce and pass their submaps to the other robots. Hence, each robot can process a smaller mapping area than the mapping of the entire environment, and it can reduce the memory space of individual robots for the mapping process. Moreover, the experimental results show that the distributed submap building does not degrade the mapping performance.

Implementation of ROS Nodes: The ROS implementation enables the proposed methods and techniques to work with the actual robotics middleware. As they are implemented as ROS nodes, the proposed methods can run with the ROS middleware, which can run on the actual robot's hardware and system. The 3D robotics simulation by the Gazebo simulator shows that the ROS nodes can perform the proposed techniques and methods in the cavern environment modeled based on an actual lava tube.

7.2 Future Work

Although the proposed methods have been implemented in the actual robotic system of the ROS and demonstrated in a 3D simulation by using the Gazebo simulator, some characteristics of the hardware and the environment were simplified or assumed to be ideal. To apply the proposed methods to actual robotic hardware in the real world situation, more realistic characteristics and system designs will need to be considered.

Therefore, an immediate next step of this study will be to make the design of the hardware and physical characteristics of the robots in the 3D simulation reflect the real world more closely. For example, instead of the generic odometry with noises, the visual odometry can be implemented with cameras observing the texture of the terrains of the simulated environment. Additional physical restrictions such as the battery limits can also be considered in the simulation. Currently, the navigation and autonomy of the robots in the 3D simulation are controlled to continuously fly and hover in the environment. However, because the battery life is limited in reality, the robots have limited flight time in practice. It can be incorporated into the simulation by designing the robot control that allows the robots to temporarily land on the ground to reduce the flight time and save the battery power.

In addition, the variations of the environment structure will also need to be considered. The environment model used in the simulation was a tunnel-like cave with a width of approximately 20 meters and gradual slopes. Since this study focused on SLAM and state estimation, the motion control was simplified and designed based on this environment structure; the robots were controlled to keep the same distance from the other robots, forming a meshed formation mainly horizontally.

This motion control may not work correctly if the environment is different in size and has significant structural changes vertically, such as steep slopes and vertical pathways. For the general case of the environment structure, the motion control will need to be improved to optimize the distance from their neighbors and form a three-dimensional meshed formation. Because the improved motion control will not rely on structural assumptions, it will also be applicable in fundamentally different environments, such as apartment buildings.

The ultimate goal of future work will be to demonstrate the proposed methods in a real environment. A set of ROS nodes for each robot will run on each robot and interact with others through wireless communication. While the protocols for the actual interactions have been designed and demonstrated in the simulation, unpredicted issues might be identified once the methods are implemented and deployed in the actual world situation. In addition, it will be practically desirable to switch to ROS2 from the original ROS or ROS1. ROS1 requires a centralized core process to serve the middleware functionality. It did not matter for the simulation in this work since the robot's communication was deliberately limited to the interactions through the simulator. However, it would not be reasonable to use ROS1 in the real environment. As ROS2 allows the robots to run their processes in a purely decentralized manner, it will be preferable for the actual hardware implementation.

BIBLIOGRAPHY

[1] NASA Planetary Pits and Caves Analog Dataset. https://ti.arc.nasa.gov/dataset/caves/.

[2] J. E. Bleacher, R. Greeley, D. A. Williams, S. C. Werner, E. Hauber, and G. Neukum. Olympus Mons, Mars: Inferred Changes in Late Amazonian Aged Effusive Activity from Lava Flow Mapping of Mars Express High Resolution Stereo Camera Data. *Journal of Geophysical Research: Solid Earth*, 112(4), 2007.

[3] G. E. Cushing, T. N. Titus, J. J. Wynne, and P. R. Christensen. THEMIS Observes Possible Cave Skylights on Mars. *Geophysical Research Letters*, 34, 2007.

[4] G. E. Cushing and T. N. Titus. Caves on Mars: Candidate Sites for Astrobiological Exploration. In *Astrobiology Science Conference 2010: Evolution and Life: Surviving Catastrophes and Extremes on Earth and Beyond*, volume 1538 of *LPI Contributions*, page 5414, April 2010.

[5] G. E. Cushing. Candidate Cave Entrances on Mars. *Journal of Cave and Karst Studies*, 74(1):33–47, 2012.

[6] K. E. Williams, C. P. McKay, O. B. Toon, and J. W. Head. Do Ice Caves Exist on Mars? *ıcarus*, 209:358–368, October 2010.

[7] J. Zhao, J. Huang, M. D. Kraft, L. Xiao, and Y. Jiang. Ridge-Like Lava Tube Systems in Southeast Tharsis, Mars. *Geomorphology*, 295:831–839, October 2017.

[8] N. Fairfield. *Localization, Mapping, and Planning in 3D Environments*. PhD thesis, Carnegie Mellon University, 2009.

[9] C. Stachniss. *Robotic Mapping and Exploration*, volume 55. Springer, 2009.

[10] A. Bachrach, A. D. Winter, R. He, G. Hemann, S. Prentice, and N. Roy. RANGE - Robust Autonomous Navigation in GPS-denied Environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 1096–1097, May 2010.

[11] S. Kohlbrecher, O. V. Stryk, J. Meyer, and U. Klingauf. A Flexible and Scalable SLAM System With Full 3D Motion Estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, November 2011.

[12] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue. *IEEE Robotics and Automation Magazine*, 19(3):46–56, 2012.

[13] Y. Liu and G. Nejat. Robotic Urban Search and Rescue: A Survey from the Control Perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, November 2013.

[14] D. Bi, Y. Yu, Y. Shen, and M. Claudio. Implementation of Trapped Personnel Detection and Evacuation Guidance in Indoor Fire Scene Based on Quadrotor UAV. In *2016 International Conference on Intelligent Control and Computer Application (ICCA 2016)*. Atlantis Press, January 2016.

[15] Y. Bi, H. Qin, M. Shan, J. Li, W. Liu, M. Lan, and B. M. Chen. An Autonomous Quadrotor for Indoor Exploration With Laser Scanner and Depth Camera. In *2016 12th IEEE International Conference on Control and Automation (ICCA)*, pages 50–55, June 2016.

[16] J. Martínez-Carranza, R. Bostock, S. Willcox, I. Cowling, and W. Mayol-Cuevas. Indoor MAV Auto-Retrieval Using Fast 6D Relocalisation. *Advanced Robotics*, 30(2):119–130, January 2016.

[17] F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero. Semi-Autonomous Teleoperation of UAVs in Search and Rescue Scenarios. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1066–1074, June 2017.

[18] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, and R. Siegwart. Aerial Service Robots for Visual Inspection of Thermal Power Plant Boiler Systems. In *Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference On*, pages 70–75. IEEE, 2012.

[19] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma. Emergency Response to the Nuclear Accident at the Fukushima Daiichi Nuclear Power Plants Using Mobile Rescue Robots: Emergency Response to the Fukushima Nuclear Accident Using Rescue Robots. *Journal of Field Robotics*, 30(1):44–63, January 2013.

[20] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart. A UAV System for Inspection of Industrial Facilities. In *Aerospace Conference, 2013 IEEE*, pages 1–8. IEEE, 2013.

[21] T. Özaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Inspection of Penstocks and Featureless Tunnel-Like Environments Using Micro UAVs. In *Field and Service Robotics*, pages 123–136. Springer, 2015.

[22] T. Özaslan, K. Mohta, J. Keller, Y. Mulgaonkar, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood. Towards Fully Autonomous Visual Inspection of Dark Featureless Dam Penstocks Using MAVs. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4998–5005, October 2016.

[23] M. Faria, I. Maza, and A. Viguria. Analysis of Data Structures and Exploration Techniques Applied to Large 3D Marine Structures Using UAS. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1277–1284, June 2017.

[24] V. R. Oberbeck, W. L. Quaide, and R. Greeley. On the Origin of Lunar Sinuous Rilles. *Modern Geology*, 1, 1969.

[25] F. Horz. Lava Tubes: Potential Shelters for Habitats. *Lunar bases and space activities of the 21st century*, pages 405–412, 1985.

[26] G. Heiken, D. Vaniman, and B. M. French. *Lunar Sourcebook: A User's Guide to the Moon*. CUP Archive, 1991.

[27] C. R. Coombs and B. R. Hawke. A Search for Intact Lava Tubes on the Moon: Possible Lunar Base Habitats. In W. W. Mendell, J. W. Alred, L. S. Bell, M. J. Cintala, T. M. Crabb, R. H. Durrett, B. R. Finney, H. A. Franklin, J. R. French, and J. S. Greenberg, editors, *Lunar Bases and Space Activities of the 21st Century*, September 1992.

[28] M. S. Robinson, S. M. Brylow, M. Tschimmel, D. Humm, S. J. Lawrence, P. C. Thomas, B. W. Denevi, E. Bowman-Cisneros, J. Zerr, M. A. Ravine, M. A. Caplinger, F. T. Ghaemi, J. A. Schaffner, M. C. Malin, P. Mahanti, A. Bartels, J. Anderson, T. N. Tran, E. M. Eliason, A. S. McEwen, E. Turtle, B. L. Jolliff, and H. Hiesinger. Lunar Reconnaissance Orbiter Camera (LROC) Instrument Overview. *Space Science Reviews*, 150(1):81–124, 2010.

[29] M. S. Robinson, J. W. Ashley, A. K. Boyd, R. V. Wagner, E. J. Speyerer, B. R. Hawke, H. Hiesinger, and C. H. V. D. Bogert. Confirmation of Sublunarean Voids and Thin Layering in Mare Deposits. *Planetary and Space Science*, 69(1):18 – 27, 2012.

[30] R. V. Wagner and M. S. Robinson. Distribution, Formation Mechanisms, and Significance of Lunar Pits. *Icarus*, 237:52 – 60, 2014.

[31] A. Daga, C. Allen, M. Battler, J. Burke, I. Crawford, R. Leveille, S. Simon, and L. Tan. Lunar and Martian Lava Tube Exploration as Part of an Overall Scientific Survey, 2009.

[32] R. J. Léveillé and S. Datta. Lava Tubes and Basaltic Caves as Astrobiological Targets on Earth and Mars: A Review. *Planetary and Space Science*, 58(4):592 – 598, 2010.

[33] S. B. Kesner, J. Plante, P. J. Boston, T. Fabian, and S. Dubowsky. Mobility and Power Feasibility of a Microbot Team System for Extraterrestrial Cave Exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4893–4898, 2007.

[34] J. Thangavelautham, M. S. Robinson, A. Taits, T. J. McKinney, S. Amidan, and A. Polak. Flying, Hopping Pit-bots for Cave and Lava Tube Exploration on the Moon and Mars. In *2nd International Workshop on Instrumentation for Planetary Missions*, Greenbelt, MD, 2014.

[35] S. Dubowsky, K. Iagnemma, S. Liberatore, D. Lambeth, J. Plante, and P. Boston. A Concept Mission: Microbots for Large-Scale Planetary Surface and Subsurface Exploration. *AIP Conference Proceedings*, 746:1449–1458, 2005.

[36] S. Dubowsky, S. Kesner, J. Plante, and P. Boston. Hopping Mobility Concept for Search and Rescue Robots. *The Industrial Robot*, 35(3):238–245, 2008.

[37] C. Lange and A. Seeni. Study on Strategies for Planetary Exploration Within the HG-project "Planetary Evolution and Life". In *IEEE Aerospace Conference Proceedings*, pages 1–11, 2011.

[38] A. Husain, H. Jones, B. Kannan, U. Wong, T. Pimentel, S. Tang, S. Daftry, S. Huber, and W. L. Whittaker. Mapping Planetary Caves With an Autonomous, Heterogeneous Robot Team. *IEEE Aerospace Conference Proceedings*, 2013.

[39] N. Ghafoor, E. Choi, G. Ravindran, J. Bakambu, T. Barfoot, R. Richards, and C. Sallaberger. Robotic Assistance, Mobility & Vision Systems – Enabling Technology for Early Human-robotic Lunar Exploration. In *58th International Astronautical Congress*, volume 2, Hyderabad, India, 2007.

[40] W. Whittaker. Technologies Enabling Exploration of Skylights , Lava Tubes and Caves. Technical Report NNX11AR42G, NASA, US, 2012.

[41] W. Fink, V. R. Baker, D. Schulze-Makuch, C. W. Hamilton, and M. A. Tarbell. Autonomous Exploration of Planetary Lava Tubes Using a Multi-Rover Framework. In *IEEE Aerospace Conference Proceedings*, volume 2015-June, pages 1–9, 2015.

[42] R. Whittaker. Exploration of Planetary Skylights and Tunnels. Technical report, Atrobotic Technology Inc./Carnegie Mellon University, Pittsburgh, 2014.

[43] H. Kalita, S. Morad, A. Ravindran, and J. Thangavelautham. Path Planning and Navigation Inside Off-World Lava Tubes and Caves. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1311–1318, April 2018. ISSN: 2153-3598.

[44] Drones Could Scope Out Martian Real Estate, December 2015. http://www.popsci.com/drones-to-scope-out-martian-real-estate.

[45] W. Tabib, M. Corah, N. Michael, and R. Whittaker. Computationally Efficient Information-Theoretic Exploration of Pits and Caves. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3722–3727, Daejeon, Korea, 2016.

[46] W. Tabib. *Approximate Continuous Belief Distributionsfor Exploration*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 2019.

[47] K. Snyder, E. Amoroso, A. Horchler, and F. Kitchell. AstroNav: Robust, High Rate SLAM for Planetary Exploration. In *16th ACM/IEEE International Conference on Information Processing in Sensor Networks · IPSN 2017*, April 2017.

[48] P. Lee, E. Kommedal, A. Horchler, E. Amoroso, K. Snyder, and A. F. Birgisson. Lofthellir Lava Tube Ice Cave, Iceland: Subsurface Micro-Glaciers, Rockfalls, Drone Lidar 3d-Mapping, and Implications for the Exploration of Potential Ice-Rich Lava Tubes on the Moon and Mars. In *Lunar and Planetary Science Conference*, volume 50, 2019.

[49] J. Falker, N. Zeitlin, R. Mueller, and M. Dupuis. Asteroid and Lava Tube In Situ Resource Utilization (ISRU) Prospecting Free Flyer Project. Technical report, NASA, 2016.

[50] J. Balaram and P. Tokumaru. Rotorcrafts for Mars Exploration. In *11th International Planetary Probe Workshop*, Pasadena, CA, 2014.

[51] J. Balaram, M. Aung, and M. P. Golombek. The Ingenuity Helicopter on the Perseverance Rover. *Space Science Reviews*, 217(4):56, May 2021.

[52] M. A. Batalin and G. S. Sukhatme. Efficient Exploration Without Localization. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 2714–2719 vol.2, September 2003.

[53] M. A. Batalin, G. S. Sukhatme, and M. Hattig. Mobile Robot Navigation Using a Sensor Network. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 1, pages 636–641 Vol.1, April 2004.

[54] M. A. Batalin and G. S. Sukhatme. Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. *Telecommunication Systems*, 26(2-4):181–196, June 2004.

[55] M. A. Batalin and G. S. Sukhatme. The Design and Analysis of an Efficient Local Algorithm for Coverage and Exploration Based on Sensor Network Deployment. *IEEE Transactions on Robotics*, 23(4):661–675, August 2007.

[56] A. Gasparri, B. Krishnamachari, and G. S. Sukhatme. A Framework for Multi-Robot Node Coverage in Sensor Networks. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):281–305, April 2008.

[57] J. Capitán Fernández, J. R. Martinez-de Dios, I. Maza, F. F. Ramon, and A. Ollero. Ten Years of Cooperation Between Mobile Robots and Sensor Networks. *International Journal of Advanced Robotic Systems*, 12(6):70, June 2015.

[58] J. A. Sauter, K. Bixler, S. Kitchen, and R. Chase. RF Emitter Localization With Robotic Swarms. In *Unmanned Systems Technology XXII*, volume 11425, pages 76–88. SPIE, April 2020.

[59] Y. Chen, L. Zhao, K. M. B. Lee, C. Yoo, S. Huang, and R. Fitch. Broadcast Your Weaknesses: Cooperative Active Pose-Graph SLAM for Multiple Robots. *IEEE Robotics and Automation Letters*, 5(2):2200–2207, April 2020. Conference Name: IEEE Robotics and Automation Letters.

[60] A. Cunningham and F. Dellaert. Large Scale Experimental Design for Decentralized SLAM. In *Unmanned Systems Technology XIV*, volume 8387. International Society for Optics and Photonics, 2012.

[61] E. Nettleton. *Decentralised Architectures for Tracking and Navigation With Multiple Flight Vehicles*. Thesis, University of Sydney, 2003.

[62] A. Makarenko and H. Durrant-Whyte. Decentralized Data Fusion and Control in Active Sensor Networks. In *Proceedings of the Seventh International Conference on Information Fusion*, volume 1, pages 479–486, 2004.

[63] A. Makarenko, A. Brooks, S. Williams, H. Durrant-Whyte, and B. Grocholsky. A Decentralized Architecture for Active Sensor Networks. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 2, pages 1097–1102 Vol.2, April 2004.

[64] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[65] B. Yamauchi. A Frontier-Based Approach for Autonomous Exploration. In *, 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings*, pages 146–151, July 1997.

[66] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated Multi-Robot Exploration Using a Segmentation of the Environment. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1160–1165, September 2008.

[67] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. The Sensor-based Random Graph Method for Cooperative Robot Exploration. *IEEE/ASME Transactions on Mechatronics*, 14(2):163–175, April 2009.

[68] C. Stachniss, G. Grisetti, and W. Burgard. Information Gain-Based Exploration Using Rao-Blackwellized Particle Filters. In *Robotics: Science and Systems*, volume 2, pages 65–72, 2005.

[69] S. J. Moorehead, R. Simmons, and W. L. Whittaker. Autonomous Exploration Using Multiple Sources of Information. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 3098–3103 vol.3, 2001.

[70] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information Based Adaptive Robotic Exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545 vol.1, 2002.

[71] C. Potthast and G. S. Sukhatme. A Probabilistic Framework for Next Best View Estimation in a Cluttered Environment. *Journal of Visual Communication and Image Representation*, 25(1):148–164, January 2014.

[72] N. Fairfield, D. Wettergreen, and G. Kantor. Segmented SLAM in Three-Dimensional Environments. *Journal of Field Robotics*, 27(1):85–103, 2010.

[73] E. Biagioni. Ubiquitous Interpersonal Communication Over Ad-hoc Networks and the Internet. In *2014 47th Hawaii International Conference on System Sciences*, pages 5144–5153, January 2014.

[74] T. Idota, E. Biagioni, and K. Binsted. Swarm Exploration of Extraterrestrial Lava Tubes With Ad-Hoc Communications. In *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pages 163–168, December 2018. ISSN: 2380-7636.

[75] J. Faigl, T. Krajník, J. Chudoba, L. Přeučil, and M. Saska. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *2013 IEEE International Conference on Robotics and Automation*, pages 993–998. IEEE, 2013.

[76] A. Franchi, G. Oriolo, and P. Stegagno. Mutual Localization in Multi-Robot Systems Using Anonymous Relative Measurements. *The International Journal of Robotics Research*, 32(11):1302–1322, 2013.

[77] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar. A Distributed Optimization Framework for Localization and Formation Control: Applications to Vision-Based Measurements. *IEEE Control Systems Magazine*, 36(4):22–44, 2016.

[78] R. Nandakumar, V. Iyer, and S. Gollakota. 3d Localization for Sub-Centimeter Sized Devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 108–119, 2018.

[79] S. Saeedi, M. Trentini, M. Seto, and H. Li. Multiple-Robot Simultaneous Localization and Mapping: A Review. *Journal of Field Robotics*, 33(1):3–46, 2016.

[80] L. Carlone, M. Kaouk Ng, J. Du, B. Bona, and M. Indri. Rao-Blackwellized Particle Filters Multi Robot SLAM With Unknown Initial Correspondences and Limited Communication. In *2010 IEEE International Conference on Robotics and Automation*, pages 243–249, May 2010. ISSN: 1050-4729.

[81] Y. Bar-Shalom. Update With Out-of-Sequence Measurements in Tracking: Exact Solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):769–777, July 2002.

[82] K. Y. Leung, T. D. Barfoot, and H. H. Liu. Decentralized Localization of Sparsely-Communicating Robot Networks: A Centralized-Equivalent Approach. *IEEE Transactions on Robotics*, 26(1):62–77, 2009.

[83] F. Bourgault and H. F. Durrant-Whyte. Communication in General Decentralized Filters and the Coordinated Search Strategy. In *In Proc. Of FUSION'04*, pages 723–770, 2004.

[84] S. I. Roumeliotis and G. A. Bekey. Distributed Multirobot Localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795, October 2002.

[85] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard. Recursive Decentralized Collaborative Localization for Sparsely Communicating Robots. In *Robotics: Science and Systems*. New York, NY, USA, 2016.

[86] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard. Recursive Decentralized Localization for Multi-Robot Systems With Asynchronous Pairwise Communication. *The International Journal of Robotics Research*, 37(10):1152–1167, September 2018.

[87] A. Bahr, M. R. Walter, and J. J. Leonard. Consistent Cooperative Localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 3415–3422, May 2009.

[88] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli. Distributed Maximum a Posteriori Estimation for Multi-Robot Cooperative Localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 1402–1409, May 2009.

[89] L. Chen, P. O. Arambel, and R. K. Mehra. Estimation Under Unknown Correlation: Covariance Intersection Revisited. *IEEE Transactions on Automatic Control*, 47(11):1879–1882, 2002.

[90] J. K. Uhlmann. Covariance Consistency Methods for Fault-Tolerant Distributed Data Fusion. *Information Fusion*, 4(3):201–215, 2003.

[91] M. Reinhardt, B. Noack, and U. D. Hanebeck. Closed-Form Optimization of Covariance Intersection for Low-Dimensional Matrices. In *2012 15th International Conference on Information Fusion*, pages 1891–1896. IEEE, 2012.

[92] S. J. Julier and J. K. Uhlmann. Using Covariance Intersection for SLAM. *Robotics and Autonomous Systems*, 55(1):3–20, 2007.

[93] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, and S. I. Roumeliotis. Decentralized Multi-Robot Cooperative Localization Using Covariance Intersection. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1412–1417. IEEE, 2013.

[94] Z. Song and K. Mohseni. Hierarchical Underwater Localization in Dominating Background Flow Fields. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3361, November 2013. ISSN: 2153-0866.

[95] Z. Song and K. Mohseni. FACON: A Flow-Aided Cooperative Navigation Scheme. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6251–6256, September 2017. ISSN: 2153-0866.

[96] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. A Wiley-Interscience publication. Wiley-Interscience, Hoboken, N.J, 2nd ed. edition, 2006. Publication Title: Elements of information theory.

[97] T. Idota and K. Baek. Conservative Data Exchange for Decentralized Cooperative Localization. In *2020 17th International Conference on Ubiquitous Robots (UR)*, pages 472–478, June 2020. ISSN: 2325-033X.

[98] T. Idota. Two-Dim-Sim. https://github.com/tidota/two-dim-sim.

[99] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3):189–206, April 2013.

[100] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[101] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[102] K. Singh and K. Fujimura. Map Making by Cooperating Mobile Robots. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 254–259 vol.2, May 1993.

[103] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-Robot Exploration Controlled by a Market Economy. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 3016–3023, 2002.

[104] A. Howard, M. J. Mataric, and G. S. Sukhatme. An Incremental Deployment Algorithm for Mobile Robot Teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2849–2854 vol.3, 2002.

[105] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. Towards Multi-Vehicle Simultaneous Localisation and Mapping. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 2743–2748, 2002.

[106] D. Rodriguez-Losada, F. Matia, and A. Jimenez. Local Maps Fusion for Real Time Multirobot Indoor Simultaneous Localization and Mapping. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 2, pages 1308–1313 Vol.2, April 2004.

[107] S. Thrun. A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots. *The International Journal of Robotics Research*, 20(5):335–363, May 2001.

[108] J. Jessup, S. N. Givigi, and A. Beaulieu. Merging of Octree Based 3D Occupancy Grid Maps. In *2014 IEEE International Systems Conference Proceedings*, pages 371–377, March 2014.

[109] J. Jessup, S. N. Givigi, and A. Beaulieu. Robust and Efficient Multirobot 3-D Mapping Merging With Octree-Based Occupancy Grids. *IEEE Systems Journal*, 11(3):1723–1732, September 2017.

[110] L. Contreras, O. Kermorgant, and P. Martinet. Efficient Decentralized Collaborative Mapping for Outdoor Environments. In *International Conference on Robotic Computing*, January 2018.

[111] A. Howard. Multi-Robot Mapping Using Manifold Representations. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 4, pages 4198–4203 Vol.4, April 2004.

[112] M. Pfingsthorn, B. Slamet, and A. Visser. A Scalable Hybrid Multi-Robot SLAM Method for Highly Detailed Maps. In *Robot Soccer World Cup*, pages 457–464. Springer, 2007.

[113] H. J. Chang, C. S. G. Lee, Y. C. Hu, and Y. Lu. Multi-Robot SLAM With Topological/Metric Maps. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1467–1472, October 2007.

[114] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena. An Online Multi-Robot SLAM System for 3D LiDARs. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1004–1011, September 2017. ISSN: 2153-0866.

[115] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV. In *2012*

IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4557–4564, October 2012.

[116] J. Engel, J. Sturm, and D. Cremers. Scale-Aware Navigation of a Low-Cost Quadrocopter With a Monocular Camera. *Robotics and Autonomous Systems*, 62(11):1646–1656, November 2014.

[117] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. In *Robotics Research*, pages 235–252. Springer, 2017.

[118] S. Hong, H. Ko, and J. Kim. VICP: Velocity Updating Iterative Closest Point Algorithm. In *2010 IEEE International Conference on Robotics and Automation*, pages 1893–1898, May 2010. ISSN: 1050-4729.

[119] F. Moosmann and C. Stiller. Velodyne Slam. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 393–398. IEEE, 2011.

[120] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[121] K. P. Nelson. Assessing Probabilistic Inference by Comparing the Generalized Mean of the Model and Source Probabilities. *Entropy*, 19(6):286, June 2017. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[122] Robot Operating System (ROS). https://www.ros.org/.

[123] Gazebo Simulator. http://gazebosim.org/.

[124] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. RotorS—A Modular Gazebo MAV Simulator Framework. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, pages 595–625. Springer International Publishing, Cham, 2016. Section: RotorS—A Modular Gazebo MAV Simulator Framework.

[125] T. Idota. Poisson-Sampling. https://github.com/tidota/poisson-sampling.

[126] I. Kuzminykh, A. Snihurov, and A. Carlsson. Testing of Communication Range in ZigBee Technology. In *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pages 133–136, February 2017.

[127] T. Idota. Map-Analysis. https://github.com/tidota/map-analysis.

[128] R. B. Rusu and S. Cousins. 3D Is Here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011. ISSN: 1050-4729.

[129] CloudCompare, 2021. http://www.cloudcompare.org/.