Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Instituto Universitario de Investigación
en Ingeniería de Aragón
Universidad Zaragoza

Departamento de
Informática e
Ingeniería de Sistemas
Universidad Zaragoza

Programa Oficial de Posgrado en Ingeniería Informática

Distributed Algorithms on Robotic Networks for Coordination in Perception Tasks

Rosario Aragüés Muñoz

Director: Carlos Sagüés Blázquiz

Instituto de Investigación en Ingeniería de Aragón
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Enero 2012

Contents

Abstract

The increasing interest in multi-robot applications is motivated by the wealth of possibilities offered by teams of robots cooperatively performing collective tasks. The efficiency and robustness of these teams goes well beyond what individual robots can do. In these scenarios, distributed strategies attract a high attention, especially in applications which are inherently distributed in space, time or functionality. These distributed schemas do not only reduce the completion time of the task due to the parallel operation, but also present a natural robustness to failures due to the redundancy. Our research is focused on distributed applications for perception tasks. Perception is of high importance in robotics, since almost all robotic applications require the robot team to interact with the environment. Then, if a robot is not able to obtain an environmental representation from others, or an a priori representation is not available, it must posses perception capabilities to sense its surroundings. Perception has been long studied for single robot systems and a lot of research has been carried out in the fields of localization, map building and exploration. Among the different sensors that can be used to perceive the environment, we are interested in visual perception using conventional or omnidirectional cameras. Each individual robot perceives the portion of the environment where it is operating. In order to make decisions in a coordinated way, the robots must fuse their local observations into a global map. We can distinguish between centralized and decentralized or distributed approaches. In distributed systems, all robots play the same role, and therefore the computations can be distributed among all the robots. In addition, distributed systems are naturally more robust to individual failures. We are interested in map merging solutions for robotic systems with range limited communication, and where the computations are distributed among the robots.

This thesis has been focused in identifying the different issues that appear in distributed perception scenarios and proposing solutions to all these issues, with a special interest in robots equipped with cameras. In particular, (i) Firstly, to investigate in the visual perception for a single robot; (ii) To propose methods for fusing the visual information acquired by the robots; (iii) To address the problem of establishing correspondences between the elements observed by different robots; (iv) To analyze the network localization problem from relative measurements; and (v) To propose control motion strategies to improve the perception of the environment.

During this thesis we have made important contributions to the field of distributed perception: We have proposed algorithms for merging maps acquired by a robot team with limited communication in a distributed way. We have considered both the static and dynamic cases. We have considered all the topics that appear in a map merging scenario

and have made contributions in all of them. The data association has been discussed for the first time in the context of distributed robot teams with limited communication. The initial correspondence or localization problem consists of reaching a consensus on a global reference frame. This global frame will be used by the robot team during their operation. In addition, we have made contributions to the problem of coordination for perception in two aspects. First, we have proposed a method so that the robots decide their next motions so that the global map is improved (it is more accurate). Second, we have proposed a method that allows the robot to compute an important parameter related to the network topology, which characterizes the speed of the previous distributed algorithm and thus establishes the rate at which the different algorithms converge to the values of interest. Therefore, controlling the network topology so that this parameter is improved, or so that it does not fall below a pre-given value, let us have a deeper control of the quality of our methods.

Part of the results obtained during this thesis have been published in international journals with high impact in the robotics community: [8], Robotics and Autonomous Systems journal. Several results have been presented in international conferences: [10], Robotics: Science and Systems 2010; [7], [3], IEEE International Conference on Robotics and Automation, years 2010 and 2011; [6], IEEE/RSJ International Conference on Intelligent Robots and Systems, year 2009; [11], International Conference on Informatics in Control, Automation and Robotics, year 2008. Some results have been presented in international workshops: [5], Workshop on Network Robot Systems, IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, year 2008. The most recent results have been submitted to international conferences ([12], American Control Conference) and to journals ([9], submitted to IEEE Transactions on Robotics; [93], submitted to IEEE Transaction on Pattern Analysis and Machine Intelligence; [4], submitted to Systems & Control Letters) and are currently under review. Additionally, we have participated in some research works in collaboration with international universities [35–37, 147].

Chapter 1
Introduction

1.1 Motivation

The increasing interest in multi-robot applications is motivated by the wealth of possibilities offered by teams of robots cooperatively performing collective tasks. The efficiency and robustness of these teams goes beyond what individual robots can do. In these scenarios, distributed strategies attract a high attention, especially in applications which are inherently distributed in space, time or functionality. These distributed schemas do not only reduce the completion time of the task due to the parallel operation, but also present a natural robustness to failures due to the redundancy. In addition to the classical issues associated to the operation of individual robots, these scenarios introduce novel challenges specific to the coordination of the members of the robot team. Several distributed algorithms are based on behaviors observed in nature. It has been observed that certain groups of animals are capable of deploying over a given region, assuming a specified pattern, achieving rendezvous at a common point, or jointly initiating motion or changing direction in a synchronized way (Fig. 1.1). In the robotics literature, these behaviors are often called deployment, pattern formation, rendezvous, and flocking, respectively. Species achieve synchronized behavior, with limited sensing or communication between individuals, and without apparently following the instructions of a group leader. Robotic researchers have intensively investigated on coordination strategies for

Figure 1.1: Examples of flocking and pattern formation observed in animals.

multi-robot systems (Fig. 1.2) capable of imitating these collective behaviors. In partic-

ular, it is worth mentioning the following strategies: rendezvous, which consists of the robots getting together at a certain location; deployment or coverage, which consists of deploying the robot team over the region of interest, and agreement, which consists of reaching consensus upon the value of some variable. Agreement has a special interest and recently it has been shown that several multi-robot strategies, including pattern formation and rendezvous, can be transformed into an agreement problem.

Figure 1.2: Examples of multi-robot teams.

Our research is focused on distributed applications for perception tasks. Perception is of high importance in robotics, since almost all robotic applications require the robot team to interact with the environment. Then, if a robot is not able to obtain an environmental representation from others, or an a priori representation is not available, it must posses perception capabilities to sense its surroundings. Perception has been long studied for single robot systems and a lot of research has been carried out in the fields of localization, map building and exploration. Among the different sensors that can be used to perceive the environment, we are interested in visual perception using conventional or omnidirectional cameras (Figs. 1.3). While the first kind of cameras (Fig. 1.4) are widely

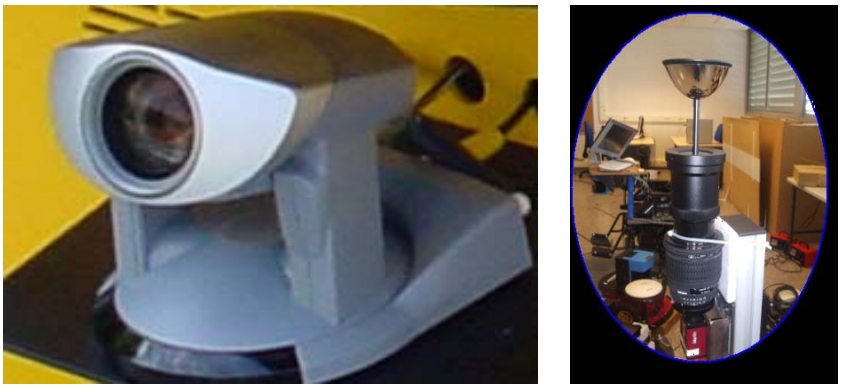
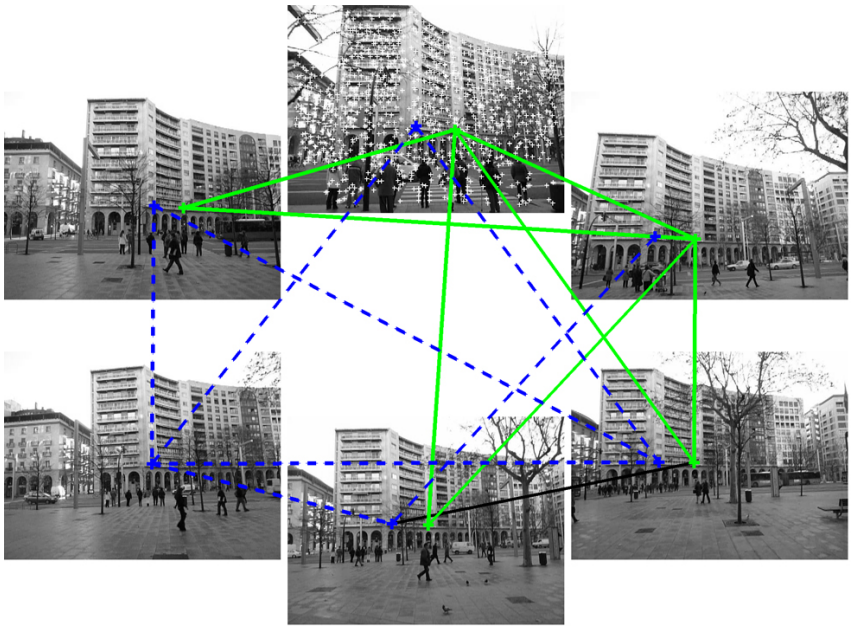


Figure 1.3: Examples of a conventional and an omnidirectional cameras.

known and used in any area, omnidirectional devices are very popular in robotic applications. These cameras are able to capture visual information within 360 degrees around the robot due to the use of an hyperbolic mirror (Fig. 1.5). We manage bearing-only



;

Figure 1.4: Examples of images taken by a team of 6 robots moving in formation equipped with a conventional camera. Crosses are features extracted from the images and lines between images represent feature matches.

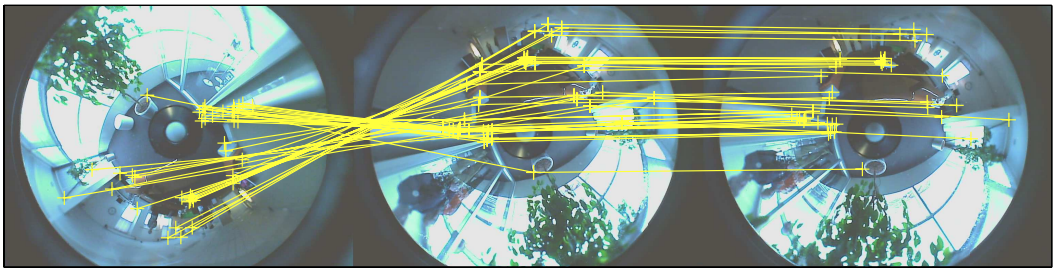


Figure 1.5: Examples of omnidirectional images. Crosses are features extracted form the images, and lines between images represent features matches.

information, which is the kind of information provided by cameras through the projection of landmarks which are in the scene. In order to recover the position of these landmarks in the world, multiple observations taken from different positions must be combined. The manipulation of bearing data is an important issue in robotics. Compared with information extracted from other sensors, such as lasers, bearing information is complicated to use. However, the multiple benefits of using cameras have motivated the interest in the researchers. These benefits include the property that cameras are able to sense quite distant features so that the sensing is not restricted to a limited range. An additional kind of cameras of high interest are RGB-D devices (Fig. 1.6). They provide both regular RGB (Fig. 1.7, first row) and depth image information (Fig. 1.7, second row). Thus, it is possible to compute the landmark 3D position from a single image (Fig. 1.7, third row).

Robots sense the environment and combine the bearing data to build representations of their surroundings in the form of stochastic maps. We use landmark-based representa-

Figure 1.6: Example of a RGB-D camera.

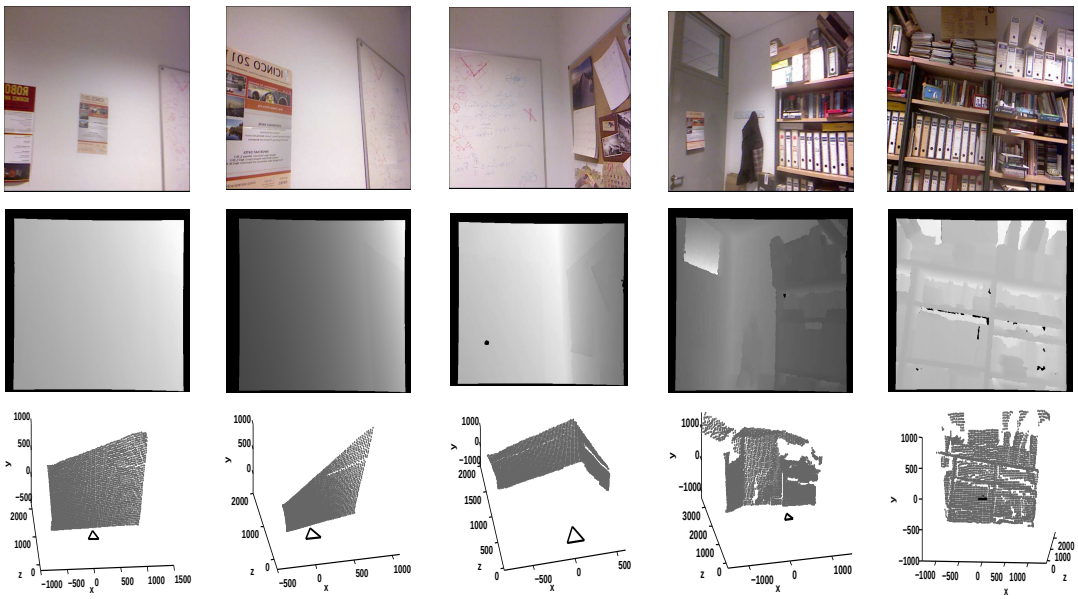


Figure 1.7: An example of the images obtained with the RGB-D sensor.

tions, where the map is a set of estimates of the positions of the observed landmarks. Each individual robot perceives the portion of the environment where it is operating. In order to make decisions in a coordinated way, the robots must fuse their local observations into a global map. We can distinguish between centralized and decentralized or distributed approaches. Centralized strategies, where a central node compiles all the information from other robots, performs the computations, and propagates the processed information or decisions to the other nodes, have several drawbacks. The whole system can fail if the central node fails, leader selection algorithms may be needed, and a (direct or indirect) communication of all agents with the central system may be required. On the other hand, in distributed systems, all robots play the same role, and therefore the computations can be distributed among all the agents. In addition, distributed systems are naturally more robust to individual failures. In distributed scenarios you cannot assume that the agents can communicate with all other robots at every time instant. A more realistic situation is when, at any time instant, robots can communicate only with a limited number of other robots, e.g., agents within a specific distance. These situations can be best modeled

using communication graphs, where nodes correspond to the agents and edges represent communication capabilities between the robots. Additionally, since agents are moving, the topology of the graph may vary along the time, given rise to switching topologies. We are interested in map merging solutions for robotic systems with range limited communication, and where the computations are distributed among the robots. We consider that a strategy is distributed when

- it does not rely on any particular communication topology and it is robust to changes in the topology;
- every robot in the team computes and obtains the global information;
- every robot plays the same role, making the system robust to individual failures;
- information is exchanged exclusively between neighbors and there are no broadcast messages or flooding methods.

1.2 Classical Approaches

Multi-robot systems have been deeply researched during the last years. A general overview of the achieved results, and the current and future research lines in distributed multi-robot systems can be found in [106,107,123]. Important results have been obtained for the cooperative control strategies mentioned in the previous section. We provide here the following references for the rendezvous [44, 61, 89], the deployment and coverage [45, 89], and the formation control problems [50, 62, 79, 135, 145], as some examples within the variety of different existing works. The consensus or averaging problem has a special relevance in multi-robot systems. It has been shown that the consensus problem is connected to diverse applications in multi-robot systems, including sensor fusion, flocking, formation control, or rendezvous, among others [102, 103]. Several ideas presented along this document are built on consensus results in the books [27, 113].

Many existent solutions for single robot perception have been extended to multi-robot scenarios under centralized schemes, full communication between the robots, or broadcasting methods. In [117] a single global map is updated by all the robots. Robots search for features in the global map that have been observed by themselves along the exploration. Then, they use these coincident features to compute implicit measurements (the difference between the Cartesian coordinates of equal features must be zero) and use these constrains to update the map. In [56] maps are represented as constraint graphs, where nodes are scans measured from a robot pose and edges represent the difference between pairs of robot poses. Robot to robot measurements are used to merge two local maps into a single map. An optimization phase must be carried out in order to transform the constraint graph into a Cartesian map. [42] also represents the global map using a graph. Nodes are local metric maps and edges describe relative positions between adjacent local maps. The map merging process consists of adding an edge between the maps. Global optimization techniques are applied to obtain the global metric map. [159] merges two maps into a single one using robot to robot measurements to align the two maps and

then detecting duplicated landmarks and imposing the implicit measurement constraints. Particle filters have been generalized to multi-robot systems assuming that the robots broadcast their controls and their observations [69]. The Constrained Local Submap Filter has been extended to the multi-robot case assuming that each robot builds a local submap and broadcasts it, or transmits it to a central agent [149]. Methods based on graph maps of laser scans [56, 78, 110] make each robot build a new node and broadcast it. The same solution could be applied for many existing submap approaches [108]. The previous methods require that each robot has the capability to communicate with all other robots at every time instant or with a central agent, i.e., they impose centralized scenarios. As previously mentioned, we are instead interested in distributed scenarios due to their robustness to robot or link failures, and due to their natural capability to operate under limited communication.

Distributed estimation methods [1, 40, 65, 71, 96, 100, 101, 143] maintain a joint estimate of a system that evolves with time by combining noisy observations taken by the sensor network. Early approaches sum the measurements from the different agents in IF (Information Filter) form. If the network is complete [96], then the resulting estimator is equivalent to the centralized one. In general networks the problems of cyclic updates or double counting information appear when nodes sum the same piece of data more than once. The use of the channel filter [65, 143] avoids these problems in networks with a tree structure. The Covariance Intersection method [71] produces consistent but highly conservative estimates in general networks. More recent approaches [1, 40, 100, 101] use distributed consensus filters to average the measurements taken by the nodes. The interest of distributed averaging is that the problems of double counting of information and cyclic updates are avoided. They, however, suffer from the delayed data problem that takes place when the nodes execute the state prediction without having incorporated all the measurements available at the current step [34]. For general communication schemes [100], the delayed data problem leads to an approximate KF (Kalman Filter) estimator. An interesting solution is given in [101] but its convergence is proved in the absence of observation and system noises. In the algorithm proposed in [40], authors prove that the nodes' estimates are consistent, although these estimates have disagreement. Other algorithms have been proposed that require the previous offline computation of the gains and weights of the algorithm [1]. The main limitation of all the previous works is that they consider linear systems without inputs, and where the evolution of the system is known by all the robots. We are however interested in a wider class of systems, without the previous restrictions.

A related scenario are sensor fusion systems [30, 31, 59, 88, 151–153], where measurements acquired by several sensors are fused in a distributed fashion. Sensor fusion systems differ from the distributed perception scenario that we consider in this thesis in several aspects. First, sensor fusion approaches consider that the successive measurements, in our case local maps, from the same robot must be independent. In a map merging scenario this does not hold, since the local map of a robot is an evolution of any of its previous maps. Second, sensors usually observe a set of variables which are a priori known by the sensor network, e.g., temperature, humidity, etc. However, in distributed perception scenarios, robots discover elements in the environment dynamically, as they operate. Thus,

it is not possible to predict which elements will be detected and inform the robot team of these elements before starting the exploration. In addition, distributed perception methods must address specific challenges such as associating the elements observed by the robots in a globally consistent way, or computing the relative poses of the robots and establishing a common reference frame for the whole robot team.

The data association problem consists of establishing correspondences between different measurements or estimates of a common element. Traditional data association methods, like the Nearest Neighbor and Maximum Likelihood [64, 72, 159], the Joint Compatibility Branch and Bound (JCBB) [97], or the Combined Constraint Data Association [16] are designed for single robot systems. They operate on two sets of elements, one containing the feature estimates and the other one containing the current observations. Methods based on submaps [29, 114, 149] use these data association algorithms by transforming one map into an observation of a second one. Multi-robot approaches have not fully addressed the problem of data association. Many methods rely on broadcasting controls and observations or submaps, see e.g., [56, 64, 69, 78, 83, 110], and solve the data association using a cycle-free order, thus essentially reducing the problem to that of the single robot scenario. However, in a distributed map merging scenario, the robots may fuse their maps with any other robot's map in any order and at any time. Therefore, it is not possible to force a specific order for solving the data association between the local maps. In the distributed perception scenario considered in this thesis, it is of high interest to let each robot solve its own local association with its neighbors in the communication graph. This scenario is more flexible but may lead to inconsistent global data associations in the presence of cycles in the communication graph. These inconsistencies are detected when chains of local associations give rise to two features from one robot being associated among them. These situations must be correctly identified and solved before merging the data. Inconsistent associations have already been discussed in the computer vision literature [53], although in centralized scenarios.

The problem of estimating the common reference frame for the team of robots is motivated by the fact that, in general, the robots start at unknown locations and do not know their relative poses. This information can be recovered by comparing their local maps and looking for overlapping regions. This approach, known as map alignment, has been deeply investigated and interesting solutions have been presented for feature-based [137] and occupancy grid [38, 39] maps. However, it has the inconvenience that its results depend on the accumulated uncertainty in the local maps. Alternatively, the relative poses between the robots can be explicitly measured [120, 158, 159]. These methods present the benefit that the obtained results do not depend on the uncertainties in the local maps. They also allow the robots to compute their relative poses when there is no overlapping between their maps, or even if they do not actually have a map. The previous methods give the relative position of a pair of robots. After that, a distributed method is required to let the robots agree on a global reference frame and obtain their positions in this frame. This problem is known as distributed network localization. Several network localization algorithms rely on range-only [2, 32], or bearing-only [122] relative measurements of positions. Alternatively, each agent can locally combine its observations and build an estimate of the relative full-position of its neighbors using e.g., the approach de-

scribed in [120, 141] for 3D scenarios. When full-position measurements are available, the localization problem becomes linear and can thus be solved by using linear optimization methods [18, 119]. There exist works that compute not only the agents' positions but also their orientations, [57], and that track the agents' poses [76]. It is also possible to use a position estimation algorithm combined with an attitude synchronization [95, 121] or a motion coordination [43] strategy to previously align the robot orientations. Formation control [43, 52, 70, 82] and network localization are related problems. While localization algorithms compute agent positions that satisfy the inter-agent restrictions, in formation control problems the agents actually move to these positions. The goal formation is defined by a set of inter-agent restrictions (range-only, bearing-only, full-positions, or relative poses). Although some works discuss the effects of measurement noises in the final result [43], formation algorithms usually assume that both, the measurements and the inter-agent restrictions are noise free [52, 70, 82]. Thus, additional analysis is necessary in noisy localization scenarios. The noisy nature of the relative pose measurements has already been taken into account in the field of cooperative localization. Here, a robotic team moves along an environment while estimating their poses. Most of the time, each robot relies on its proprioceptive measurements. When two robots meet, they obtain a noisy measurement of their relative pose and update their estimates accordingly [118]. During this rendezvous, each robot must be able to communicate with all the other robots in the team in order to update its estimate. In order to improve the usage of the network, recently a more efficient algorithm based on quantization [140] has been presented. Cooperative localization approaches, however, assume that an initial guess on the robot poses exists.

An important issue that arises in multi-robot scenarios consists of placing the robots at positions where they are more useful. Extensive research has been focused on the optimization of facilities [98, 99, 132], and on coverage problems [45, 89], where a group of robots is optimally placed in an environment of interest to achieve maximum coverage. Here, we are interested in controlling the robot motions to maximize the information collected about the scene. This problem is highly related to exploration guided by information and active sensing, which has been investigated for both single robot [131, 136] and multi-robot systems [28]. The previous works are based on grid maps, where frontier cells dividing between already explored and unknown sections can be easily detected. Robots evaluate a cost function on this small subset of destinations and make decisions propagating small pieces of information with the other robots. The exploration [115, 116] and feature tracking [155] problems turn out to be more complicated for landmark-based representations, since the number of candidate destinations is infinite. It is common the use of global optimization methods, where robots search for the best position to reduce the whole map uncertainty. Every robot makes decisions based on its current local estimate of the global map and propagates its observations to the other robots so that they can update their maps. These approaches result in weak robot coordination, because without a common global map estimate, different robots may end up exploring exactly the same regions. In addition, many of these solutions use gradient methods to find minima on the cost function. Gradient algorithms are computationally expensive since the gradient must be reevaluated at every step. Besides, they may find local minima, and the step

size adjustment is complicated. Alternatively, clustering methods can be used to select a finite subset of candidate positions [150]. Instead of choosing frontier points, as it is done in [150], we are interested in looking for already explored places which present big uncertainties. We focus on one step strategies instead of considering path planning or trajectory optimization methods [84, 85]. These methods use a larger time horizon and consider the cost function for multiple successive robot motions, and thus they present important scaling problems for the multi-robot case.

The multi-robot perception can also be improved by speeding up the map merging algorithms. These distributed methods have a rate of convergence that depends on the algebraic connectivity of the network. Thus, it would be of high interest for the robots to estimate this algebraic connectivity in a distributed fashion. Since distributed algorithms are typically iterative and may take several iterations to converge, it would be useful for the robots to have upper and lower bounds for the estimated algebraic connectivity at every step. Note that some applications, such as the adaptive triggered consensus algorithm in [124], require the robots to know upper or lower bounds of the algebraic connectivity. Here we focus on distributed methods that iteratively compute the algebraic connectivity, and that give lower and upper bounds at each iteration. Connectivity control methods establish robot motions that preserve or maximize some network connectivity property. The $k-$connectivity matrix of the graph is computed in a centralized fashion in [156]. There are several distributed methods that compute spanning subgraphs [157], the left Laplacian eigenvector with eigenvalue 1 for directed unbalanced graphs [112], or the first four moments (mean, variance, skewness and kurtosis) of the Laplacian eigenvalue spectrum [111]. In [126], the motion control strategy maximizes the algebraic connectivity without actually computing it. Although the previous control methods improve the network connectivity, they do not characterize any particular eigenvalue of the Laplacian as required in our case. A method that computes and tracks the eigenvalues of the network topology is given in [58]. Robots execute a local interaction rule that makes their states oscillate at frequencies corresponding to these eigenvalues, and use the Fast Fourier Transform (FFT) on their states to identify these eigenvalues. The main limitation of this work is that the proper adjustment of the FFT, so that the eigenvalues can be correctly identified, is nontrivial. In addition, some robots may observe only a subset of the eigenvalues and thus they need to execute additional coordination rules for propagating their data. Several solutions to the computation of the Laplacian spectra rely on the power iteration method or variations [63, 74, 154]. Power iteration [68] selects an initial vector and then repeatedly multiplies it by a matrix and normalizes it. This vector converges to the eigenvector associated to the leading eigenvalue (the one with the greatest absolute value) of the matrix. The original matrix can be previously deflated so that a particular eigenvalue becomes the leading one. A continuous-time version of the power iteration is proposed in [154] for computing the Fiedler eigenvector, which is the one associated to the algebraic connectivity of the graph. In [74] the orthogonal iteration method is used for simultaneously computing the k leading eigenvectors of a matrix. This algorithm can be seen as a generalization of the power iteration where, at each step, the k vectors are multiplied by the matrix and ortonormalized. The previous method is used in [63] for computing the Fiedler eigenvector. The interest of power iteration methods is that each robot only needs

to maintain its own component within the estimate of the eigenvector. Then, the product of this vector by the Laplacian can be executed locally by the robots. However, the main limitation of these approaches consists on the normalization and orthonormalization of the vectors at each step. For [74], it involves a gossip-based information aggregation algorithm [73] and for [154] a distributed averaging method [59]. Therefore, several iterations of the previous algorithms must be executed by the robots between consecutive steps of the power method in order to ensure that they have achieved the required accuracy in the vector normalization. Besides, the previous methods only ensure convergence but they do not give any upper or lower bound relating the true algebraic connectivity and the estimates at each iteration.

1.3 Objectives

This thesis has been focused in identifying the different issues that appear in distributed perception scenarios and proposing solutions to all these issues, with a special interest in robots equipped with cameras. In particular,

(i) Firstly, investigate the management of bearing only data and visual perception;

(ii) Propose methods for fusing the visual information acquired by the different robots;

(iii) Address the problem of establishing correspondences between the elements observed by different robots;

(iv) Analyze the network localization problem from relative measurements; and

(v) Propose control motion strategies to improve the perception of the environment.

We were specially interested in providing distributed solutions that allow the robot team to operate in scenarios with limited communication. Our goal was to provide formal proofs of the performance of the algorithms in the previous scenarios. In addition, we were interested in validating our algorithms under real data as well.

This thesis has been developed in the Departamento de Informática e Ingeniería de Sistemas (DIIS) in the Universidad de Zaragoza,
http://diis.unizar.es/
within the Instituto de Investigacion e Ingenieria de Aragon,
http://i3a.unizar.es/
and associated to the Robotics, Perception and Real Time (ROPERT) Group,
http://robots.unizar.es/html/home.php
that researches on robot navigation, perception and communication. This thesis has been associated to the R&D Nacional project NERO (NEtworked mobile RObots for service and intervention tasks, MEC DPI2006-07928), to the European project URUS (Ubiquitous Networking Robotics in Urban Settings, European Union, IST Program, 6FP, IST-1-045062-URUS-STP), and to the R&D National project TESSEO (TEams of robots for Service and SEcurity missiOns, MEC DPI2009-08126). Within these projects, this thesis has been focused on multi-robot systems, cooperative perception and distributed

map merging. This research has been founded by the grant MEC BES-2007-14772 (grant for the formation of researchers, FPI). Part of this research has been carried out in collaboration between the Universidad de Zaragoza and several international universities and institutes, being supported by the program of short research stays (EEBB FPI). Specifically, in the University of California, San Diego, US (April 2008, 3 months); again in the University of California, San Diego, US (March 2009, 3 months); in the Politecnico di Torino, Italy (April 2010, 4 months); and in the Royal Institute of Technology, Sweden (March 2011, 3 months).

1.4 Document Organization

This document is organized as follows:

Chapter 1 introduces the problem addressed in this thesis, the classical approaches to solve it, and states the contributions.

Chapter 2 addresses the problem of visual perception. We discuss and propose several visual information management methods and evaluate their performance. These methods allow each robot to combine the visual information obtained with its camera and build a local map of the portion of the environment explored by itself.

Chapter 3 merges the information acquired by each robot in the network to build a global representation of the environment. This map merging is carried out after the robots finish their exploration. We assume that the ground-truth data association is available at the robots. We further assume that all the robots share a common reference frame and that they know their pose in this frame.

Chapter 4 solves the dynamic map merging problem. Robots explore the environment and, simultaneously, fuse their local maps and build the global map of the environment. Therefore, robots have a representation of the environment beyond its local map during all their operation. The fusion of the local observations of all the team members leads to a merged map that contains more precise information and more features. This global map enables other multi-robot tasks such as cooperative exploration, navigation, or obstacle avoidance.

Chapter 5 addresses the problem of establishing correspondences between the elements in the environment observed by different robots. Robots compute the associations with their neighbors using classical matching methods. We propose distributed algorithms that allow each robot to propagate this local data and obtain the global data association relating its features with the ones of all the other robots in the network. In addition, they are able to identify and correct associations which are inconsistent in the global level.

Chapter 6 discusses the problem of estimating the positions of the robots in a common reference frame. Robots compute the relative position of their neighbors using classical methods. From this information, they build the global frame and compute their positions in this frame in a distributed fashion. We consider scenarios with both noise-free and noisy measurements, and discuss the use of an anchor node to define the global frame and well as more sophisticated methods.

Chapter 7 studies how the robot motions affect their perception of the environment. This has two aspects: First, we analyze how they can improve the precision of the ob-

served elements in a cooperative fashion. Second, we investigate on how to speed up the convergence of the distributed algorithms proposed in this thesis. This can be achieved by controlling the network topology. We propose distributed methods that allow each robot to compute the algebraic connectivity of the network topology, that specifies this convergence speed.

Chapter 8 evaluates the performance of some of our proposals in real scenarios.

Chapter 9 presents the conclusions of this thesis.

The document finishes with a set of appendices. Appendix A contains several well known results on average consensus algorithms which are included here for this document to be self-contained. Appendix B describes a public dataset that we have used in several of our experiments.

Within each chapter, we include a brief description of its contents, an introduction and a discussion of the related work, a description of the problem and our proposal, specific simulations, and the conclusions.

1.5 Contributions

During this thesis we have made important contributions to the field of distributed perception:

- The proposal of algorithms for merging maps acquired by a robot team with limited communication in a distributed way. We have considered both the static and dynamic cases.
- We have considered all the topics that appear in a map merging scenario and have made contributions in all of them. The data association, to our knowledge, has been discussed for the first time in the context of distributed robot teams with limited communication.
- The initial correspondence or localization problem consists of reaching a consensus on a global reference frame. This global frame will be used by the robot team during their operation. We have proposed distributed algorithms for different scenarios.
- In addition, we have made additional contributions to the problem of coordination for perception in two aspects. First, we have proposed a method so that the robots decide their next motions to improve the global map. Second, we have proposed a method that allows the robots to compute an important parameter related to the network topology, which characterizes the speed at which the different distributed algorithms converge to the values of interest.

The results about static map merging have been published in the journal Robotics and Autonomous Systems [8], and in the Workshop on Network Robot Systems, IEEE/RSJ Int. Conf. on Intelligent Robots & Systems [5]. The results regarding the dynamic map merging case have been presented in the IEEE International Conference on Robotics and Automation [7] and an extended version has been submitted to the journal IEEE Transactions on Robotics [9] and is currently under review.

The basic detection method and the resolution based on trees have been presented in the conference Robotics: Science and Systems [10]. An extension with the feature labeling appears as part of a paper [8] published in the journal Robotics and Autonomous Systems. The improved detection algorithm and the resolution algorithm based on the maximum error cut, together with several experiments with real data, have been submitted to the journal IEEE Transaction on Pattern Analysis and Machine Intelligence and are currently under review [93].

The noise-free pose localization appears as part of the paper [8] published in the journal Robotics and Autonomous Systems. The pose localization from noisy measurements algorithms was presented in the conference IEEE International Conference on Robotics and Automation [3]. The centroid-based position localization has been submitted to the Systems & Control Letters journal [4] and is currently under review.

The map precision improvement method appears in the proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems [6]. The results regarding the convergence speed improvement have been submitted to the American Control Conference [12] and are under review.

Part of the research has been carried out in collaboration between the Universidad de Zaragoza and several international universities and institutes, being supported by the program of short research stays (EEBB FPI). Specifically, in the University of California, San Diego, US (April 2008, 3 months); again in the University of California, San Diego, US (March 2009, 3 months); in the Politecnico di Torino, Italy (April 2010, 4 months); and in the Royal Institute of Technology, Sweden (March 2011, 3 months).

The list of publications produced during this PhD follows.

- **International Journals:**

  R. Aragues, J. Cortes, and C. Sagues. Distributed consensus algorithms for merging feature-based maps with limited communication. *Robotics and Autonomous Systems*, 59(3-4):163-180, 2011.

  R. Aragues, J. Cortes, and C. Sagues. Distributed consensus on robot networks for dynamically merging feature-based maps. *IEEE Transactions on Robotics*, 2011, *submitted*.

  E. Montijano R. Aragues and C. Sagues. Distributed multi-view matching in networks with limited communications. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2011, *submitted*.

  R. Aragues, L. Carlone, C. Sagues, and G. Calafiore. Distributed centroid estimation from noisy relative measurements. *Systems & Control Letters*, 2011, *submitted*.

- **International Conferences:**

  R. Aragues and C. Sagues. Parameterization and initialization of bearing-only information: a discussion. In *Int. Conf. on Informatics in Control, Automation and Robotics*, volume RA-1, pages 252-261, Funchal, Portugal, May 2008.

R. Aragues, J. Cortes, and C. Sagues. Motion control strategies for improved multi robot perception. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1065-1070, St. Louis, USA, October 2009.

R. Aragues, J. Cortes, and C. Sagues. Dynamic consensus for merging visual maps under limited communications. In *IEEE Int. Conf. on Robotics and Automation*, pages 3032-3037, Anchorage, AK, May 2010.

R. Aragues, E. Montijano, and C. Sagues. Consistent data association in multi-robot systems with limited communications. In *Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

R. Aragues, L. Carlone, G. Calafiore, and C. Sagues. Multi agent localization from noisy relative pose measurements. In *IEEE Int. Conf. on Robotics and Automation*, pages 364-369, Shanghai, China, May 2011.

R. Aragues, G. Shi, D. V. Dimarogonas, C. Sagues, and K. H. Johansson. Distributed algebraic connectivity estimation for adaptive event-triggered consensus. In *American Control Conference*, 2012, *submitted*.

- **International Workshops:**

  R. Aragues, J. Cortes, and C. Sagues. Distributed map merging in a robotic network. In *Workshop on Network Robot Systems, IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Nice, France, September 2008.

Chapter 2
Visual Information Management

In this chapter we discuss feature parameterization and initialization for bearing-only data obtained from vision sensors. The performance of the methods is analyzed under different robot motions and depth of the features. The results are evaluated in terms of the sensitivity to step size and performance under ill conditioned situations. The problem studied refers to robots moving on the plane, sensing the environment and extracting bearing-only information from uncalibrated cameras to recover the position of the landmarks and its own localization.

2.1 Introduction

The manipulation of bearing information is an important issue in robotics. Bearing-only data is the kind of information provided by cameras through the projection of the scene. In order to recover the position of the landmarks in the world, multiple observations taken from different positions must be combined. Compared with information extracted from other sensors such as lasers, bearing information is complicated to use. However, the multiple benefits of using cameras have motivated the interest in the researchers. These benefits include their capability to sense quite distant features so that the sensing is not restricted to a limited range. This sensing of the environment in the form of bearing information may be used for many applications such as the Simultaneous Localization and Mapping (SLAM), which consists of the computation of the landmark localization in the environment and the calculation of the own robot pose. Algorithms which use bearing information must deal with the problem of creating representations for features by the combination of bearing data. The problem of feature parameterization and feature initialization are of big importance here.

A classical feature parameterization approach is the *cartesian* [14, 46, 75, 80]. In the *depth* parameterization [47] features are stored as an starting point of the ray where the feature lays, the inclination of the ray and the depth. The *inverse-depth* is an alternative similar to the *depth* parameterization, but using the inverse of the depth instead [92]. Some approaches use no explicit feature parameterization and instead represent landmarks as constraints between three robot poses [139].

With regard to the feature initialization, *undelayed* techniques immediately introduce features in the map so that they can be used to improve the robot estimation [46, 80, 92, 139] while *delayed* methods defer the introduction into the map until the features

are near-Gaussian [14, 75]. Delayed approaches often create temporal representations for landmarks which are maintained in separate filters and evolve with the incorporation of new observations of these landmarks until they are finally introduced into the map [47].

The problem of depth computation for landmarks is afforded in two separate ways. Some approaches create depth representation from only one bearing,W assuming an approximate value for it. These techniques are able to cover depths from the position were the landmark was observed until infinity or until a maximum depth within the workspace [47, 80, 92]. The other approach to depth computation is the combination of observations taken from different robot poses, where triangulation techniques are used to recover the depth [14, 75].

The interest of this work refers to the comparison of the bearing-only data representations and initialization techniques, analyzed for different robot motions relative to depth of the landmarks in the scene. Two feature parameterizations are studied. The first is an standard cartesian parameterization, where features are described by their (x, y) position. The alternative representation is an adaptation of the inverse-depth in [92] to the 2D situation. Besides, both undelayed and delayed strategies for feature initialization are used and their performance is compared in different scenarios. The problem studied in this chapter refers to robots moving on the plane, sensing the environment and extracting bearing-only information from uncalibrated images to recover the position of the landmarks and its own localization. As a result of this investigation, some theoretical solutions are proposed, and their validity is supported by an exhaustive experimentation using simulated data.

2.2 Problem Description

The problem studied in this chapter is related to the use of bearing-only information for the SLAM problem using EKF. The robot moves on the plane and elements in the map are represented by their 2D coordinates. Robot observes landmarks within a field of view of 360° due to the use of omnidirectional cameras and obtains bearing-only measurements. Odometry is used to predict robot motion in every step. The Extended Kalman Filter (EKF) is a widely used technique in these problems and a lot of information can be found in the literature. The data association problem is not discussed in this chapter. The Joint Compatibility Branch and Bound (JCBB) [97] is used to select the observations which are used in the filter update.

In order to make the reading easy, along the chapter we use the index $i \in \mathbb{N}$ to refer to robots and the index $j \in \mathbb{N}$ to refer to features. We let $\mathbf{x}_{r_i} = [x_{r_i}, y_{r_i}, \theta_{r_i}]^T$ be the i-th robot pose, where $x_{r_i}, y_{r_i} \in \mathbb{R}$ and $\theta_{r_i} \in [-\pi, \pi]$. When there is no confusion, the subscript i is omitted. The position of the j-th feature in the map is represented by $\mathbf{x}_j = [x_j, y_j]^T$ or $\mathbf{x}_j = [x_j, y_j, \theta_j, \rho_j]^T$ for respectively the cartesian and the inverse-depth parameterizations, where $x_j, y_j \in \mathbb{R}$, $\theta_j \in [-\pi, \pi]$, and $\rho_j \in \mathbb{R}_{\geq 0}$. We let $\mathbf{x}$ be the state vector containing the current robot pose $\mathbf{x}_r$ and the positions of the n landmarks,

$$\mathbf{x} = [\mathbf{x}_r, \mathbf{x}_1 ... \mathbf{x}_n]^T ,$$

and $\mathbf{P}$ be its associated covariance matrix. The measurement taken from robot pose i to feature j is denoted by z_{ij}, or by z_j when the robot that took the measurement is not specified.

2.3 Feature Parameterization

The feature parameterization refers to the way in which features are stored in the state vector $\mathbf{x}$. In this section the cartesian and the inverse-depth landmark parameterizations are compared. We discuss whether their landmark representations and observation models are near-Gaussian and near-linear or not.

Cartesian parameterizations represent features by their (x, y)-coordinates. This parameterization is very intuitive since the feature position within the map can be easily obtained. The initialization of features in this cartesian parameterization is problematic due to the nonlinearity of the triangulation techniques used to recover its position based on the observations taken from different robots poses. It can be easily shown that bearings generate bigger uncertainties as landmark positions go away from the camera. The observation model for a feature $\mathbf{x}_j = [x_j, y_j]^T$ observed from a robot pose $\mathbf{x}_r = [x_r, y_r, \theta_r]^T$ is [14]

$$z_j = \mathbf{h}(\mathbf{x}_r, \mathbf{x}_j) = \arctan\left(\frac{y_r - y_j}{x_r - x_j}\right) - \theta_r.$$

Inverse-depth parameterizations represent each feature $\mathbf{x}_j$ as a ray starting at the position where the feature was firstly observed (x_j, y_j) with a global bearing θ_j and a depth of $1/\rho_j$. Let $\mathbf{m}_j$ and R_r be

$$\mathbf{m}_j = \begin{bmatrix} \cos\theta_j \\ \sin\theta_j \end{bmatrix}, \qquad R_r = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\sin\theta_r & \cos\theta_r \end{bmatrix}.$$

The position in cartesian coordinates of the j-th feature in the state vector with inverse-depth representation $\mathbf{x}_j = [x_j, y_j, \theta_j, \rho_j]^T$ is given by

$$[x_j, y_j]^T + 1/\rho_j \ \mathbf{m}_j,$$

and its inverse-depth observation model is

$$h_j = \text{atan2}(h_j^y, h_j^x), \tag{2.1}$$

where $\mathbf{h}_j^{xy} = [h_j^x, h_j^y]^T$ are the coordinates of the feature j in the robot reference $\mathbf{x}_r = [x_r, y_r, \theta_r]^T$,

$$\mathbf{h}_j^{xy} = \begin{bmatrix} h_j^x \\ h_j^y \end{bmatrix} = R_r \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} + 1/\rho_j \ \mathbf{m}_j - \begin{bmatrix} x_r \\ y_r \end{bmatrix} \right). \tag{2.2}$$

Provided that $\rho_i > 0$, the inverse-depth observation model in (2.1) remains valid if the following expression is used instead of equation 2.2

$$\mathbf{h}_j^{xy} = \begin{bmatrix} h_j^x \\ h_j^y \end{bmatrix} = R_r \left(\rho_i \left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_r \\ y_r \end{bmatrix} \right) + \mathbf{m}_i \right).$$

As advantage with respect to the cartesian parameterization, the inverse-depth observation model is near linear. Additionally, landmarks at infinity, $\rho_i = 0$, or uncertainties that extend to infinity can be represented. The main drawback of the inverse-depth is that features are over-parameterized, and therefore the size of the covariance matrix is greater.

2.4 Depth Computation and Feature Initialization

The feature initialization consists of combining the observations of a landmark to create an estimate (mean and covariance) of its position, and introducing it into the map for the first time. The feature initialization problem of bearing-only data is due to the fact that features are only partially observable. Since a measurement only gives information about the direction towards the landmark, then two or more observations must be combined in order to recover the depth of the landmark. However, there are some situations where the depth cannot be recovered.

Theorem 2.4.1. *Let $\mathbf{x}_{r_1}$ be a robot pose and $\mathbf{x}_{r_2}$ be a second pose translated but not rotated with respect to $\mathbf{x}_{r_1}$. Let z_{1j}, z_{2j} be observations of a feature j taken from respectively $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$, and let α be $\alpha = z_{1j} - z_{2j}$. Let d_p and d_t be the translation from $\mathbf{x}_{r_1}$ to $\mathbf{x}_{r_2}$ on respectively a perpendicular and a paralell directions to z_{1j}. Without loss of generality, let d_t be equal to zero. The landmark depth d_j (distance between $\mathbf{x}_{r_1}$ and the landmark j) can be totally determined from*

$$d_j = d_p / \tan \alpha.$$

Corollary 2.4.2. *The problem of depth computation from pure translation motions has the following properties,*

(i) this is an undetermined problem $(0/0)$ when simultaneously $d_p = 0$ and $\alpha = 0 + k\pi$, for $k \in \mathbb{Z}$;

(ii) this problem remains undetermined independently of the magnitude of d_t;

(iii) the landmark is at infinity if simultaneously $\alpha = 0 + k\pi$ for $k \in \mathbb{Z}$ and d_p is different of zero.

Theorem 2.4.3. *Let $\mathbf{x}_{r_1}$ be a robot pose and $\mathbf{x}_{r_2}$ be a second pose rotated but not translated with respect to $\mathbf{x}_{r_1}$. Let z_{1j}, z_{2j} be observations of a feature j taken from respectively $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$. The robot rotation θ_{r_2} can be absolutely determined as*

$$\theta_{r_2} = z_{1i} - z_{2i}.$$

Corollary 2.4.4. *The problem of depth computation from general robot motions has the following properties,*

(i) given a pure rotation motion, feature depth cannot be recovered;

(ii) given a translation and rotation motion with landmarks of infinite depth, the robot rotation can be computed from $z_{1i} - z_{2i}$ for any $d_p < \infty$ and robot translation cannot be recovered.

Based on the previous theorems, ill-conditioned situations are identified:

(a) Depth of features aligned with robot trajectory cannot be recovered. This situations is formalized in Corollary 2.4.2 (i)-(ii).

(b) Depth cannot be recovered with pure rotation motions as shown in Corollary 2.4.4 (i).

(c) Landmarks at infinity give robot orientation, but no translation information can be obtained from them (Corollary 2.4.4 (ii)).

Feature estimates calculated when the depth computation problem is ill-conditioned present high covariances and great estimation errors which may cause linealization problems. Once a feature has been wrongly initialized, new observations taken from robot poses not aligned with the feature will not be able to correct its position. If a cartesian parameterization is used, an additional problem is that features with infinite depth cannot be represented and their initialization must be deferred.

2.4.1 Undelayed Initialization

The undelayed initialization consists of introducing landmarks into the system the first time the landmark is observed. This technique presents many benefits since the information attached to a landmark can be used earlier and it allows the use of landmarks which may never been initialized if a delayed strategy is used.Since the first time a landmark is observed only bearing information is available, undelayed techniques must deal with the problem of creating a representation for the depth and its associated uncertainty.

If an inverse-depth parameterization is used, landmarks are introduced using a fixed initial depth and an initial uncertainty covering all depths from some d_{min} to infinity. This initial depth must be adjusted depending on the workspace. Since cartesian parameterizations require low covariances, an undelayed initialization is only possible if multiple hypothesis in depth are created [80, 81, 128]. All these approaches present a high complexity and map size. Due to this complexity, approaches using undelayed initialization together with cartesian parameterization are no longer analyzed in this chapter.

2.4.2 Delayed with Two Observations

This delayed method combines the first two observations of a landmark to recover its position using a triangulation algorithm. This is a not purely delayed technique, since

there are no conditions which must be satisfied by the observations in order for the landmark to be initialized, and all landmarks are introduced in the map provided that they are observed from at least two robots poses. The main benefit of this initialization strategy is that the solution is independent on the workspace. However, triangulation algorithms used to recover the landmark position are highly non-linear and, depending on the arrangement of robot poses and features, the problem may be ill-conditioned.

If a cartesian parameterization is used, the recovered feature position must be near-Gaussian and covariances must be small. For this reason, additional tests are used to check that features satisfy these conditions. If features are parameterized using inverse-depth, this strategy may suppose a benefit in the sense that it is independent on the size of the scene. Therefore higher covariances in the estimates are admissible and recovered features are near-Gaussian even for low parallaxes.

2.4.3 Delayed until Condition

In a pure delayed initialization technique, observations of landmarks are accumulated and its initialization is deferred until a condition of Gaussianity is satisfied; then observations are used to create a representation for the feature [14, 75].

If a delayed initialization is used, some landmarks may never been initialized. Since the information provided by landmarks cannot been used until the landmark is initialized, a delayed technique decreases the amount of information available to improve robot the pose. Many delayed techniques present a high computational cost to calculate the condition, and have their own problems and limitations. The main benefit is that the representation for the landmark is more accurate and reliable than the obtained by an undelayed strategy.

2.5 Studied Methods

Along this chapter, a comparison of cartesian and inverse-depth parameterizations combined with delayed and undelayed initialization techniques is carried out. The following studied combinations have been selected because are the most commonly used, being also simple and of low computational complexity.

2.5.1 Inverse-Depth Undelayed

The following method is an adaptation to the 2D situation of the technique described in [92]. Each feature j is introduced into the map using a single observation. The current robot pose $\mathbf{x}_r$ is used together with the measurement z_j and an initial depth parameterized in *inverse-depth* ρ_0 to get the feature representation $\mathbf{x}_j$. This depth is worked out using a minimal distance $d_{\min}$ which must be selected depending on the workspace,

$$\rho_{\min} = \frac{1}{d_{\min}},\ \rho_0 = \frac{\rho_{\min}}{2},\ \sigma_\rho = \frac{\rho_{\min}}{4},$$

where $\rho_{\min}$ is the inverse of depth, ρ_0 is the initial inverse-depth, which is the middle value of the interval $[0, \rho_{\min}]$, and σ_ρ is the standard deviation used to initialize ρ_0. It is selected so that the 95% of ρ belongs to the interval $[\rho_0 - 2\sigma_\rho, \rho_0 + 2\sigma_\rho] = [0, \rho_{min}]$. The initial value of the feature is calculated as

$$\mathbf{x}_j = \mathbf{g}_j(\mathbf{x}_r, z_j, \rho_0) = [x_r, y_r, \theta_r + z_j, \rho_0]^T.$$

2.5.2 Inverse-Depth Delayed with Two Observations

As a proposal, an inverse-depth parameterization [92] is combined with a delayed initialization technique where two observations z_{1j}, z_{2j} of a feature j taken from different robot poses $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$ are used to retrieve the feature position $\mathbf{x}_j$. Here, the second observation z_{2j} is used to calculate the initial depth ρ_0 for the feature,

$$\mathbf{x}_j = \mathbf{g}(\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, z_{1j}, z_{2j}) = [x_{r_2}, y_{r_2}, \theta_{r_2} + z_{2i}, \rho_0]^T, \quad \rho_0 = \frac{s_2\ c_1 - c_2\ s_1}{c_1\ (y_{r_1} - y_{r_2}) - s_1\ (x_{r_1} - x_{r_2})},$$

where c_i, s_i, with $i \in \{1, \ldots, 2\}$ are

$$c_i = \cos(\theta_i + z_{ij}), \qquad s_i = \sin(\theta_i + z_{ij}). \tag{2.3}$$

An additional check is used in order to detect situations where inverse-depth cannot be recovered and intersections take place in the opposite direction of the observation. In these situations, the initialization is deferred.

2.5.3 Cartesian Delayed with Two Observations

Given the first two observations z_{1j}, z_{2j} of a landmark j taken from robot poses $\mathbf{x}_{r_1}, \mathbf{x}_{r_2}$, the landmark position $\mathbf{x}_j$ is calculated as follows [14]

$$x_j = g_j^x(\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, z_{1j}, z_{2j}) = \frac{x_{r_1}\ s_1\ c_2 - x_{r_2}\ s_2\ c_1 - (y_{r_1} - y_{r_2})\ c_1\ c_2}{s_1\ c_2 - s_2\ c_1},$$
$$y_j = g_j^y(\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, z_{1j}, z_{2j}) = \frac{y_{r_2}\ s_1\ c_2 - y_{r_1}\ s_2\ c_1 + (x_{r_1} - x_{r_2})\ s_1\ s_2}{s_1\ c_2 - s_2\ c_1},$$

where c_i, s_i, with $i \in \{1, \ldots, 2\}$ are given by (2.3). Similarly a test is used to check that features can be recovered and intersections of bearings are not in the opposite direction of the observations.

2.5.4 Cartesian/Inverse-Depth Delayed until Finite Depth

A delayed strategy is proposed where feature initialization is deferred until finite uncertainty in depth can be estimated. This is achieved by a test which compares two observation rays and checks if they are parallel. This situation is characterized by Corollaries 2.4.2 and 2.4.4. When observation rays are parallel, the uncertainty in depth of the recovered landmark extends to infinity and the initialization is deferred. This test is especially useful when a cartesian parameterization is used, since infinite depths cannot

been modeled. Let $\mathbf{x}_{r_i}$ for $i = 1, 2$ be the two robot poses where observations z_{ij} to a landmark j were taken.

The global bearings α_{ij} to the landmark are

$$\alpha_{ij} = \theta_{r_i} + z_{ij}, \tag{2.4}$$

for $i = 1, 2$. If we name $S_{\alpha_{ij}}$ the linearized propagated covariance for bearing α_{ij} then the Chi-squared test for Finite Depth is expressed as

$$\frac{(\alpha_{1j} - \alpha_{2j})^2}{S_{\alpha 1j} + S_{\alpha_{2j}}} > \chi^2_{0.99,1\text{d.o.f}}.$$

2.5.5 Cartesian/Inverse-Depth Delayed until Feature Not Aligned

The initialization of features aligned with the robot trajectory is a problematic issue when working with bearing-only data. When a feature is observed from two robot poses which are aligned with the feature, it is not possible to make a right depth initialization. Corollary 2.4.2 gives a formal explanation of this situation: a feature is aligned with the robot trajectory when the observation rays are parallel and the robot translation takes place in a direction which is parallel to the observation.

Let $\mathbf{x}_{r_i}$ for $i = 1, 2$ be the two robot poses where observations to a landmark j were taken and α_{ij} and $S_{\alpha_{ij}}$ be respectively the global bearings to the landmark given by (2.4) and their linearized propagated covariances. Let θ_t be the global inclination of the robot trajectory from $\mathbf{x}_{r_1}$ to $\mathbf{x}_{r_2}$,

$$\theta_t = \arctan\left(\frac{y_{r_2} - y_{r_1}}{x_{r_2} - x_{r_1}}\right),$$

and S_{θ_t} be its linearized propagated covariance.

Observation rays are parallel when

$$\frac{(\alpha_{1j} - \alpha_{2j})^2}{S_{\alpha 1j} + S_{\alpha_{2j}}} \leq \chi^2_{0.99,1d.o.f},$$

and the trajectory is parallel to the observation rays when

$$\frac{(\theta_t - \alpha_{ij})^2}{S_{\theta t} + S_{\alpha_{ij}}} \leq \chi^2_{0.99,1d.o.f},$$

for $j = 1, 2$.

The initialization of features is deferred until a pair of observations is available where the feature is not aligned with the trajectory. This delayed technique is less restrictive than the explained in section 2.5.4 and is specially useful for the inverse-depth parameterization since it allows the initialization and the use of features of infinite depth.

2.6 Discussion

In order to analyze the performance of the different parameterizations and initialization techniques, some experiments have been designed so that the performance and robustness of the algorithms to deal with bearing-only data can be analyzed. The experimentation and result analysis is carried out using a simulator which presents many benefits. First of all, *e*xactly the same experiment can be solved by several algorithms so that results are fully comparable. Besides, ground truth information is available and therefore obtained results can be compared with the true situation.

In the simulated experiments, an observation noise with an standard deviation of 0.125 degrees is used. Features are placed on the walls of a squared room. An initialization to the system is introduced from three robot poses and the first 5 observed landmarks. It is based on SFM techniques with the Trifocal Tensor [120]. The data association problem is not discussed in this chapter and data association is supposed to be perfect. Algorithms have been tested in different scenarios and under different conditions of *visibility*, *trajectory* and *step sizes*. The sensor visibility affects to the number of visible landmarks. Two possibilities are evaluated: total visibility, where all features are visible from all robot poses, and section visibility, where the workspace is divided into four sections and, in every step, robot observes the features within its section and a few from the neighborhood in order to connect the sections. Observe that when the visibility is total, no loop closing takes place and distant features are used.

As stated in Section 2.4, the robot trajectory has an important influence on depth computation in such a way that if landmark is on the direction of robot translation, depth computation is an undetermined problem. Two trajectories have been evaluated. The first is a squared trajectory composed by several pure translation motions and four 90° pure rotations. In this trajectory some features are aligned with the robot movement for many steps. The odometry noise is introduced as a function of the step size (st) and it can be seen in Table 2.1, columns Pure translation and Pure rotation. The second trajectory is circular. The robot describes a circumference when moving along the environment which gives rise to mixed rotations and translations. No feature in the map is really aligned with the trajectory, although for small step sizes features may seem to be in the direction of the robot translation. The standard deviations of the odometry noise are shown in Table 2.1 (column Mixed motion). The Step Size st determines the distance (in meters)

Table 2.1: Odometry noise relative to the step size (st).

Standard deviation	Pure translation	Pure rotation	Mixed motion
x_r	0.01 st	0.03 st	0.03 st
y_r	0.01 st	0.03 st	0.03 st
θ_r	2°	2.5°	2.5°

between two consecutive robot poses. This is the parameter which affects the most the behavior of the algorithms. Step sizes of 0.125 m, 0.250 m, 0.5 m and 1 m are tested.

The variables used in order to analyze the performance of an algorithm are listed below.

- **Final divergence**: Percent of results where the final robot pose diverges from its estimation. The condition which is tested for each component (x_r, y_r, θ_r) independently can be written as

$$\frac{(a-\widehat{a})^2}{P_a} \leq \chi^2_{0.99,1},$$

being a the ground-truth for x_r, y_r, θ_r, whereas $\widehat{a}$ and P_a are the estimated value and covariance for x_r, y_r and θ_r.

- **Map consistency**: Percent of features in the final map whose estimation is consistent with the ground truth. A feature j is considered inconsistent if its estimated x_j or y_j have a great estimation error,

$$\frac{|a-\widehat{a}|}{+\sqrt{P_a\ \chi^2_{0.99,1}}} \leq 1.5$$

being a, $\widehat{a}$ and P_a respectively the ground-truth, the estimate and covariance of either the x_j or y_j coordinates.

- **Trajectory divergence**: Percent of steps in the trajectory where the robot pose estimate (x_r, y_r, θ_r) diverges.

- **Feature initialization step**: Average of the number of steps needed to initialize a feature, computed as the difference between the step when a feature is observed for the first time, and the step when the feature is introduced into the map.

- **Feature usage**: Average of the feature usage per step, computed as the percent of features used in the filter update versus the number of features observed.

- **Map consistency per step**: Average of the percent of consistent features in the map in every step.

Additionally, information related to the precision and error of the final robot pose, the trajectory and the final map has also been studied.

A total of 160 experiments have been designed, and all of them have been solved using the algorithms discussed in Section 2.5. For the inverse-depth undelayed, a minimal depth $d_{min} = 0.5m$ is used. The results are analyzed in three different blocks. In the first we compare the cartesian delayed algorithms. In the second, we compare all inverse-depth delayed approaches, and in the third block a general comparison is carried out where the best of the cartesian delayed algorithms and the inverse-depth delayed method which performs better are compared to the inverse-depth undelayed algorithm.

The results obtained by the cartesian delayed algorithms can be found in Fig. 2.1. The cartesian delayed until finite depth (**xy-f**) algorithm performs better than the delayed with two observations (**xy-d**) and the delayed until features not aligned (**xy-l**) methods. The

final divergence (Fig. 2.1.a) and trajectory divergence (Fig. 2.1.c) are the lowest for all step sizes, the map consistency (Figs. 2.1.b, .f) are the highest, and the number of features used to update (Fig. 2.1.e) is higher than the used by the other cartesian algorithms for all step sizes even though this algorithm needs more steps to initialize a feature (Fig. 2.1.d).

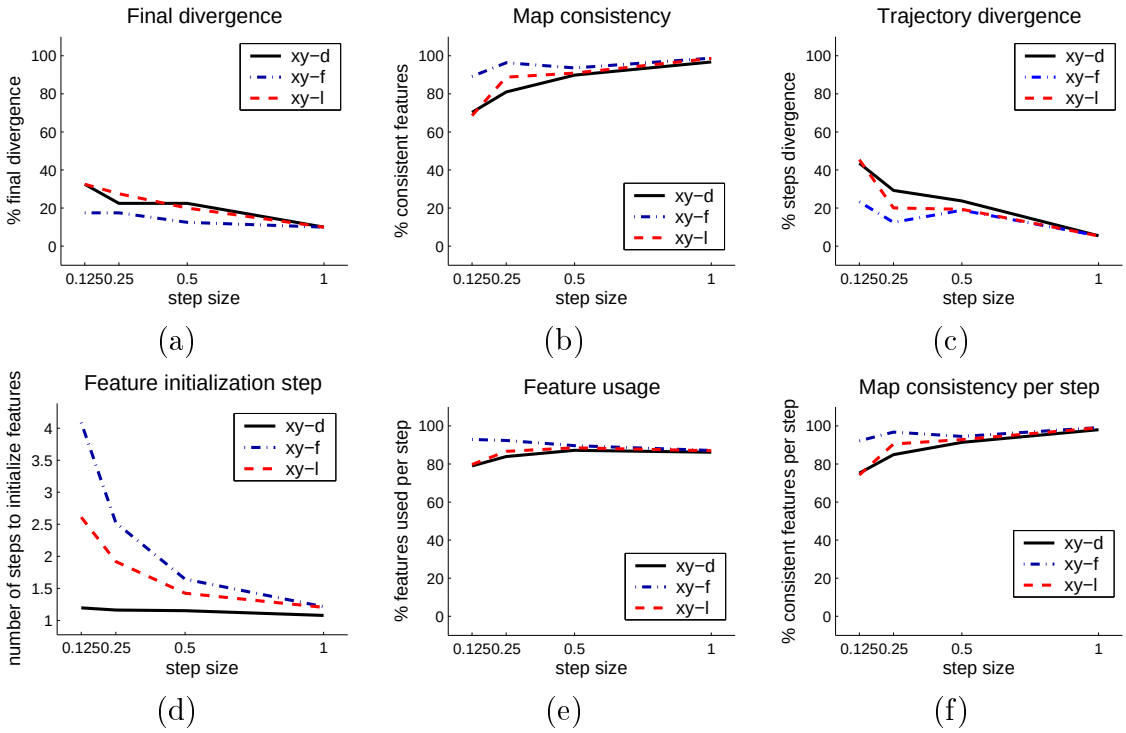


Figure 2.1: Cartesian delayed techniques comparison. Analysis of the results for different step sizes (x-axis). The algorithms used are *cartesian delayed*. **xy-d**: *with two observations*. **xy-f**: *until finite depth*. **xy-l**: *until feature not aligned with robot poses*.

From the study of the results obtained by the inverse-depth delayed algorithms, we can observe that all algorithms performed in a very similar way (Fig. 2.2). The final divergence (Fig. 2.2.a), map consistency (Fig. 2.2.b), trajectory divergence (Fig. 2.2.c), feature usage (Fig. 2.2.e), and map consistency per step (Fig. 2.2.f) results are similar for all inverse-depth delayed algorithms for the different step sizes. Only the feature initialization step (Fig. 2.2.d) differs due to the use of the different delayed strategies.

An especial study is carried out in order to compare the capability of the inverse-depth algorithms to deal with features which are observed during many steps as aligned with the trajectory. The most critical situation is when the robot moves following an squared trajectory and only observes landmarks within its section. In this situation the problematic features are F12, F23, and F34 (Fig. 2.3). In this figure, the ground-truth robot trajectory and landmark positions are displayed in red, while the estimates and uncertainties calculated by the algorithms are drawn in blue. As can be observed, both the trajectory and the landmark positions have been correctly estimated in all cases. However, features F12, F23 and F34 present high uncertainty (Fig. 2.3.a) when the algorithm used

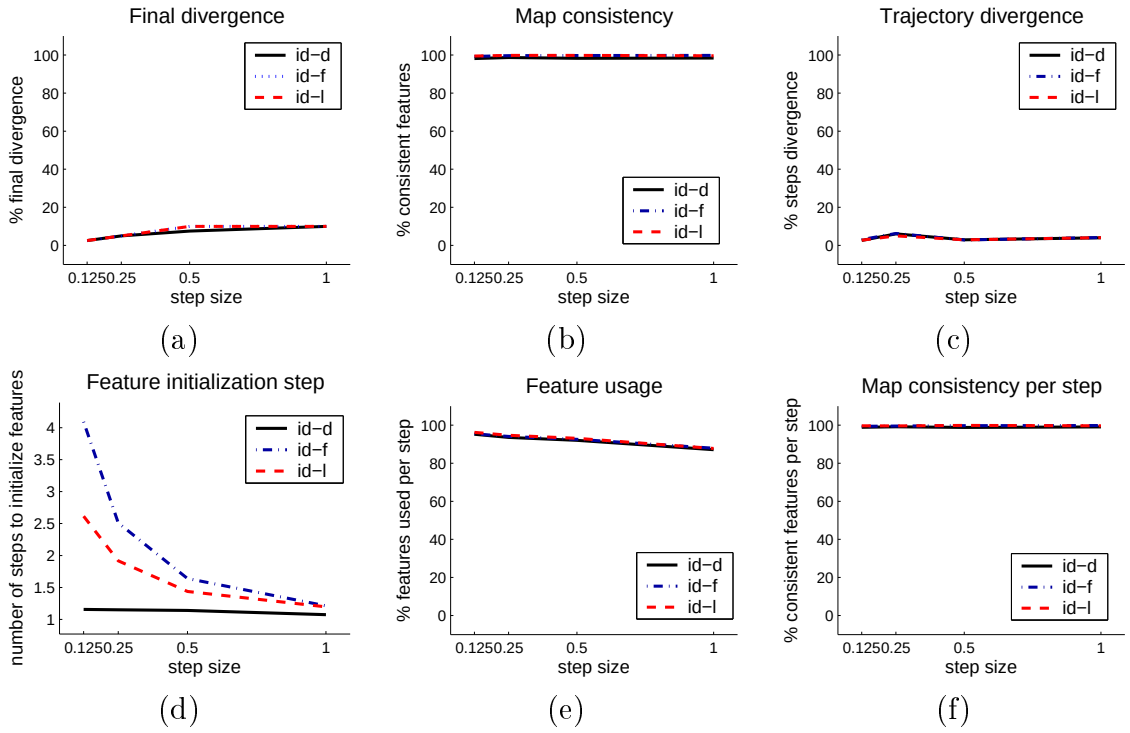
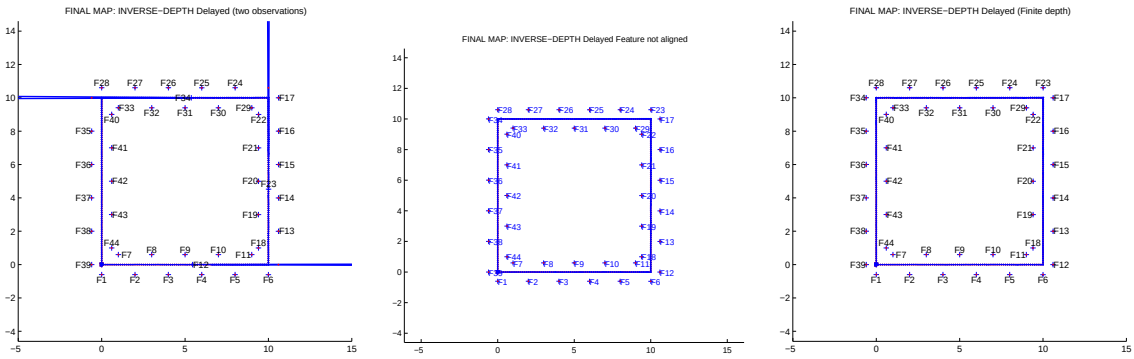


Figure 2.2: Inverse-depth delayed comparison. Analysis of the results for different step sizes (x-axis). The algorithms are: **id-d**: *with two observations.* **id-f**: *until finite depth.* **id-l**: *until feature not aligned with robot poses.*

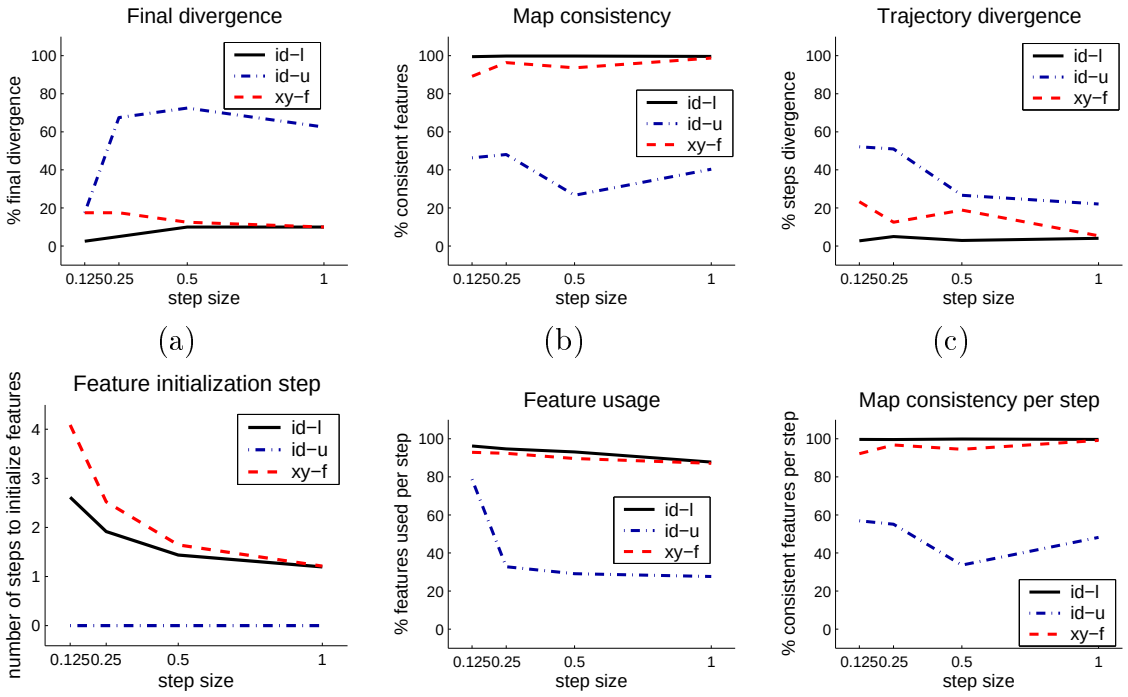
is the inverse-depth with two observations (**id-d**). Paying attention to the problematic features (F12, F23, F34) in Fig. 2.3 we can observe the results of an earlier initialization of features aligned with the trajectory. Even though their initial estimate and covariance correctly represent the feature position, posterior observations are not able to correct its position due to the huge innovation. The inverse-depth delayed until finite depth (**id-f**) and the inverse-depth delayed until feature not aligned with robot poses (**id-l**) algorithms performed in a similar way. However, the second is preferred because of its capability to initialize and use features of infinite depth.

Finally, the inverse-depth undelayed, with $d_{min} = 0.5m$, the inverse-depth delayed until feature not aligned with robot poses, and the cartesian delayed until finite depth algorithms are globally compared and their results are analyzed in order to find the one which performs better. As can be observed in Fig. 2.4, the behavior of the inverse-depth undelayed algorithm (**id-u**) is seriously affected by the step size. For the smallest step size (0.125m), almost all experiments converged in the last robot pose (Fig. 2.4.a) while for the other step sizes, many experiments diverged. The number of consistent features in the final map (Fig. 2.4.b) is lower than for the other algorithms. This behavior is also observed for the number of consistent features per step (Fig. 2.4.f). The behavior of the cartesian delayed until finite depth (**xy-f**) algorithm is not so much affected by the step size, although a better performance is observed when the step size increases. Its final divergence (Fig. 2.4.a) is slightly higher for smaller step sizes. The number of



(a) (b) (c)

Figure 2.3: (a) Inverse-depth delayed with two observations. (b) Inverse-depth delayed until feature not aligned with robot poses. (c) Inverse-depth delayed until finite depth.



(a) (b) (c)

(d) (e) (f)

Figure 2.4: Global comparison. Analysis of the results for different step sizes (x-axis). The algorithms used are: **id-u**: *inverse-depth undelayed*, $d_{min} = 0.5m$. **xy-f**: *cartesian delayed until finite depth.* **id-l**: *inverse-depth delayed until feature not aligned with robot poses.*

consistent features in its final map (Fig. 2.4.b) and along the steps (Fig. 2.4.f) slightly decreases for smaller step sizes. And its feature usage (Fig. 2.4.e) remains high for all step sizes. The algorithm based on inverse-depth delayed until features not aligned (**id-l**) produced the best results exhibiting an stable behavior for all step sizes. Almost all experiments converged both during the last step (Fig. 2.4.a) and along the trajectory

(Fig. 2.4.c). Almost all features are consistent in the final map (Fig. 2.4.b) and along the steps (Fig. 2.4.f), and the feature usage is the highest (Fig. 2.4.e).

An interesting information about the features usage can be extracted from Figs. 2.4.d, .e. It can be observed that when an undelayed strategy is selected, the percent of features used to update the map at each step (Fig. 2.4.e) is much lower than the used by the delayed algorithms even though features initialization requires a lower number of steps (Fig. 2.4.d). Therefore, delayed techniques provide important benefits due to the fact that the initial estimates introduced into the map are better with lower covariance.

2.7 Conclusions

In this chapter we have discussed feature parameterization and initialization using bearing-only measurements. Both considerably affect the results of the algorithms. However, this chapter shows that even with a perfect feature parameterization, if the initialization problem is ill-conditioned the results are inconsistent. As conclusion we can state that in general situations the delayed inverse-depth until features not aligned performs competitively.

An interesting result of this study is the related to the cartesian parameterization when it is combined with a finite depth test. It was expected that cartesian algorithm based in triangulation techniques were to suffer a great degradation of their performance for small step sizes. However, results show that the algorithm delayed until finite depth with cartesian parameterization is not very sensitive to the step size and exhibits very competitive results. Other interesting conclusion is that introducing features earlier in the EKF does not mean that more/better information will be available to update the state.

In this chapter we have also stated ill-conditioned situations: a pure rotation motion and features aligned with the trajectory. None of them can be managed in any case. Some results have been given to detect these situations which will allow the algorithms to decide which data can be used in each instant.

Chapter 3
Static Map Merging

In this chapter we present a solution for merging feature-based maps in a robotic network with limited communication. We consider a team of robots that have explored an unknown environment and have built local stochastic maps of the explored region. After the exploration has taken place, the robots communicate and build a global map of the environment. This problem has been traditionally addressed using centralized schemes or broadcasting methods. The contribution of the work presented in this chapter is the design of a fully distributed approach which is implementable in scenarios with limited communication. Our solution does not rely on a particular communication topology and does not require any central node, making the system robust to individual failures. Information is exchanged exclusively between neighboring robots in the communication graph. We give worst-case performance bounds for computational complexity, memory usage, and communication load. We validate our results trough simulations.

3.1 Introduction

The increasing interest in multi-robot applications is motivated by the wealth of possibilities offered by teams of robots cooperatively performing collective tasks. The efficiency and robustness of these teams goes well beyond what individual robots can do. In addition to the classical issues associated to the operation of individual robots, these scenarios introduce novel challenges specific to the coordination of multiple robots. An important issue associated to the operation of a robot team is perception. In general, each robot just observes a portion of the environment. In order to make decisions in a coordinated way, the robots must fuse their local observations into a global map. Many existent solutions for single robot perception have been extended to multi-robot scenarios under centralized schemes, full communication between the robots, or broadcasting methods. In [117] a single global map is updated by all the robots. Robots search for features in the global map that have been observed by themselves along the exploration. Then, they use these coincident features to compute implicit measurements (the difference between the Cartesian coordinates of equal features must be zero) and use these constrains to update the map. In [56] maps are represented as constraint graphs, where nodes are scans measured from a robot pose and edges represent the difference between pairs of robot poses. Robot to robot measurements are used to merge two local maps into a single map. An optimization phase must be carried out in order to transform the constraint graph into a Cartesian

map. [42] also represents the global map using a graph. Nodes are local metric maps and edges describe relative positions between adjacent local maps. The map merging process consists of adding an edge between the maps. Global optimization techniques are applied to obtain the global metric map. [159] merges two maps into a single one using robot to robot measurements to align the two maps and then detecting duplicated landmarks and imposing the implicit measurement constraints. Particle filters have been generalized to multi-robot systems assuming that the robots broadcast their controls and their observations [69]. The Constrained Local Submap Filter has been extended to the multi-robot case assuming that each robot builds a local submap and broadcasts it, or transmits it to a central node [149]. Methods based on graph maps of laser scans [56, 78, 110] make each robot build a new node and broadcast it. The same solution could be applied for many existing submap approaches [108].

The previous methods require that each robot has the capability to communicate with all other robots at every time instant or with a central node. Centralized strategies, where a central node compiles all the information from other robots, performs the computations, and propagates the processed information or decisions to all the robots, have several drawbacks. The whole system can fail if the central node fails, leader selection algorithms may be needed, and a (direct or indirect) communication of all robots with the central system may be required. On the other hand, in distributed systems, all robots play the same role, and therefore the computations can be distributed among all the robots. In addition, distributed systems are naturally more robust to individual failures. In distributed scenarios you cannot assume that the robots can communicate with all other robots at every time instant. A more realistic situation is when, at any time instant, robots can communicate only with a limited number of other robots, e.g., robots within a specific distance. These situations can be best modeled using communication graphs, where nodes correspond to the robots and edges represent communication capabilities between them. Additionally, since robots are moving, the topology of the graph may vary along the time, given rise to switching topologies, see for instance [27]. We are interested in map merging solutions for robotic systems with range limited communication, and where the computations are distributed among the robots.

There has been an intensive recent research in distributed implementations of the Kalman Filter that make use of its information form (IF). Measurement updates in IF are additive and therefore information coming from different sensors can be fused in any order and at any time. While optimal solutions exist for complete communication networks [96], for general communication schemes [100, 130] the delayed data problem leads to an approximate KF estimator. This problem appears when the robots execute the state prediction without having incorporated all the measurements taken at the current step. As a result, their estimates become suboptimal and give rise to disagreement. The effects of this delayed data problem have been studied in [34]. A solution that reduces this disagreement has been presented [101] and its convergence has been proved in the absence of observation and system noises. However, this solution does not consider system inputs, which usually model odometry measurements in typical robotic applications. Therefore, it does not solve its associated delayed data problem. Other methods have been proposed that require the previous offline computation of the gains and weights of the algorithm

and that are only applicable when the variance of measurement noises is constant and a priori known [1]. Due to the limitations of these methods, the formulation of a multi-robot perception problem as a distributed estimator presents multiple obstacles.

Here, instead we formulate the map merging as a sensor fusion problem, where each robot can be seen as a sensor and its local map as a measurement. Instead of maintaining a global estimator, we let each robot maintain its own local estimator, i.e., build its local map using exclusively measurements acquired by itself. The information received from other robots is introduced into its estimated global map, but not into its local map. Sensor fusion approaches [31] present the inconvenience that the successive measurements, in our case local maps, from the same robot must be independent. In a map merging scenario this does not hold, since the local map of a robot is an evolution of any of its previous maps. However, since we discuss a static map merging where the maps are fused after the exploration, our approach does not suffer from this limitation. In our solution, the local maps of the robots are expressed in IF form and they are fused in an additive fashion using a consensus filter [31, 152] to provide a distributed implementation.

In this chapter, we propose a solution to the map merging problem for a robotic network modeled by a communication graph, where all computations are distributed among the robots and where, at every time step, robots only use its own (local) data and the information received from its neighbors in the graph. The solution is based on distributed average consensus algorithms for data fusion problems [129, 152].

3.2 Problem Description

Throughout the chapter we use the following notation (Table 3.1):

Table 3.1: Notation.

Symbol	Usage
i, i', j, j'	robot index
r, r', s, s'	feature index
G	index used for referring to the global reference frame and map
t	iteration number, $t \in \mathbb{N}$
f_r^i	the r feature observed by the i robot
$A_{r,s}$	the (r,s) entry of matrix A
A_{ij}	the block (i,j) of matrix A defined by blocks
$[A_{ij}]_{r,s}$	the (r,s) entry of A_{ij}
$\mathcal{M}_i$	size of the map of robot i
$\mathcal{M}_G$	size of the global map

We consider a team of $n \in \mathbb{N}$ robots with limited communication capabilities. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the undirected communication graph. The nodes are the robots, $\mathcal{V} = \{1, \ldots, n\}$. If two robots i, j can exchange information, then there is an edge $(i,j) \in \mathcal{E}$ between them. Let $\mathcal{N}_i$ be the set of neighbors of robot i,

$$\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E}, j \neq i\}.$$

The robots have explored an unknown environment. Along its operation, each robot i has observed $m_i \in \mathbb{N}$ features whose true positions are unknown. Based on its own observations, each robot i has estimated its own pose together with the positions of the features. It has built a stochastic map with $\mathcal{M}_i = \texttt{szr} + m_i\ \texttt{szf}$ elements, composed of a mean $\hat{\mathbf{x}}_i \in \mathbb{R}^{\mathcal{M}_i}$ and covariance matrix $\Sigma_i \in \mathbb{R}^{\mathcal{M}_i \times \mathcal{M}_i}$. The constants `szr`, `szf` represent the size of, respectively, a robot pose and a feature position; $\texttt{szr} = 3$ for planar motions, where the robot position (x, y) and orientation θ are estimated; $\texttt{szf} = 2$ or $\texttt{szf} = 3$ for respectively 2D or 3D environments. If $\mathbf{x}_i \in \mathbb{R}^{\mathcal{M}_i}$ is the vector with the true robot pose after the exploration and with the true positions of the m_i features, then

$$\hat{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{v}_i, \tag{3.1}$$

where $\mathbf{v}_i$ is a zero mean noise with covariance matrix Σ_i. The aim is to merge the local maps to obtain a global estimate of the map.

Let $m \in \mathbb{N}$ be the number of different features in the environment. As noted above, each robot $i \in \{1, \ldots, n\}$ has observed $m_i \leq m$ of them. Let $\mathbf{x} \in \mathbb{R}^{\mathcal{M}_G}$ contain the true poses of the n robots and the true positions of the m features, where $\mathcal{M}_G = n\ \texttt{szr} + m\ \texttt{szf}$. Let $H_i \in \{0, 1\}^{\mathcal{M}_i \times \mathcal{M}_G}$ be the observation matrix that relates the elements in $\mathbf{x}$ with the elements observed by robot i. It is a binary matrix, i.e., each entry is equal to 0 or 1, where there is at most a 1 per row. Since $\mathbf{x}_i = H_i\mathbf{x}$, eq. (3.1) becomes

$$\hat{\mathbf{x}}_i = H_i\mathbf{x} + \mathbf{v}_i,$$

where $\mathbf{v}_i$ has zero mean and covariance matrix Σ_i. Considering all the local maps together, we have that

$$\left(\hat{\mathbf{x}}_1^T \ldots \hat{\mathbf{x}}_n^T\right)^T = \left(H_1^T \ldots H_n^T\right)^T \mathbf{x} + \left(\mathbf{v}_1^T \ldots \mathbf{v}_n^T\right)^T.$$

We assume that the noises $\mathbf{v}_i$ are independent since every robot has constructed the map based on its own observations, and thus $E[\left(\mathbf{v}_1^T \ldots \mathbf{v}_n^T\right)^T \left(\mathbf{v}_1^T \ldots \mathbf{v}_n^T\right)] = \text{diag}\left(\Sigma_1, \ldots, \Sigma_n\right)$. The local map of each robot i is represented in IF form by its information matrix $I_i \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ and its information vector $\mathbf{i}_i \in \mathbb{R}^{\mathcal{M}_G}$,

$$I_i = H_i^T \Sigma_i^{-1} H_i, \qquad \mathbf{i}_i = H_i^T \Sigma_i^{-1} \hat{\mathbf{x}}_i. \tag{3.2}$$

Given the n local maps in IF form, the operation that merges their information and produces the global map,

$$I_G = \sum_{i=1}^{n} I_i, \qquad \mathbf{i}_G = \sum_{i=1}^{n} \mathbf{i}_i, \tag{3.3}$$

is additive, commutative, and associative. For this reason, merging the maps in IF form is a common practice [137]. The mean and covariance matrix of the global map are

$$\hat{\mathbf{x}}_G = (I_G)^{-1}\ \mathbf{i}_G, \qquad \Sigma_G = (I_G)^{-1}. \tag{3.4}$$

The goal is for each robot to compute the global map (3.3)-(3.4) in a distributed fashion.

3.3 Distributed Averaging

There is a rich literature in sensor networks, where the observations taken by a set of sensors are fused in IF form to build a better estimate of a variable. A widely used distributed sensor fusion method whose convergence conditions and properties have been deeply studied is proposed in [152]. Each robot i maintains the variables $\hat{I}^i_G(t) \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$, $\hat{\mathbf{i}}^i_G(t) \in \mathbb{R}^{\mathcal{M}_G}$, initialized as

$$\hat{I}^i_G(0) = H_i^T \Sigma_i^{-1} H_i, \qquad \hat{\mathbf{i}}^i_G(0) = H_i^T \Sigma_i^{-1} \hat{\mathbf{x}}_i, \tag{3.5}$$

and updated at each time step $t \geq 0$ by

$$\hat{I}^i_G(t+1) = \sum_{j=1}^{n} W_{i,j} \hat{I}^j_G(t), \qquad \hat{\mathbf{i}}^i_G(t+1) = \sum_{j=1}^{n} W_{i,j} \hat{\mathbf{i}}^j_G(t), \tag{3.6}$$

where $\hat{\mathbf{x}}_i$, Σ_i is the local map of robot i and $W_{i,j}$ are the Metropolis weights of $\mathcal{G}$ given by eq. (A.3) in Appendix A. This is a distributed averaging algorithm that, for a fixed connected communication graph $\mathcal{G}$, or a time-varying jointly connected graph, guarantees that the estimates asymptotically converge to the average of the initial states,

$$\lim_{t\to\infty} \hat{I}^i_G(t) = I_G/n, \qquad \lim_{t\to\infty} \hat{\mathbf{i}}^i_G(t) = \mathbf{i}_G/n, \tag{3.7}$$

being I_G, $\mathbf{i}_G$ the information matrix and vector of the global map (3.3). More information about averaging algorithms can be found in Appendix A. Then, the variables $\hat{\mathbf{x}}^i_G(t) \in \mathbb{R}^{\mathcal{M}_G}$ and $\hat{\Sigma}^i_G(t) \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ at each robot $i \in \mathcal{V}$ and each $t \geq 0$, defined as

$$\hat{\mathbf{x}}^i_G(t) = \left(\hat{I}^i_G(t)\right)^{-1} \hat{\mathbf{i}}^i_G(t), \qquad \hat{\Sigma}^i_G(t) = \left(\hat{I}^i_G(t)\right)^{-1}, \tag{3.8}$$

asymptotically converge to

$$\lim_{t\to\infty} \hat{\mathbf{x}}^i_G(t) = \hat{\mathbf{x}}_G, \qquad \lim_{t\to\infty} \hat{\Sigma}^i_G(t) = n\Sigma_G, \tag{3.9}$$

where $\hat{\mathbf{x}}_G$, Σ_G are the mean and covariance of the global map in (3.4). The mean $\hat{\mathbf{x}}^i_G(t)$ and covariance $\hat{\mathbf{x}}^i_G(t)$ estimated by each robot $i \in \{1, \ldots, n\}$ executing this sensor fusion algorithm have the following properties:

(i) the mean is an unbiased estimate [152] of the truth $\mathbf{x}$ and $\mathbf{E}\left[\hat{\mathbf{x}}^i_G(t)\right] = \mathbf{x}$ for all $t \geq 0$;

(ii) the temporal estimates are consistent [31, Lemma 2.1] for all $t \geq 0$ since the true uncertainty $Q^i_G(t)$ is smaller than the estimated uncertainty, $Q^i_G(t) \preceq \hat{\Sigma}^i_G(t)$.

This algorithm is fully distributed because the robots only use information about its direct neighbors in the communication graph ($W_{i,j} = 0$ if $j \notin \mathcal{N}_i$). Besides, under mild connectivity conditions, it converges to the average even if the communication graph is time-varying, $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$. The convergence is asymptotic, and hence the global map is obtained by each robot i in the limit as $t \to \infty$. However, for complete communication graphs, the strategy converges in a single iteration.

3.4 Consensus on the Global Map

Throughout this chapter we assume that the local maps have been previously expressed in a common reference and the data association has been solved. These issues are discussed in Chapters 5 and 6. In this section we address the problem of fusing the maps once they are expressed in the same reference frame, and the robots know the data association.

In the previous section we presented a sensor fusion algorithm [152] (eqs. (3.5)-(3.9)) with several interesting properties. The use of sensor fusion algorithms for merging stochastic maps has an important difficulty that must be addressed. Sensors usually observe a set of variables which are a priori known, e.g., temperature, humidity, etc. Specifically, during the initialization of the algorithm (Section 3.3, eq. (3.5)) each robot i knows the observation matrix H_i relating its measurements with the variables to be estimated. In a map merging scenario, this matrix H_i relates the local features observed by robot i with the globally observed ones. However, in our case the robots do not know the observation matrices since they do not know the whole set of features that have been observed by the robot team. Initially each robot i exclusively knows the labels $\mathcal{L}_i$ of its local features. The goal is that each robot discovers the feature labeling of all the robots through the interaction with its neighbors. We propose an algorithm where the feature labeling discovering and the data fusion are executed simultaneously.

After solving the data association (Chapter 5), each robot $i \in \mathcal{V}$ has its label set $\mathcal{L}_i = \{L_1^i, \ldots, L_{m_i}^i\}$ with the labels of its m_i features. Two features f_r^i and f_s^j from robots i, j can be fused together if and only if their labels $L_r^i = (i_\star, r_\star)$, $L_s^j = (i'_\star, r'_\star)$ have the same value,

$$i_\star = i'_\star \quad \text{and} \quad r_\star = r'_\star.$$

However, in the initialization stage in Section 3.3, (3.5), the robots do not know the label sets $\mathcal{L}_j$ from the other robots, and thus cannot compute their matrix H_i.

Throughout the discussion, we use a sorted version of the label sets $\mathcal{L}_i$ that we term label vectors L_i. In a label vector L_i, the labels $L_r^i = (i_\star, r_\star)$ appear sorted following the lexicographic order (first by $i_\star$, then by $r_\star$). We assume that initially each robot i sorts its set $\mathcal{L}_i$ to create its label vector L_i, and that it arranges its local map accordingly. The robot poses which have been estimated are always placed together at the first rows and columns of the information matrices and vectors. In this case, the labels are exclusively composed of the robot id. Let $|L_i|$ denote the number of labels in L_i. Given two label vectors L_i, L_j, we say that $L_i \subseteq L_j$ if all labels of L_i are contained in L_j. We let Γ be a function that joins two label vectors L_i, L_j and returns a new one L_{ij} with all the labels in L_i and L_j, without duplicates, and sorted as prescribed above. It is not difficult to see that the function Γ satisfies the following properties:

$$\begin{aligned}
&(i)\ \Gamma(L_i, L_j) = \Gamma(L_j, L_i);\\
&(ii)\ \Gamma(L_i, L_i) = L_i;\\
&(iii)\ \Gamma(L_i, \Gamma(L_j, L_k)) = \Gamma(\Gamma(L_i, L_j), L_k);\\
&(iv)\ L_i, L_j \subseteq \Gamma(L_i, L_j);
\end{aligned}$$

for any label vectors $L_i, L_j, L_{j'}$.

We let $\mathcal{H}$ be a function that computes the observation matrix between two label vectors L_i, L_j with $L_i \subseteq L_j$. $\mathcal{H}(L_i, L_j)$ returns a matrix $H_j^i \in \{0,1\}^{|L_i|\ \mathtt{szf} \times |L_j|\ \mathtt{szf}}$ that relates the Cartesian coordinates of the features with labels L_i and the ones with labels L_j. It can be seen that $\mathcal{H}$ has the property that, if $L_i \subseteq L_j \subseteq L_{j'}$, $\mathcal{H}(L_i, L_j) = H_j^i$, and $\mathcal{H}(L_j, L_{j'}) = H_{j'}^j$, then $H_j^i H_{j'}^j = H_{j'}^i = \mathcal{H}(L_i, L_{j'})$. Additionally, $\mathcal{H}$ also satisfies $\mathcal{H}(L_i, L_i) = I$.

Given the labels of all the robots L_j, for $j = 1, \ldots, n$, we let L_G be the global label vector which contains all the m *different* labels in $L_1, \ldots, L_n$,

$$L_G = \Gamma(\Gamma(\ldots \Gamma(L_1, L_2), \ldots), L_n). \tag{3.10}$$

Note that the same L_G is obtained if the functions Γ are applied to the label vectors L_i in any other order, as long as each L_i, $i \in \{1, \ldots, n\}$, is used at least once. Each observation matrix H_i in (3.5) is then $\mathcal{H}(L_i, L_G)$. We propose two alternative approaches to solve the problem of the global map merging. The first approach begins by reaching consensus on the global label vector L_G, and then computes the matrices H_i and executes the sensor fusion algorithm in Section 3.3. The second approach does not require the initial consensus stage on the global label vector L_G. We let each robot i start with its own L_i and, incrementally, incorporate the new labels discovered in its neighbors data, to finally discover the full L_G. Simultaneously, each robot i arranges its information matrix and vector at each iteration t according to this information.

3.5 Partially Distributed Approach

Our first approach to the problem of distributed map merging consists of the following steps:

1. **BFS tree construction:** The robots begin by constructing a BFS spanning tree in the undirected graph using a variation of the flood with termination algorithm, see e.g., [109]. As a result, all nodes in the graph know the identity of its parent and its children in the graph, and also know its role (root, leaf or regular node).

2. **Computation of the global label vector:** After this, the leaves initiate the incremental computation of the vector with the labels of all the features observed by the robots. Robots fuse the label vectors from their children with their own label vectors and send this data to their parents. When all information is available to the root, it computes the final global label vector and uses a flooding algorithm to propagate this information to all nodes in the graph.

3. **Distributed averaging:** Finally, the nodes compute the matrices H_i and initiate the basic sensor fusion algorithm in Section 3.3 to compute the global map.

Next, we describe each of these steps in more detail.

BFS Tree Construction

In order to construct the BFS tree, all nodes in the undirected graph must know if they are the root and also must know which nodes are their neighbors in the graph.

All nodes initialize 'parent id' to null and 'children set' to be the set of neighbors. The root node initiates the process sending a 'parent request' to all its neighbors. When a node receives a 'parent request' message, it checks the value of its 'parent id'; if it is null, then it updates this value to be the sender id; if the node already has a parent, it replays with a 'parent reject' message. If during a step a node receives multiple 'parent request' messages, it selects as a parent the node with the smallest id and sends a 'parent reject' message to the other nodes. Nodes remove the parent id from the list of children. When a node receives a 'parent reject' message, it updates its children set, deleting the sender from this set. A node with an empty children set is a leaf.

Some steps of this algorithm are illustrated in Fig. 3.1.

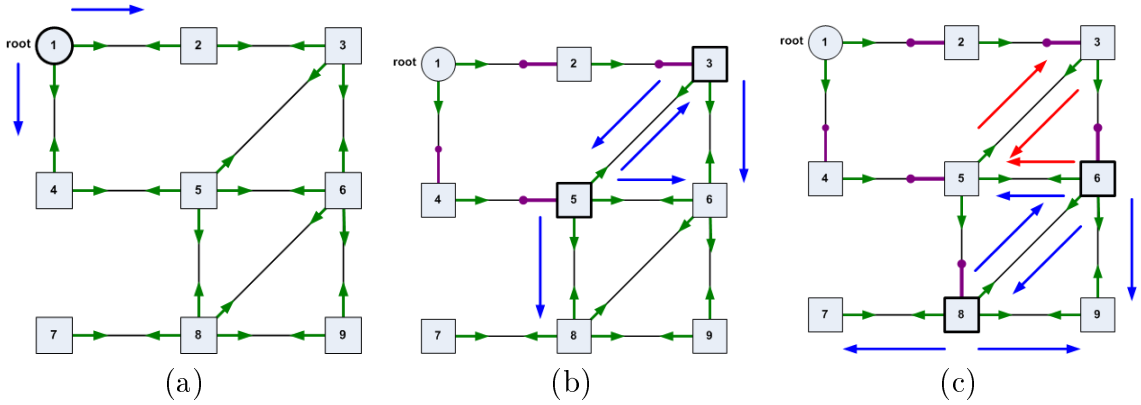


Figure 3.1: **Example of BFS tree construction**. **(a)**: The root (circle) initiates the process sending a parent request to all its children (blue arrows). **(b)**: Nodes 3 and 5 update their parent and send parent requests (blue arrows) to all their children. **(c)**: Node 3 sends a parent reject (red arrow) to node 5 as a response to the parent request received. Node 5 behaves in the same way. Node 6 selects as parent the node with the minimal identifier (node 3) and sends a parent reject to node 5; then, it sends a parent request to all its children. Node 8 updates its parent and sends a parent request to its children.

Incremental Computation of the Global Label Vector

When a node detects that it is a leaf, it starts the process of computing the global label vector. Nodes build child label vectors L_i^{child} containing their own local labels L_i and the child label vectors L_j^{child} from their child nodes j. Each leave node i builds its child label with its own local labels, $L_i^{\text{child}} = L_i$ and sends it to its parent inside an 'up' message. Each node i in the graph compiles all the child label vectors $L_{j_1}^{\text{child}}, \ldots, L_{j_{|\text{child}_i|}}^{\text{child}}$ sent by its children child_i and fuses this information with its own identity vector L_i to create its child label vector, $L_i^{\text{child}} = \Gamma(\Gamma(\Gamma(\ldots\Gamma(L_{j_1}^{\text{child}}, L_{j_2}^{\text{child}}), \ldots), L_{j_{|\text{child}_i|}}^{\text{child}}), L_i)$. Once a node i has received 'up' messages from all its children, it sends an 'up' message to its parent with the

resulting child label vector L_i^{child}. When the root has received all the information from its children, it computes the final global label vector, $L_G = L_{\text{root}}^{\text{child}}$. This global vector contains all the labels of the features observed by all the robots in the team, without repetition, and sorted in lexicographic order. Then the root sends this final vector L_G to all its children in a 'down' message. Every node that receives a 'down' message, records the global label vector L_G and propagates this information sending a 'down' message to all its children.

Some steps of this algorithm are illustrated in Fig. 3.2.

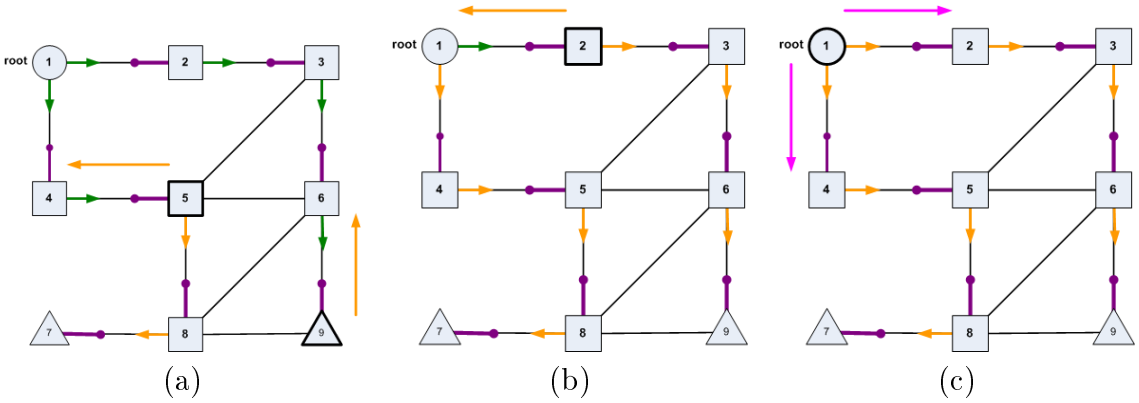


Figure 3.2: **Incremental computation of the global label vector**. **(a)**: Nodes 5 and 9 send an up message (yellow arrow) to their parent. **(b)**: The root receives an up message from node 2. It had previously received an up message from its other children (node 4). **(c)**: The root computes the final (global) parameter vector and starts a flooding process to communicate this vector to all the nodes. It sends a down message (pink arrow) to all its children.

Distributed Averaging

During the execution of the previous phase, all the robots receive the global label vector L_G. In this phase, each robot i computes the observation matrix H_i from its local L_i and the global L_G label vectors, $H_i = \mathcal{H}(L_i, L_G)$. Robots use their observation matrices H_i in eq. (3.5) for initializing their estimates of the global map information matrices and vectors $\hat{I}_G^i(t), \hat{\mathbf{i}}_G^i(t)$ and they update these estimates with the the sensor fusion algorithm (3.6).

3.6 Fully Distributed Approach

We let each robot i start with its own L_i and, incrementally, incorporate the new labels discovered in its neighbors data, to finally discover the full L_G. Each robot $i \in \{1, \ldots, n\}$ maintains a label vector $L_i(t)$ with the different labels discovered by itself up to time t. Additionally, it maintains the variables $\mathbf{i}_G^i(t) \in \mathbb{R}^{\mathcal{M}_i(t)}$ and $I_G^i(t) \in \mathbb{R}^{\mathcal{M}_i(t) \times \mathcal{M}_i(t)}$, where $\mathcal{M}_i(t)$ is the size of its estimated global map at time t, $\mathcal{M}_i(t) = |L_i(t)|$ `szf`. At $t = 0$,

robot i initializes these variables with its own local information,

$$L_i(0) = L_i, \qquad I_G^i(0) = \Sigma_i^{-1}, \qquad \mathbf{i}_G^i(0) = \Sigma_i^{-1}\hat{\mathbf{x}}_i. \tag{3.11}$$

At each iteration t, the robot incorporates into its label vector $L_i(t+1)$ the new labels discovered from its neighbors data $L_j(t)$, with $j \in \mathcal{N}_i \cup \{i\}$,

$$L_i(t+1) = \Gamma(\Gamma(\dots\Gamma(L_{j_1}(t), L_{j_2}(t)), \dots), L_{j_{\mathcal{N}_i+1}}(t)), \tag{3.12}$$

where $\{j_1, \dots, j_{\mathcal{N}_i+1}\} = \mathcal{N}_i \cup \{i\}$. Then, it arranges the variables $I_G^i(t)$ and $\mathbf{i}_G^i(t)$ accordingly, and computes the new values,

$$I_G^i(t+1) = \sum_{j=1}^{n} W_{i,j}(H_{i,t+1}^{j,t})^T I_G^j(t) H_{i,t+1}^{j,t}, \quad \mathbf{i}_G^i(t+1) = \sum_{j=1}^{n} W_{i,j}(H_{i,t+1}^{j,t})^T \mathbf{i}_G^j(t), \tag{3.13}$$

where $H_{i,t+1}^{j,t} = \mathcal{H}(L_j(t), L_i(t+1))$.

Proposition 3.6.1. *The outcomes of the algorithms* (3.11)-(3.13) *and* (3.6) *are related as follows: for all* $t \geq 0$ *and all* $i \in \{1, \dots, n\}$,

$$\hat{I}_G^i(t) = (H_G^{i,t})^T I_G^i(t) H_G^{i,t}, \qquad \hat{\mathbf{i}}_G^i(t) = (H_G^{i,t})^T \mathbf{i}_G^i(t), \tag{3.14}$$

where $H_G^{i,t} = \mathcal{H}(L_i(t), L_G)$. *Furthermore, after* $\operatorname{diam}(\mathcal{G})$ *iterations, the result of both algorithms is exactly the same, i.e., for all* $t \geq 0$ *and all* $i \in \{1, \dots, n\}$,

$$\hat{I}_G^i(\operatorname{diag}(\mathcal{G}) + t) = I_G^i(\operatorname{diag}(\mathcal{G}) + t), \qquad \hat{\mathbf{i}}_G^i(\operatorname{diag}(\mathcal{G}) + t) = \mathbf{i}_G^i(\operatorname{diag}(\mathcal{G}) + t). \tag{3.15}$$

Proof. We only present the proof for the information matrices $I_G^i(t)$ (the reasoning is analogous for the information vectors $\mathbf{i}_G^i(t)$). We reason by induction. At $t = 0$, equation (3.14) is satisfied since

$$\hat{I}_G^i(0) = H_i^T \Sigma_i^{-1} H_i = (H_G^{i,0})^T I_G^i(0) H_G^{i,0}, \tag{3.16}$$

for all $i \in \{1, \dots, n\}$. Assuming that (3.14) is true for a given t, then for all $i \in \{1, \dots, n\}$,

$$\begin{aligned} \hat{I}_G^i(t+1) &= \sum_{j=1}^{n} W_{i,j} \hat{I}_G^j(t) = \sum_{j=1}^{n} W_{i,j}(H_G^{j,t})^T I_G^j(t) H_G^{j,t} \\ &= (H_G^{i,t+1})^T \left[\sum_{j=1}^{n} (H_{i,t+1}^{j,t})^T I_G^j(t) H_{i,t+1}^{j,t} \right] H_G^{i,t+1} = (H_G^{i,t+1})^T I_G^i(t+1) H_G^{i,t+1}, \end{aligned} \tag{3.17}$$

where we have used the facts that for all $t \geq 0$ and all $i, j \in \{1, \dots, n\}$, $H_G^{j,t} = H_{i,t+1}^{j,t} H_G^{i,t+1}$ and $\sum_{j=1}^{n} W_{i,j} = 1$. This concludes the proof of (3.14). Regarding (3.15), note that after $\operatorname{diam}(\mathcal{G})$ iterations of the algorithm (3.11)-(3.13), each robot $i \in \{1, \dots, n\}$ has already incorporated into its label vector information from all the initial label vectors L_j, for $j = 1, \dots, n$. Therefore, for all $t \geq 0$, $L_i(\operatorname{diam}(\mathcal{G}) + t) = L_G$, where L_G is the global vector in (3.10). Then, $H_G^{i,\operatorname{diam}(\mathcal{G})+t} = \mathcal{H}(L_G, L_G) = I$, and the result follows. □

Using the previous algorithm, the robots compute the same global map than if they were given H_i from the beginning, with the additional benefit that the information matrices $I_G^i(t)$ can be inverted for any $t \geq 0$ and $i \in \{1, \ldots, n\}$, since they do not contain any non informative zero columns or rows. Note that in the basic map merging algorithm (Section 3.3), the information matrices $\hat{I}_G^i(t)$ are initialized with zero rows and columns for the features which are not local to robot i. Therefore, they cannot be inverted until robot i has received informative (non zero) data for all the features. In case all the robots have observed at least one exclusive feature, the robots have to wait for $t = \text{diam}(\mathcal{G})$ iterations for inverting their information matrices and recovering a temporal estimate of the global map. However, when the robots use the algorithm (3.11)-(3.13), their information matrices $I_G^i(t)$ can be inverted for any $t \geq 0$ and $i \in \{1, \ldots, n\}$ and thus the global map can be computed at any step. In the next sections, we will analyze the presented algorithm in terms of its theoretical and experimental performance. From now on, we let $\mathbf{x}_G^i(t) \in \mathbb{R}^{\mathcal{M}_i(t)}$, $\Sigma_G^i(t) \in \mathbb{R}^{\mathcal{M}_i(t) \times \mathcal{M}_i(t)}$ be the global map estimated by a robot i executing algorithm (3.11)-(3.13),

$$\mathbf{x}_G^i(t) = \left(I_G^i(t)\right)^{-1} \mathbf{i}_G^i(t), \qquad \Sigma_G^i(t) = \left(I_G^i(t)\right)^{-1}. \tag{3.18}$$

3.7 Properties

The convergence is asymptotic, and hence the global map is obtained by each robot i in the limit as $t \to \infty$. However, for complete communication graphs, the strategy converges in a single iteration. For general networks, the intermediate estimates (3.8) have interesting properties that allow their use at any time t:

(i) the intermediate estimates $\hat{\mathbf{x}}_G^i(t)$ are unbiased,

$$\mathbf{E}\left[\hat{\mathbf{x}}_G^i(t)\right] = \mathbf{x},$$

for all $t \geq 0$ and all $i \in \mathcal{V}$ such that the information matrix $\hat{I}_G^i(t)$ can be inverted [152, Theorem 3];

(ii) the numerical covariance $Q_G^i(t) \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ of $\hat{\mathbf{x}}_G^i(t)$ is bounded by the locally estimated global map covariance $\hat{\Sigma}_G^i(t)$,

$$\begin{aligned} Q_G^i(t) &= \mathbf{E}\left[\left(\hat{\mathbf{x}}_G^i(t) - \mathbf{x}\right)\left(\hat{\mathbf{x}}_G^i(t) - \mathbf{x}\right)^T\right] \\ &= \left(\hat{I}_G^i(t)\right)^{-1} \left(\sum_{j=1}^{n} ([W^t]_{i,j})^2 H_j^T \Sigma_j^{-1} H_j\right) \left(\hat{I}_G^i(t)\right)^{-1}, \end{aligned}$$

for all $t \geq 0$ and all $i \in \mathcal{V}$ such that $\hat{I}_G^i(t)$ can be inverted. Since $0 \leq [W^t]_{i,j} \leq 1$ for all $i, j \in \mathcal{V}$ and all $t \geq 0$, then $([W^t]_{i,j})^2 \leq [W^t]_{i,j}$ and

$$Q_G^i(t) \preceq \left(\hat{I}_G^i(t)\right)^{-1} = \hat{\Sigma}_G^i(t),$$

where $\preceq$ means that $\hat{\Sigma}^i_G(t) - Q^i_G(t)$ is a positive semidefinite matrix [31, Lemma 2.1]. Therefore, the numerical covariance $Q^i_G(t)$ is bounded by the locally estimated global map covariance $\hat{\Sigma}^i_G(t)$. In particular, since $\hat{\Sigma}^i_G(t) - Q^i_G(t)$ is a positive semidefinite matrix, the elements in the main diagonal satisfy $[\hat{\Sigma}^i_G(t) - Q^i_G(t)]_{r,r} \geq 0$, and thus $[\hat{\Sigma}^i_G(t)]_{r,r} \geq [Q^i_G(t)]_{r,r}$. Therefore, any decision taken by the robots based on the entries in the main diagonal of the covariance matrix, can also be taken based on $\hat{\Sigma}^i_G(t)$.

Those properties of the intermediate estimates allow the robots to make decisions based on their global map estimates at every time instant.

Here, we analyze the algorithm complexity regarding execution time, amount of communication, and memory space required. Throughout this section, let $\mathcal{M}_{\max}$ be the highest size of the local map of any robot,

$$\mathcal{M}_{\max} = \max_{i \in \{1,\dots,n\}} \mathcal{M}_i,$$

and $d_{\max}$ be the highest number of neighbors of any robot,

$$d_{\max} = \max_{i \in \{1,\dots,n\}} |\mathcal{N}_i|.$$

For time-varying topologies $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, $d_{\max}$ is defined as

$$d_{\max} = \max_{i \in \{1,\dots,n\}, t \geq 0} |\mathcal{N}_i(t)|.$$

Computational complexity per iteration and robot

The consensus initialization operations in the map merging algorithm have a computational complexity per iteration and robot of $O(\mathcal{M}^3_{\max})$ that exclusively depends on the local map size. During the general iterations, the cost is $O(d_{\max}\mathcal{M}^2_G)$ to perform the matrix additions.

Communication complexity per iteration and robot

Along the consensus iterations, each robot i exchanges its information matrix $I^i_G(t)$ with its neighbors, with a communication cost of $O(d_{\max}\mathcal{M}^2_G)$. Notice that each $I^i_G(t)$s is a sparse information matrix, where the significant coefficients are grouped around the main diagonal. Therefore, if a compression algorithm is used, the cost of exchanging $I^i_G(t)$ can be expressed as $O(n\mathcal{M}^2_{\max})$. Alternatively, it can be expressed as $O(n+m)$ if we consider the local map sizes as constants, giving rise to a total communication cost per robot of, respectively, $O(d_{\max}n\mathcal{M}^2_{\max})$ or $O(d_{\max}(n+m))$. In addition, the robots exchange the label vectors of the features, with a total cost of $O(d_{\max}m)$.

Space complexity per iteration and robot

Along the consensus algorithm, the maximal space complexity for each robot is associated to the matrix $I^i_G(t)$ that, as discussed above, can be considered $O(n\mathcal{M}^2_{\max})$ or $O(n+m)$. Additionally this algorithm requires extra storage for the label vectors, although it does not have influence in the worst-case space complexity measure.

Time complexity until completion

As we mentioned before, the consensus is asymptotically reached, which means that the time until completion is infinite. However, the convergence speed of the averaging algorithm presents a geometric rate for fixed graphs [151], [30] which depends on the second eigenvalue with the largest absolute value $|\lambda_2(W)|$ in the Metropolis weights matrix (eq. (**??**) in Appendix A). If we denote $\gamma = |\lambda_2(W)|$, it can be shown that each entry $[I^i_G(t)]_{r,s}$, $[\mathbf{i}^i_G(t)]_r$ in the information matrices and vectors estimated by the robots evolve according to

$$|[I^i_G(t)]_{r,s} - [I_G]_{r,s}| \leq (\gamma)^t \sqrt{n} \max_j \left\{ \left| [I^j_G(0)]_{r,s} - [I_G]_{r,s} \right| \right\}, \tag{3.19}$$

$$|[\mathbf{i}^i_G(t)]_r - [\mathbf{i}_G]_r| \leq (\gamma)^t \sqrt{n} \max_j \left\{ \left| [\mathbf{i}^j_G(0)]_r - [\mathbf{i}_G]_r \right| \right\}, \tag{3.20}$$

for all $i \in \{1, \ldots, n\}$, all $r, s \in \{1, \ldots, \mathcal{M}_i(t)\}$, and all $t \geq 0$.

For graphs with switching topology $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, the convergence speed is geometric if the graph has an interval of joint connectivity τ such that every subsequence

$$\{G(t_0 + 1), \ldots \; G(t_0 + \tau)\}$$

of length τ is jointly connected for all t_0 [30]. In these graphs, the τ-index of joint contractivity $\delta < 1$ is given by

$$\delta = \max_{\mathcal{W} \in \mathbb{W}_\tau} \{ |\lambda_2(\mathcal{W})| \mid \mathcal{W} \text{ primitive paracontractive} \} \tag{3.21}$$

where $\mathbb{W}_\tau$ is the set of all products of, at most, τ Metropolis matrices $W(t)$ that can be obtained in the communication graph. The convergence speed of each entry $[I^i_G(t)]_{r,s}$, $[\mathbf{i}^i_G(t)]_r$ in the information matrices and vectors estimated by the robots depends on the τ-index of joint contractivity,

$$|[I^i_G(t)]_{r,s} - [I_G]_{r,s}| \leq (\delta)^{\llcorner \frac{t}{\tau} \lrcorner} \sqrt{n} \max_j \left\{ \left| [I^j_G(0)]_{r,s} - [I_G]_{r,s} \right| \right\}, \tag{3.22}$$

$$|[\mathbf{i}^i_G(t)]_r - [\mathbf{i}_G]_r| \leq (\delta)^{\llcorner \frac{t}{\tau} \lrcorner} \sqrt{n} \max_j \left\{ \left| [\mathbf{i}^j_G(0)]_r - [\mathbf{i}_G]_r \right| \right\}, \tag{3.23}$$

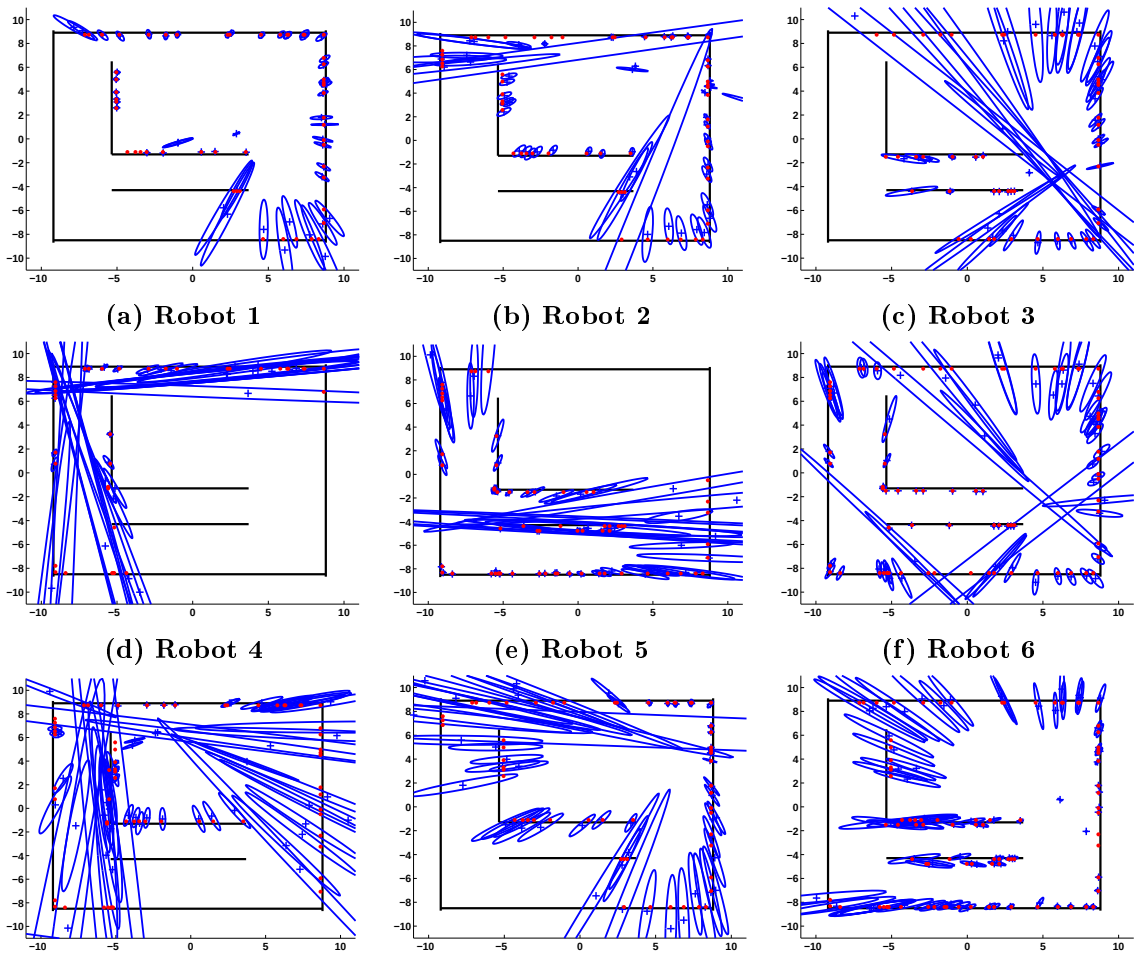
where $\llcorner \frac{t}{\tau} \lrcorner$ is the largest integer less than or equal to $\frac{t}{\tau}$.

Therefore, the convergence speed depends on the topology of the communication graph. Moreover, from the time complexity analysis, we can see that when the communication graph is complete, the robots reach consensus in one iteration. For complete communication graphs, the Metropolis matrix is $W = (1/n)\mathbf{1}\mathbf{1}^T$ and $|\lambda_2(W)| = 0$. Then, $|[I^i_G(t)]_{r,s} - [I_G]_{r,s}| \leq 0$, $|[\mathbf{i}^i_G(t)]_r - [\mathbf{i}_G]_r| \leq 0$ for all $t \geq 1$, all $i \in \{1, \ldots, n\}$, and all $r, s \in \{1, \ldots, \mathcal{M}_i(t)\}$.

3.8 Discussion

A first set of simulations has been carried out where a team composed by 9 robots has explored an environment, obtaining a set of local maps (Fig. 3.3(a-i)). Black dots represent obstacles, red dots are the ground-truth location of landmarks, blue crosses are the

estimates of the landmark positions and blue ellipses are the estimated covariance. Each robot explores only a small portion of the environment so that none robot observes all the landmarks. Due to the short trajectories followed by the robots and to the nature of bearing-only data, landmark estimates present large uncertainty in the local maps. In the simulation, robots estimate their motion based on odometry information and sense the environment using a camera device that provides bearings to the landmarks. The goal of



(a) Robot 1 **(b) Robot 2** **(c) Robot 3**

(d) Robot 4 **(e) Robot 5** **(f) Robot 6**

(g) Robot 7 **(h) Robot 8** **(i) Robot 9**

Figure 3.3: Local maps obtained by robots 1 to 9 (a-i). Red dots are the ground-truth location of landmarks. Blue crosses are the estimates of the landmark positions and blue ellipses are the estimated covariance.

the map merging process is the combination of the local maps to obtain the maximum-likelihood estimate for the global map (see Fig. 3.5). The robots exchange data according to the communication topology in Fig. 3.4. Each node executes the algorithm described in this chapter so that their estimates asymptotically approach this maximum-likelihood global map. Even thought the consensus is asymptotically reached, we can see that in practice, the convergence of the averaging algorithm is very fast, and in a few steps the estimates at every robot approach the global map. In Fig. 3.5 we show the global map

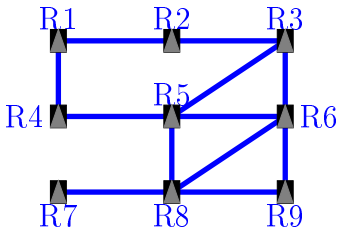
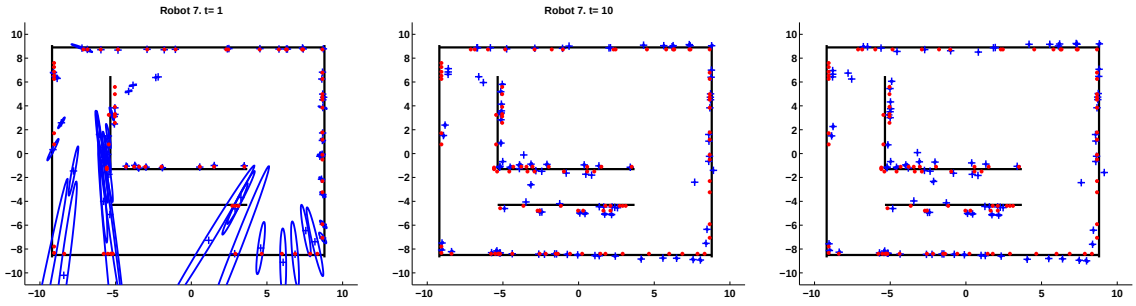


Figure 3.4: Communication graph between the 9 robots after the exploration.

estimated by robot 7 at iterations $t = 1$ and $t = 10$ compared to the centralized global map $t \to \infty$.



(a) $t = 1$ **(b)** $t = 10$ **(c)** $t \to \infty$

Figure 3.5: **(a)** Global map estimated by robot 7 at $t = 1$. **(b)** Global map estimated by robot 7 at $t = 10$. **(c)** Maximum-likelihood goal global map that is obtained by the robots as $t \to \infty$.

A second set of simulations has been carried out with a team composed by 7 robots exploring an environment of 20×20 m with 300 features, see Fig. 3.6. Each robot executes 70 motion steps along a path of approximately 30 m. The robots estimate their motion based on odometry information that is corrupted with a noise of standard deviation $\sigma_x, \sigma_y = 0.4\ cm$ for the translations and $\sigma_\theta = 1$ degree for the orientations. They sense the environment using an omnidirectional camera that gives bearing measurement to features within 360 degrees around the robot and within a distance of 6 m. The measurements are corrupted with a noise of 0.5 degrees standard deviation. Each robot explores the environment and builds its local map (Fig. 3.8). Due to the presence of obstacles (gray areas), each robot may have not observed some landmarks. Besides, the precision of the estimated positions (blue crosses and ellipses) of the landmarks depends on the trajectory followed by each robot. The 7 robots compute the global map as described in Section 3.6, under the communication graph in Fig. 3.7. Here, we just display the global map $\mathbf{x}_G^i(t), \Sigma_G^i(t)$ at robot 2 after 5 iterations. We provide a deeper analysis of the performance of the map fusion algorithm under real data and different communication networks below. Since the communication graph has a high connectivity, in a few iterations each robot has received information from any other robot. After 5 iterations, the global map at robot 2 already contains precise estimates of the whole explored environment (Fig. 3.9).

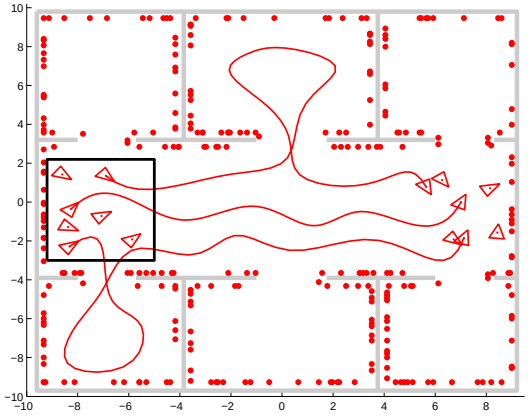


Figure 3.6: A team of 7 robots explore an environment of 20×20 m. Gray areas are walls and red dots are the ground-truth location of landmarks. Initially, the robots are placed in the black box region. They explore the environment and build their maps. We display the trajectories followed by robots 2, 3, and 5, together with the final poses of the 7 robots.

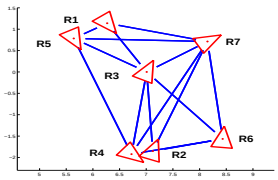


Figure 3.7: Communication graph $\mathcal{G}$ associated to the final robot poses in Fig. 3.6. There is a link (blue solid line) between any pair of robot poses (red triangles) that are within a distance of 3 m.

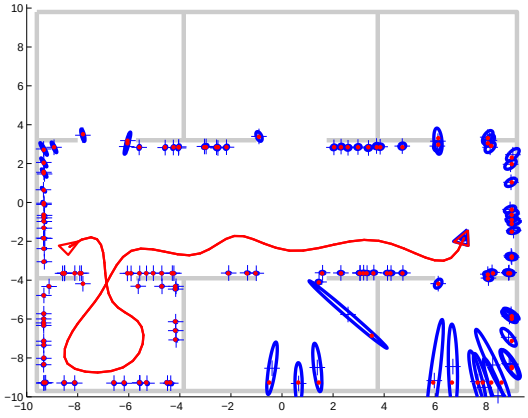


Figure 3.8: Local map estimated by robot 2. The landmarks close to its trajectory (red line) have been estimated (blue crosses and ellipses) with a high precision. Besides, its estimated positions (blue crosses) are very close to the ground truth locations (red dots). Due to the presence of obstacles (gray areas) some of the landmarks have not been observed, or have been estimated with high uncertainty.

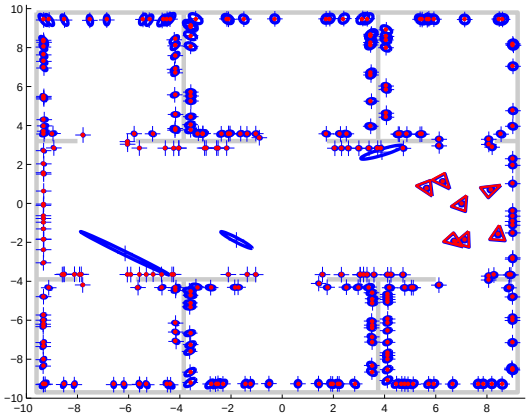


Figure 3.9: Global map $\mathbf{x}_G^i(t), \Sigma_G^i(t)$ estimated by robot 2 after $t = 5$ iterations. Red dots are the ground truth position of the features while blue crosses and ellipses are their estimated positions. Red triangles are the ground truth poses of the 7 robots after the exploration, and blue triangles are their estimated poses in the global map of robot 2.

3.9 Conclusions

We have presented a new method for merging stochastic feature-based maps acquired by a team of robots for scenarios with limited communication. The robots explore an environment and build their local maps. When they finish the exploration, they fuse their local maps and build a global map. The whole method is fully decentralized, relying exclusively on local interactions between neighboring robots. Under fixed connected communication graphs, or time-varying jointly connected topologies, the estimates at each robot asymptotically converge to the global map. Moreover, the intermediate estimates at each robot present interesting properties that allow their use at any time: the mean of the global map estimated by each robot is unbiased at each iteration; the numerical covariance of the global map estimated by each robot, which cannot be locally computed, is bounded by the locally computed covariance. The algorithm is robust to changes in the communication topology and to link failures. We have studied the performance of the method for robots equipped with omnidirectional cameras in a simulated environment. Additional experiments with real data acquired with conventional cameras, link failures, and changes in the topology, can be found in Chapter 8. To the best of our knowledge this is the first method that solves the map merging problem in a fully decentralized way.

Chapter 4
Dynamic Map Merging

In the previous chapter we presented a distributed algorithm for merging feature-based maps in a robot network after the exploration has taken place. In this chapter we discuss the dynamic map merging case. Along its operation, each robot observes the environment and builds and maintains its local map. Simultaneously, the robots communicate and build a global map of the environment. The communication between the robots is limited, and, at every time instant, each robot can only exchange data with its neighboring robots. In our contribution to the problem of dynamic map merging we provide a fully distributed solution which does not rely on any particular communication topology and is robust to changes in the topology. Each robot computes and tracks the global map based on local interactions with its neighbors. Under mild connectivity conditions on the communication graph, the algorithm asymptotically converges to the global map. In addition, we give means of its convergence speed according to the information increase in the local maps. We validate our proposal with several experiments.

4.1 Introduction

As previously said, there is an great interest in multi-robot perception in scenarios where a robot team operates in an unknown environment and individual robots only observe a portion of it. In such situations, it is of interest for each robot to have a representation of the environment beyond its local map. The fusion of the local observations of all the team members leads to a merged map that contains more precise information and more features. In a static map merging scenario, the information fusion is carried out after the exploration. Dynamic solutions, where the information is fused while the robots operate, are more interesting. They enable other multi-robot tasks such as cooperative exploration, navigation, or obstacle avoidance. In this chapter, we study the problem of dynamic map merging, where each robot's communication radius is limited, and hence the communication topology is not complete.

While multi-robot localization under communication constraints has received some attention [86, 118], most of the existing multi-robot map merging solutions are extensions of the single robot case under centralized schemes, all-to-all communication among the robots, or broadcasting methods. Particle filters [69] have been generalized to multi-robot systems assuming that the robots broadcast their controls and their observations. In multi-robot submap filters [149] and graph maps of laser scans [146] approaches, each

robot builds a local submap and sends it by broadcast to all the other robots or to a central node. The same solution could be applied for many existing submapping methods [108]. However, in robot network scenarios, distributed approaches are often necessary because of limited communication, switching topologies, link failures, and limited bandwidth.

Distributed estimation methods [1, 40, 65, 71, 96, 100, 101, 143] maintain a joint estimate of a system that evolves with time by combining noisy observations taken by the sensor network. Early approaches sum the measurements from the different robots in IF (Information Filter) form. If the network is complete [96], then the resulting estimator is equivalent to the centralized one. In general networks the problems of cyclic updates or double counting information appear when robots sum the same piece of data more than once. The use of the channel filter [65, 143] avoids these problems in networks with a tree structure. The Covariance Intersection method [71] produces consistent but highly conservative estimates in general networks. More recent approaches [1, 40, 100, 101] use distributed consensus filters to average the measurements taken by the robots. The interest of distributed averaging is that the problems of double counting the information and cyclic updates are avoided. They, however, suffer from the delayed data problem that takes place when the robots execute the state prediction without having incorporated all the measurements taken at the current step [34]. For general communication schemes [100], the delayed data problem leads to an approximate KF (Kalman Filter) estimator. An interesting solution is given in [101] but its convergence is proved in the absence of observation and system noises. In the algorithm proposed in [40], authors prove that the robots' estimates are consistent, although these estimates have disagreement. Other algorithms have been proposed that require the previous offline computation of the gains and weights of the algorithm [1]. The main limitation of all the previous works is that they consider linear systems without inputs, and where the evolution of the system is known by all the robots. Here instead we are interested in more general scenarios, without the previous restrictions. We allow each robot to build its map by using system models not necessarily linear or known by the other robots, or where the robot odometry is modeled as an input, among others. A recent work that does not suffer from the previous limitations is given in [87]. Here each robot records its own measurements and odometry, as well as the observations and odometry from any other robot it encounters. Despite being very interesting and going beyond the state-of-art, this work has the drawback that robots must maintain an unbounded amount of memory, which depends on the time between meetings. Moreover, if a single robot fails or leaves the network, the whole system fails, and the data association is not discussed. In our case, the information fusion is carried out on the local maps of the robots for which efficient distributed data association methods [10] already exist in the literature. Our approach is similar to a sensor fusion problem, although the classical assumption [31, 153] that successive measurements taken by a sensor must be independent does not hold here.

We propose a dynamic map merging method where robots average their maps instead of their raw measurements. The local maps of the robots are expressed in IF form and they are fused in an additive fashion using the consensus filter. We build on ideas from consensus algorithms that allow the introduction of new information, regardless of the independence between successive measurements [59, 88]. We use a discrete-time version

of the PI algorithm which is more appropriate for the robot systems we consider. As weight matrices we use the Metropolis weights [152] which have been shown to perform quite good in multi-agent systems [31, 40, 153] and that have the benefit that they can be locally computed by the robots.

The contributions of this chapter are the following: (i) the proposal of the dynamic consensus strategy where, at each step, a discrete-time version of the PI algorithm is executed; (ii) the careful study of the convergence rate of the dynamic consensus strategy; (iii) the applications of this study to characterize the errors in the map merging and to understand the trade-offs between the number of iterations and the performance of the algorithm; and (iv) the implementation for feature-based maps taking into account the possibly different features discovered by each robot during the exploration.

4.2 Problem Description

Throughout the chapter we let n be the number of robots. Indices i, j refer to robots, r, s to elements within the maps, and G to the global map. The superscript $k \in \mathbb{N}$ is used for exploration steps and $t \in \mathbb{N}$ for iteration numbers. We let $\mathbf{I}$ be the $n \times n$ identity matrix, and $\mathbf{0}$ be a $n \times n$ matrix with all its elements equal to zero. When they are followed by a subindex $n_1 \times n_2$, this specifies their dimensions. We let $\mathbf{1} \in \mathbb{R}^n$ be a column vector with all entries equal to 1. Given a matrix W, $[W]_{i,j}$ denotes its (i, j) entry, $\lambda_i(W)$ refers to its $i-$th eigenvalue with associated eigenvector $\mathbf{v}_i(W)$, and $\lambda_{\text{eff}}(W)$ is the modulus of its eigenvalue with the second largest absolute value.

We consider a team of $n \in \mathbb{N}$ robots exploring an unknown environment. At the exploration step k, each robot i has observed $m_i^k \in \mathbb{N}$ features and it has estimated its own pose together with the positions of the features. Let the constants `szr` and `szf` represent the size of respectively a robot pose and a feature position[1]. The estimates at each robot $i \in \{1, \dots, n\}$ and each step k are stored into a stochastic map with mean $\hat{\mathbf{x}}_i^k \in \mathbb{R}^{\mathcal{M}_i^k}$ and covariance matrix $\Sigma_i^k \in \mathbb{R}^{\mathcal{M}_i^k \times \mathcal{M}_i^k}$, being $\mathcal{M}_i^k = \texttt{szr} + m_i^k \texttt{ szf}$. Let $\mathbf{x}_i^k \in \mathbb{R}^{\mathcal{M}_i^k}$ contain the true robot pose and the true positions of the m_i^k features, then

$$\hat{\mathbf{x}}_i^k = \mathbf{x}_i^k + \mathbf{v}_i^k, \tag{4.1}$$

where $\mathbf{v}_i^k$ is a zero mean noise with covariance matrix Σ_i^k.

If at step k the information from the n robots was available, e.g., at a central node, then the global map combining the information of the local maps at the n robots at step k could be computed. Let $m \in \mathbb{N}$ be the number of different features in the environment and $\mathbf{x} \in \mathbb{R}^{\mathcal{M}_G}$ be the vector with the true poses of the n robots and the true positions of the m features, being $\mathcal{M}_G = n \texttt{ szr} + m \texttt{ szf}$. Each robot $i \in \{1, \dots, n\}$ at step k has observed $m_i^k \leq m$ features and we let $H_i^k \in \{0, 1\}^{\mathcal{M}_i^k \times \mathcal{M}_G}$ be the observation matrix that relates the elements in $\mathbf{x}$ and $\mathbf{x}_i^k$ so that $\mathbf{x}_i^k = H_i^k \mathbf{x}$. The local map of each robot i (4.1) is a partial observation of $\mathbf{x}$,

$$\hat{\mathbf{x}}_i^k = H_i^k \mathbf{x} + \mathbf{v}_i^k, \tag{4.2}$$

[1] e.g., $\texttt{szr} = 3$ when the robot pose is composed of its planar position (x, y) and orientation θ; $\texttt{szf} = 2$ or $\texttt{szf} = 3$ for respectively 2D or 3D environments.

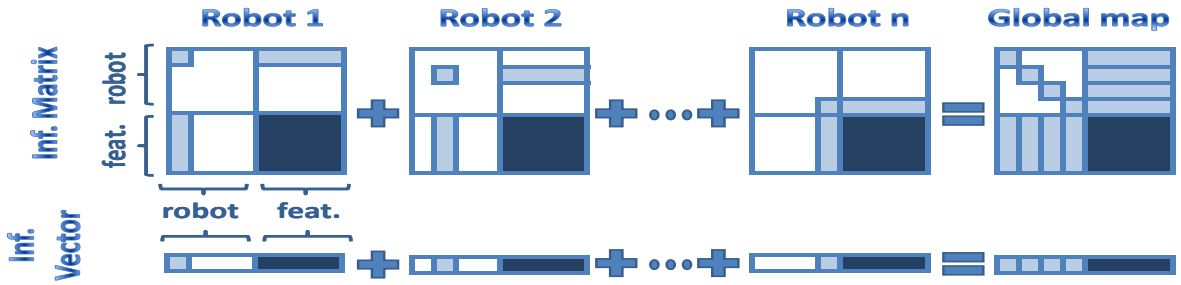


Figure 4.1: Centralized merging of the local maps of n robots in IF form. The global information matrix I_G^k and vector $\mathbf{i}_G^k$ are the addition of the local ones I_i^k, $\mathbf{i}_i^k$, for $i \in \{1, \ldots, n\}$. The first rows and columns contain the robot poses, and the last ones, the feature positions. The elements with zero value are displayed in white. Each robot i has information of its own pose (light blue) and of the feature positions (dark blue), but it has no information of any other robot poses $j \neq i$ (white).

where we assume that the noises $\mathbf{v}_i^k, \mathbf{v}_j^{k'}$ are independent for different robots $i \neq j$ and all $k, k' \in \mathbb{N}$, since every robot has constructed the map based on its own observations. Note that since the local map of a robot i at step k is an evolution of its map at any previous step $k' < k$, then the noises $\mathbf{v}_i^k, \mathbf{v}_i^{k'}$ are not independent. Let $I_i^k \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ and $\mathbf{i}_i^k \in \mathbb{R}^{\mathcal{M}_G}$ be the information matrix and vector of the local map at robot i and step k in IF form,

$$I_i^k = (H_i^k)^T (\Sigma_i^k)^{-1} H_i^k, \qquad \mathbf{i}_i^k = (H_i^k)^T (\Sigma_i^k)^{-1} \hat{\mathbf{x}}_i^k, \tag{4.3}$$

for $i \in \{1, \ldots, n\}$. The information matrix I_G^k and vector $\mathbf{i}_G^k$ of the global map at step k in IF form are

$$I_G^k = \sum_{i=1}^{n} I_i^k, \qquad \mathbf{i}_G^k = \sum_{i=1}^{n} \mathbf{i}_i^k. \tag{4.4}$$

The previous operation is additive, commutative, and associative. For this reason, merging the maps in IF form is a common practice [138]. Equivalently, the global map at step k can be expressed by its mean and covariance matrix,

$$\hat{\mathbf{x}}_G^k = (I_G^k)^{-1}\, \mathbf{i}_G^k, \qquad \Sigma_G^k = (I_G^k)^{-1}. \tag{4.5}$$

Maps in information form have the property that entries (r, s) and r in the information matrix I_i^k and vector $\mathbf{i}_i^k$ associated to the elements not observed by robot i are zero (Fig. 4.1, white elements). Consider a feature observed by several robots $\mathcal{R} \subseteq \{1, \ldots, n\}$ (Fig. 4.1, dark blue area). The associated entries (r, s) and r in the global map $[I_G^k]_{r,s}$, $[\mathbf{i}_G^k]_r$ (4.4) are the addition of the different values $[I_i^k]_{r,s}$, $[\mathbf{i}_i^k]_r$, for $i \in \mathcal{R}$,

$$[I_G^k]_{r,s} = \sum_{i \in \mathcal{R}} [I_i^k]_{r,s}, \qquad [\mathbf{i}_G^k]_r = \sum_{i \in \mathcal{R}} [\mathbf{i}_i^k]_r.$$

Here, each robot i reaches a consensus between its own and the others' values $[I_j^k]_{r,s}$, $[\mathbf{i}_j^k]_r$, for $j \in \mathcal{R}$. Consider now the estimated pose of a robot i. It was exclusively observed by i, and thus for any other robot $j \neq i$ the associated entries (r,s) and r are zero, $[I_j^k]_{r,s} = 0$, $[\mathbf{i}_j^k]_r = 0$. Only robot i is providing information of these entries for the global map,

$$[I_G^k]_{r,s} = [I_i^k]_{r,s}, \qquad\qquad [\mathbf{i}_G^k]_r = [\mathbf{i}_i^k]_r,$$

and thus it is clear that here there is no need for consensus. The dynamic map merging problem can be separated into two parts. The first part consists of propagating the rows and columns of I_i^k, $\mathbf{i}_i^k$ associated the pose of a robot j. Any other robot $i \neq j$ just incorporates this data into its global map. The second part, which consists of reaching a consensus on the entries associated exclusively to features, is discussed along the following sections.

Problem 4.2.1. *We consider $n \in \mathbb{N}$ robots exploring and acquiring local maps at some exploration steps $k = 1, 2, \ldots$ as in eqs. (4.1)-(4.3). The communication is range-limited and two robots can exchange data only if they are close enough. We let $\mathcal{G}^k = (\mathcal{V}, \mathcal{E}^k)$ be the undirected communication graph at step k. The nodes are the robots, $\mathcal{V} = \{1, \ldots, n\}$. If robots i, j can communicate then there is an edge between them, $(i,j) \in \mathcal{E}^k$. The set of neighbors $\mathcal{N}_i^k$ of robot i at step k is*

$$\mathcal{N}_i^k = \{j \mid (i,j) \in \mathcal{E}^k, j \neq i\}.$$

The goal is the design of distributed algorithms so that each robot $i \in \mathcal{V}$ computes and tracks the global map in eqs. (4.4)-(4.5) based on local interactions with its neighbors $\mathcal{N}_i^k$.

4.2.1 Proportional Integral (PI) Averaging Algorithm

We propose a map merging algorithm based on averaging algorithms. They have become very popular in sensor networks due to their capability to reach agreement in a distributed way. In praticular, we use a discrete version of the Proportional Integral (PI) estimator in [59], where each robot $i \in \{1, \ldots, n\}$ has an input $u_i \in \mathbb{R}$. It maintains variables $x_i(t), w_i(t) \in \mathbb{R}$ and updates them by the following rule,

$$\begin{aligned} \dot{x}_i(t) &= -\gamma\, x_i(t) - \sum_{j \neq i} [W_P]_{i,j}\ [x_i(t) - x_j(t)] + \sum_{j \neq i} [W_I]_{j,i}\ [w_i(t) - w_j(t)] + \gamma\, u_i, \\ \dot{w}_i(t) &= -\sum_{j \neq i} [W_I]_{i,j}\ [x_i(t) - x_j(t)]\,, \end{aligned} \tag{4.6}$$

where W_P and W_I are respectively the proportional and the integral weights matrices. Those weights are compatible with the graph, so that $[W_P]_{i,j} = [W_P]_{j,i} = [W_I]_{i,j} = [W_I]_{j,i} = 0$ if robots i and j cannot communicate, $j \notin \mathcal{N}_i$. The parameter $\gamma > 0$ establishes the rate at which new information replaces old information. It allows the network to slowly forget errors introduced by robots entering or leaving the network.

If the inputs and variables at the n robots are considered simultaneously, $\mathbf{u} \in \mathbb{R}^n = (u_1, \dots, u_n)^T$, $\mathbf{x} \in \mathbb{R}^n = (x_1, \dots, x_n)^T$, $\mathbf{w} \in \mathbb{R}^n = (w_1, \dots, w_n)^T$, then the PI estimator can be expressed in matrix form as follows,

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{w}}(t) \end{bmatrix} = \begin{bmatrix} -\gamma I - L_P & L_I^T \\ -L_I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{w}(t) \end{bmatrix} + \begin{bmatrix} \gamma I \\ 0 \end{bmatrix} \mathbf{u}, \tag{4.7}$$

where L_P and L_I are the Laplacian associated to respectively the proportional W_P and the integral W_I weight matrices,

$$L_P = \operatorname{diag}(W_P \mathbf{1}) - W_P, \qquad L_I = \operatorname{diag}(W_I \mathbf{1}) - W_I.$$

Let $\mathbf{e}_\mathbf{x}(t) \in \mathbb{R}^n$ be the error vector,

$$e_\mathbf{x}(t) = \mathbf{x}(t) - \frac{\mathbf{1}\mathbf{1}^T}{n}\mathbf{u},$$

and Π be the matrix $\Pi = I - \frac{\mathbf{1}\mathbf{1}^T}{n}$. If each robot $i \in \{1, \dots, n\}$ executes the PI algorithm (4.6) with γ, L_P and L_I so that

$$\operatorname{rank}(L_I) = n - 1, \tag{4.8a}$$

$$\varepsilon \in \mathbb{R} \text{ is such that } \Pi(L_P + L_P^T)\Pi \succeq 2\varepsilon\Pi, \tag{4.8b}$$

$$\gamma > 0 \text{ is chosen such that } \gamma + \varepsilon > 0, \tag{4.8c}$$

then, for any input $\mathbf{u}$ and any initial states $\mathbf{x}(0), \mathbf{w}(0)$, the error vector $e_\mathbf{x}(t)$ converges to 0 exponentially as $t \to \infty$ [59, Theorem 5].

The variables $x_i(t)$ asymptotically reach the same value for all $i \in \mathcal{V}$, i.e., they reach a consensus, and moreover, the consensus value is the average of the inputs. Observe that each robot i updates its variables $x_i(t), w_i(t)$ using local information since the weights matrices have zero entries for non-neighboring robots, $[W_P]i, j = [W_I]i, j = 0$ when $j \notin \mathcal{N}_i$.

A common choice for the weights matrices in distributed averaging are the Metropolis weights $W_M \in \mathbb{R}^{n \times n}$ [152] given by eq. (A.3) in Appendix A, where each robot can compute the weights that affect its evolution using only local information. The Metropolis weights matrix W_M is compatible with the graph, $[W_M]_{i,j} = [W_M]_{j,i} = 0$ if robots i and j cannot communicate. Besides, for undirected graphs, W_M is symmetric and doubly stochastic $W_M = W_M^T, W_M \mathbf{1} = \mathbf{1}, \mathbf{1}^T W_M = \mathbf{1}^T$. It has an eigenvalue at 1, and all its other eigenvalues $\lambda(W_M) \in (-1, 1)$. Its associated Laplacian,

$$L_M = \operatorname{diag}(W_M \mathbf{1}) - W_M = I - W_M, \tag{4.9}$$

is symmetric and positive semidefinite [27, Theorem 1.37]. It has an eigenvalue at 0, and all the others $\lambda(L_M) \in (0, 2)$. L_M satisfies (4.8a) for connected graphs. It also satisfies (4.8b) for $\varepsilon = 0$ taking into account that $\Pi(L_M + L_M^T)\Pi = 2L_M$ since L_M is symmetric, and that L_M is positive semidefinite. Then, condition (4.8c) reduces to $\gamma > 0$. Thus, for connected graphs, the PI algorithm (4.7) with symmetric Metropolis weights matrices $W_P = W_I = W_M$ converges to the average of the inputs.

Along this chapter, we consider a discrete version of the PI algorithm, with equal and symmetric Laplacian matrices $L_W \in \mathbb{R}^n$ so that $L_P = L_I = L_W$, $L_W^T = L_W$. We let W

be its associated weights matrix, $L_W = \text{diag}(W\mathbf{1}) - W$. Then, we extend these results to the particular choice of the Metropolis weights matrix $W = W_M$. We use the notation $\lambda_i(A)$ for the i eigenvalue of a matrix A, and $\mathbf{v}_i(A)$ for its associated eigenvector. When the matrix is not specified, λ_i refers to the L_W. We let $\mathbf{r} = \frac{1}{\sqrt{n}}\mathbf{1}$ be the eigenvector of L_W associated to the eigenvalue 0, and $[\mathbf{r}\ S_2\ \dots\ S_n] = [\mathbf{r}\ S]$ be a basis of eigenvectors of L_W.

4.3 Consensus on Constant Scalar Inputs

We start by considering a simplified version of Problem 4.2.1, where there is a single exploration step k. Instead of an information matrix and a vector, each robot $i \in \mathcal{V}$ has a single scalar input $u_i \in \mathbb{R}$. The global data $x_G \in \mathbb{R}$ is the sum of the inputs u_i and we let $x_{avg} \in \mathbb{R}$ be their average,

$$x_G = \sum_{i=1}^{n} u_i, \qquad x_{avg} = \frac{1}{n}\sum_{i=1}^{n} u_i = \frac{1}{n}x_G. \tag{4.10}$$

The goal is that each robot $i \in \mathcal{V}$ computes an estimate $x_i(t) \in \mathbb{R}$ of x_{avg} by local interactions with its neighbors $\mathcal{N}_i$.

The previous simplified problem can be solved by distributed consensus algorithms [113] for systems with constant inputs. In particular, we analyze in depth a discrete version of the Proportional Integral (PI) estimator [59] in the context of dynamic consensus. As we will show, the capabilities of the PI for re-using past information are crucial for the considered map merging scenario.

Discrete-time algorithms are more appropriate for the robot systems we consider. In this section we analyze a discrete-time version of the PI algorithm (4.7) with equal, symmetric, positive semidefinite Laplacian matrices $L_W \in \mathbb{R}^n$ so that $L_P = L_I = L_W$, $L_W^T = L_W$ and we let W be its associated weight matrix, $L_W = \text{diag}(W\mathbf{1}) - W$. We analyze its convergence properties and its convergence speed depending on the step size h and the parameter γ. The theoretical results we give are general for any weighting matrix. We later extend them to the particular choice of the Metropolis weight matrix $W = W_M$ (eq. (A.3) in Appendix A) and its Laplacian matrix $L_W = L_M$ given by eq. (4.9). From now on, we let $\mathbf{r} \in \mathbb{R}^n$ be the eigenvector of L_W associated to the eigenvalue $\lambda_1(L_W) = 0$,

$$\mathbf{r} = \mathbf{1}/\sqrt{n}. \tag{4.11}$$

We let $S_2, \dots, S_n$ be the remaining $n-1$ eigenvectors of L_W so that $[\mathbf{r}\ S_2 \dots S_n] = [\mathbf{r}\ S]$ is a basis of eigenvectors of L_W,

$$[\mathbf{r}\ S]^T L_W [\mathbf{r}\ S] = \text{diag}\left(\lambda_1(L_W), \dots, \lambda_n(L_W)\right), \tag{4.12}$$

with the eigenvalues sorted as $\lambda_1(L_W) \leq, \dots, \leq \lambda_n(L_W)$. This orthonormal basis exists since L_W is symmetric with real entries. For connected communication graphs, all the other eigenvalues $\lambda_2(L_W), \dots, \lambda_n(L_W)$ are strictly greater than zero and we let $L_W^{(-1)}$ be

$$L_W^{(-1)} = (\mathbf{I} - \mathbf{r}\mathbf{r}^T)\left(L_W + \mathbf{r}\mathbf{r}^T\right)^{-1}(\mathbf{I} - \mathbf{r}\mathbf{r}^T). \tag{4.13}$$

For all $i \in \mathcal{V}$ we let $b_i = b(\lambda_i(L_W))$ be

$$b_i = b(\lambda_i(L_W)) = \sqrt{(\gamma + \lambda_i(L_W))^2 - (2\ \lambda_i(L_W))^2}. \tag{4.14}$$

The discrete-time consensus algorithm with constant scalar inputs, with equal and symmetric Laplacian matrices L_W, and step size $h > 0$ is given by

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{w}(t+1) \end{bmatrix} = A \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{w}(t) \end{bmatrix} + \begin{bmatrix} h\gamma \mathbf{I} \\ \mathbf{0} \end{bmatrix} \mathbf{u}, \quad \text{with} \tag{4.15}$$

$$A = \mathbf{I}_{2n\times 2n} + h \begin{bmatrix} -\gamma \mathbf{I} - \mu L_W & L_W \\ -L_W & \mathbf{0} \end{bmatrix}. \tag{4.16}$$

The parameter $\mu > 0$ weights the relative effects of the Proportional and Integral components. In the following we focus on the study of the system for the case $\mu = 1$ and we give conditions on parameters h, γ that ensure the convergence in real scenarios. The optimal combination of the proportional and integral weighting matrices depends on the graph, and on h and γ. The analysis of the properties for each case can be done as a replication of the theoretical analysis presented here. Note that this algorithm is fully distributed as each robot updates its states using information from its immediate neighbors. Along this section we will show that under certain conditions, the states at the robots asymptotically converge to the average of the inputs, $x_i(t) \to x_{avg}$ as $t \to \infty$; equivalently in vector form, that $\mathbf{x}(t) \to \mathbf{1}x_{avg}$. We let the vectors $\mathbf{x}_*$ and $\mathbf{w}_* \in \mathbb{R}^n$ be

$$\mathbf{x}_* = \mathbf{r}\mathbf{r}^T\mathbf{u} = \mathbf{1}x_{avg}, \qquad \mathbf{w}_* = \mathbf{r}\mathbf{r}^T\mathbf{w}(0) - \gamma L_W^{(-1)}\mathbf{u}, \tag{4.17}$$

where $\mathbf{r}$ and $L_W^{(-1)}$ are given by eqs (4.11) and (4.13).

In order to analyze the convergence conditions and convergence speed of algorithm (4.15), we first analyze the eigenvalues of the system matrix A in eq. (4.16). The following result establishes a relationship between them and the eigenvalues of the Laplacian L_W associated to the weight matrix.

Proposition 4.3.1. *For each eigenvalue $\lambda_i(L_W)$ of the Laplacian L_W associated to the weight matrix, there exist eigenvalues $\lambda_i(A)$ and $\lambda_{n+i}(A)$ of the system matrix A in eq.* (4.16)*,*

$$\lambda_i(A) = 1 - h\left(\gamma + \lambda_i(L_W) + b_i\right)/2, \quad \lambda_{n+i}(A) = 1 - h\left(\gamma + \lambda_i(L_W) - b_i\right)/2, \tag{4.18}$$

for $i \in \mathcal{V}$, being b_i given by (4.14)*. Note that for $\lambda_1(L_W) = 0$, eq.* (4.18) *gives $\lambda_1(A) = 1 - h\gamma$ and $\lambda_{n+1}(A) = 1$.*

Proof. Denote $Z = \begin{bmatrix} -\gamma \mathbf{I} - L_W & L_W \\ -L_W & \mathbf{0} \end{bmatrix}$, such that $A = \mathbf{I}_{2n\times 2n} + hZ$. The relationship between the eigenvalues and eigenvectors of A and Z for all $i \in \{1, \ldots, 2n\}$ is

$$\lambda_i(A) = 1 + h\lambda_i(Z), \qquad \mathbf{v}_i(A) = \mathbf{v}_i(Z). \tag{4.19}$$

We define the change of basis $Y = P^T Z P$, with

$$P = \begin{bmatrix} \mathbf{r}\ S_2 \dots S_n & \mathbf{0} \\ \mathbf{0} & \mathbf{r}\ S_2 \dots S_n \end{bmatrix}, \tag{4.20}$$

where $[\mathbf{r}\ S_2 \dots S_n]$ is an orthonormal basis of eigenvectors of L_W as in eq. (4.12), so that the eigenvalues and eigenvectors of Z and Y are related by

$$\lambda_i(Z) = \lambda_i(Y), \qquad \mathbf{v}_i(Z) = P\ \mathbf{v}_i(Y). \tag{4.21}$$

We focus on the matrix Y,

$$Y = \begin{bmatrix} -\gamma \mathbf{I} - [\mathbf{r}\ S]^T L_W [\mathbf{r}\ S] & [\mathbf{r}\ S]^T L_W [\mathbf{r}\ S] \\ -[\mathbf{r}\ S]^T L_W [\mathbf{r}\ S] & \mathbf{0} \end{bmatrix}, \tag{4.22}$$

which because of eq. (4.12) is equal to

$$Y = \begin{bmatrix} \mathrm{diag}(-\gamma - \lambda_1, \dots, -\gamma - \lambda_n) & \mathrm{diag}(\lambda_1, \dots, \lambda_n) \\ -\mathrm{diag}(\lambda_1, \dots, \lambda_n) & \mathbf{0} \end{bmatrix},$$

where each $\lambda_i = \lambda_i(L_W)$ and we have omitted (L_W) for clarity. By solving for $Y\mathbf{v}_i(Y) = \lambda_i(Y)\mathbf{v}_i(Y)$, we get the following expression for the eigenvalues of Y, for all $i \in \mathcal{V}$:

$$\lambda_i(Y) = -\frac{\gamma + \lambda_i + b_i}{2}, \qquad \lambda_{n+i}(Y) = -\frac{\gamma + \lambda_i - b_i}{2}. \tag{4.23}$$

Its eigenvectors $\mathbf{v}_i(Y)$, $\mathbf{v}_{n+i}(Y)$ have all its elements equal to zero but the $i-$th and the $(n+i)-$th components,

$$\begin{aligned} [\mathbf{v}_i(Y)]_i &= 1, & [\mathbf{v}_i(Y)]_{n+i} &= -\lambda_i(L_W)/\lambda_i(Y), \\ [\mathbf{v}_{n+i}(Y)]_i &= 1, & [\mathbf{v}_{n+i}(Y)]_{n+i} &= -\lambda_i(L_W)/\lambda_{n+i}(Y). \end{aligned} \tag{4.24}$$

Combining eqs. (4.19),(4.21),(4.23), the expression for the eigenvalues of A in (4.18) is obtained. □

Proposition 4.3.2. *If the step size h and the parameter γ satisfy*

$$\gamma \geq 3/2\ \lambda_n(L_W), \tag{4.25a}$$
$$h\gamma < 3/2, \tag{4.25b}$$

where $\lambda_n(L_W)$ is the maximum eigenvalue of L_W, then all the eigenvalues of A in (4.18) are real. For connected communication graphs, all of them but $\lambda_{n+1}(A) = 1$ have modulus strictly less than one.

Proof. The eigenvalues of A are related to the ones of L_W by (4.18) as stated by Proposition 4.3.1. All the eigenvalues $\lambda_i(L_W)$ are real and positive because L_W is positive semidefinite. Since both γ and h are real, the imaginary part of $\lambda_i(A)$ is $\pm \mathrm{Im}\,[b_i]$, which is 0 because of (4.25a). As a result, all the eigenvalues of A are real.

Regarding the modulus, first note that for connected graphs $\lambda_i > 0$ for all $i \in \{2,\dots,n\}$. Due to (4.25a) b_i given by eq. (4.14) satisfies $\gamma \leq b_i \leq 2/\sqrt{3}\gamma$, and both $\lambda_i(Y)$ and $\lambda_{n+i}(Y)$ in eq. (4.23) are decreasing functions satisfying $-4/3\ \gamma \leq \lambda_i(Y) < -\gamma$, $-1/3\ \gamma \leq \lambda_{n+i}(Y) < 0$, for all $i \in \{2,\dots,n\}$.

We first consider the eigenvalue $\lambda_1(A) = 1 - h\gamma$. It is strictly less than 1 since $h > 0$, $\gamma > 0$, and it is greater than $-1/2$ because of (4.25b). Then, its modulus is strictly less than 1. For all $i \in \{2,\dots n\}$, both $\lambda_i(A) = 1 + h\lambda_i(Y) < 1$ and $\lambda_{n+i}(A) = 1 + h\lambda_{n+i}(Y) < 1$, since $h > 0$ and $\lambda_i(Y) < 0, \lambda_{n+i}(Y) < 0$. Besides, $\lambda_i(A) \geq 1 - 4/3h\gamma > -1$, and $\lambda_{n+i}(A) \geq 1 - 1/3h\gamma > 1/2$. Then the modulus of both $\lambda_i(A)$ and $\lambda_{n+i}(A)$ are strictly less than 1. Finally, $\lambda_{n+1}(A) = 1$ as stated in Proposition 4.3.1. □

In particular, the selection of $\gamma \geq 3$ and $h < 3/(2\gamma)$ when the Metropolis Laplacian matrix L_M (eq. (4.9)) is used, satisfies Proposition 4.3.2 for any connected communication graph, since its eigenvalues satisfy $\lambda_1(L_M) = 0$ and $0 < \lambda_i(L_M) < 2$ for all $i \in \{2,\dots,n\}$.

We discuss now which one is the second eigenvalue $\lambda_{\text{eff}}(A)$ of A with maximum absolute value. Observe that $\lambda_{n+i}(A) \geq 1/2$ is decreasing, thus the greatest absolute value of $\lambda_{n+i}(A)$ for $i \in \{2,\dots,n\}$ is associated to $\lambda_{n+2}(A)$. Also $\lambda_i(A)$ is decreasing and takes both positive and negative values. For all i such that $\lambda_i(A) \geq 0$, the associated $\lambda_{n+i}(A)$ has greater modulus. For all i such that $\lambda_i(A) < 0$, the maximum absolute value is associated to $\lambda_n(A)$. We conclude that $\lambda_{\text{eff}}(A) = \max\{\lambda_{n+2}(A), -\lambda_n(A)\}$.

At this point we are ready to prove the convergence of algorithm (4.15) and to characterize its convergence speed.

Theorem 4.3.3. *Let L_W be the positive semidefinite Laplacian matrix associated to the connected undirected communication graph $\mathcal{G}$. Let us consider that the robots execute algorithm* (4.15) *with a step size $h > 0$ and a parameter $\gamma > 0$ as in Proposition 4.3.2. Then, for any input $\mathbf{u} \in \mathbb{R}^n$ and any initial states $\mathbf{x}(0) \in \mathbb{R}^n$, $\mathbf{w}(0) \in \mathbb{R}^n$, the states $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{w}(t) \in \mathbb{R}^n$ of the consensus algorithm* (4.15) *converge exponentially to*

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_*, \qquad \lim_{t\to\infty} \mathbf{w}(t) = \mathbf{w}_*, \tag{4.26}$$

as $t \to \infty$, where $\mathbf{x}_*$ and $\mathbf{w}_*$ are given by (4.17). *Moreover, if we let $\beta = 2\sqrt{10}/3$, then the error vector $e_{\mathbf{xw}}(t) = \left(\mathbf{x}(t)^T, \mathbf{w}(t)^T\right)^T - \left(\mathbf{x}_*^T, \mathbf{w}_*^T\right)^T$ after t iterations[2] satisfies*

$$\|e_{\mathbf{xw}}(t)\|_2 \leq \beta\lambda_{\text{eff}}^t(A)\, \|e_{\mathbf{xw}}(0)\|_2 . \tag{4.27}$$

Proof. First we prove the convergence. Let us assume that the relation in (4.27) is true. Since h and γ satisfy conditions (4.25a)-(4.25b) then, as stated by Proposition 4.3.2, $|\lambda_{n+1}(A)| = 1$ and the other eigenvalues have modulus strictly less than one $|\lambda_i(A)| < 1$. In particular, this is true for $\lambda_{\text{eff}}(A)$, and thus $\lambda_{\text{eff}}^t(A)$ tends to 0 as $t \to \infty$ and the norm of the error $\|e_{\mathbf{xw}}(t)\|_2$ converges to zero.

[2]All along this chapter, we characterize the convergence speed for an even t in order to give more accurate bounds.

Next, we prove that the error vector satisfies (4.27). The discrete time consensus algorithm (4.15) expressed in terms of the error $e_{\mathbf{xw}}(t)$ is

$$e_{\mathbf{xw}}(t+1) = A\ e_{\mathbf{xw}}(t), \qquad e_{\mathbf{xw}}(t) = A^t\ e_{\mathbf{xw}}(0).$$

We define the following change of basis,

$$C = P^T A P = \mathbf{I} + hY,$$

where P and Y are given by (4.20) and (4.22), and we let $e_{\mathbf{zy}}$ be the error in the new coordinates,

$$e_{\mathbf{zy}}(t) = P^T\ e_{\mathbf{xw}}(t), \qquad e_{\mathbf{xw}}(t) = P\ e_{\mathbf{zy}}(t), \tag{4.28}$$

which has the same Euclidean norm, $\|e_{\mathbf{xw}}(t)\|_2 = \|e_{\mathbf{zy}}(t)\|_2$. We focus on the system in the new coordinates,

$$e_{\mathbf{zy}}(t+1) = C\ e_{\mathbf{zy}}(t), \qquad e_{\mathbf{zy}}(t) = C^t\ e_{\mathbf{zy}}(0),$$

where the initial error is $e_{\mathbf{zy}}(0) = (\mathbf{z}(0)^T, \mathbf{y}(0)^T)^T - (\mathbf{z}_*^T, \mathbf{y}_*^T)^T$. By applying the change of basis to (4.26), the limit values in the new coordinates are

$$\mathbf{z}_* = \begin{bmatrix} \mathbf{r}^T\mathbf{u} \\ \mathbf{0}_{n-1} \end{bmatrix}, \qquad \mathbf{y}_* = \begin{bmatrix} \mathbf{r}^T\mathbf{w}(0) \\ -\gamma(S^T L_W S)^{-1} S^T \mathbf{u} \end{bmatrix}, \tag{4.29}$$

where $\mathbf{r}^T\mathbf{w}(0) = [\mathbf{y}(0)]_1$. Note that as a result the component $[e_{\mathbf{zy}}(0)]_{n+1}$ of the error in the new coordinates is zero. Let us decompose the initial error into a linear combination of the eigenvectors of C,

$$e_{\mathbf{zy}}(0) = \sum_{i=1}^{2n} a_i\ \mathbf{v}_i(C), \tag{4.30}$$

where it can be seen that for all $i \in \{1, \ldots, 2n\}$

$$\lambda_i(C) = \lambda_i(A), \text{ and} \qquad \mathbf{v}_i(C) = \mathbf{v}_i(Y), \tag{4.31}$$

being $\lambda_i(A)$ and $\mathbf{v}_i(C)$ given by respectively eqs. (4.18) and (4.24). Now we compute the coefficients a_i, a_{n+i} in eq. (4.30) as follows. Each pair $[e_{\mathbf{zy}}(0)]_i$, $[e_{\mathbf{zy}}(0)]_{n+i}$ of elements in $e_{\mathbf{zy}}(0)$ give two equations on a_i, a_{n+i}, for $i \in \mathcal{V}$,

$$\begin{aligned} [e_{\mathbf{zy}}(0)]_i &= a_i + a_{n+i}, \\ [e_{\mathbf{zy}}(0)]_{n+i} &= -a_i\lambda_i(L_W)/\lambda_i(Y) - a_{n+i}\lambda_i(L_W)/\lambda_{n+i}(Y), \end{aligned}$$

For $i = 1$, the previous equations give $[e_{\mathbf{zy}}(0)]_1 = a_1 + a_{n+1}$ and $[e_{\mathbf{zy}}(0)]_{n+1} = 0$. Thus, we can chose the first coefficient a_1 to be the first element in the error vector, $a_1 = [e_{\mathbf{zy}}(0)]_1$,

and its associated $a_{n+1} = 0$. Proceeding in a similar fashion with the remaining coefficients a_i, a_{n+i}, for $i \in \{2, \ldots, n\}$, we get

$$\begin{aligned} a_i &= -\frac{\lambda_i(L_W)}{b_i}\left([e_{\mathbf{zy}}(0)]_{n+i} + \frac{\lambda_i(Y)}{\lambda_i(L_W)}[e_{\mathbf{zy}}(0)]_i\right), \\ a_{n+i} &= \frac{\lambda_i(L_W)}{b_i}\left([e_{\mathbf{zy}}(0)]_{n+i} + \frac{\lambda_{n+i}(Y)}{\lambda_i(L_W)}[e_{\mathbf{zy}}(0)]_i\right), \end{aligned} \tag{4.32}$$

where b_i is given by (4.14). With the initial error decomposed as in (4.30), the error after t iterations can be expressed as follows

$$e_{\mathbf{zy}}(t) = C^t\ e_{\mathbf{zy}}(0) = \sum_{i=1}^{2n} a_i\ \lambda_i(C)^t\ \mathbf{v}_i(C),$$

which combined with (4.31) and (4.32) gives

$$[e_{\mathbf{zy}}(t)]_1 = (\lambda_1(A))^t\ \ [e_{\mathbf{zy}}(0)]_1, \qquad\qquad [e_{\mathbf{zy}}(t)]_{n+1} = 0,$$

and for all $i \in \{2, \ldots, n\}$,

$$\begin{aligned} [e_{\mathbf{zy}}(t)]_i &= c_{i,n+i}\ [e_{\mathbf{zy}}(0)]_{n+i} + c_{i,i}\ [e_{\mathbf{zy}}(0)]_i, \\ [e_{\mathbf{zy}}(t)]_{n+i} &= -c_{i,n+i}\ [e_{\mathbf{zy}}(0)]_i + c_{n+i,n+i}\ [e_{\mathbf{zy}}(0)]_{n+i}, \end{aligned} \tag{4.33}$$

with

$$\begin{aligned} c_{i,n+i} &= \lambda_i(L_W)\left[(\lambda_{n+i}(A))^t - (\lambda_i(A))^t\right]/b_i, \\ c_{i,i} &= \left[\lambda_{n+i}(Y)(\lambda_{n+i}(A))^t - \lambda_i(Y)(\lambda_i(A))^t\right]/b_i, \\ c_{n+i,n+i} &= \left[-\lambda_i(Y)(\lambda_{n+i}(A))^t + \lambda_{n+i}(Y)(\lambda_i(A))^t\right]/b_i. \end{aligned} \tag{4.34}$$

The squared Euclidean norm $\|e_{\mathbf{zy}}(t)\|_2^2$ of the error vector at iteration t, is given by

$$\begin{aligned} \|e_{\mathbf{zy}}(t)\|_2^2 &= \sum_{i=1}^{n}([e_{\mathbf{zy}}(t)]_i)^2 + ([e_{\mathbf{zy}}(t)]_{n+i})^2 = (\lambda_1(A))^{2t}([e_{\mathbf{zy}}(0)]_1)^2 + \\ &+ \sum_{i=2}^{n}(c_{i,n+i}^2 + c_{i,i}^2)([e_{\mathbf{zy}}(0)]_i)^2 + \sum_{i=2}^{n}(c_{i,n+i}^2 + c_{n+i,n+i}^2)([e_{\mathbf{zy}}(0)]_{n+i})^2 + \\ &+ \sum_{i=2}^{n} 2c_{i,n+i}(c_{i,i} - c_{n+i,n+i})[e_{\mathbf{zy}}(0)]_i[e_{\mathbf{zy}}(0)]_{n+i}, \end{aligned} \tag{4.35}$$

where

$$2c_{i,n+i}(c_{i,i} - c_{n+i,n+i}) = -\frac{2\lambda_i(L_W)(\gamma + \lambda_i(L_W))}{b_i^2}\left((\lambda_{n+i}(A))^t - (\lambda_i(A))^t\right)^2.$$

Note that when $k_1 a$ and $k_2 b$ have the same sign, then $|k_1 a - k_2 b| \leq \max\{k_1, k_2\} \max\{a, b\}$. By taking into account that both $(\lambda_{n+i}(A))^t$ and $(\lambda_i(A))^t \geq 0$ for t even, and that $1/b_i \leq 1/\gamma$, $\lambda_i \leq 2/3\gamma$, $\max\{-\lambda_i(Y), -\lambda_{n+i}(Y)\} \leq 4/3\gamma$, then it can be seen that

$$\begin{aligned} c_{i,n+i}^2 &\leq (2/3)^2 \lambda_{\text{eff}}^{2t}(A), \\ \max\{c_{n+i,n+i}^2, c_{i,i}^2\} &\leq (4/3)^2 \lambda_{\text{eff}}^{2t}(A), \\ |2c_{i,n+i}(c_{i,i} - c_{n+i,n+i})| &\leq (20/3^2) \lambda_{\text{eff}}^{2t}(A). \end{aligned}$$

In addition,

$$\sum_{i=2}^{n} |[e_{\mathbf{zy}}(0)]_i [e_{\mathbf{zy}}(0)]_{n+i}| \leq \sum_{i=2}^{n} (\max\{|[e_{\mathbf{zy}}(0)]_i|, |[e_{\mathbf{zy}}(0)]_{n+i}|\})^2 \leq \|e_{\mathbf{zy}}(0)\|_2^2 .$$

Combining the previous results, we get

$$\|e_{\mathbf{zy}}(t)\|_2^2 \leq 40/3^2 \lambda_{\text{eff}}^{2t}(A) \, \|e_{\mathbf{zy}}(0)\|_2^2 . \tag{4.36}$$

Then, $\|e_{\mathbf{xw}}(t)\|_2 = \|e_{\mathbf{zy}}(t)\|_2$ satisfies

$$\|e_{\mathbf{xw}}(t)\|_2 \leq 2\sqrt{10}/3 \; \lambda_{\text{eff}}^{t}(A) \, \|e_{\mathbf{xw}}(0)\|_2 , \tag{4.37}$$

as in eq. (4.27) and the proof is completed. □

The convergence speed in Theorem 4.3.3 depends on $\lambda_{\text{eff}}(A) = \max\{\lambda_{n+2}(A), -\lambda_n(A)\}$, which is related to the eigenvalues $\lambda_2(L_W), \lambda_n(L_W)$ of the Laplacian L_W of the communication graph. These eigenvalues depend on the graph topology and require global information of the network. In Chapter 7.3 we propose a distributed algorithm that allows each robot to compute the leading eigenvalue $\lambda_n(L_W)$ and the algebraic connectivity $\lambda_2(L_W)$ of the Laplacian; alternatively, other distributed methods [58, 63] could be used. Then, the robots can compute $\lambda_n(A)$, $\lambda_{n+2}(A)$ and find the one with the largest absolute value. In this case, they can also compute the optimal step size h^* such that $-\lambda_n(A) = \lambda_{n+2}(A)$,

$$h^* = 4/(2\gamma + \lambda_n(L_W) + \lambda_2(L_W) + b_n - b_2). \tag{4.38}$$

4.4 Dynamic Averaging Strategy

Now we consider the dynamic scenario, where each robot i observes its input u_i^k, whose value varies along the steps $k = 1, \ldots, K$. The goal is that each robot computes and tracks the average of the inputs $\frac{1}{n}\sum_{i=1}^{n} u_i^k$ up to step k.

We adopt an strategy where, each step $k = 1, \ldots, K$, the robots run the consensus algorithm in Section 4.3, to compute the average of the inputs up to step k (Fig.4.2). They initialize their consensus filters with their previous average estimate. They execute a total number of L consensus iterations, divided into l iterations per input update step, and the remaining $L - l(K-1)$ after the last step.

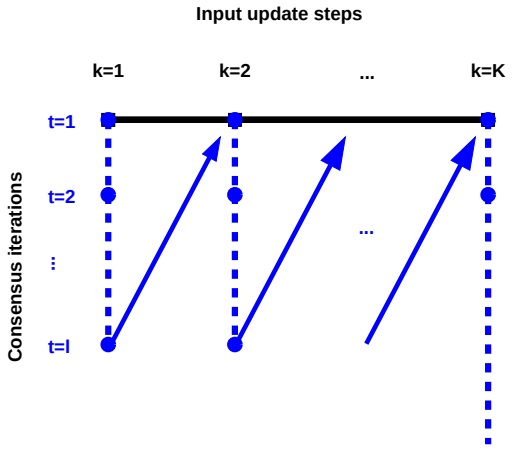


Figure 4.2: **Dynamic consensus**. Each robot $i \in \{1, \ldots, n\}$ observes its inputs u_i^k at the input update steps $k = 1, \ldots, K$. Between any two input update steps, they execute l consensus iterations to compute the average of the inputs. Then, they use these average estimates as an initial solution for the consensus iterations in the next step.

Remark 4.4.1. *Throughout this section, we consider that the maximum number of consensus iterations L is limited by the problem requirements and it is a priori given to the robots. This value L may depend, e.g., on the amount of energy each robot has for carrying out its operation, the power consumption of each data exchange operation, and the energy assigned to other robot tasks. We consider that the number of iterations per step l is also established a priori. It may be selected so that the timespan of input update steps is the desired one, taking into account the time consumed by the computation and communication operations executed by the robots.*

In case the robot team does not have any of the previous limitations, then the robots can select the desired l_*^k for each step so that their estimates reach a certain precision. For instance, if their goal is maintaining a relative estimation error of ϵ at each step k, $\left\|e_{\mathbf{xw}}^k(l_*^k)\right\|_2 \;/\; \left\|e_{\mathbf{xw}}^k(0)\right\|_2 \leq \epsilon$, then, from eq. (4.27), the desired value of l_k^* would be

$$l_*^k \geq \left(\log\left(\epsilon\right) - \log\left(\beta\right)\right) / \log\left(\lambda_{\text{eff}}(A)\right).$$

We do not specify the number of local observation-estimation iterations carried out by each robot between consecutive steps k and $k+1$. Using this strategy, if a map update step starts, and a robot is not ready for transmitting its updated local map, it can act as if it was disconnected from the communication network. □

From now on, we add the superscript k to the Laplacian L_W^k which its associated to the communication graph $\mathcal{G}^k = \left(\mathcal{V}, \mathcal{E}^k\right)$ at step k and we let $[\mathbf{r}\ S_2^k \ldots S_n^k] = [\mathbf{r}\ S^k]$ be a basis of eigenvectors of L_W^k as in eq. (4.12). Note that the eigenvector $\mathbf{r}$ is common to all the Laplacians L_W^k. We let A^k be the system matrix associated to L_W^k given by (4.16). We also add the superscript k to the inputs $\mathbf{u}^k = (u_1^k, \ldots, u_n^k)$ and the states $\mathbf{x}^k(t) = (x_1^k(t), \ldots, x_n^k(t))$, $\mathbf{w}^k(t) = (w_1^k(t), \ldots, w_n^k(t))$ of the consensus algorithm with constant inputs (4.15) to identify the associated step k. We define $\mathbf{x}_*^k$ and $\mathbf{w}_*^k \in \mathbb{R}^n$ as we

did in the previous section but using the variables at step k,

$$\mathbf{x}_*^k = \mathbf{r}\mathbf{r}^T\mathbf{u}^k = \mathbf{1}x_{avg}^k, \qquad \mathbf{w}_*^k = \mathbf{r}\mathbf{r}^T\mathbf{w}^k(0) - \gamma(L_W^k)^{(-1)}\mathbf{u}^k, \tag{4.39}$$

being $\mathbf{r}$, $(L_W^k)^{(-1)}$ as in (4.11), (4.13), and let $\lambda_\star$ be

$$\lambda_\star = \max_{k\in\{1,\dots,K\}} \lambda_{\text{eff}}(A^k). \tag{4.40}$$

The proposed dynamic consensus algorithm is detailed in Algorithm 4.4.1, where the step size $h > 0$ and parameter $\gamma > 0$ of the consensus algorithm with constant inputs (4.15) (Algorithm 4.4.1, lines 6 and 13) are as in Proposition 4.3.2 for all k. In the same way that the consensus algorithm with constant inputs was fully distributed, the dynamic consensus algorithm is distributed as each robot updates its data by local interactions with its neighbors.

Algorithm 4.4.1 *Dynamic consensus algorithm* - Robot i

1: – *Initialization* at $k = 1$
2: $x_i^k(0) = 0, w_i^k(0) = 0$, $u_i^k \leftarrow$ current local map
3: –*Algorithm*
4: **for** each step $k = 1, \dots, K-1$ **do**
5: execute algorithm (4.15) for $t = l$ iterations:
6: $[x_i^k(t), w_i^k(t)] = \text{consensus\_alg}\left(u_i^k, x_i^k(0), w_i^k(0)\right)$
7: initialize the states with the previous estimates:
8: $x_i^k(0) = x_i^k(t)$, $w_i^k(0) = w_i^k(t)$,
9: $u_i^k \leftarrow$ current local map
10: **end for**
11: – *Final step* at $k = K$
12: execute (4.15) for the remaining $t = L - (K-1)l$ iterations:
13: $[x_i^k(t), w_i^k(t)] = \text{consensus\_alg}\left(u_i^k, x_i^k(0), w_i^k(0)\right)$

The rate of convergence for the dynamic consensus algorithm (Algorithm 4.4.1) depends on (i) the initial input and graph at $k = 1$, and (ii) the changes on both the input and the graph topology during consecutive steps. We let α and σ represent this information,

$$\alpha = \alpha_k \text{ for } k = 1, \qquad \sigma = \max_{k\in\{1,\dots,K-1\}} \sigma_k, \tag{4.41}$$

$$\begin{aligned} \text{with } \alpha_k &= \left(\|\mathbf{r}^T\mathbf{u}^k\|_2^2 + \gamma^2\|(L_W^k)^{(-1)}\mathbf{u}^k\|_2^2\right)^{1/2}, \\ \text{and } \sigma_k &= (\|\mathbf{r}^T(\mathbf{u}^k - \mathbf{u}^{k+1})\|_2^2 + \gamma^2\|(L_W^k)^{(-1)}\mathbf{u}^k - (L_W^{k+1})^{(-1)}\mathbf{u}^{k+1}\|_2^2)^{1/2}. \end{aligned} \tag{4.42}$$

As the following result states, under mild connectivity conditions on the communication graphs $\mathcal{G}^k$, the states $x_i^k(t)$ at each robot i correctly track the average of the inputs x_{avg}^k for each k.

Theorem 4.4.2. *Assume all robots in $\mathcal{V}$ execute the dynamic consensus strategy detailed in Algorithm 4.4.1 and that their undirected communication graphs $\mathcal{G}^k$ are connected for any step $k \in \{1, \dots, K\}$. Then, the states $\mathbf{x}^k(t) \in \mathbb{R}^n$, $\mathbf{w}^k(t) \in \mathbb{R}^n$ of Algorithm 4.4.1 converge exponentially to*

$$\lim_{t\to\infty} \mathbf{x}^k(t) = \mathbf{x}_*^k, \qquad \lim_{t\to\infty} \mathbf{w}^k(t) = \mathbf{w}_*^k, \tag{4.43}$$

as $t \to \infty$, where $\mathbf{x}_*^k$ and $\mathbf{w}_*^k$ are given by (4.39). Moreover, the error vector $e_{\mathbf{xw}}^k(t) = [(\mathbf{x}^k(t))^T, (\mathbf{w}^k(t))^T]^T - [(\mathbf{x}_*^k)^T, (\mathbf{w}_*^k)^T]^T$ for each step $k \in \{1, \dots, K\}$ after t iterations, with t even, satisfies

$$\begin{aligned} \left\|e_{\mathbf{xw}}^k(t)\right\|_\infty \leq \left\|e_{\mathbf{xw}}^k(t)\right\|_2 \leq \alpha f_k(t) + \sigma g_k(t), \\ f_k(t) = \beta^k \lambda_\star^{t+(k-1)l}, \quad g_k(t) = \beta \lambda_\star^t \sum_{p=0}^{k-2} (\beta \lambda_\star^l)^p, \end{aligned} \tag{4.44}$$

where l is the number of iterations of the consensus algorithm with constant inputs executed per input update step, $\beta = 2\sqrt{10}/3$, and $\lambda_\star$, α and σ are given by eqs. (4.40) and (4.41).

Proof. The convergence of the states $\mathbf{x}^k(t) \in \mathbb{R}^n$ and $\mathbf{w}^k(t) \in \mathbb{R}^n$ to $\mathbf{x}_*^k$ and $\mathbf{w}_*^k$ in (4.39) is a consequence of Theorem 4.3.3. Regarding the convergence rate, as stated by Theorem 4.3.3 the error vector after l iterations satisfies

$$\left\|e_{\mathbf{xw}}^k(l)\right\|_2 \leq \beta \lambda_\star^l \left\|e_{\mathbf{xw}}^k(0)\right\|_2 . \tag{4.45}$$

The final error vector $e_{\mathbf{xw}}^k(l)$ at step k and the initial error vector $e_{\mathbf{xw}}^{k+1}(0)$ at the next step $k+1$ are related as follows:

$$e_{\mathbf{xw}}^{k+1}(0) = e_{\mathbf{xw}}^k(l) + \begin{bmatrix} \mathbf{x}_*^k \\ \mathbf{w}_*^k \end{bmatrix} - \begin{bmatrix} \mathbf{x}_*^{k+1} \\ \mathbf{w}_*^{k+1} \end{bmatrix}. \tag{4.46}$$

Combining the previous results we get

$$\begin{aligned} \left\|e_{\mathbf{xw}}^k(t)\right\|_2 &\leq \beta^k \lambda_\star^{t+(k-1)l} \left\|e_{\mathbf{xw}}^1(0)\right\|_2 \\ &+ \beta \lambda_\star^t \sum_{p=0}^{k-2} (\beta \lambda_\star^l)^p \left\| \begin{bmatrix} \mathbf{x}_*^{k-p-1} - \mathbf{x}_*^{k-p} \\ \mathbf{w}_*^{k-p-1} - \mathbf{w}_*^{k-p} \end{bmatrix} \right\|_2, \end{aligned} \tag{4.47}$$

where at step $k = 1$ the states are initialized with zeros (Algorithm 4.4.1, line 2), and thus the initial error at step $k = 1$ and iteration $t = 0$ is $e_{\mathbf{xw}}^1(0) = [(-\mathbf{x}_*^1)^T, (-\mathbf{w}_*^1)^T]^T$. We compute the norm of the initial error $\|e_{\mathbf{xw}}^1(0)\|_2$ and obtain α in eq. (4.41), where we have used the fact that $\left\|(a^T, b^T)^T\right\|_2^2 = \|a\|_2^2 + \|b\|_2^2$. Proceeding in a similar fashion, for $k = 1, \dots, K-1$, the norms $\left\|((\mathbf{x}_*^k - \mathbf{x}_*^{k+1})^T, (\mathbf{w}_*^k - \mathbf{w}_*^{k+1})^T)^T\right\|_2$ are equal to σ_k in eq. (4.42), and thus they are smaller than σ in eq. (4.41). We finally obtain the expression in eq. (4.44) and the proof is completed. □

The interest of the proposed dynamic consensus algorithm is that the robots use the estimates at the previous step $k-1$ for initializing their estimates at step k. As it can be seen from the rate of convergence,

$$\beta^k \lambda_\star^{t+(k-1)l} \alpha_1 + \beta \lambda_\star^t \sum_{p=0}^{k-2} (\beta \lambda_\star^l)^p \sigma_{k-p-1},$$

errors associated to previous steps $\beta^k \lambda_\star^{t+(k-1)l} \alpha_1$, and $\beta \lambda_\star^t (\beta \lambda_\star^l)^{k-p-1} \sigma_p$, for $p = 1, \ldots, k-2$, are small since they have already been reduced by the execution of the algorithm. The error associated to the last step $\beta \lambda_\star^t \sigma_{k-1}$ depends on the variation of the input and graph topology between steps $k-1$ and k. Consider instead a zero-initialization strategy, where at each step k the robots discard their old estimates and compute the average at step k from scratch (initializing their estimates with zeros). The rate of convergence of this zero-initialization strategy would be given by

$$\beta \lambda_\star^t \alpha_k.$$

Note that the term α_k given by eq. (4.41) depends on the input and the graph itself. Therefore, if the variation of inputs and graph topologies σ_k are small compared to the input itself α_k, then the dynamic consensus algorithm is preferable to the zero-initialization strategy.

Theorem 4.4.2 can further be used to analyze the behavior of the algorithm under changes in the communication graph. So far we have assumed that at each step k, during the l iterations employed by the robots to reach consensus on the average of the input $\mathbf{u}^k$, the graph $\mathcal{G}^k$ remains fixed. Now consider instead that after $t < l$ iterations we let the graph change. This is equivalent to having a new step $k+1$ with a smaller l, and with the new graph $\mathcal{G}^{k+1}$ and with the same input, $\mathbf{u}^k = \mathbf{u}^{k+1}$. In this case, the additional error introduced due to the graph change σ_{k+1} in eq. (4.42) is $\gamma \|(L_W^{k+1})^{(-1)} \mathbf{u}^k\|_2$. Therefore, as long as the changes in the topology σ_{k+1} are small and slow enough compared to the number of iterations t and l, the algorithm will correctly track the average of the inputs.

4.5 Dynamic Map Merging Algorithm

We adapt the dynamic consensus strategy (Algorithm 4.4.1) presented in Section 4.4 to operate on matrices and vectors instead of on scalar inputs. This generalization is key for merging feature-based stochastic IF maps.

The local maps to be merged given by (4.3) are represented by a $\mathcal{M}_G \times \mathcal{M}_G$ information matrix and an information vector of size $\mathcal{M}_G$. The robots execute in parallel many instances of Algorithm 4.4.1 on each entry (r, s) within its information matrix and on each r entry within its information vector, for $r, s \in \{1, \ldots, \mathcal{M}_G\}$[3]. Let us add the subscripts $\{I, r, s\}$ or $\{\mathbf{i}, r\}$ to the variables $u_i^k, x_i^k(t), w_i^k(t)$ to identify the instance we are

[3]Initially, this would suppose a total of $\mathcal{M}_G^2 + \mathcal{M}_G$ instances of the consensus algorithm. However, since the information matrix is symmetric, it only has $\frac{1}{2}\mathcal{M}_G(\mathcal{M}_G + 1)$ different entries. Therefore, the robots actually execute $\mathcal{M}_G + \frac{1}{2}\mathcal{M}_G(\mathcal{M}_G + 1)$ instances of the consensus algorithm instead of $\mathcal{M}_G + \mathcal{M}_G^2$.

referring to. At step k, each entry within the information matrix $[I_i^k]_{r,s}$ and vector $[\mathbf{i}_i^k]_r$ of the local map of robot $i \in \mathcal{V}$ (4.3) is used as an input for an instance of Algorithm 4.4.1,

$$u_{i\ \{I,r,s\}}^k = [I_i^k]_{r,s}, \qquad u_{i\ \{\mathbf{i},r\}}^k = [\mathbf{i}_i^k]_r,$$

for $r, s \in \{1, \ldots, \mathcal{M}_G\}$. The states $x_i^k(t)$ of all the instances of Algorithm 4.4.1 at iteration t and robot $i \in \mathcal{V}$ are arranged into the temporal information matrix $I_i^k(t) \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ and vector $\mathbf{i}_i^k(t) \in \mathbb{R}^{\mathcal{M}_G}$ as follows,

$$[I_i^k(t)]_{r,s} = x_i^k(t)_{\ \{I,r,s\}}, \qquad [\mathbf{i}_i^k(t)]_r = x_i^k(t)_{\ \{\mathbf{i},r\}},$$

for $r, s \in \{1, \ldots, \mathcal{M}_G\}$. For each robot $i \in \mathcal{V}$, $k \in \{1, \ldots, K\}$, $t = 0, 1, \ldots$, we define its estimate of the mean $\hat{\mathbf{x}}_i^k(t) \in \mathbb{R}^{\mathcal{M}_G}$ and covariance $\Sigma_i^k(t) \in \mathbb{R}^{\mathcal{M}_G \times \mathcal{M}_G}$ of the global map, at iteration t and step k as

$$\hat{\mathbf{x}}_i^k(t) = \left(I_i^k(t)\right)^{-1} \mathbf{i}_i^k(t), \qquad \Sigma_i^k(t) = \left(I_i^k(t)\right)^{-1} / n. \tag{4.48}$$

Recall our discussion in Section 4.2 about the two parts of the dynamic map merging: (i) propagating the rows and columns of I_i^k, $\mathbf{i}_i^k$ associated the pose of a robot j; and (ii) reaching consensus on the entries associated exclusively to features through the instances of Algorithm 4.4.1. In eq. (4.48) we are assuming that the information concerning the poses of the robots has already been received by the robots and incorporated into their information matrices and vectors.

For simplicity, we are presenting the structures of the information matrices and vectors $\mathbf{i}_i^k(t)$, $I_i^k(t)$, as fixed and known by all the robots. Actually, the robots discover the features observed by the others in the messages exchanged at each iteration, see Chapter 5 for a detailed discussion of this issue. Robots use this information to introduce new columns and rows into their information matrices and vectors. If a robot has never received information of a feature, e.g., cause it has been observed by a distant robot, it simply does not have any space for it in its information matrix and vector. As a result, the information matrices and vectors do not contain non-informative zero rows and columns. Therefore, $I_i^k(t)$ in eq. (4.48) can be inverted at each iteration of the algorithm and thus the global map can always be estimated.

4.6 Convergence for Fixed Networks

Corollary 4.6.1. *Assume all the robots $i \in \mathcal{V}$ execute the dynamic consensus (Algorithm 4.4.1) on each entry of its information matrix and vector as detailed above, and assume that their undirected communication graphs $\mathcal{G}^k$ are connected for all $k \in \{1, \ldots, K\}$. Then, for the last step K, the mean $\hat{\mathbf{x}}_i^K(t)$ and covariance $\Sigma_i^K(t)$ at each robot $i \in \mathcal{V}$ asymptotically converge to the mean $\hat{\mathbf{x}}_G^K$ and covariance Σ_G^K of the global map given by* (4.5),

$$\lim_{t\to\infty} \hat{\mathbf{x}}_i^K(t) = \hat{\mathbf{x}}_G^K, \qquad \lim_{t\to\infty} \Sigma_i^K(t) = \Sigma_G^K. \tag{4.49}$$

4.7 Convergence Speed for Fixed Networks

Corollary 4.7.1. *Assume all the robots $i \in \mathcal{V}$ execute the dynamic consensus (Algorithm 4.4.1) on each entry of its information matrix and vector as detailed above, and assume that their undirected communication graphs $\mathcal{G}^k$ are connected for all $k \in \{1, \dots, K\}$. Let I^k_{avg}, $\mathbf{i}^k_{avg}$ be the average of the local maps in IF form at step k,*

$$I^k_{avg} = \frac{1}{n}\sum_{j=1}^{n} I^k_j, \qquad \mathbf{i}^k_{avg} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{i}^k_j. \tag{4.50}$$

Let $\alpha_{\{I,r,s\}}$ and $\sigma_{\{I_r,s\}}$, be defined as α, σ in (4.41) for the inputs $u^k_{i\ \{I,r,s\}} = [I^k_i]_{r,s}$; equivalently $\alpha_{\{\mathbf{i},r\}}$, $\sigma_{\{\mathbf{i},r\}}$ for the inputs $u^k_{i\ \{\mathbf{i},r\}} = [\mathbf{i}^k_i]_r$. We let α_I and σ_I, respectively $\alpha_{\mathbf{i}}$ and $\sigma_{\mathbf{i}}$, be the maximum over all the entries of I, respectively of $\alpha_{\mathbf{i}}$,

$$\begin{aligned} \alpha_I &= \max_{r,s\in\{1,\dots,\mathcal{M}_G\}} \alpha_{\{I,r,s\}}, & \sigma_I &= \max_{r,s\in\{1,\dots,\mathcal{M}_G\}} \sigma_{\{I,r,s\}}, \\ \alpha_{\mathbf{i}} &= \max_{r\in\{1,\dots,\mathcal{M}_G\}} \alpha_{\{\mathbf{i},r\}}, & \sigma_{\mathbf{i}} &= \max_{r\in\{1,\dots,\mathcal{M}_G\}} \sigma_{\{\mathbf{i},r\}}. \end{aligned} \tag{4.51}$$

Then, for all $i \in \mathcal{V}$, $k \in \{1, \dots, K\}$, $r, s \in \{1, \dots, \mathcal{M}_G\}$, $t \geq 0$, the entry $[I^k_i(t)]_{r,s}$ within the information matrix and the entry $[\mathbf{i}^k_i(t)]_r$ within the information vector estimated by robot i after t iterations satisfy

$$\begin{aligned} |[I^k_i(t)]_{r,s} - [I^k_{avg}]_{r,s}| &\leq \alpha_I f_k(t) + \sigma_I g_k(t), \\ |[\mathbf{i}^k_i(t)]_r - [\mathbf{i}^k_{avg}]_r| &\leq \alpha_{\mathbf{i}} f_k(t) + \sigma_{\mathbf{i}} g_k(t), \end{aligned} \tag{4.52}$$

where the convergence speed expressions $f_k(t)$, $g_k(t)$, are defined in Theorem 4.4.2, eq. (4.44).

4.8 Properties of the Partial Estimates

An interesting property of this map merging algorithm is that the temporal global maps $\hat{\theta}^k_i(t)$ estimated at each robot i, are unbiased estimates of the true feature positions $\mathbf{x}$. As a result, the robots do not need to wait for any specific number of iterations of the map merging algorithm. Instead, they can make decisions on their temporal global map estimates whenever they need.

Proposition 4.8.1. *The estimates of the global map mean $\hat{\mathbf{x}}^k_i(t)$, for each robot $i \in \mathcal{V}$, at a step $k \in \{1, \dots, K\}$, after t iterations of the dynamic consensus algorithm, are unbiased estimates of the true feature positions $\mathbf{x}$,*

$$\mathrm{E}\left[\hat{\mathbf{x}}^k_i(t)\right] = \mathrm{E}\left[\left(I^k_i(t)\right)^{-1}\mathbf{i}^k_i(t)\right] = \mathbf{x}. \tag{4.53}$$

Proof. The temporal values of $I^k_i(t)$, $\mathbf{i}^k_i(t)$, that evolve according to Algorithm 4.4.1, can be alternatively expressed as a function of the inputs $I^1_j, \dots, I^K_j$, $\mathbf{i}^1_j, \dots, \mathbf{i}^K_j$, (4.3), and the

initial states. Since the states at $k = 1$ and $t = 0$ are zero (Algorithm 4.4.1, line 2), then $I_i^k(t)$ and $\mathbf{i}_i^k(t)$ are

$$\begin{aligned} I_i^k(t) &= \sum_{j=1}^{n} [\Phi(k,t)]_{i,j}\ I_j^k + \sum_{p=1}^{k-1}\sum_{j=1}^{n} [\Omega(k,t,p)]_{i,j}\ I_j^p, \\ \mathbf{i}_i^k(t) &= \sum_{j=1}^{n} [\Phi(k,t)]_{i,j}\ \mathbf{i}_j^k + \sum_{p=1}^{k-1}\sum_{j=1}^{n} [\Omega(k,t,p)]_{i,j}\ \mathbf{i}_j^p, \end{aligned} \tag{4.54}$$

where the matrices $\Phi(k,t), \Omega(k,t,p), \Psi(t_1,t_2) \in \mathbb{R}^{2n\times 2n}$ are

$$\begin{aligned} \Phi(k,t) &= \sum_{\tau=1}^{t} \Psi\left(\tau + (k-1)l, t-1+(k-1)l\right) \begin{bmatrix} h\gamma\mathbf{I} \\ \mathbf{0} \end{bmatrix}, \\ \Omega(k,t,p) &= \sum_{\tau=1}^{l} \Psi\left(\tau + (p-1)l, t-1+(k-1)l\right) \begin{bmatrix} h\gamma\mathbf{I} \\ \mathbf{0} \end{bmatrix}, \\ \Psi(t_1,t_2) &= A(t_2)\dots A(t_1+1)A(t_1), \end{aligned}$$

and $A(t+kl) = A^k(t)$ is the system matrix associated to the iteration t and step k given by (4.16). Since the local maps $\hat{\mathbf{x}}_j^k$ at each robot j are an estimate of the true $\mathbf{x}$ (4.2),

$$\hat{\mathbf{x}}_j^k = H_i^k\mathbf{x} + \mathbf{v}_j^k, \text{ with} \qquad\qquad \mathrm{E}\left[\mathbf{v}_j^k\right] = 0,$$

then the inputs $\mathbf{i}_j^k = (H_j^k)^T(\Sigma_j^k)^{-1}\hat{\mathbf{x}}_j^k$ are

$$\mathbf{i}_j^k = (H_j^k)^T(\Sigma_j^k)^{-1}\mathbf{v}_j^k + I_j^k\mathbf{x}. \tag{4.55}$$

Combining eqs. (4.54) and (4.55), variables $\mathbf{i}_i^k(t)$ are given by

$$\begin{aligned} \mathbf{i}_i^k(t) = &\sum_{j=1}^{n} [\Phi(k,t)]_{i,j}(H_j^k)^T(\Sigma_j^k)^{-1}\mathbf{v}_j^k + \\ &\sum_{p=1}^{k-1}\sum_{j=1}^{n} [\Omega(k,t,p)]_{i,j}(H_j^{k-p})^T(\Sigma_j^{k-p})^{-1}\mathbf{v}_j^{k-p} + \\ &\left(\sum_{j=1}^{n} [\Phi(k,t)]_{i,j}\ I_j^k + \sum_{p=1}^{k-1}\sum_{j=1}^{n} [\Omega(k,t,p)]_{i,j}\ I_j^p\right)\mathbf{x}, \end{aligned}$$

where the last term is exactly $I_i^k(t)\mathbf{x}$, with $I_i^k(t)$ as in eq. (4.54). Then $\hat{\mathbf{x}}_i^k(t) = (I_i^k(t))^{-1}\mathbf{i}_i^k(t)$ is

$$\begin{aligned} \hat{\mathbf{x}}_i^k(t) = \mathbf{x} &+ (I_i^k(t))^{-1}\left(\sum_{j=1}^{n} [\Phi(k,t)]_{i,j}(H_j^k)^T(\Sigma_j^k)^{-1}\mathbf{v}_j^k\right) + \\ &(I_i^k(t))^{-1}\left(\sum_{p=1}^{k-1}\sum_{j=1}^{n} [\Omega(k,t,p)]_{i,j}(H_j^{k-p})^T(\Sigma_j^{k-p})^{-1}\mathbf{v}_j^{k-p}\right). \end{aligned}$$

Since the noises $\mathbf{v}_j^k$ have zero mean for all $k \in \{1,\dots,K\}$ and all $j \in \mathcal{V}$, the expected value of $\hat{\mathbf{x}}_i^k(t)$ is $\mathbf{x}$. □

Note that this property holds also for time-varying graphs, where $A(t+kl)$ is different for each iteration t and each step k. For fixed graphs, $\Psi(t_1+kl, t_2+kl)$ is simply $(A^k)^{t_2-t_1+1}$.

4.9 Discussion

A set of experiments has been carried out with a team composed by 9 robots that explore the region inside the black box in Fig. 4.3. The robots run a total of $K=5$ map update

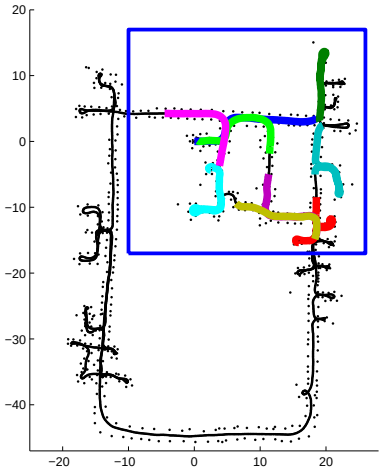
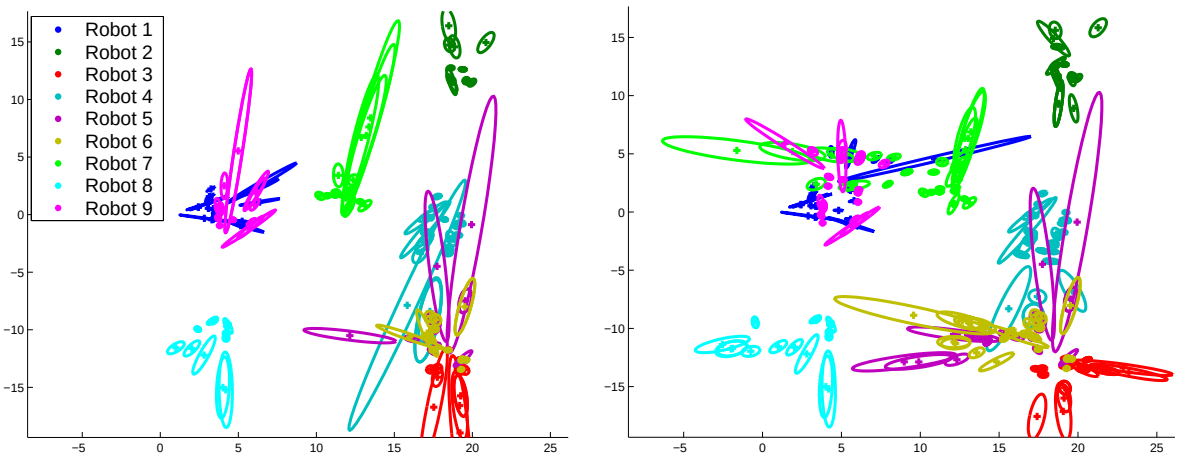


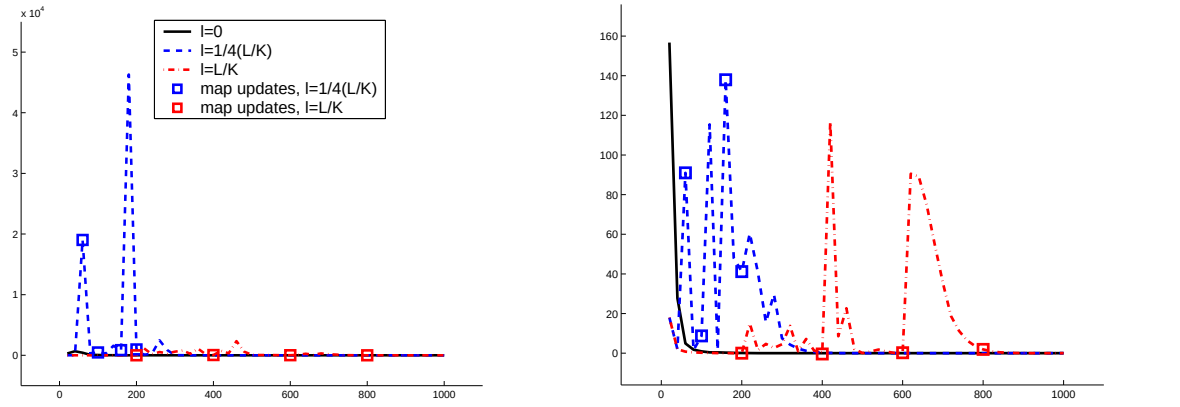
Figure 4.3: Trajectories followed by 9 robots. They cover a region of 30m x 30m of the scene [60]. In order to give an idea of the scene structure, we display in black the path and a set of artificial landmarks (black dots) placed on both sides of the trajectory, which are not used in the experiment. Here, the rooms can be identified since robots enter and leave them describing short trajectories. The long, straight motions correspond to corridors.

steps. Between consecutive map update steps k, $k+1$, each robot performs 20 steps of a bearing-only SLAM algorithm [11]. We display in different colors the 9 local maps for the map update steps $k=1$, and $k=2$. (Fig. 4.4). The robots execute a total of $L=1000$ consensus iterations. We experiment with 3 different configurations. In the first one, the robots execute all the consensus iterations after the last map update step, $l=0$. This is equivalent to a static map merging. In the second case, we use $l=\frac{1}{4}(L/K)$, and in the last one, we use an equal number of iterations per step $l=(L/K)$. The best results are obtained with the first configuration $l=0$ (Fig. 4.5), since the robots only need to agree on the last map $k=K$. In the second case, $l=\frac{1}{4}(L/K)$, the robots employ their first 50 iterations on reaching consensus on the first map $k=1$. Then, every 50 iterations, the input maps change again. They start to reach consensus on the map $k=K$ after the iteration 200. And after that, they converge very fast to the global map. In the last configuration, $l=(L/K)$, the robots use more consensus iterations than in $l=\frac{1}{4}(L/K)$ for the maps at $k=1,2,\ldots,4$. Their estimates of these temporal maps are better. However, the robots start to estimate the last map $k=K$ after iteration 800. After the L iterations, the global maps $\hat{\theta}_i^K(L)$, $\Sigma_{\theta_i}^K(L)$, computed by the dynamic map merging



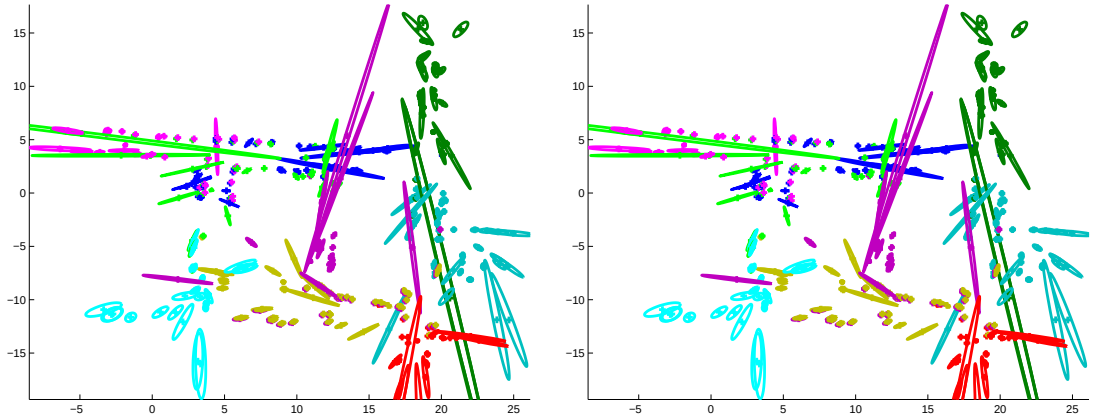
(a) $k = 1$ **(b)** $k = 2$

Figure 4.4: Local maps of the 9 robots used for the update steps $k = 1$ (a) and $k = 2$ (b). They have been obtained after, respectively, 20 and 40 SLAM steps. They are expressed in a common reference frame. As it can be seen, in $k = 2$ the feature estimates have been updated. In addition, the robots have introduced new features into their local maps.



(a) $||\hat{\theta}_i^k(t) - \hat{\theta}_c^k||$ **along the** L **iterations** **(b)** $||\mathrm{Tr}(\Sigma^k_{\hat{\theta}_i}(t)) - \mathrm{Tr}\left(\Sigma_c^k(t)\right)||$ **along the** L **iterations**

Figure 4.5: Estimation errors at robot 1 along the L consensus iterations. (a) $||\hat{\theta}_i^k(t) - \hat{\theta}_c^k||$, (b) $||\mathrm{Tr}(\Sigma^k_{\hat{\theta}_i}(t)) - \mathrm{Tr}\left(\Sigma_c^k(t)\right)||$. The configuration $l = 0$ (black solid line) employs all the iterations in reaching consensus on the last map $k = 5$. In the configuration $l = \frac{1}{4}(L/K)$ (blue dashed line), every 50 iterations the local maps change (blue squares). The robots start the consensus on the last map after the iteration 200. In the last configuration $l = (L/K)$ (red dash-dotted line), the map update steps start every 200 iterations (red squares). The consensus on the last map begins at iteration 800.



(a) Robot 1 **(b) Centralized system**

Figure 4.6: Global map estimated by robot $i = 1$ at the last consensus iteration, $\hat{\theta}_i^K(L)$, $\Sigma_{\hat{\theta}_i}^K(L)$, for the configuration $l = \frac{1}{4}(L/K)$ (a), and global map that would be obtained by a centralized system, $\hat{\theta}_G^K$,Σ_G^K (b). We display in different colors the sections that correspond to different initial local maps.

algorithm, are very close to the global map $\hat{\theta}_G^K$, Σ_G^K (4.5) that would be obtained by a centralized system (Fig.4.6 b). We show the global map at robot 1, for the $l = \frac{1}{4}(L/K)$ configuration (Fig.4.6 a), which is very similar to the maps computed by the other robots. Similar results have been obtained using the other configurations.

4.10 Conclusions

In this chapter we have presented an algorithm for dynamically merging visual maps in a robot network with limited communication. This algorithm allows the robots to have a better map of the environment containing the features observed by any other robot in the team. Thus, it helps the coordination of the team in several multi-robot tasks such as exploration or rescue. The algorithm correctly propagates the new information added by the robots to their local maps. We have shown that, with the proposed strategy, the robots correctly track the global map. At the final step, they obtain the last global map, which contains the last updated information at all the robots.

Future extensions of this work are related to the improvement of the communication network usage. The number of consensus iterations may be optimized by a proper selection of the weights and μ in eq. (4.16) or by controlling the network topology to maximize its connectivity. The amount of information exchanged by the robots can be improved by applying submapping ideas or by sending only the most informative elements.

Chapter 5
Distributed Data Association

In this chapter we address the data association problem of features observed by the robots in a network with limited communications. At every time instant, each robot can only exchange data with a subset of the robots, its neighbors. Initially, each robot solves a local data association with each of its neighbors. After that, the robots execute the proposed algorithm to agree on a data association between all their local observations which is globally consistent. One inconsistency appears when chains of local associations give rise to two features from one robot being associated among them. The contribution of this work is the decentralized detection and resolution of these inconsistencies. We provide a fully decentralized solution to the problem. This solution does not rely on any particular communication topology. Every robot plays the same role, making the system robust to individual failures. Information is exchanged exclusively between neighbors. In a finite number of iterations, the algorithm finishes with a data association which is free of inconsistent associations. We show the performance of the proposed algorithms through exhaustive experimentation.

5.1 Introduction

In the commented multi-robot systems, a team of robots cooperatively perform some task in a more efficient way than a single robot would do. In this chapter, we address the data association problem. It consists of establishing correspondences between different measurements or estimates of a common element. It is of high interest in localization, mapping, exploration, and tracking applications [15]. The Nearest Neighbor (NN), and the Maximum Likelihood (ML), are widely used methods which associate each observation with its closest feature in terms of the Euclidean or the Mahalanobis distance [159], [72], [64]. Other popular method is the Joint Compatibility Branch and Bound (JCBB) [97], which considers the compatibility of many associations simultaneously. The Combined Constraint Data Association [16] builds a graph where the nodes are individually compatible associations and the edges relate binary compatible assignments. Over this graph, a Maximal Common Subgraph problem is solved for finding the maximum clique in the graph. Scan matching and Iterative Closest Point (ICP) [41] are popular methods for comparing two laser scans. Other methods, like the Multiple Hypothesis Tracking, and the Joint Probabilistic Data Association, maintain many association hypothesis instead of selecting one of them. And exist many variations of these

techniques that combine RANSAC [55] for higher robustness.

In solutions based on submaps, one of them is usually transformed into an observation of another. The local submaps are merged with the global map following a sequence [149], or in a hierarchical binary tree fashion [29]. All the mentioned data association approaches, operate on elements from two sets. One set usually contains the current observations, and the other one consists of the feature estimates. These sets may be two images, two laser scans, or two probabilistic maps.

Lately, many localization, mapping, and exploration algorithms for multi-robot systems have been presented. However, they have not fully addressed the problem of multi-robot data association. Some solutions have been presented for merging two maps [137], [159] that do not consider a higher number of robots. Many approaches rely on broadcasting all controls and observations measured by the robots. Then, the data association is solved like in a single robot scenario, using scan matching and ICP for laser scans [69], [56, 78, 110], or NN, ML, and visual methods for feature-based maps [64], [83]. In these methods, the problem of inconsistent data associations is avoided by forcing a cycle-free merging order. This limitation has also been detected in the computer vision literature. In [53] they approach an inconsistent association problem for identifying equal regions in different views. They consider a centralized scenario, where each 2 views are compared among them in a 2-by-2 way. Then, their results are arranged on a graph where associations are propagated and conflicts are solved. The work in [48], from the target tracking literature, simultaneously considers the association of all local maps. It uses an expectation-maximization method for both computing the data association and the final global map. The main limitation of this work is that the data from all sensors needs to be processed together, what implies a centralized scheme, or a broadcast method.

All the previous methods rely on centralized schemes, full communication between the robots, or broadcasting methods. However, in multi-robot systems, distributed approaches are more interesting. They present a natural robustness to individual failures since there are no central nodes. Besides, they do not rely on any particular communication scheme, and they are robust to changes in the topology. On the other hand, distributed algorithms introduce an additional level of complexity in the algorithm design. Although the robots make decisions based on their local data, the system must exhibit a global behavior. In this chapter, we address the data association problem for distributed robot systems. Each of our robots posses a local observation of the environment. Instead of forcing a specific order for associating their observations, we allow the robots compute its data association with each of its neighbors in the graph. Although this scenario is more flexible, it may lead to inconsistent global data associations in the presence of cycles in the communication graph. These inconsistencies are detected when chains of local associations give rise to two features from one robot being associated among them. These situations must be correctly identified and solved before merging the data. Otherwise, the merging process would be wrong and could not be undone. In this chapter, we approach a distributed data association, under limited communications. Instead of comparing any 2 local observations among them, only the local observations of neighboring robots can be compared. Besides, there is no central node that has knowledge of all the local associations and each robot exclusively knows the associations computed by itself. Then, each

robot updates its local information by communicating with its neighbors. We present an algorithm where, finally, each robot is capable of detecting and solving any inconsistent association that involves any of its features.

5.2 Problem Description

We consider a robotic team composed of $n \in \mathbb{N}$ robots. The n robots have communication capabilities to exchange information with the other robots. However, these communications are limited. Let $\mathcal{G}_{com} = (\mathcal{V}_{com}, \mathcal{E}_{com})$ be the undirected communication graph. The nodes are the robots, $\mathcal{V}_{com} = \{1, \ldots, n\}$. If two robots i, j can exchange information then there is an edge between them, $(i, j) \in \mathcal{E}_{com}$. Let $\mathcal{N}_i$ be the set of neighbors of robot i,

$$\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}_{com}\}.$$

Each robot i has observed a set $\mathcal{S}_i$ of m_i features,

$$\mathcal{S}_i = \{f_1^i, \ldots, f_{m_i}^i\}.$$

It can compute the data association between its own set $\mathcal{S}_i$, and the sets of its neighbors $\mathcal{S}_j$, with $j \in \mathcal{N}_i$. However, these data associations are not perfect. There may appear inconsistent data associations relating different features from the same set $\mathcal{S}_i$ (Fig. 5.1). If the robots merge their data as soon as they solve the local data association, inconsistent associations cannot be managed since the merging cannot be undone. The goal of our algorithm is to detect and resolve these inconsistent associations before executing the merging.

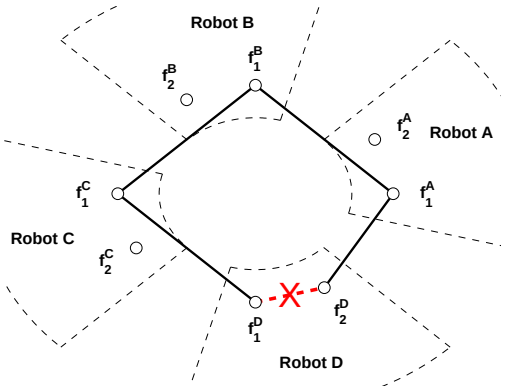


Figure 5.1: Robots A, B, C and D associate their features comparing their maps in a two-by-two way. Robot A associates its feature f_1^A with f_2^D and with f_1^B; robot B associates f_1^B with f_1^C; robot C associates f_1^C with f_1^D (solid lines). As a result, there is a path (dashed line) between f_1^D and f_2^D. This is an inconsistent situation. Finding this path would require the knowledge of the whole association graph.

In order to make the reading easy, along the chapter we use the indices i, j and k to refer to robots and indices r, r', s, s', to refer to features. The r^{th} feature observed by the

i^{th} robot is denoted as f_r^i. Given a matrix A, the notations $A_{r,s}$ and $[A]_{r,s}$ correspond to the (r,s) entry of the matrix, whereas A_{ij} denotes the (i,j) block when the matrix is defined by blocks. We let $\mathbf{I}_k$ be the $k \times k$ identity matrix, and $\mathbf{0}_{k_1 \times k_2}$ a $k_1 \times k_2$ matrix with all entries equal to zero.

5.2.1 Matching between two cameras

Let F be a function that computes the data association between any two sets of features, $\mathcal{S}_i$ and $\mathcal{S}_j$, and returns an association matrix $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$ where

$$[\mathbf{A}_{ij}]_{r,s} = \begin{cases} 1 & \text{if } f_r^i \text{ and } f_s^j \text{ are associated,} \\ 0 & \text{otherwise,} \end{cases}$$

for $r = 1, \ldots, m_i$ and $s = 1, \ldots, m_j$. We assume that F satisfies the following conditions.

Assumption 5.2.1 (Self Association)**.** *When F is applied to the same set $\mathcal{S}_i$, it returns the identity, $F(\mathcal{S}_i, \mathcal{S}_i) = \boldsymbol{A}_{ii} = \mathbf{I}$.*

Assumption 5.2.2 (Unique Association)**.** *The returned association $\boldsymbol{A}_{ij}$ has the property that the features are associated in a one-to-one way,*

$$\sum_{r=1}^{m_i} [\boldsymbol{A}_{ij}]_{r,s} \leq 1 \text{ and } \sum_{s=1}^{m_j} [\boldsymbol{A}_{ij}]_{r,s} \leq 1,$$

for all $r = 1, \ldots, m_i$ and $s = 1, \ldots, m_j$.

Assumption 5.2.3 (Symmetric Association)**.** *Robots i and j associate their features in the same way. Given two sets $\mathcal{S}_i$ and $\mathcal{S}_j$ it holds that $F(\mathcal{S}_i, \mathcal{S}_j) = \boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}^T = (F(\mathcal{S}_j, \mathcal{S}_i))^T$.*

Additionally, the local matching function may give information of the quality of each associations. The management of this information is discussed in Section 5.7.

We do not make any assumptions about the sets of features used by the cameras. However, we point out that the better the initial matching is, the better the global matching will be. Examples of features and matching functions that can be used in our method are: Lines or invariant descriptors matched with epipolar or homography constraints [66]; 3D points computed using mapping techniques matched with the Joint Compatibility Branch and Bound (JCBB) [97]; Image templates of people, faces, objects, etc. matched with sums of absolute differences of the pixels or correlation methods [104].

5.2.2 Centralized matching between n cameras

Let us consider now the situation in which there are n cameras and a central unit with the n sets of features available. In this case F can be applied to all the pairs of sets of features, $\mathcal{S}_i$, $\mathcal{S}_j$, for $i, j \in \{1, \ldots, n\}$. The results of all the associations can be represented by an undirected graph $\mathcal{G}_{cen} = (\mathcal{F}_{cen}, \mathcal{E}_{cen})$. Each node in $\mathcal{F}_{cen}$ is a feature f_r^i, for $i = 1, \ldots, n$, $r = 1, \ldots, m_i$. There is an edge between two features f_r^i, f_s^j iff $[\mathbf{A}_{ij}]_{r,s} = 1$.

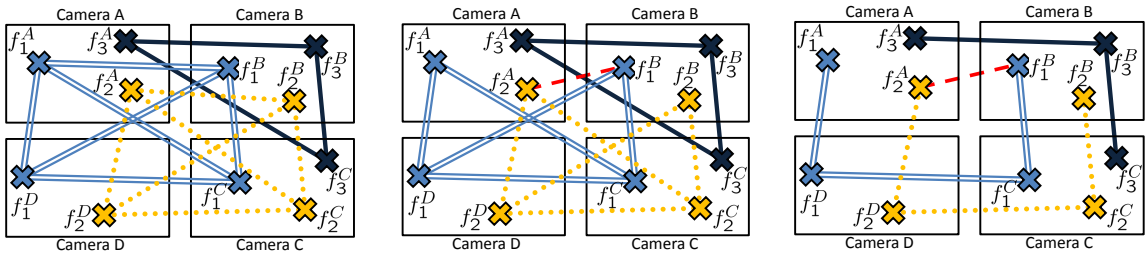


Figure 5.2: Different association graphs. (a) Centralized matching with perfect association function. The graph is formed by disjoint cliques. (b) Centralized matching with imperfect association. Some links are missed, (f_1^A, f_1^B) and (f_2^A, f_2^B), and spurious links appear, (f_2^A, f_1^B). As a consequence, a subset of the features form a *conflictive set*. (c) Matching with limited communications. Now, the links between A and C, and B and D, cannot be computed because they are not neighbors in $\mathcal{G}_{com}$. Moreover, the information available to each camera is just the one provided by its neighbors.

For a perfect matching function, the graph $\mathcal{G}_{cen}$ exclusively contains disjoint cliques, identifying features observed by multiple cameras (Fig. 5.2 (a)). However, in real situations, the matching function will miss some matches and will consider as good correspondences some spurious matches (Fig. 5.2 (b)). As a consequence, inconsistent associations relating different features from the same set $\mathcal{S}_i$ may appear.

Definition 5.2.4. *An* association set *is a set of features such that they form a connected component in* $\mathcal{G}_{cen}$. *Such set is a* conflictive set *or an* inconsistent association *if there exists a path in* $\mathcal{G}_{cen}$ *between two or more features observed by the same camera. A feature is* inconsistent *or* conflictive *if it belongs to an inconsistent association.*

Centralized solutions to overcome this problem are found in [54], [13]. The latter one is also well suited for a distributed implementation but yet requires that any pair of images can be matched. In camera networks this implies global communications, which is not always possible.

5.2.3 Matching between n cameras with limited communications

Let us consider now that there is no central unit with all the information and there are n robots, each one with a camera and a process unit with limited communication capabilities. The robots are scattered forming a network with communications described with the undirected communication graph $\mathcal{G}_{com} = (\mathcal{V}_{com}, \mathcal{E}_{com})$ introduced at the beginning of this section.

In this case, due to communication restrictions, local matches can only be found within direct neighbors. As a consequence, the matching graph computed in this situation will be a subgraph of the centralized one, $\mathcal{G}_{dis} = (\mathcal{F}_{dis}, \mathcal{E}_{dis}) \subseteq \mathcal{G}_{cen}$, (Fig. 5.2 (c)). It has the same set of nodes, $\mathcal{F}_{dis} = \mathcal{F}_{cen}$, but it has an edge between two features f_r^i, f_s^j only if the edge exists in $\mathcal{G}_{cen}$ and the robots i and j are neighbors in the communication graph,

$$\mathcal{E}_{dis} = \{(f_r^i, f_s^j) \mid (f_r^i, f_s^j) \in \mathcal{E}_{cen} \wedge (i,j) \in \mathcal{E}_{com}\}.$$

Along this chapter, we name m_{sum} the number of features, $|\mathcal{F}_{dis}| = \sum_{i=1}^{n} m_i = m_{sum}$. We name d_f the diameter of $\mathcal{G}_{dis}$, the length of the longest path between any two nodes in $\mathcal{G}_{dis}$, and we name d_v the diameter of the communication graph, $\mathcal{G}_{com}$. The diameters satisfy $d_f \leq m_{sum}$ and $d_v \leq n$. We name $\mathbf{A} \in \mathbb{N}^{m_{sum} \times m_{sum}}$ the adjacency matrix of $\mathcal{G}_{dis}$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \dots & \mathbf{A}_{nn} \end{bmatrix}, \tag{5.1}$$

where

$$\mathbf{A}_{ij} = \begin{cases} F(\mathcal{S}_i, \mathcal{S}_j) & \text{if } j \in \{\mathcal{N}_i \cup i\}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{5.2}$$

Let us note that in this case none of the robots has the information of the whole matrix. Robot i has only available the sub-matrix corresponding to its own local matches $\mathbf{A}_{ij}, j = 1, \dots, n$. Under these circumstances the problem is formulated as follows: Given a network with communications defined by a graph, $\mathcal{G}_{com}$, and an association matrix $\mathbf{A}$ scattered over the network find the global matches and the possible inconsistencies in a decentralized way. In case there are conflicts, find alternative associations free of them.

5.3 Propagation of Local Associations and Detection of Inconsistencies

Considering definition 5.2.4 we observe that in order to detect an inconsistent association it is required to compute the paths that exist among the elements in $\mathcal{G}_{dis}$. As the following lemma states [27], given a graph $\mathcal{G}_{dis}$, the powers of its adjacency matrix contains the information about the number of paths existing between the nodes of $\mathcal{G}_{dis}$:

Lemma 5.3.1 (Lemma 1.32 [27])**.** *Let $\mathcal{G}_{dis}$ be a weighted graph of order $|\mathcal{V}|$ with unweighted adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, and possibly with self loops. For all $i, j \in \{1, \dots, |\mathcal{V}|\}$ and $t \in \mathbb{N}$ the (i,j) entry of the t^{th} power of $\boldsymbol{A}$, $\boldsymbol{A}^t$, equals the number of paths of length t (including paths with self loops) from node i to node j.*

The computation of the powers of $\mathbf{A}$ requires, a priori, the information about the whole matrix. We show now that this computation can also be done in a decentralized manner. Let each robot $i \in \mathcal{V}_{com}$ maintain the blocks within $\mathbf{A}^t$ associated to its own features, $X_{ij}(t) \in \mathbb{N}^{m_i \times m_j}$, $j = 1, \dots, n$, $t \geq 0$,, which are initialized as

$$X_{ij}(0) = \begin{cases} \mathbf{I}, & j = i, \\ \mathbf{0}, & j \neq i, \end{cases} \tag{5.3}$$

and are updated, at each time step, with the following algorithm

$$X_{ij}(t+1) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} X_{kj}(t), \tag{5.4}$$

with $\mathbf{A}_{ik}$ as defined in (5.2). It is observed that the algorithm is fully distributed because the robots only use information about its direct neighbors in the communication graph.

Theorem 5.3.2. *Let $[\boldsymbol{A}^t]_{ij} \in \mathbb{N}^{m_i \times m_j}$ be the block within $\boldsymbol{A}^t$ related to the associations between robot i and robot j. The matrices $X_{ij}(t)$ computed by each robot i using the decentralized algorithm (5.4) are exactly the sub-matrices $[\boldsymbol{A}^t]_{ij}$,*

$$X_{ij}(t) = [\boldsymbol{A}^t]_{ij}, \tag{5.5}$$

for all $i, j \in \{1, \ldots, n\}$ and all $t \in \mathbb{N}$.

Proof. The proof is done using induction. First we show that eq. (5.5) is satisfied for $t = 0$. In this case we have that $\mathbf{A}^0 = \mathbf{I}$, thus for all $i, j \in \{1, \ldots, n\}$, $[\mathbf{A}^0]_{ii} = \mathbf{I}$ and $[\mathbf{A}^0]_{ij} = \mathbf{0}$, which is exactly the initial value of the variables X_{ij} (eq. (5.3)).

Now we have that for any $t > 0$,

$$[\mathbf{A}^t]_{ij} = \sum_{k=1}^{n} \mathbf{A}_{ik}[\mathbf{A}^{t-1}]_{kj} = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik}[\mathbf{A}^{t-1}]_{kj},$$

because $\mathbf{A}_{ik} = \mathbf{0}$ for $k \notin \{\mathcal{N}_i \cup i\}$. Assuming that for all $i, j \in \{1, \ldots, n\}$ and a given $t > 0$, $X_{ij}(t-1) = [\mathbf{A}^{t-1}]_{ij}$ is true, then

$$X_{ij}(t) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} X_{kj}(t-1) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik}[\mathbf{A}^{t-1}]_{kj} = [\mathbf{A}^t]_{ij}.$$

Then, by induction, $X_{ij}(t) = [\mathbf{A}^t]_{ij}$ is true for all $t > 0$. □

Corollary 5.3.3. *The variables $X_{ij}(t)$ contain the information about all the paths of length t between features observed by robots i and j.*

Proof. By direct application of Lemma 5.3.1. □

Analyzing the previous algorithm the first issue to deal with is how to simplify the computation of the matrices in order to avoid high powers of $\mathbf{A}$. In the case we are studying it is just required to know if there is a path between two elements in $\mathcal{G}_{dis}$ and not how many paths are. This means that in this situation it is enough that $[X_{ij}(t)]_{r,s} > 0$ in order to know that features f_r^i and f_s^j are connected by a path. Another issue is to decide when the algorithm in (5.4) must stop. Since the maximum length of a path between any two nodes in a graph is its diameter, then after d_f iterations the algorithm should stop. However, in general situations the robots will not know neither d_f nor m_{sum}, which makes this decision hard to be made a priori.

Definition 5.3.4. *We will say that two matrices $\boldsymbol{A}$ and $\bar{\boldsymbol{A}}$ of the same dimensions are equivalent, $\boldsymbol{A} \sim \bar{\boldsymbol{A}}$, if for all r and s it holds*

$$[\boldsymbol{A}]_{r,s} > 0 \Leftrightarrow [\bar{\boldsymbol{A}}]_{r,s} > 0 \text{ and } [\boldsymbol{A}]_{r,s} = 0 \Leftrightarrow [\bar{\boldsymbol{A}}]_{r,s} = 0.$$

In practice any equivalent matrix to the $X_{ij}(t)$ will provide the required information, which allows us to simplify the computations simply by changing any positive value in the matrices by 1. Moreover, the equivalency is also used to find a criterion to stop the algorithm:

Proposition 5.3.5. *For a robot i, let t_i be the first time instant, t, such that $X_{ij}(t) \sim X_{ij}(t-1)$ for all $j = 1, \ldots, n$. Then robot i can stop to execute the algorithm at time t_i.*

Proof. Let $\bar{X}_{ij}(t)$ be the components in $X_{ij}(t)$, such that $[X_{ij}(t-1)]_{r,s} = 0$ and $[X_{ij}(t)]_{r,s} > 0$. The cardinal, $|\bar{X}_{ij}(t)|$, represents the number of features $f_s^j \in \mathcal{S}_j$ such that the minimum path length in $\mathcal{G}_{dis}$ between them and one feature $f_r^i \in \mathcal{S}_i$ is t. At time t_i, $X_{ij}(t_i) \sim X_{ij}(t_i - 1)$ $\forall j$ for the first time, and then $\sum_{j=1}^{n} |\bar{X}_{ij}(t_i)| = 0$ because no component has changed its value from zero to a positive. This means that there is no path of minimum distance t_i linking any feature f_r^i with any other feature in $\mathcal{G}_{dis}$. By the physical properties of a path, it is obvious that if there are no features at minimum distance t_i, it will be impossible that a feature is at minimum distance $t_i + 1$ and all the paths that connect features of robot i with any other feature have been found. □

Corollary 5.3.6. *All the robots end the execution of the iteration rule* (5.4) *in at most in $d_f + 1$ iterations.*

Proof. Recalling that the maximum distance between two nodes in $\mathcal{G}_{dis}$ is the diameter of the graph, denoted by d_f, then $\sum_{j=1}^{n} |\bar{X}_{ij}(d_f + 1)| = 0$ for all $i = 1, \ldots, n$. □

If a robot j at time t does not receive the information $X_{ij}(t)$ from robot i then it will use the last matrix received, because robot i has already finished computing its paths and $X_{ij}(t) \sim X_{ij}(t-1)$.

It remains to analyze which features are conflictive and which are not. Each robot has the information about all the association paths of its features and the features of the rest of the robots in the network in the different variables $X_{ij}(t_i)$. The robots detect all the conflictive features using two simple rules. A feature f_r^i is conflictive if and only if one of the following conditions are satisfied:

(i) There exists other feature $f_{r'}^i$, with $r \neq r'$, such that

$$[X_{ii}(t_i)]_{r,r'} > 0; \tag{5.6}$$

(ii) There exist features f_s^j and $f_{s'}^j$, $s \neq s'$, such that

$$[X_{ij}(t_i)]_{r,s} > 0 \text{ and } [X_{ij}(t_i)]_{r,s'} > 0. \tag{5.7}$$

In conclusion, the proposed algorithm will be able to find all the inconsistencies in a finite number of iterations. The algorithm is decentralized and it is based only on local interactions between the robots. Each robot only needs to know its local data associations. It updates its information based on the data exchanged with its neighbors. When the algorithm finishes, each robot i can extract from its own matrices $X_{ij}(t_i)$ all the information of any conflict that involves any of its features. If the robot has any conflictive feature, it also knows the rest of features that belong to the conflictive set independently of the robot that observed such features.

5.4 Improved Detection Algorithm

The previous detection method has several drawbacks. The powers of $\mathbf{A}$ may contain large values. However, in practice we do not require to compute all the paths of length t between the features. In this case it is just required to know if there is a path between two elements in $\mathcal{G}_{dis}$. Moreover, the method does not exploit the local information of features belonging to the same robot. In this section we propose a new algorithm that overcomes these limitations reducing the complexity of the operations, the number of required steps to execute it and the amount of transmitted information.

Let $\mathbf{y}_r^i(0) = \{[\mathbf{A}_{i1}]_{r,1}, \ldots, [\mathbf{A}_{in}]_{r,m_n}\} \in \{0,1\}^{m_{sum}}$ be the row associated with feature f_r^i and $[\mathbf{y}_r^i(0)]_u, u = 1, \ldots, m_{sum}$, the u^{th} component in the row. The new algorithm to compute the paths between features is:

Algorithm 5.4.1 Inconsistency Detection - Robot i

Ensure: All the inconsistencies are found

1: **repeat**
2: Send $\mathbf{y}_r^i(t)$, $r \in \mathcal{S}_i$ to all $j \in \mathcal{N}_i$
3: Receive all $\mathbf{y}_s^j(t)$, $j \in \mathcal{N}_i$, $s \in \mathcal{S}_j$
4: **for all** $r \in \mathcal{S}_i$ **do**
5: **for all** $j \in \mathcal{N}_i, s \in \mathcal{S}_j \mid [\mathbf{A}_{ij}]_{r,s} = 1$ **do**
6: $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t) \vee \mathbf{y}_s^j(t)$
7: **end for**
8: **for all** $r' \in \mathcal{S}_i$ satisfying that $\exists\ u \in \{1, \ldots, m_{sum}\}$ such that $[\mathbf{y}_r^i(t)]_u = [\mathbf{y}_{r'}^i(t)]_u = 1$ **do**
9: $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t) \vee \mathbf{y}_{r'}^i(t)$
10: **end for**
11: **end for**
12: **until** $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t), \forall r \in \mathcal{S}_i$

The algorithm computes the logical "*or*" operation of rows of neighbor robots and common matches. Lines 5-7 are equivalent to compute the powers of $\mathbf{A}$ using logical values instead of integers. The second part of the update, lines 8-10, speeds up the process by also considering that when two or more of features observed by the same robot share a common third feature observed by a different robot, then eventually they will be associated with each other.

The new algorithm reduces the complexity of the operations with respect to [10] by replacing the products of matrices by logical operations between rows. This reduction allows to avoid the large numbers that may appear when computing high powers of the adjacency matrix and also allows us to reduce the amount of transmitted data.

Proposition 5.4.1. *The amount of information exchanged by the network during the whole execution of Algorithm 5.4.1 can be upper bounded by* $2m_{sum}^2$.

Proof. By the definition of the operations in lines 6 and 9 of Algorithm 5.4.1 we can see that the components of $\mathbf{y}_r^i$ change their value at most once during the whole execution, for

all $i \in \mathcal{V}_{com}, r \in \mathcal{S}_i$. This means that it is not necessary for the robots to send the whole blocks $\mathbf{y}_r^i$ to their neighbors but just the indices of the components that have changed their value from *false* to *true*. Each element can be identified by two data, the row and the column and there are a total of m_{sum}^2 elements, the size of $\mathbf{A}$. Therefore, in the worst case, the amount of transmitted information through the network during the whole execution of the algorithm now is $2m_{sum}^2$. □

Regarding the correctness of the algorithm we have the following result:

Proposition 5.4.2. *After execution of Algorithm 5.4.1 all the paths between features have been found and they are available to all the robots with features involved in them.*

Proof. Let $y_r^i(t)$ be the number of components in $\mathbf{y}_r^i(t)$, such that $[\mathbf{y}_r^i(t-1)]_u = 0$ and $[\mathbf{y}_r^i(t)]_u = 1, u = 1, \ldots, m_{sum}$. This number represents the number of new paths found in $\mathcal{G}_{dis}$ at time instant t that include the features in $\mathcal{S}_i$. These new paths come either from the execution of line 6, or the execution of line 9.

Let t_i be the first time instant such that $\mathbf{y}_r^i(t_i) = \mathbf{y}_r^i(t_i - 1) \ \forall r$ and $y_r^i(t_i) = 0$ because no component has changed its value from zero to one for any of the features. This means that, for any feature in $\mathcal{S}_i$, there are no new paths with other features. By the physical properties of a path, it is obvious that if there are no new features at minimum distance t_i, it will be impossible that a new feature is at minimum distance $t_i + 1$. In addition, if no new paths at distance $t_i + 1$ can be found, line 9 of Algorithm 5.4.1 will not find new paths either. At this point the condition of line 12 is true and the algorithm ends. Since the solution of the algorithm is equivalent to the computation of the powers of the adjacency matrix, because of line 6, this also means that all the paths that connect features of robot i with any other feature have been found. □

When one robot j at time t does not receive the information $\mathbf{y}_r^i(t), r = 1, \ldots, m_i$ from robot i then it will use the last information it had, because it means that robot i has not found new paths and $\mathbf{y}_r^i(t) = \mathbf{y}_r^i(t-1)$.

Finally let us analyze the number of iterations that the algorithm requires to finish:

Theorem 5.4.3. *All the robots end the execution of the Algorithm 5.4.1 in at most* $\min(d_f, 2n)$ *iterations.*

Proof. We already know that the algorithm finishes in at most d_f iterations. In the case that the matching does not contain any inconsistency $d_f \leq 2n$ and the result is valid.

Now let us suppose that there is one inconsistency. This implies that the communication graph, $\mathcal{G}_{com}$, contains one cycle of arbitrary length, ℓ. We divide the number of iterations in three parts. First $n - \ell$ iterations are required to ensure that the information of all the features belonging to the robots outside the cycle reaches at least one robot in the cycle.

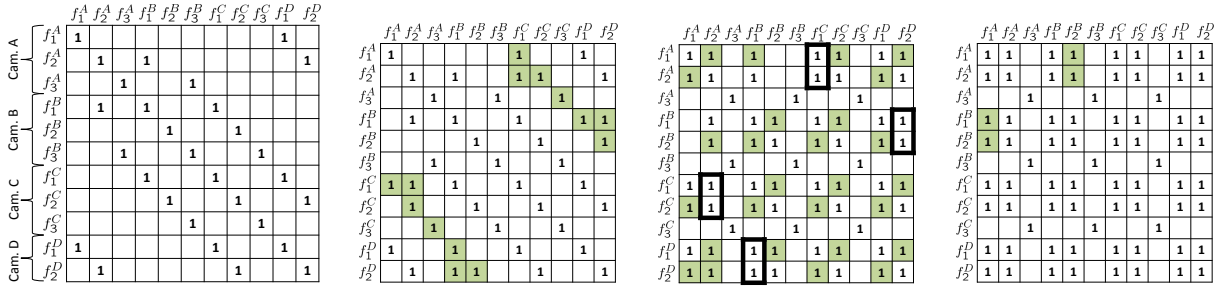
The second part requires $\frac{3}{4}\ell + 1$ iterations. In the worst case, the diameter of the subgraph defined by the cycle is $\ell/2$ and only $\ell + 1$ features in the cycle form the inconsistency, which means that only one robot will execute, at some point, lines 8-10 of Algorithm 5.4.1. It is clear that after $\ell/2 + 1$ iterations there will be at least two robots in the cycle, at maximum distance from each other $(\ell/2)$, with all the information. One

of the robots, the one with the inconsistency, will obtain the information from the execution of 8-10 in Algorithm 5.4.1. The other robot is the one with the common feature, u, detected in lines 8-10 of the algorithm by the first one. After this point $\ell/4$ iterations are required to share this information with the rest of the robots in the cycle and we can ensure that all the robots in the cycle have all the information about the inconsistency. If there are more than $\ell+1$ features inside the cycle forming the inconsistency the result is still valid.

With all the robots in the cycle knowing all the features that form the inconsistency, the number of additional iterations required to transmit the information to the rest of the network is upper bounded again by $n-\ell$. If we sum all the iterations we obtain $2n-\frac{5}{4}\ell+1$. Since the minimum length of a cycle is 3 the above quantity is always lower than $2n$. □

Let us remark that this bound is conservative because it does not take into account that during the initial $n-\ell$ and the final $n-\ell$ iterations the cycle is also exchanging information.

5.4.1 Example of execution



(a) (b) (c) (d)

Figure 5.3: Example of execution of the decentralized algorithm for detection of inconsistencies applied to Fig. 5.2 (c). A detailed explanation can be found in section 3.2.

Figure 5.3 shows an example of how the algorithm is applied. The example shows the execution of the algorithm for the associations shown in Fig. 5.2 (c). Each robot has only the information about the rows corresponding to the features they have observed. In Fig. 5.3 (a) the matrix with the local matches found by all the cameras can be seen. The zeros have been omitted in the figure for a better representation. For simplicity here we will only explain the process for the robot A. After the first round of communications and the execution of lines 5-7 of Algorithm 5.4.1 the rows have the form of Fig. 5.3 (b). The components with green background are the new paths found by the algorithm. For the case of the camera A, the first feature, f_1^A, is matched with the first feature of robot D, which is a direct neighbor of A, thus, $\mathbf{y}_1^A(2) = \mathbf{y}_1^A(1) \vee \mathbf{y}_1^D(1)$. The second feature is matched with f_1^B and f_2^D so $\mathbf{y}_2^A(2) = \mathbf{y}_2^A(1) \vee \mathbf{y}_1^B(1) \vee \mathbf{y}_2^D(1)$. Finally $\mathbf{y}_3^A(2) = \mathbf{y}_3^A(1) \vee \mathbf{y}_3^B(1)$. After that robot A detects that f_1^A and f_2^A share a common match with f_1^C. Therefore it executes lines 8-10 of the algorithm with these two features, as shown in Fig. 5.3 (c). Now the process is repeated, obtaining the matrix in Fig. 5.3 (d). The algorithm has

found all associations using only $3 < d_f = 7$ iterations. At this point the robot A knows that f_1^A and f_2^A belong to one inconsistent association with features $f_1^B, f_2^B, f_1^C, f_2^C, f_1^D$ and f_2^D.

5.5 Resolution Algorithm based on Trees

The resolution of inconsistent associations consists of deleting edges from $\mathcal{G}_{dis}$ so that the resulting graph is conflict-free.

Definition 5.5.1. *Let C denote the number of conflictive sets in $\mathcal{G}_{dis}$. We say a conflictive set $\mathcal{C}$ is* detectable *by a robot i if there exists a $r \in \{1, \dots, m_i\}$ such that $f_r^i \in C$. The set of robots that detect a conflictive set $\mathcal{C}$ is $R \subseteq \mathcal{V}_{com}$. The number of features from each robot $i \in R$ involved in $\mathcal{C}$ is $\tilde{m}_i$. We say $\mathcal{G}_{dis}$ is* conflict-free *if $C = 0$.*

All the edges whose deletion transforms $\mathcal{G}_{dis}$ into a conflict-free graph, belong to any of the C conflictive sets of $\mathcal{G}_{dis}$. Since the conflictive sets are disjoint, they can be considered separately. From now on, we focus on the resolution of one of the conflictive sets $\mathcal{C}$. The other conflictive sets are managed in the same way. The resolution problem consists of partitioning $\mathcal{C}$ into a set of disjoint conflict-free components $\mathcal{C}_q$ such that

$$\underset{q}{\cup} \mathcal{C}_q = \mathcal{C}, \text{ and } \mathcal{C}_q \cap \mathcal{C}_{q'} = \emptyset,$$

for all $q, q' = 1, 2, \dots$. The number of such conflict-free components is a priori unknown and it will be discussed later in this section.

Obtaining an optimal partition that minimizes the number of deleted edges is complicated. If there were only two inconsistent features f_r^i, $f_{r'}^i$, it could be approached as a max-flow min-cut problem [105]. However, in general there will be more inconsistent features, $\tilde{m}_i \geq 2$, within $\mathcal{C}$ associated to a robot $i \in R$. Besides, there may also be $\tilde{m}_j \geq 2$ inconsistent features belonging to a different robot $j \in R$. The application of [105] separately to any pair of inconsistent features does not necessarily produce an optimal partition. It may happen that a single edge deletion simultaneously resolves more than one inconsistent association. Therefore, an optimal solution should consider multiple combinations of edge deletions, what makes the problem computationally intractable, and imposes a centralized scheme. We propose a resolution algorithm that is not optimal but is efficient and is proven to be correct. Besides, it allows a decentralized computation.

Proposition 5.5.2. *Let R be the set of robots that detect $\mathcal{C}$. Let $i_\star$ be the robot with the most features involved in $\mathcal{C}$,*

$$i_\star = \arg\max_{i \in R} \tilde{m}_i. \tag{5.8}$$

The number of conflict-free components in which $\mathcal{C}$ can be decomposed is lower bounded by $\tilde{m}_{i_\star}$.

Proof. Each conflict-free component can contain, at most, one feature from a robot $i \in R$. Then there must be, at least, $\max_{i \in R} \tilde{m}_i = \tilde{m}_{i_\star}$ components. □

Algorithm 5.5.1 Spanning Trees - Robot i

1: – *Initialization*
2: **for** each conflictive set $\mathcal{C}$ for which i is root ($i = i_\star$) **do**
3: create $\tilde{m}_{i_\star}$ components
4: assign each inconsistent feature $f_r^{i_\star} \in \mathcal{C}$ to a different component $\mathcal{C}_q$
5: send component request to all its neighboring features
6: **end for**
7:
8: – *Algorithm*
9: **for** each component request from f_s^j to f_r^i **do**
10: **if** (b) or (c) **then**
11: $[\mathbf{A}_{ij}]_{r,s} = 0$
12: send reject message to j
13: **else if** (d) **then**
14: assign f_r^i to the component
15: send component request to all its neighboring features
16: **end if**
17: **end for**
18: **for** each component reject from f_s^j to f_r^i **do**
19: $[\mathbf{A}_{ij}]_{r,s} = 0$
20: **end for**

The resolution algorithm constructs $\tilde{m}_{i_\star}$ conflict-free components using a strategy close to a BFS tree construction. Initially, each robot i detects the conflictive sets for which it is the root using its local information $X_{i1}(t_i), \ldots, X_{in}(t_i)$. The root robot for a conflictive set is the one with the most inconsistent features involved. In case two robots have the same number of inconsistent features, the one with the lowest robot id is selected. Then, each robot executes the resolution algorithm (Algorithm 5.5.1).

The root robot creates $\tilde{m}_{i_\star}$ components and initializes each component $\mathcal{C}_q$ with one of its features $f^{i_\star} \in \mathcal{C}$. Then, it tries to add to each component $\mathcal{C}_q$ the features directly associated to $f^{i_\star} \in \mathcal{C}_q$. Let us consider that f_s^j has been assigned to $\mathcal{C}_q$. For all f_r^i such that $[\mathbf{A}_{ij}]_{r,s} = 1$, robot j sends a component request message to robot i. When robot i receives it, it may happen that

(a) f_r^i is already assigned to $\mathcal{C}_q$;

(b) f_r^i is assigned to a different component;

(c) other feature $f_{r'}^i$ is already assigned to $\mathcal{C}_q$;

(d) f_r^i is unassigned and no feature in i is assigned to $\mathcal{C}_q$.

In case (a), f_r^i already belongs to the component $\mathcal{C}_q$ and robot i does nothing. In cases (b) and (c), f_r^i cannot be added to $\mathcal{C}_q$; robot i deletes the edge $[\mathbf{A}_{ij}]_{r,s}$ and replies with a reject message to robot j; when j receives the reject message, it deletes the equivalent

edge $[\mathbf{A}_{ji}]_{s,r}$. In case (d), robot i assigns its feature f_r^i to the component $\mathcal{C}_q$ and the process is repeated.

Theorem 5.5.3. *Let us consider that each robot $i \in \mathcal{V}_{com}$ executes the decentralized resolution algorithm (Algorithm 5.5.1) on $\mathcal{G}_{dis}$, obtaining $\mathcal{G}'_{dis}$,*

(i) after $t = n$ iterations no new features are added to any component $\mathcal{C}_q$ and the algorithm finishes;

(ii) each obtained $\mathcal{C}_q$ is a connected component in $\mathcal{G}'_{dis}$;

(iii) $\mathcal{C}_q$ is conflict free;

(iv) $\mathcal{C}_q$ contains at least two features;

for all $q \in \{1, \ldots, \tilde{m}_{i_\star}\}$ and all conflictive sets.

Proof. (i) The maximal depth of a conflict-free component is n since, if there were more features, at least two of them would belong to the same robot. Then, after at most n iterations of this algorithm, no more features are added to any component $\mathcal{C}_q$ and the algorithm finishes.

(ii) There is a path in $\mathcal{G}_{dis}$ between any two features belonging to a conflictive set $\mathcal{C}$. Therefore, there is also a path in $\mathcal{G}_{dis}$ between any two features assigned to the same component $\mathcal{C}_q$. Since the algorithm does not delete edges from $\mathcal{G}_{dis}$ within a component (case (a)), then $\mathcal{C}_q$ it is also connected in $\mathcal{G}'_{dis}$. Since none feature can be assigned to more than one component (case (b)), the components are disjoint. Therefore, $\mathcal{C}_q$ is a connected component in $\mathcal{G}'_{dis}$.

(iii) By construction, two features from the same robot are never assigned to the same component $\mathcal{C}_q$ (case (c)). Therefore, each component is conflict-free.

(iv) Each conflictive set has more than one feature. Because of Assumptions 5.2.1 and 5.2.2, each feature and its neighbors are conflict free. Therefore, each component $\mathcal{C}_q$ contains, at least, its originating feature, and a neighboring feature. Thus, it has at least two features. □

Corollary 5.5.4. *After executing Algorithm 5.5.1, the size of each conflict set $\mathcal{C}$ is reduced by at least 2 $\tilde{m}_{i_\star}$, where $\tilde{m}_{i_\star} \geq 2$.*

When the algorithm finishes, each original conflictive set $\mathcal{C}$ has been partitioned into $\tilde{m}_{i_\star}$ conflict-free components. It may happen that a subset of features remains unassigned. These features may still be conflictive in $\mathcal{G}'_{dis}$. The detection algorithm can be executed on the subgraph defined by this smaller subset of features.

Proposition 5.5.5. *Consider each robot i iteratively executes the detection (Section 5.4) and the resolution (Section 5.5) algorithms. Then, in a finite number of iterations, all conflictive sets disappear.*

Proof. After each execution of the resolution algorithm, the size of each conflict set $\mathcal{C}$ is reduced by, at least, 2 $\tilde{m}_{i_\star} \geq 4$ (Corollary 5.5.4). Then, in a finite number of iterations, it happens that $|\mathcal{C}| < 4$. A set with 3 features $f_r^i, f_{r'}^i, f_s^j$ cannot be conflictive; this would require the existence of edges (f_r^i, f_s^j) and $(f_{r'}^i, f_s^j)$, what is impossible (Assumption 5.2.2). A set with 2 features cannot be conflictive (Assumptions 5.2.1 and 5.2.2), and a set with a single feature cannot be inconsistent by definition. Therefore, there will be no remaining inconsistencies or conflictive sets. □

The main interest of the presented resolution algorithm, is that it is fully decentralized and it works on local information. Each robot uses its own $X_{ij}(t_i)$ for detecting the root robot of each conflictive set. During the resolution algorithm, the decisions and actions taken by each robot are based on its local associations $\mathbf{A}_{ij}$, and the components assigned to its local features. Moreover, each robot is responsible of deleting the edges from its local association matrices $\mathbf{A}_{ij}$, with $j \in \{1, \ldots, n\}$. In addition, the presented algorithm works in finite time. Let us note that although we presented the algorithm for a single conflictive set, all conflictive sets are managed in parallel.

5.6 Feature Labeling

Simultaneously to the data association process, the robots assign labels to their features. After checking feature f_r^i is consistent, robot i assigns it a label $L_r^i = (i_\star, r_\star) \in \mathbb{N}^2$ composed of a robot identifier $i_\star$ and a feature index $r_\star$ as follows. Assume f_r^i and features $f_s^j, f_{s'}^{j'}, \ldots$ form a consistent association set in $\mathcal{G}_{dis}$ and thus they are observations of a common landmark in the environment taken by robots $i, j, j', \ldots$. Among all the candidates $(i, r), (j, s), (j', s'), \ldots$, a unique label $(i_\star, r_\star)$ is selected by the robots, e.g., the one with the lowest robot id. Then, robot i assigns this label to f_r^i, $L_r^i = (i_\star, r_\star)$; the other robots $j, j', \ldots$, proceed in a similar way so that finally,

$$L_r^i = L_s^j = L_{s'}^{j'} = \cdots = (i_\star, r_\star)\,.$$

We say a feature f_r^i is *exclusive* if it is isolated in $\mathcal{G}_{dis}$, corresponding to a landmark observed by a single robot i; in this case, its label L_r^i is simply (i, r). Otherwise, we say f_r^i is non-exclusive and it may either be *consistent* or conflictive. Consistent features are labeled as explained above, whereas robots wait until conflicts are *resolved* for labeling its conflictive features. The data association and labeling process finishes with an association graph $\mathcal{G}_{dis}$ free of any inconsistent association and with all the features labeled. When the algorithm finishes, two features f_r^i, f_s^j have the same label, $L_r^i = L_s^j$, iff they are connected by a path in the resulting conflict-free $\mathcal{G}_{dis}$. The decentralized data association and labeling algorithm is summarized in Algorithm 5.6.1. This strategy makes use of two subroutines to detect features and resolve inconsistencies that we explained in the previous sections.

Throughout this section, we use $\tilde{\mathcal{S}}_i \subseteq \mathcal{S}_i$ for the set of unlabeled features at robot $i \in \{1, \ldots, n\}$ and let $|\tilde{\mathcal{S}}_i|$ be its cardinality, i.e., the number of unlabeled features at robot i. The set of labels $\mathcal{L}_i$ consists of the labels L_r^i already assigned to the features $f_r^i \in \mathcal{S}_i \setminus \tilde{\mathcal{S}}_i$. Given a matrix X_{ij} of size $|\tilde{\mathcal{S}}_i| \times |\tilde{\mathcal{S}}_j|$, we define the function $\bar{r} = \mathrm{row}\,(f_r^i)$

that takes an unlabeled feature $f_r^i \in \tilde{\mathcal{S}}_i$ and returns its associated row in X_{ij}, with $\bar{r} \in \{1, \dots, |\tilde{\mathcal{S}}_i|\}$. Equivalently, we define the function $\bar{s} = \text{col}(f_s^j)$ for features in $\tilde{\mathcal{S}}_j$. We let $\tilde{\mathbf{A}}_{ij} \in \mathbb{N}^{|\tilde{\mathcal{S}}_i| \times |\tilde{\mathcal{S}}_j|}$ be like the local association matrix $\mathbf{A}_{ij}$, but containing exclusively the rows and columns of the unlabeled features of robots i and j.

Algorithm 5.6.1 Data association and labeling - Robot i

$\tilde{\mathcal{S}}_i \leftarrow \{f_1^i, \dots, f_{m_i}^i\}$, $\mathcal{L}_i \leftarrow \emptyset$
Solve the local data association
ASSIGN_LABEL($L_r^i = (i, r)$, f_r^i) to each *exclusive* feature f_r^i
while $|\tilde{\mathcal{S}}_i| > 0$ **do**
    Run the detection algorithm
    Find each *consistent* feature f_r^i and its root $f_{r_\star}^{i_\star}$
    ASSIGN_LABEL($L_r^i = (i_\star, r_\star)$, f_r^i)
    Run the resolution algorithm
    Find each *resolved* feature f_r^i and its component id $[i_\star, r_\star]$
    ASSIGN_LABEL($L_r^i = (i_\star, r_\star)$, f_r^i)
    Find each *exclusive* feature f_r^i
    ASSIGN_LABEL($L_r^i = (i, r)$, f_r^i)
end while
function ASSIGN_LABEL(L_r^i, f_r^i)
    $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{L_r^i\}$, $\tilde{\mathcal{S}}_i \leftarrow \tilde{\mathcal{S}}_i \setminus \{f_r^i\}$
end function

Initially, all the features of each robot i are unlabeled,

$$\tilde{\mathcal{S}}_i = \{f_1^i, \dots, f_{m_i}^i\}, \qquad \mathcal{L}_i = \emptyset.$$

Each robot i solves a local data association with each of its neighbors $j \in \mathcal{N}_i$ and obtains the association matrix $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$. Then, the robot locally detects its *exclusive* features f_r^i which have not been associated to any other feature,

$$[\mathbf{A}_{ij}]_{r,s} = 0 \qquad \text{for all } j \in \mathcal{N}_i,\ j \neq i, \text{ and all } s \in \{1, \dots, m_j\}. \tag{5.9}$$

Since an exclusive feature f_r^i is always consistent, robot i assigns a label L_r^i to it composed of its own robot id and feature index and removes it from the set of unlabeled features,

$$L_r^i = (i, r), \qquad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \qquad \tilde{\mathcal{S}}_i = \tilde{\mathcal{S}}_i \setminus \{f_r^i\}. \tag{5.10}$$

Since its unlabeled features in $\tilde{\mathcal{S}}_i$ may be conflictive, it executes the detection algorithm 5.4 on this subset.

The detection algorithm is executed on the subgraph of $\mathcal{G}_{dis}$ involving the features in $\tilde{\mathcal{S}}_i$, for $i \in \{1, \dots, n\}$. When it finishes, robot i has the power matrices $X_{ij} \in \mathbb{N}^{|\tilde{\mathcal{S}}_i| \times |\tilde{\mathcal{S}}_j|}$, for $j = 1, \dots, n$, which contain the entries in $\mathbf{A}^{\text{diam}(\mathcal{G}_{dis})}$ associated to the features in $\tilde{\mathcal{S}}_i$ and $\tilde{\mathcal{S}}_j$. There is a path between f_r^i and f_s^j iff

$$[X_{ij}]_{\bar{r},\bar{s}} > 0, \tag{5.11}$$

being $\bar{r} = \text{row}(f_r^i)$ and $\bar{s} = \text{col}(f_s^j)$. These matrices give robot i the information about all the association paths of its features and the features of the rest of the robots in the network.

Then, each robot i detects its *consistent* features. After a feature f_r^i has been classified as consistent, its robot i proceeds to assign it a label. Here we show how robot i decides the feature label $(i_\star, r_\star)$. Let us first give a general definition of the root robot of an either consistent or conflictive association set.

Definition 5.6.1. *The* root *robot $i_\star$ for an association set is the one that has the most features in it. In case there are multiple candidates, it is the one with the lowest identifier. Equivalently, we define the* root *features $f_{r_\star}^{i_\star}, f_{r'_\star}^{i_\star}, \dots$ as the features from the root robot that belong to the association set.*

Using the power matrices $X_{i1}, \dots, X_{in}$, robot i can find the number of features $\tilde{m}_j$ from a second robot j that belong to the same association set than f_r^i with $\bar{r} = \text{row}(f_r^i)$ as follows,

$$\tilde{m}_j = \left| \left\{ f_s^j \mid [X_{ij}]_{\bar{r},\bar{s}} > 0, \text{ with } \bar{s} = \text{col}(f_s^j) \right\} \right|. \tag{5.12}$$

If we let $\tilde{m}_\star$ be the maximum $\tilde{m}_j$ for $j \in \{1, \dots, n\}$, then the root robot $i_\star$ and root features $f_{r_\star}^{i_\star}, f_{r'_\star}^{i_\star}, \dots$ for the association set of f_r^i with $\bar{r} = \text{row}(f_r^i)$ are

$$i_\star = \min \left\{ j \mid \tilde{m}_j = \tilde{m}_\star \right\}, \quad \{r_\star, r'_\star, \dots\} = \left\{ s \mid [X_{ii_\star}]_{\bar{r},\bar{s}} > 0 \text{ with } \bar{s} = \text{col}(f_s^{i_\star}) \right\}. \tag{5.13}$$

When f_r^i belongs to a consistent set, the root $i_\star$ corresponds to the robot with a single feature $f_{r_\star}^{i_\star}$ in the association set that has the lowest identifier,

$$i_\star = \min \left\{ j \mid [X_{ij}]_{\bar{r},\bar{s}} > 0 \text{ for some } \bar{s} \in \{1, \dots, |\tilde{\mathcal{S}}_j|\} \right\}, \quad r_\star = \left\{ s \mid [X_{ii_\star}]_{\bar{r},\bar{s}} > 0 \text{ with } \bar{s} = \text{col}(f_s^{i_\star}) \right\}, \tag{5.14}$$

where $\bar{r} = \text{row}(f_r^i)$. Robot i assigns to its feature f_r^i the label $L_r^i = (i_\star, r_\star)$ and removes it from the set of unlabeled features,

$$L_r^i = (i_\star, r_\star), \qquad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \qquad \tilde{\mathcal{S}}_i = \tilde{\mathcal{S}}_i \setminus \{f_r^i\}. \tag{5.15}$$

Thus, all features in the association set are assigned the same label. The robots proceed with all its consistent features in a similar fashion. For the features classified as conflictive, the resolution method (Algorithm 5.5.1) presented in the previous section is executed to solve the inconsistencies.

Let each component $\mathcal{C}_q$ in Algorithm 5.5.1 have the identifier $(i_\star, r_\star)$ composed of the root robot $i_\star$ and root feature $r_\star$ responsible of creating the component. When the resolution algorithm finishes, each feature f_r^i that has been assigned to a component $(i_\star, r_\star)$ has become consistent due to the edge removals. We say that such features are *resolved*. Thus, all the resolved features with the same component id form a consistent association set. Each robot i uses the component id of f_r^i as its label,

$$L_r^i = (i_\star, r_\star), \qquad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \qquad \tilde{\mathcal{S}}_i = \tilde{\mathcal{S}}_i \setminus \{f_r^i\}. \tag{5.16}$$

Additionally, due to edge removal, some unlabeled features $f_r^i \in \tilde{\mathcal{S}}_i$ may have become exclusive. Robot i detects such features f_r^i by checking that

$$[\tilde{\mathbf{A}}_{ij}]_{\bar{r},\bar{s}} = 0, \qquad \text{for all } j \in \mathcal{N}_i,\ j \neq i,\ \text{all } \bar{s} \in \{1, \ldots, |\tilde{\mathcal{S}}_j|\},$$

being $\bar{r} = \text{row}(f_r^i)$, and it manages them as in (5.10). The remaining features may still be conflictive. Each robot i executes a new detection-resolution iteration on these still unlabeled features $\tilde{\mathcal{S}}_i$.

In a finite number of iterations, all features of all robots have been labeled, and the algorithm finishes. The interest of the presented algorithm is that it is fully decentralized and works on local information. Each robot i uses its own S_{ij} and X_{ij} to classify its features.

5.7 Resolution Algorithm based on the Maximum Error Cut

The previous resolution algorithm has the advantage of solving all the inconsistencies in an easy way. However, the algorithm does not use information about the quality of the matches, and therefore is not optimal and the provided solution is arbitrary.

Most of the matching functions in the literature are based on errors between the matched features. These errors can be used to find a better partition of $\mathcal{C}$. Let $\mathbf{E}$ be the weighted symmetric association matrix

$$[\mathbf{E}]_{r,s} = \begin{cases} e_{rs} & \text{if } [\mathbf{A}]_{r,s} = 1, \\ -1 & \text{otherwise,} \end{cases} \tag{5.17}$$

with e_{rs} the error of the match between f_r and f_s.

Assumption 5.7.1. *The error between matches satisfies:*

- $e_{rr} = 0, \forall r$;
- *Errors are non negative,* $e_{rs} \geq 0, \forall r, s$;
- *Errors are symmetric,* $e_{rs} = e_{sr}, \forall r, s$;
- *Errors of different matches are different,* $e_{rs} = e_{r's'} \Leftrightarrow [r = r' \wedge s = s'] \vee [r = s' \wedge s = r']$;

Since the inconsistency is already known there is no need to use the whole matrix but just the sub-matrix related with the inconsistency, $\mathbf{E}_{\mathcal{C}}$. Although all the errors in $\mathbf{E}_{\mathcal{C}}$ are small enough to pass the matching between pairs of images, we can assume that the largest error in the path between two conflictive features is, with most probability, related to the spurious match.

Definition 5.7.2. *Given two conflictive features, we define a bridge as a* single link *whose deletion makes the conflict between those two features disappear.*

Note that not all the links in one inconsistency are bridges. There are links that, if deleted, would not break the inconsistency because:

- They do not belong to the path between the features to separate;
- They belong to the path, but they also belong to a cycle in the association graph, and therefore, they are not bridges.

Our goal is, for each pair of conflictive features, find and delete the bridge that links them with the maximum error.

Algorithm 5.7.1 shows our solution to find the bridges using local interactions. Along the section we explain in detail how it works. As we did in the detection algorithm, let each

Algorithm 5.7.1 *Maximum Error Cut* - Robot i

Require: Set of $\mathcal{C}$ different conflictive sets
Ensure: $\mathcal{G}_{dis}$ is *conflict free*
for all $\mathcal{C}$ **do**
  – *Error transmission*
  $\mathbf{z}_r(0) = \{[\mathbf{E}_\mathcal{C}]_{r,1}, \ldots, [\mathbf{E}_\mathcal{C}]_{r,c}\}$, $r = 1, \ldots, \tilde{m}_i$
  repeat
    $\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C},\ [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0}(\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs})$
  until $\mathbf{z}_r(t+1) = \mathbf{z}_r(t)$, $\forall r \in \tilde{m}_i$
  – *Link Deletion*
  while robot i has conflictive features r and r' **do**
    Find the bridges (s, s') :
      (a) $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}$, $s \neq s'$,
      (b) For all $s'' \neq s$, $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_{s''}$,
      (c) For all $s'' \neq s'$, $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_{s''}$
    Select the bridge with largest error
    Send message to break it
  end while
end for

robot initialize its own rows of elements as $\mathbf{z}_r(0) = \{[\mathbf{E}_\mathcal{C}]_{r,1}, \ldots, [\mathbf{E}_\mathcal{C}]_{r,c}\}$, $r \in \{1, \ldots, \tilde{m}_i\}$. Each robot manages the $\tilde{m}_i$ rows corresponding to the conflictive features it has observed. The update rule executed by every robot and every feature is

$$\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C},\ [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0}(\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs}), \tag{5.18}$$

where the maximum is done element to element and $\mathbf{P}_{rs}$ is the permutation matrix of the columns r and s. We have dropped the super indices corresponding to robots because the limited communications are implicit in the error caused by direct associations, eq. (5.17).

Proposition 5.7.3. *The dynamic system defined in* (5.18) *converges in a finite number of iterations and for any* $r, s \in \mathcal{C}$ *such that* $[\boldsymbol{E}_\mathcal{C}]_{r,s} \geq 0$ *the final value of* $\boldsymbol{z}_r$ *is the same than* $\boldsymbol{z}_s\boldsymbol{P}_{rs}$.

Proof. The features involved in the inconsistency form a strongly connected graph. For a given graph, the max consensus update is proved to converge in a finite number of iterations [27]. For any $r, s \in \mathcal{C}$ such that $[\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0$, by eq. (5.18) and the symmetry of $\mathbf{E}_{\mathcal{C}}$, the final consensus values of $\mathbf{z}_r$ and $\mathbf{z}_s$ satisfy, element to element, that

$$\mathbf{z}_r \geq \mathbf{z}_s \mathbf{P}_{rs} \text{ and } \mathbf{z}_s \geq \mathbf{z}_r \mathbf{P}_{sr} \tag{5.19}$$

Using the properties of the permutation matrices, $\mathbf{P}_{rs} = \mathbf{P}_{sr} = \mathbf{P}_{sr}^{-1}$, we see that $\mathbf{z}_s \mathbf{P}_{rs} \geq \mathbf{z}_r$, which combined with eq. (5.19) yields to $\mathbf{z}_r = \mathbf{z}_s \mathbf{P}_{rs}$. □

Let us see the convergence values of the different elements. Considering again eq. (5.18) for a given feature f_r, we can express it as a function of its elements, the u^{th} component, $[\mathbf{z}_r(t+1)]_u$, is updated as follows:

$$[\mathbf{z}_r(t+1)]_u = \begin{cases} \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_s) \text{ if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge u = r \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_r) \text{ if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge u = s \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_u) \text{ if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge r \neq u \neq s \end{cases}, \tag{5.20}$$

where the two first rows are due to the permutations. Let us first analyze the case in which the inconsistency does not contain any cycle.

Theorem 5.7.4. *If $\mathcal{C}$ is cycle free, then:*

(i) For any $r \in \mathcal{C}$, $[\boldsymbol{z}_r(t)]_r = 0, \forall t \geq 0$.

(ii) $[\boldsymbol{z}_r(t)]_{s'} \rightarrow [\boldsymbol{E}_{\mathcal{C}}]_{r',s'} = e_{r's'}$, where

$$r' = \underset{[\boldsymbol{A}]_{r'',s'}=1}{\arg\min}\, d(r, r''),$$

and $d(r, r'')$ is the distance in links to reach node r'' starting from node r. In other words, $f_{r'}$ is the closest feature to f_r directly associated to $f_{s'}$.

Proof. For any feature, f_r, taking into account eq. (5.20), the update of the r^{th} element of $\mathbf{z}_r$, $[\mathbf{z}_r(t+1)]_r$, is

$$[\mathbf{z}_r(t+1)]_r = \max_{s \in \mathcal{C},\ [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0} ([\mathbf{z}_r(t)]_r, [\mathbf{z}_s(t)]_s).$$

Recalling the first point in assumption 5.7.1, the initial value of $[\mathbf{z}_r(0)]_r = e_{rr} = 0$, for all r, then $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$.

The inconsistency does not have any cycles and there is a path between any two features, the conflict is a spanning tree. Let us consider one link, $(f_{r'}, f_{s'})$. The link creates a partition of $\mathcal{C}$ in two strongly connected, disjoint subsets

$$\mathcal{C}_{r'} = \{r \mid d(r, r') < d(r, s')\},$$

$$\mathcal{C}_{s'} = \{s \mid d(s, s') < d(s, r')\}.$$

In the above equations it is clear that $r' \in \mathcal{C}_{r'}$ and $s' \in \mathcal{C}_{s'}$.

We will focus now on the values of the s'^{th} element of the state vector for the nodes in $\mathcal{C}_{r'}$ and the r'^{th} element for the nodes in $\mathcal{C}_{s'}$,

$$[\mathbf{z}_r(t)]_{s'}, r \in \mathcal{C}_{r'}, \text{ and } [\mathbf{z}_s(t)]_{r'}, s \in \mathcal{C}_{s'}.$$

In the first case, for any $r \in \mathcal{C}_r \setminus r'$, update rule (5.20) is equal to

$$[\mathbf{z}_r(t+1)]_{s'} = \max_{r'' \in \mathcal{C}_{r'},\ [\mathbf{E}_C]_{r,r''} \geq 0} ([\mathbf{z}_r(t)]_{s'}, [\mathbf{z}_{r''}(t)]_{s'}).$$

because $r \neq s' \neq r''$. The nodes in $\mathcal{C}_{s'}$ are not taken into account because that would mean that $\mathcal{C}$ has a cycle. The special case of feature $f_{r'}$ has an update rule equal to

$$[\mathbf{z}_{r'}(t+1)]_{s'} = \max_{r \in \mathcal{C}_{r'}, [\mathbf{E}_C]_{r',r} \geq 0} ([\mathbf{z}_{r'}(t)]_{s'}, [\mathbf{z}_r(t)]_{s'}, [\mathbf{z}_{s'}(t)]_{r'}).$$

In a similar way the updates for features in $\mathcal{C}_s$ are

$$[\mathbf{z}_s(t+1)]_{r'} = \max_{s'' \in \mathcal{C}_{s'},\ [\mathbf{E}_C]_{s,s''} \geq 0} ([\mathbf{z}_s(t)]_{r'}, [\mathbf{z}_{s''}(t)]_{r'})$$
$$[\mathbf{z}_{s'}(t+1)]_{r'} = \max_{s \in \mathcal{C}_{s'}, [\mathbf{E}_C]_{s',s} \geq 0} ([\mathbf{z}_{s'}(t)]_{r'}, [\mathbf{z}_s(t)]_{r'}, [\mathbf{z}_{r'}(t)]_{s'}).$$

Considering together all the equations and the connectedness of $\mathcal{C}_{r'}$ and $\mathcal{C}_{s'}$, all these elements form a connected component and they will converge to

$$\max_{r \in \mathcal{C}_{r'},\ s \in \mathcal{C}_{s'}} ([\mathbf{z}_r(0)]_{s'}, [\mathbf{z}_s(0)]_{r'}),$$

Since all the features $r \in \mathcal{C}_{r'} \setminus r'$ are not associated with $f_{s'}$, $[\mathbf{z}_r(0)]_{s'} = -1$. Analogously, for all the features $s \in \mathcal{C}_{s'} \setminus s'$, $[\mathbf{z}_s(0)]_{r'} = -1$. Finally, for the features r' and s', by the second and third point of assumption 5.7.1, $[\mathbf{z}_{r'}(0)]_{s'} = e_{r's'} = e_{s'r'} = [\mathbf{z}_{s'}(0)]_{r'} \geq 0 > -1$. Therefore this subset of c elements of the state vectors converge to the error of the link $(f_{r'}, f_{s'})$, $e_{r's'}$. From Proposition 5.7.3 we can also see that for any $r \in \mathcal{C}_{r'}$, $[\mathbf{z}_r]_s, s \in \mathcal{C}_{s'} \setminus s'$, will converge to the final value of $[\mathbf{z}_{s'}]_s$.

The same argument applies for the rest of the links and the proof is complete. □

Let us see what happens now in the presence of cycles in the inconsistency.

Theorem 5.7.5. *Let us suppose the inconsistency has a cycle involving ℓ features. Let $\mathcal{C}_\ell$ be the subset of features that belong to the cycle. After the execution of* (5.18) *it holds that:*

(i) $\forall r', s' \in \mathcal{C}_\ell, s' \neq r'$

$$[\boldsymbol{z}_{r'}]_{s'} \to \max_{r,s \in \mathcal{C}_\ell} e_{rs}.$$

(ii) $\forall r' \notin \mathcal{C}_\ell,\ s' \in \mathcal{C}_\ell,\ s' \neq \arg\min_{s \in \mathcal{C}_\ell} d(r', s)$,

$$[\boldsymbol{z}_{r'}]_{s'} \to \max_{r,s \in \mathcal{C}_\ell} e_{rs}.$$

Proof. In the proof we will denote $r_1, \ldots, r_\ell$, the set of features in $\mathcal{C}_\ell$. Without a loss of generality we will assume that the links that form the cycle are $(f_{r_1}, f_{r_2}), (f_{r_2}, f_{r_3}) \ldots, (f_{r_\ell}, f_{r_1})$. For an easy reading of the proof of this result we will omit the time indices in the update equations. Let us consider the update rule (5.20) for element r_2 of feature f_{r_1},

$$[\mathbf{z}_{r_1}]_{r_2} = \max([\mathbf{z}_{r_1}]_{r_2}, [\mathbf{z}_{r_2}]_{r_1}, [\mathbf{z}_{r_\ell}]_{r_2}),$$

where we have also omitted other possible features that are directly linked to f_{r_1} because if they are also linked to f_{r2} they belong to $\mathcal{C}_\ell$ and if not they do not affect to the final result.

From the above equation we observe that $[\mathbf{z}_{r_1}]_{r_2}$ depends on the value of $[\mathbf{z}_{r_2}]_{r_1}$. At the same time this value is updated with

$$[\mathbf{z}_{r_2}]_{r_1} = \max([\mathbf{z}_{r_2}]_{r_1}, [\mathbf{z}_{r_1}]_{r_2}, [\mathbf{z}_{r_3}]_{r_1}),$$

which depends on the value of $[\mathbf{z}_{r_3}]_{r_1}$. If we keep with the chain of associations we reach the point in which $[\mathbf{z}_{r_{\ell-1}}]_{r_1}$ depends on $[\mathbf{z}_{r_\ell}]_{r_1}$, which has update rule equal to

$$[\mathbf{z}_{r_\ell}]_{r_1} = \max([\mathbf{z}_{r_\ell}]_{r_1}, [\mathbf{z}_{r_{\ell-1}}]_{r_1}, [\mathbf{z}_{r_1}]_{r_\ell}).$$

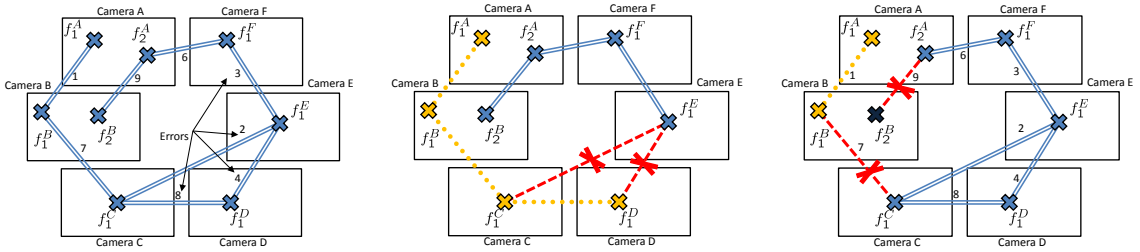
As we have proved in Proposition 5.7.3, in the end $[\mathbf{z}_{r_1}]_{r_2} = [\mathbf{z}_{r_2}]_{r_1}$, $[\mathbf{z}_{r_2}]_{r_1} = [\mathbf{z}_{r_3}]_{r_1}, \ldots,$ $[\mathbf{z}_{r_{\ell-1}}]_{r_1} = [\mathbf{z}_{r_\ell}]_{r_1}$ and $[\mathbf{z}_{r_\ell}]_{r_1} = [\mathbf{z}_{r_1}]_{r_\ell}$ because they are direct neighbors. This means that after the execution of enough iterations of (5.18), $[\mathbf{z}_{r_1}]_{r_2} = [\mathbf{z}_{r_1}]_{r_\ell} = [\mathbf{z}_r]_{r_1}, \forall r \in \mathcal{C}_\ell \setminus r_1$. By applying the same argument for any other feature in $\mathcal{C}_\ell$ we conclude that after the execution of the update, for any $r \in \mathcal{C}_\ell$, $[\mathbf{z}_r]_{r'} = [\mathbf{z}_r]_{r''}, \forall r', r'' \in \mathcal{C}_\ell \setminus r$. Thus, each feature inside the cycle will end with $\ell - 1$ elements in its state vector with the same value (the maximum of all the considered links) and (i) is true. If there are any additional links inside the cycle the result is the same including in the max consensus the weights of these links.

Now let us consider the rest of the features in the inconsistency, $\bar{\mathcal{C}}_\ell = \mathcal{C} \setminus \mathcal{C}_\ell$. Given a feature $s \in \bar{\mathcal{C}}_\ell$ two things can happen:

- $\exists$ unique $r \in \mathcal{C}_\ell$ such that f_r and f_s are directly associated;
- s is not directly associated with any feature in $\mathcal{C}_\ell$ but there exists at least one path of features $\in \bar{\mathcal{C}}_\ell$ that ends in a unique feature $r \in \mathcal{C}_\ell$;

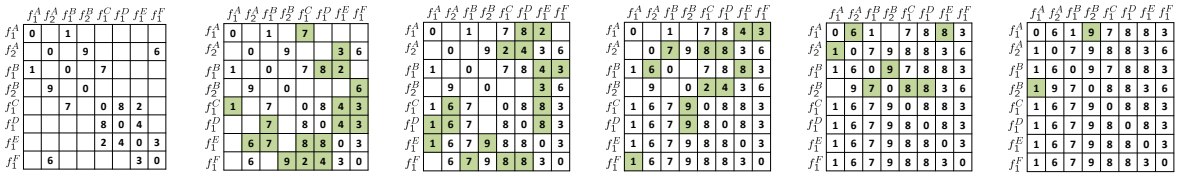
The uniqueness of r comes from the fact that if there were another feature $r' \in \mathcal{C}_\ell$, reachable from s without passing through r, that would mean that s is also part of the cycle. Note that this does not discard the possibility that r and s belong to another cycle different than $\mathcal{C}_\ell$.

As we have seen in the proof of Theorem 5.7.4, due to the fact that r is the only connection with $\mathcal{C}_\ell$, for any $r' \in \mathcal{C}_\ell \setminus r$, $[\mathbf{z}_s]_{r'}$ will have final value equal to $[\mathbf{z}_r]_{r'}$ which proves (ii). On the other hand $[\mathbf{z}_s]_r$ will have the value of the link that connects it to feature r or, if f_r and f_s belong to another cycle different than $\mathcal{C}_\ell$, the maximum error of all the links that form the second cycle. In both cases, doing a change in the names of the indices, we can see that (ii) is also true. □



(a) Inconsistency (b) *Spanning Trees* (c) *Maximum Error Cut*

Figure 5.4: Example of execution of the resolution of one inconsistency using the two approaches. (a) Inconsistency. (b) Solution obtained using the *Spanning Trees* algorithm. (c) Solution obtained using the *Maximum Error Cut* approach. A detailed explanation can be found in section 4.3.



(a) (b) (c) (d) (e) (f)

Figure 5.5: Example of execution of the algorithm (5.18) for the inconsistency in Fig. 5.4 (a). Each subfigure (a)-(f) represents a new step of the algorithm. In 6 steps the robots with inconsistent features are able to decide which links delete to solve them. For more details see Section 4.3.

At this point we are ready to define the bridges in terms of the variables $\mathbf{z}_r$ and to propose a criterion to select the bridge to break. The bridges, $(f_s, f_{s'})$, for any pair of conflictive features f_r and $f_{r'}$ satisfy

(a) $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}, s \neq s'$,

(b) for all $s'' \neq s$, $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_{s''}$,

(c) for all $s'' \neq s'$, $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_{s''}$,

The first condition comes from Theorem 5.7.4 and the other two come from Theorem 5.7.5. Note that for any bridge, the error of the bridge is the same as the value of $[\mathbf{z}_r]_s$, $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'} = e_{ss'}$. Therefore, each node can look in a local way at its own rows and choose the best bridge that breaks the conflict, the one with the largest error. In case one robot has more than two features in the same conflict, finding the optimal cut becomes NP-hard. In this chapter we use a greedy approach that returns good results. Our solution chooses two of the $\tilde{m}_i$ inconsistent features and selects the best bridge for them. The bridge separates all the $\tilde{m}_i$ features in two disconnected subsets. The process is repeated with each of the subsets until the inconsistencies are solved.

Note that we are considering only single-link deletions. Cycles in the association graph are sets of features strongly associated, and therefore, it is better not to break links there. If two conflictive features belong to the same cycle, then there are no bridges. However, the algorithm is also able to detect this situation and the *Spanning Trees* can be used to solve the conflict.

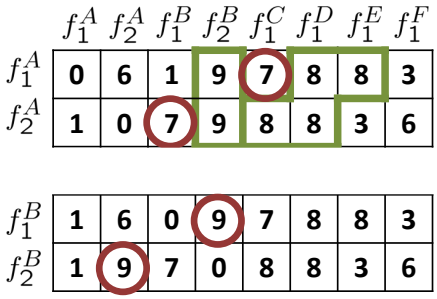


Figure 5.6: Decision about which links should be deleted to solve the inconsistency. Robot B chooses the link (f_2^A, f_2^B). Robot A discards the elements with values 8 and 9 because they belong to a cycle and to a link that does not solve its inconsistency respectively. The match between f_1^B and f_1^C solves the inconsistency and has the largest error.

In conclusion, this algorithm is able to detect in a local way the best bridge to break each inconsistency. This provides a more solid criterion to solve the inconsistencies than just cutting arbitrary edges. Each robot is able to detect which set of links is best to cut in order to solve the conflicts regarding its own features. The algorithm also finishes in finite time and does not require much additional bandwidth because, as in the detection algorithm, the amount of transmitted information can be optimized.

5.7.1 Example of execution

Let us consider one inconsistency as the one depicted in Fig. 5.4 (a) where the communication graph is a ring with an additional link between robots C and E. The *Spanning Trees* solution is shown in Fig. 5.4 (b). In this case the root camera to manage the inconsistency is the camera A. For each feature, camera A instantiates a different spanning tree. After 2 communications rounds, robots C and E send a request to D and also among them. f_1^D gets attached to f_1^C and the other links are broken. After this point the algorithm has ended its execution and the new association graph is conflict free.

Figure 5.4 (c) shows the solution obtained using the *Maximum Error Cut* algorithm. The evolution of the $\mathbf{z}_r$ vectors is shown in Figure 5.7. Each one of the subfigures (a)-(f) represents a new iteration of the algorithm in (5.18). The -1 values are omitted for clarity. As an example of how it works, the third row in subfigure 5.7 (b), corresponding to f_1^B, executes (5.18) with the first and fifth rows, sent by robots A and C because of features f_1^A and f_1^C. Robot B permutes the first and third element of the vector sent by robot A and the third and fifth element of vector sent by robot C and chooses the maximum (element to element) of the three vectors. As a result the sixth and seventh position (features f_1^D and f_1^E) change their values. It is interesting to observe how for the cycle all the elements in the different vectors are receiving the value "8", corresponding to the largest value within the cycle. Once rule (5.18) has finished, robots A and B look for the bridges to break their inconsistencies (Fig. 5.6). For robot B the best bridge is the one matching features f_2^A and f_2^B. For the robot A this is not a bridge because both features have the same value in the same element. The next largest value is also discarded because it belongs to a cycle. Finally, the bridge with error 7 is selected and the match between

f_1^B and f_1^C is deleted.

5.8 Discussion

In order to show the performance of the algorithm, we have carried out several simulations with a team composed by 7 robots exploring an environment of 20×20 m with 300 features, see Fig. 5.7. Each robot executes 70 motion steps along a path of approximately 30 m. The robots estimate their motion based on odometry information that is corrupted with a noise of standard deviation $\sigma_x, \sigma_y = 0.4$ cm for the translations and $\sigma_\theta = 1$ degree for the orientations. They sense the environment using an omnidirectional camera that gives bearing measurement to features within 360 degrees around the robot and within a distance of 6 m. The measurements are corrupted with a noise of 0.5 degrees standard deviation. Each robot explores the environment and builds its local (Fig. 5.8).

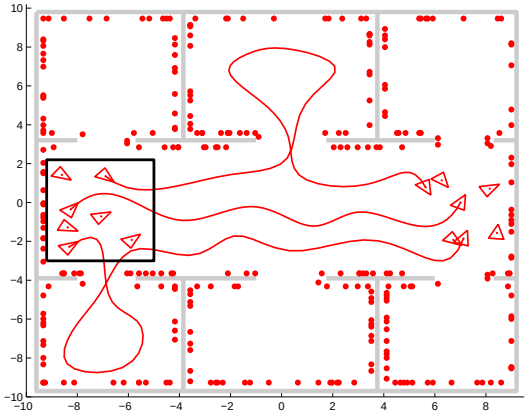


Figure 5.7: A team of 7 robots explore an environment of 20×20 m. Gray areas are walls and red dots are the ground-truth location of landmarks. Initially, the robots are placed in the black box region. From these initial poses, they reach consensus on the global reference frame. After that, they explore the environment and build their maps according this global reference frame. We display the trajectories followed by robots 2, 3, and 5, together with the final poses of the 7 robots.

Due to the presence of obstacles (gray areas), each robot may have not observed some landmarks. Besides, the precision of the estimated positions (blue crosses and ellipses) of the landmarks depends on the trajectory followed by each robot.

When they finish the exploration, they execute the consistent data association algorithm under the communication graph in Fig. 5.9. We assume that a pair of robots can exchange information if they are within a distance of 3 m. The local data associations $F(\mathcal{S}_i, \mathcal{S}_j)$ are obtained by applying the JCBB method [97] to the local maps of any pair of neighboring robots $(i, j) \in \mathcal{E}_{com}$. Since all the trajectories followed by the robots traverse the main corridor (Fig. 5.7) there is a high overlapping between their local maps (Table 5.1). Given any 2 local maps with approx. 122 features, there are approximately 89 true matches (ground truth). We can see that, although the local data association

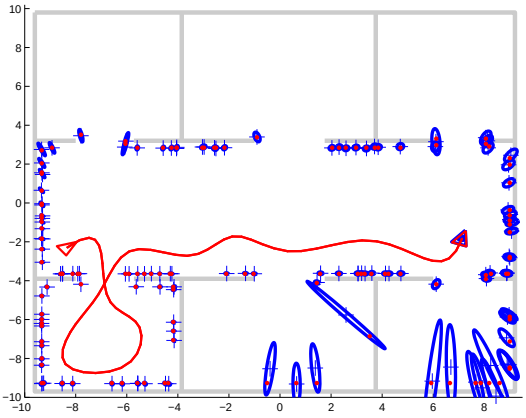


Figure 5.8: Local map estimated by robot 2. The landmarks close to its trajectory (red line) have been estimated (blue crosses and ellipses) with a high precision. Besides, its estimated positions (blue crosses) are very close to the ground truth locations (red dots). Due to the presence of obstacles (gray areas) some of the landmarks have not been observed, or have been estimated with high uncertainty.

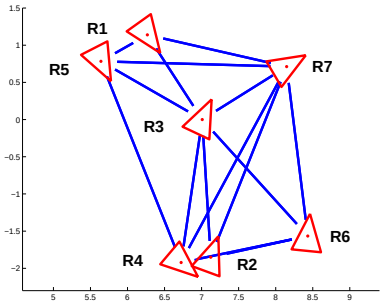


Figure 5.9: Communication graph associated to the final robot poses in Fig. 5.7. There is a link (blue solid line) between any pair of robot poses (red triangles) that are within a distance of 3 m.

method has found a high amount of the ground truth links (good links or true positives), it has also missed a few of them (missing links or false negatives). In addition, some additional links have been detected that link together different features (spurious links or false positives).

From the 858 features within all the local maps, there are 300 different features in the ground truth sense (association sets). From them, 184 were observed by a single robot (ground truth exclusive features), and the remaining where observed by around 6 robots (ground truth size of the remaining association sets). As previously stated, there is a high overlapping between the maps, and each non-exclusive feature has been observed by almost every robot. In the data association graph $\mathcal{G}_{dis}$ however, only 296 association sets have been obtained, which means that different features have been mixed up together. There are 184 exclusive features (ground truth exclusive features), although the local data association algorithm has found 187 exclusive features. These additional 3 exclusive features appear due to the presence of the three outliers, the features with high covariance

ellipses in Fig. 5.10. Since their positions have been wrongly estimated, the local data association method has failed to correctly associate them.

Table 5.1: Local data associations.

Features	**Per local map**	**Total**
Features observed	122	**858**
Data associations	**Per pair of local maps**	**Total**
Links (ground truth)	89	2860
Links	88	2820
Good links	85	2750
Missing links	3	110
Spurious links	2	70
Association sets	**Obtained**	**Ground truth**
Association sets	**296**	**300**
Exclusive features	**187**	**184**
Non-exclusive assoc.	**109**	116
Size of non-exclusive	6.1	5.8

The robots execute Algorithm 5.6.1 on the non-exclusive features to detect and solve any inconsistent associations. From the 109 non exclusive association sets, 102 of them are consistent, and its associated 591 features are classified as consistent (Table 5.2). The remaining 7 sets are conflictive, and they have associated 80 conflictive features. After executing the resolution algorithm on the 80 conflictive features, all of them are resolved and the process finishes. The original 7 conflictive sets are partitioned into 14 consistent non-exclusive sets. Due to these additional sets, the number of consistent non-exclusive association sets (Table. 5.2, third row), which initially was 102 (Table. 5.2 first row), is increased into 116 (102 + 14) after executing the algorithm. Equivalently, the number of consistent non-exclusive features (Table. 5.2, fourth row) which was 591 (Table. 5.2, second row) becomes 671 (591 + 80) since the 80 inconsistent features are resolved.

Table 5.2: Detection and resolution of inconsistent associations.

Detection	**Conflictive**	**Consistent non-exclusive**	**Consistent exclusive**
Association sets	7	102	187
Features	80	591	187
Resolution	**Conflictive**	**Consistent non-exclusive**	**Consistent exclusive**
Association sets	0	116 **(+14)**	187
Features	0	671 **(+80)**	187

The comparison between the final data association graph and the ground truth information can be seen in Table 5.3. Since the resolution algorithm is based on link deletion, the number of links here is lower than in Table 5.1. However, the number of association sets is closer to the ground truth results. From the 303 obtained association sets, 3 of them are due to the three outliers in Fig.5.10. Thus, there are 300 remaining association

sets, which is exactly the same number of association sets in the ground truth data. The same behavior is observed regarding their sizes. This means that the resulting associations are similar to the ground truth ones in spite of the fact that they have less links. From the 26 links erased from $\mathcal{G}_{dis}$, 22 were spurious links, and only 4 where good links that now are missing.

Table 5.3: Results after detecting and solving the inconsistencies.

Features	**Per local map**	**Total**
Features observed	122	858
Data associations	**Per pair of local maps**	**Total**
Links (ground truth)	89	2860
Links	87	2794 **(-26)**
Good links	85	2746 **(-4)**
Missing links	3	114 **(+4)**
Spurious links	2	48 **(-22)**
Association sets	**Obtained**	**Ground truth**
Association sets	303	300
Exclusive features	187	184
Non-exclusive assoc.	116	116
Size of non-exclusive	5.7	5.8

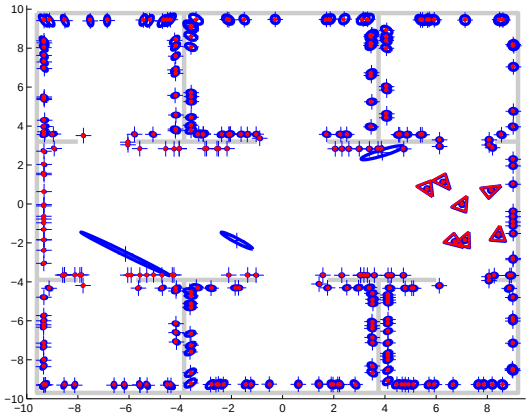


Figure 5.10: Global map $\mathbf{x}_G^i(t), \Sigma_G^i(t)$ estimated by robot 2 after $t = 5$ iterations. Red dots are the ground truth position of the features while blue crosses and ellipses are their estimated positions. Red triangles are the ground truth poses of the 7 robots after the exploration, and blue triangles are their estimated poses in the global map of robot 2. The three landmarks with high covariance ellipses are outliers that have been wrongly estimated by one of the robots. They have been classified by the algorithm as exclusive, giving rise to the presence of 3 additional exclusive sets in the final association map (303) compared to the ground truth information (300).

After associating their features, the 7 robots compute the global map as described in Chapter 3, under the communication graph in Fig. 5.9. Since the communication graph

has a high connectivity, in a few iterations each robot has received information from any other robot. After 5 iterations, the global map at robot 2 already contains precise estimates of the whole explored environment (Fig. 5.10).

5.9 Conclusions

We have presented a new technique to match several sets of features observed by a team of robots in a consistent way under limited communications. Local associations are found only within robots that are neighbors in the communication graph. After that, a fully decentralized method to compute all the paths between local associations is carried out, allowing the robots to detect all the inconsistencies related with their observations. For every conflictive set detected, in a second step the method is able to delete local associations to break the conflict using only local communications. The whole method is proved to finish in a finite amount of time finding and solving all the inconsistent associations. Experimental results show the performance of the method in scenarios with great interest in robotic tasks. To the best of our knowledge this is the first algorithm that is able to do it.

Chapter 6
Distributed Localization

As discussed at the beginning of this document, one of the issues in a distributed perception scenario is that, usually, robots start at unknown poses and do not share any reference frame. In this chapter we address the problem of distributed localization, which consists of establishing this common frame and computing the robots' poses relative to this frame. Each robot is capable of measuring the relative pose of its neighboring robots. However, it does not know the poses of far robots, and it can only exchange data using the range-limited communication network. In this chapter, we discuss three different scenarios and propose distributed algorithms for them. In the first scenario, each robot measures the planar position and orientation of nearby robots relative to its own frame, being the measurements noise free. In the second case, we address the same problem but assuming that the measurements are noisy. In the third case, the robots measure and estimate their positions, i.e., they do not compute their orientations, being the measurements corrupted with noises. We discuss the cases that the common frame is set at an anchor robot, and that this common frame is placed at the centroid of the robot team. The presented algorithms have the interesting property that can be executed in a distributed fashion. They allow each robot to recover its pose using exclusively local information and local interactions with its neighbors. Besides, they only require each robot to maintain an estimate of its own pose. Thus, the memory load of the algorithm is low compared to methods where each robot must also estimate the positions or poses of any other robot.

6.1 Introduction

Multi-robot tasks, such as pattern formation [22, 144] or collision avoidance [127], often require the knowledge of the robots' positions in a common reference frame. Typically, robots start at unknown locations, they do not share any common frame, and they can only measure the relative positions of nearby robots. We address the localization problem, which consists of combining these relative measurements to build an estimate of the robots' positions in a common frame.

Several localization algorithms rely on range-only [2, 32, 33], or bearing-only [122] relative measurements of the robots' poses. Other approaches assume that robots measure the full state of their nearby robots. The relative full-pose of a pair of robots can be obtained, for instance, by comparing their local maps [38,39,137] and looking for overlapping

regions. This approach, known as map alignment, presents a high computational cost and its results depend on the accumulated uncertainty in the local maps. Alternatively, each robot can locally combine several observations to build an estimate of the relative poses. The 2D relative pose can be retrieved from at least five noisy distance measurements and four noisy displacements [158]. Bearing-only measurements can be also used to recover the 2D relative pose in vision systems [120]. The 3D case has also been analyzed for distance and bearing, bearing-only, and distance-only observations [141]. These methods present the benefit that the obtained results do not depend on the uncertainties in the local maps. They also allow the robots to compute their relative poses when there is no overlapping between their maps, or even if they do not actually have a map.

Network localization algorithms properly combine the previous relative measurements to produce an estimate of the robots' poses. Some distributed algorithms compute both the positions and orientations but assume that the relative measurements are noise free, e.g., [57] where each robot reaches an agreement on the centroid of the network expressed in its local reference frame. Other methods compute exclusively the robot positions but not their orientations, and consider noisy relative measurements of the robot positions [18,119]. This latter localization problem can be solved by using linear optimization methods [18, 119]. Although these works do not consider the robots' orientations, they can also be applied to such cases provided that the robots have previously executed an attitude synchronization [95,121] or a motion coordination [43] strategy to align their orientations.

Cooperative localization algorithms [76, 118, 140] do not just compute the network localization once, but also track the robots positions. These algorithms, however, usually assume that an initial guess on the robot poses exists.

Formation control [43,52,70,82] and network localization are related problems. While localization algorithms compute robot positions that satisfy the inter-robot restrictions, in formation control problems the robots actually move to these positions. The goal formation is defined by a set of inter-robot restrictions (range-only, bearing-only, full-positions, or relative poses). Although some works discuss the effects of measurement noises in the final result [43], formation algorithms usually assume that both, the measurements and the inter-robot restrictions are noise free [52,70,82]. Thus additional analysis is necessary in noisy localization scenarios.

Both, formation control and localization problems can be solved up to a rotation and a translation. This ambiguity disappears when the positions of a subset of *anchor* robots is given in some absolute reference frame. The range-only case [2] requires at least three non-collinear anchors for planar scenarios. The density and placement of anchors has an important effect on the accuracy of the solution for the bearing-only case [122]. In the full-position case a single anchor is enough. Its placement influences the accuracy of the final results and it is common to analyze the estimation errors at the robots as a function of their distances to the anchor [20]. However, it is common to assume that the first robot is the anchor placed at the origin of the common reference frame and make the other robots compute their positions relative to the anchor.

In this chapter we focus on network localization methods where robots measure the relative full-pose of their neighbors. We assume that one of the methods presented in the

second paragraph [120, 141, 158] is executed by the robots to compute the relative pose measurements and their associated covariances. Since these methods do not require the robots to have a map, it can be executed at any time. In particular, we execute it at an initial stage, prior to any exploration taking place. The communication graph during this initial stage must be connected. We consider scenarios with both noisy and noise-free relative measurements. For the noisy case, we assume that these measurements are independent since they are acquired individually by the robots. We do not further discuss cooperative localization algorithms, since in a map merging scenario it is enough for the robots to compute the global frame and their poses once. In addition, in this chapter we discuss the selection of the common reference frame. We consider the cases that the common frame is one of the robots (anchor-based), and that the common frame is the centroid.

First, we discuss a scenario where each robot measures the noise-free pose of its nearby robots in its own reference frame. We build in [142], where a camera network reaches an agreement on the pose of an object observed from each camera. When the cameras do not share a common reference frame, each robot uses the relative pose of its neighbors to transform their estimates into its own reference frame. We show that this algorithm allows the robots to compute their poses relative to the centroid of the team in a distributed fashion. Then, we present an algorithm for noisy measurements of relative poses. The robots estimate their poses relative to an anchor node in a distributed fashion. Finally, we discuss the noisy position estimate case. We present a distributed algorithm that allows the robots to simultaneously compute the centroid of the team and their positions relative to the centroid. We show that when the centroid of the team is selected as the common frame, the estimates are more precise than with any anchor selection.

In order to make the reading easy, along the chapter we use the indices i, j to refer to robots and indices e, e' to refer to edges. An edge e starting at robot i and ending at robot j is represented by $e = (i, j)$. Given a matrix A, the notations $A_{r,s}$ and $[A]_{r,s}$ corresponds to the (r, s) entry of the matrix. We let $\otimes$ be the Kronecker product, $\mathbf{I}_r$ be the identity matrix of size $r \times r$, and $\mathbf{0}_{r\times s}$ be a $r \times s$ matrix with all entries equal to zero. A matrix A defined by blocks A_{ij} is denoted $A = [A_{ij}]$. The operation $A = \text{blkDiag}(B_1, \ldots, B_r)$ returns a matrix A defined by blocks with $A_{ii} = B_i$ and $A_{ij} = \mathbf{0}$ for $i \neq j$.

6.2 Problem Description

The problem addressed in this chapter consists of computing the localization of a network of $n \in \mathbb{N}$ robots from relative measurements. We consider three different scenarios.

In the first scenario, the goal is to compute the planar poses of $n \in \mathbb{N}$ robots $\{\mathbf{p}_1^G, \ldots, \mathbf{p}_n^G\}$ expressed in the global frame G, where $\mathbf{p}_i^G = \left[x_i^G, y_i^G, \theta_i^G\right] \in SE(3)$ for $i \in \{1, \ldots, n\}$, given $m \in \mathbb{N}$ measurements of relative poses between robots. The robots measure the planar pose (position and orientation) of nearby robots expressed on their own reference frame, being the measurements noise-free. We let $\mathbf{p}_j^i \in SE(3)$ be the pose of a robot j relative to robot i. This information is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V} = \{1, \ldots, n\}$ are the robots, and $\mathcal{E}$ contains the m relative measurements, $|\mathcal{E}| = m$. There is an edge $e = (i, j) \in \mathcal{E}$ from i to j if robot i has a

relative measurement of the state of robot j. We assume that the measurement graph $\mathcal{G}$ is directed and weakly connected, and that an robot i can exchange data with both its in and out neighbors $\mathcal{N}_i$ so that the associated communication graph is undirected,

$$\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E} \text{ or } (j,i) \in \mathcal{E}\}.$$

We let $\mathcal{A} \in \{0,1,-1\}^{n \times m}$ be the *negative* incidence matrix of the measurement graph,

$$\mathcal{A}_{i,e} = \begin{cases} -1 & \text{if } e = (i,j) \\ 1 & \text{if } e = (j,i) \\ 0 & \text{otherwise} \end{cases}, \text{ for } i \in \{1,\dots,n\}, e \in \{1,\dots,m\}, \tag{6.1}$$

and we let $\mathcal{W}_{i,j}$ be the Metropolis weights defined in eq. (A.3) in Appendix A associated to $\mathcal{G}$. The localization problem consists of estimating the states of the n robots from the relative measurements. Any solution can be determined only up to a rotation and a translation, i.e., several equivalent solutions can be obtained depending on the reference frame selected. We compute the robot poses relative to a global frame G whose position is the centroid of the team, and whose orientation is the Karcher mean [91] of the robots orientations as explained in Section 6.3. We assume that the orientations of the robots satisfy $-\pi/2 < \theta_i < \pi/2$ for all $i \in \mathcal{V}$, so that the average orientation is well defined [91].

Alternatively [18], one of the robots $a \in \mathcal{V}$, e.g., the first one $a = 1$, can be established as an anchor with state $\mathbf{p}_a^a = \mathbf{0}_{3\times 1}$, and the poses of the non-anchor robots can be expressed relative to the anchor. We call such approaches anchor-based and add the superscript a to their associated variables. We let $\mathcal{V}^a = \mathcal{V} \setminus \{a\}$ be the set of non-anchor nodes and matrix $\mathcal{A}^a \in \{0,1,-1\}^{n-1 \times m}$ be the result of deleting the row associated to node a from $\mathcal{A}$ in eq. (6.1). This is the case considered in our second scenario, where we address the anchor-based localization problem for the case that the relative measurements are noisy. Each edge $e = (i,j) \in \mathcal{E}$ in the relative measurements graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has associated noisy measurements of the orientation $\mathbf{z}_e^\theta$ and the position $\mathbf{z}_e^{xy}$ of robot j relative to robot i, with associated covariance matrices $\Sigma_{\mathbf{z}_e^\theta}$ and $\Sigma_{\mathbf{z}_e^{xy}}$. We assume that the measurements are independent since they were acquired individually by the robots. The goal is to estimate the robot poses $\hat{\mathbf{p}}_i^a$ of the non-anchor robots $i \in \mathcal{V}^a$ relative to the anchor a from the noisy relative measurements. As in the previous scenario, here we also assume that the orientations of the robots satisfy $-\pi/2 < \theta_i < \pi/2$ for all $i \in \mathcal{V}$.

In the third scenario, instead of computing robot poses, we consider that each robot $i \in \mathcal{V}$ has a $p-$dimensional state $\mathbf{x}_i \in \mathbb{R}^p$, and that the measurement $\mathbf{z}_e \in \mathbb{R}^p$ associated to an edge $e = (i,j) \in \mathcal{E}$ relates the states of robots i and j as follows

$$\mathbf{z}_e = \mathbf{x}_j - \mathbf{x}_i + \mathbf{v}_e,$$

where $\mathbf{v}_e \sim N(\mathbf{0}_{p\times p}, \Sigma_{\mathbf{z}_e})$ is a Gaussian additive noise. Thus, we solve a position localization problem, although the proposed method can be alternatively applied for estimating speeds, accelerations, or current times. In addition, this method can be used in a pose localization scenario, provided that the robots have previously executed an attitude synchronization [95, 121] or a motion coordination [43] strategy to align their orientations. We estimate the states $\hat{\mathbf{x}}_i^{cen}$ of the robots $i \in \mathcal{V}$ relative to the centroid of the states, and compare this representation with a classical anchor-based one $\hat{\mathbf{x}}_i^a$. In the following sections we explain in detail the three scenarios.

6.3 Noise-free Pose Localization

In this section we address the problem of establishing a common reference frame for a robot team in a planar environment and obtaining the robot poses in this frame. The robots measure the relative poses (relative positions and orientations) of nearby robots and we consider the case that the measurements are noise-free. We propose a distributed strategy where the robots use the relative measurements to compute their pose relative to the global frame. We define the global frame as the centroid of the robot positions and the Karcher mean [91] of their orientations. We assume that the orientations of the robots satisfy $0 < \theta_i < \pi$ for all $i \in \mathcal{V}$, so that the average orientation is well defined [91].

In this section, we represent the pose $\mathbf{p}_i^G = (R_i^G, T_i^G) \in SE(3)$ of robot i expressed in the global frame G, by the following rotation matrix R_i^G and translation vector T_i^G,

$$R_i^G = \begin{bmatrix} \cos\theta_i^G & -\sin\theta_i^G & 0 \\ \sin\theta_i^G & \cos\theta_i^G & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad T_i^G = \begin{bmatrix} x_i^G \\ y_i^G \\ 0 \end{bmatrix}, \tag{6.2}$$

where $z = 0$ since the robots move on the plane, and the planar rotations are around the z-axis. We let $\mathbf{p}_j^i = (R_j^i, T_j^i) \in SE(3)$ be the pose of a robot j relative to robot i. Recall that as previously stated relative measurements are noise-free, thus we use $\mathbf{p}_j^i$ indistinctively for the true and the measured relative poses. Along this section we use the generic world reference frame $\mathtt{w}$ as a tool for providing the theoretical results and definitions, and we let $\mathbf{p}_i^{\mathtt{w}} = (R_i^{\mathtt{w}}, T_i^{\mathtt{w}}) \in SE(3)$ be the pose of robot i expressed in frame $\mathtt{w}$. Note that this frame, which is unknown by the robots, is only used for deriving our proposal and it is not needed for executing the proposed algorithms. Instead, the proposed algorithms exclusively rely on the relative poses $\mathbf{p}_j^i$ between neighboring robots.

As previously stated, we define the global frame G as the centroid of the robot positions and the Karcher mean [91] of their orientations. We begin by discussing what the centroid of the robot team is. Suppose we are given the true robot poses $\mathbf{p}_i^{\mathtt{w}} = (R_i^{\mathtt{w}}, T_i^{\mathtt{w}})$ in a world reference frame $\mathtt{w}$. Then the position of the centroid $T_G^{\mathtt{w}}$ in frame $\mathtt{w}$ is

$$T_G^{\mathtt{w}} = \frac{1}{n} \sum_{j=1}^{n} T_j^{\mathtt{w}}. \tag{6.3}$$

The equivalent expression for the orientation of the global reference frame $R_G^{\mathtt{w}}$ requires further explanations on averages of rotations and it is discussed later in this section. Let $(R_{\mathtt{w}}^G, T_{\mathtt{w}}^G)$ be the inverse of $(R_G^{\mathtt{w}}, T_G^{\mathtt{w}})$, i.e., the pose of the world frame $\mathtt{w}$ expressed in the centroid frame, $R_{\mathtt{w}}^G = (R_G^{\mathtt{w}})^T$, $T_{\mathtt{w}}^G = -(R_G^{\mathtt{w}})^T T_G^{\mathtt{w}}$. Then, the pose of each robot $i \in \mathcal{V}$ with respect to the centroid frame $\mathbf{p}_i^G = (R_i^G, T_i^G)$ is

$$R_i^G = R_{\mathtt{w}}^G R_i^{\mathtt{w}}, \qquad T_i^G = R_{\mathtt{w}}^G T_i^{\mathtt{w}} + T_{\mathtt{w}}^G. \tag{6.4}$$

The centroid G representation does not depend on the original reference frame $\mathtt{w}$ selected. The same robot poses relative to the centroid $\mathbf{p}_i^G$ are obtained regardless of the reference $\mathtt{w}$ used in (6.3)-(6.4). Moreover, as we show in this section, the robots do not need to know any world reference frame $\mathtt{w}$ in order to compute their poses $\mathbf{p}_i^G = (R_i^G, T_i^G)$ with

respect to G (6.4). More specifically, we let each robot i compute the pose of the centroid $\mathbf{p}_G^i = (R_G^i, T_G^i)$ in its own reference frame; and $\mathbf{p}_i^G$ can be obtained from $\mathbf{p}_G^i$ and vice versa as follows:

$$R_i^G = \left(R_G^i\right)^T, \qquad T_i^G = -\left(R_G^i\right)^T T_G^i, \qquad R_G^i = \left(R_i^G\right)^T, \qquad T_G^i = -\left(R_i^G\right)^T T_i^G. \tag{6.5}$$

We first present the distributed algorithm for the positions and then we explain the agreement on the orientations. Both algorithms exclusively rely on the relative poses $\mathbf{p}_j^i$ between neighboring robots, here denoted by $(R_j^i, T_j^i) \in SE(3)$.

We are ready to present the distributed algorithm executed by the robots to agree on the position of the global reference frame, computed as the centroid G of the robot team. Let each robot $i \in \mathcal{V}$ have an estimate of the position of the centroid in its own reference frame $T_G^i(t) \in \mathbb{R}^3$ which is initialized as $T_G^i(0) = \mathbf{0}$ and updated each time step $t \in \mathbb{N}$ according to

$$T_G^i(t+1) = \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j}(R_j^i T_G^j(t) + T_j^i), \tag{6.6}$$

where $\mathcal{W}_{i,j}$ are the Metropolis weights associated to the relative measurements graph $\mathcal{G}$ as defined in eq. (A.3) in Appendix A. Then, we have the following result.

Proposition 6.3.1. *Assume $\mathcal{G}$ is connected. Then, if each robot $i \in \mathcal{V}$ executes the algorithm* (6.6)*, as $t \to \infty$, each $T_G^i(t)$ with $i \in \mathcal{V}$ tends to the position of the centroid in robot's i reference T_G^i as defined by eqs.* (6.3)-(6.5)*,*

$$\lim_{t \to \infty} T_G^i(t) = T_G^i. \tag{6.7}$$

Proof. First of all, we show that, by making a proper change of variables, the update rule (6.6) is actually an averaging algorithm (see Appendix A) and thus the states at the robots asymptotically reach consensus. Consider the change of variables,

$$\mathcal{T}_i(t) = R_i^{\mathtt{w}} T_G^i(t) + T_i^{\mathtt{w}}, \qquad\qquad T_G^i(t) = \left(R_i^{\mathtt{w}}\right)^T \mathcal{T}_i(t) - \left(R_i^{\mathtt{w}}\right)^T T_i^{\mathtt{w}}, \tag{6.8}$$

that expresses each $T_G^i(t)$ in a world frame $\mathtt{w}$, being $(R_i^{\mathtt{w}}, T_i^{\mathtt{w}})$ the robot poses in frame $\mathtt{w}$. We apply this change of variables to our system (6.6),

$$\begin{aligned} \mathcal{T}_i(t+1) &= \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j} R_i^{\mathtt{w}} R_j^i R_{\mathtt{w}}^j \mathcal{T}_j(t) + \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j}(R_i^{\mathtt{w}} R_j^i T_{\mathtt{w}}^j + R_i^{\mathtt{w}} T_j^i + T_i^{\mathtt{w}}) \\ &= \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j} \mathcal{T}_j(t), \end{aligned} \tag{6.9}$$

since $\sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j} = 1$ for all $i \in \mathcal{V}$, $R_i^{\mathtt{w}} R_j^i R_{\mathtt{w}}^j = R_{\mathtt{w}}^{\mathtt{w}} = \mathbf{I}$, and $R_i^{\mathtt{w}} R_j^i T_{\mathtt{w}}^j + R_i^{\mathtt{w}} T_j^i + T_i^{\mathtt{w}} = T_{\mathtt{w}}^{\mathtt{w}} = \mathbf{0}$, for all $i, j \in \mathcal{V}$. Thus, (6.9) is an averaging algorithm like (A.1) in Appendix A, which converges to (A.2) the average of the initial states,

$$\lim_{t \to \infty} \mathcal{T}_i(t) = \frac{1}{n} \sum_{j=1}^n \mathcal{T}_j(0) = \frac{1}{n} \sum_{j=1}^n R_j^{\mathtt{w}} T_G^j(0) + T_j^{\mathtt{w}} = \frac{1}{n} \sum_{j=1}^n T_j^{\mathtt{w}} = T_G^{\mathtt{w}}, \tag{6.10}$$

which is the position of the centroid (6.3) in the world frame $\mathtt{w}$. Let us now reverse the change of variables (6.8) and use equations (6.4), (6.5) and (6.10),

$$\begin{aligned}\lim_{t\to\infty} T_G^i(t) &= (R_i^{\mathtt{w}})^T T_G^{\mathtt{w}} - (R_i^{\mathtt{w}})^T T_i^{\mathtt{w}} = -(R_i^{\mathtt{w}})^T \left(R_{\mathtt{w}}^G\right)^T T_{\mathtt{w}}^G - (R_i^{\mathtt{w}})^T \left(R_{\mathtt{w}}^G\right)^T R_{\mathtt{w}}^G T_i^{\mathtt{w}} \\ &= -\left(R_i^G\right)^T \left(R_{\mathtt{w}}^G T_i^{\mathtt{w}} + T_{\mathtt{w}}^G\right) = -\left(R_i^G\right)^T T_i^G = T_G^i, \end{aligned} \tag{6.11}$$

and the proof is complete. □

Let us now discuss the agreement on the orientation of the global reference frame. We use the Karcher mean [91] to compute the average of the robot orientations. Given the robot orientations $R_i^{\mathtt{w}}$ in some frame $\mathtt{w}$, the orientation of the global frame $R_G^{\mathtt{w}}$ is the Karcher mean given by,

$$R_G^{\mathtt{w}} = \arg\min_{R^{\mathtt{w}}} \sum_{i=1}^{n} d^2(R^{\mathtt{w}}, R_i^{\mathtt{w}}), \tag{6.12}$$

where $d^2(R_i^{\mathtt{w}}, R)$ is the Riemannian square distance between $R^{\mathtt{w}}$ and $R_i^{\mathtt{w}}$ given by

$$d^2(R^{\mathtt{w}}, R_i^{\mathtt{w}}) = -\frac{1}{2}\mathrm{Tr}\big\{[\log((R^{\mathtt{w}})^T R_i^{\mathtt{w}})]^2\big\}. \tag{6.13}$$

Here log is the logarithmic map $\log : SO(3) \to so(3)$ defined by

$$\log(R) = \begin{cases} \mathbf{0}_{3\times 3} & \text{if } \beta = 0, \\ \frac{\beta}{2\sin\beta}(R - R^T) & \text{if } \beta \neq 0, \end{cases} \tag{6.14}$$

where $\beta = \arccos\left(\frac{\mathrm{Tr}\{R\}-1}{2}\right)$. The term $\sum_{i=1}^n d^2(R_i^{\mathtt{w}}, R)$ in (6.12) can be seen as a cost function to be minimized. Recalling that the initial orientations θ_i are located in a geodesic ball of radius less than $\pi/2$, then this cost function restricted to the geodesic ball is convex, and the Karcher mean is well defined [91].

The orientation of the global frame $R_G^i(t)$ relative to each robot i is then computed using a distributed consensus algorithm on $SO(3)$ [142] combined with the Metropolis weights as defined in eq. (A.3) in Appendix A. Robot i initializes its variable $R_G^i(0) = \mathbf{I}$ and updates it at each $t \in \mathbb{N}$ by

$$R_G^i(t+1) = R_G^i(t)\exp(\mathbf{u}_i(t)), \qquad \mathbf{u}_i(t) = \sum_{j\in\mathcal{N}_i\cup\{i\}} \mathcal{W}_{i,j}\log(R_G^i(t)^T R_j^i R_G^j(t)), \tag{6.15}$$

where $\mathcal{W}_{i,j}$ are the Metropolis weights associated to the graph $\mathcal{G}$ given by eq. (A.3) in Appendix A, and exp is the exponential map $\exp : so(3) \to SO(3)$ defined by

$$\exp(\mathbf{u}_i(t)) = \begin{cases} \mathbf{I} & \text{if } \alpha = 0, \\ \mathbf{I} + \frac{\sin\alpha}{\alpha}\mathbf{u}_i(t) + \frac{1-\cos\alpha}{\alpha^2}\mathbf{u}_i^2(t) & \text{if } \alpha \neq 0, \end{cases}$$

where $\alpha = \sqrt{\frac{1}{2}\mathrm{Tr}\{\mathbf{u}_i^T(t)\mathbf{u}_i(t)\}}$. Given that the robot orientations are planar and within a range of $\pm\frac{\pi}{2}$, and that the graph $\mathcal{G}$ is connected, this algorithm converges to the unique Karcher mean expressed in each robot's reference frame

$$\lim_{t\to\infty} R_i^{\mathtt{w}} R_G^i(t) = R_G^{\mathtt{w}}, \tag{6.16}$$

for all $i \in \mathcal{V}$.

After executing this algorithm for order $n^2 \log(\epsilon^{-1})$ iterations each robot obtains its pose relative to the global frame with a precision ϵ (see Appendix A). Robots can then express their local maps according to this common reference frame.

6.4 Noisy Pose Localization

The problem addressed in this section consists of computing the planar poses of $n \in \mathbb{N}$ robots $\{\mathbf{p}_1^a, \ldots, \mathbf{p}_n^a\}$, where $\mathbf{p}_i^a = [x_i^a, y_i^a, \theta_i^a]$ for $i \in \{1, \ldots, n\}$, relative to an anchor robot a, given $m \in \mathbb{N}$ noisy measurements of relative poses between robots. There is a single anchor node $a \in \mathcal{V}$ which is placed at the pose $\mathbf{p}_a^a = \mathbf{0}_{3\times 1}$. By convention, we let the anchor be the first node, $a = 1$, and denote $\mathcal{V}^a = \mathcal{V} \setminus \{a\}$ the set of non-anchor nodes. The robots measure the planar pose (position and orientation) of nearby robots expressed on their own reference frame. In the previous section, we assumed the measurements were noise-free. Here, we instead consider that they are corrupted with noises.

Each edge $e = (i, j) \in \mathcal{E}$ in the relative measurements graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has associated noisy measurements of the orientation $\mathbf{z}_e^\theta$ and the position $\mathbf{z}_e^{xy}$ of robot j relative to robot i, with associated covariance matrices $\Sigma_{\mathbf{z}_e^\theta}$ and $\Sigma_{\mathbf{z}_e^{xy}}$. We let $\mathbf{z}_\theta \in \mathbb{R}^m$, $\mathbf{z}_{xy} \in \mathbb{R}^{2m}$, $\Sigma_{\mathbf{z}_\theta} \in \mathbb{R}^{m\times m}$ and $\Sigma_{\mathbf{z}_{xy}} \in \mathbb{R}^{2m\times 2m}$ contain information of the m measurements,

$$\mathbf{z}_\theta = (\mathbf{z}_1^\theta, \ldots, \mathbf{z}_m^\theta)^T, \qquad \mathbf{z}_{xy} = ((\mathbf{z}_1^{xy})^T, \ldots, (\mathbf{z}_m^{xy})^T)^T,$$
$$\Sigma_{\mathbf{z}_\theta} = \mathrm{Diag}(\Sigma_{\mathbf{z}_1^\theta}, \ldots \Sigma_{\mathbf{z}_m^\theta}), \qquad \Sigma_{\mathbf{z}_{xy}} = \mathrm{blkDiag}(\Sigma_{\mathbf{z}_1^{xy}}, \ldots \Sigma_{\mathbf{z}_m^{xy}}).$$

We assume that the measurements are independent since they were acquired individually by the robots. Thus, the goal is that each robot $i \in \mathcal{V}$ estimates its pose $\hat{\mathbf{p}}_i^a$ relative to this anchor.

This problem is solved by using a three-phases strategy:

Phase 1: Compute a suboptimal estimate of the robot orientations $\tilde{\theta}_{\mathcal{V}}^a \in \mathbb{R}^n$ relative to the anchor a for all the robots in $\mathcal{V}$;

Phase 2: Express the position measurements $\mathbf{z}_{xy}$ of the robots in terms of the previously computed orientations;

Phase 3: Compute the estimated poses of the robots $\hat{\mathbf{p}}_{\mathcal{V}}^a = ((\hat{\mathbf{x}}_{\mathcal{V}}^a)^T, (\hat{\theta}_{\mathcal{V}}^a)^T)^T$.

During the rest of the section, we analyze each of these phases and present a distributed implementation.

6.4.1 Centralized algorithm

Phase 1

During this first phase, an initial estimate of the robot orientations $\tilde{\theta}_{\mathcal{V}^a} \in \mathbb{R}^{n-1}$ relative to the anchor a is obtained. This estimate is computed based exclusively on the orientation

measurements $\mathbf{z}_\theta \in \mathbb{R}^m$ with covariance $\Sigma_{\mathbf{z}_\theta} \in \mathbb{R}^{m\times m}$. When the orientations measurements are considered alone and they belong to $\pm\frac{\pi}{2}$, the estimation problem becomes linear, and the estimated orientations are given by the Weighted Least Squares,

$$\tilde{\theta}^a_{\mathcal{V}^a} = \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} \mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} \mathbf{z}_\theta, \qquad \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} = \left(\mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} (\mathcal{A}^a)^T\right)^{-1}, \tag{6.17}$$

where $\mathcal{A}^a \in \{0,1,-1\}^{n-1\times m}$ is the result of deleting the row associated to the anchor a from the incidence matrix $\mathcal{A}$ of the measurement graph in eq. (6.1). Recall that the orientation of the anchor is set to zero, $\tilde{\theta}^a_i = 0$ for $i = a$. We let $\tilde{\theta}^a_{\mathcal{V}} \in \mathbb{R}^n$ and $\Sigma_{\tilde{\theta}^a_{\mathcal{V}}} \mathbb{R}^{n\times n}$ contain the orientation of all the robots in $\mathcal{V}$, including the anchor a,

$$\tilde{\theta}^a_{\mathcal{V}} = (0, (\tilde{\theta}^a_{\mathcal{V}^a})^T)^T, \qquad \Sigma_{\tilde{\theta}^a_{\mathcal{V}}} = \mathrm{Diag}(0, \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}}). \tag{6.18}$$

Phase 2

Each relative position measurement $\mathbf{z}^\theta_e$ associated to the edge $e = (i,j)$, was originally expressed in the local coordinates of robot i. During the second phase, these measurements are transformed into a common orientation using the previously computed $\tilde{\theta}^a_{\mathcal{V}}$.

For each edge $e = (i,j) \in \mathcal{E}$ we let $\tilde{R}_e \in \mathbb{R}^{2\times 2}$ and $\tilde{S}_e \in \mathbb{R}^{2\times 2}$ be the following matrices associated to the orientation $\tilde{\theta}_i$ of robot i,

$$\tilde{R}_e = \mathcal{R}(\tilde{\theta}^a_i) = \begin{bmatrix} \cos\tilde{\theta}^a_i & -\sin\tilde{\theta}^a_i \\ \sin\tilde{\theta}^a_i & \cos\tilde{\theta}^a_i \end{bmatrix}, \qquad \tilde{S}_e = \mathcal{S}(\tilde{\theta}^a_i) = \begin{bmatrix} -\sin\tilde{\theta}^a_i & \cos\tilde{\theta}^a_i \\ -\cos\tilde{\theta}^a_i & -\sin\tilde{\theta}^a_i \end{bmatrix}, \tag{6.19}$$

and let the block diagonal matrix $\tilde{R} \in \mathbb{R}^{2m\times 2m}$ compile information from the m edges,

$$\tilde{R} = \mathcal{R}(\tilde{\theta}^a_{\mathcal{V}}) = \mathrm{blkDiag}(\tilde{R}_1, \ldots, \tilde{R}_m). \tag{6.20}$$

The updated pose measurements in the global coordinates $\mathbf{w} \in \mathbb{R}^{2m+(n-1)}$ and their associated covariance $\Sigma_{\mathbf{w}}$ are

$$\begin{aligned} \mathbf{w} &= \begin{bmatrix} \tilde{\mathbf{z}}_{xy} \\ \tilde{\theta}_{\mathcal{V}^a} \end{bmatrix} = \begin{bmatrix} \tilde{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{z}_{xy} \\ \tilde{\theta}_{\mathcal{V}^a} \end{bmatrix}, \\ \Sigma_{\mathbf{w}} &= \begin{bmatrix} K & J \\ \mathbf{0} & \mathbf{I}_{n-1} \end{bmatrix} \begin{bmatrix} \Sigma_{\mathbf{z}_{xy}} & \mathbf{0} \\ \mathbf{0} & \Sigma_{\tilde{\theta}_{\mathcal{V}^a}} \end{bmatrix} \begin{bmatrix} K^T & \mathbf{0} \\ J^T & \mathbf{I}_{n-1} \end{bmatrix}, \end{aligned} \tag{6.21}$$

where $K \in \mathbb{R}^{2m\times 2m}$ and $J \in \mathbb{R}^{2m\times(n-1)}$ are the jacobians of the transformation with respect to respectively, $\mathbf{z}_{xy}$ and $\hat{\theta}$,

$$K = \tilde{R}, \text{ and } \quad J_{e,i} = \tilde{S}_e\, \mathbf{z}^{xy}_e \text{ if } e = (i,j) \text{ for some } j, \text{ and } J_{e,i} = \mathbf{0}_{2\times 1} \text{ otherwise.} \tag{6.22}$$

Phase 3

During the last phase, the positions of the robots $\hat{\mathbf{x}}^a_{\mathcal{V}^a} \in \mathbb{R}^{2(n-1)}$ relative to the anchor node a are computed, and an improved version $\hat{\theta}^a_{\mathcal{V}^a} \in \mathbb{R}^{n-1}$ of the previous orientations

$\tilde{\theta}^a_{\mathcal{V}^a}$ is obtained. Let $\hat{\mathbf{p}}^a_{\mathcal{V}^a} \in \mathbb{R}^{3(n-1)}$ contain both the positions and orientations of the non-anchor robots,

$$\hat{\mathbf{p}}^a_{\mathcal{V}^a} = \begin{bmatrix} \hat{\mathbf{x}}^a_{\mathcal{V}^a} \\ \hat{\theta}^a_{\mathcal{V}^a} \end{bmatrix} = \Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}} B \Sigma_{\mathbf{w}}^{-1} \mathbf{w}, \qquad \Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}} = \left(B \Sigma_{\mathbf{w}}^{-1} B^T\right)^{-1}, \tag{6.23}$$

where $B = \text{blkDiag}\left((\mathcal{A}^a \otimes \mathbf{I}_2), \mathbf{I}_{n-1}\right)$, and $\Sigma_{\mathbf{w}}$ and $\mathbf{w}$ are given by (6.21). The estimated poses $\hat{\mathbf{p}}^a_{\mathcal{V}} \in \mathbb{R}^{3n}$ of all the robots in $\mathcal{V}$, including the anchor a, are given by

$$\hat{\mathbf{p}}^a_{\mathcal{V}} = (\mathbf{0}^T_{3\times 1}, (\hat{\mathbf{p}}^a_{\mathcal{V}^a})^T)^T, \qquad \Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}}} = \text{blkDiag}(\mathbf{0}_{3\times 3}, \Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}}}). \tag{6.24}$$

Algorithm

Considering the three phases together, the estimated positions $\hat{\mathbf{x}}^a_{\mathcal{V}^a}$ and orientations $\hat{\theta}^a_{\mathcal{V}^a}$ of the non-anchor robots are

$$\begin{aligned} \hat{\mathbf{x}}^a_{\mathcal{V}^a} &= L^{-1}(\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}} \left(\mathbf{I}_{2m} + J \Sigma_{\hat{\theta}^a_{\mathcal{V}^a}} J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} E\right) \tilde{R}\ \mathbf{z}_{xy}, \\ \hat{\theta}^a_{\mathcal{V}^a} &= (\mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} (\mathcal{A}^a)^T)^{-1} \mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} \mathbf{z}_\theta + \Sigma_{\hat{\theta}^a_{\mathcal{V}^a}} J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} E\ \tilde{R}\ \mathbf{z}_{xy}, \text{ where} \end{aligned} \tag{6.25}$$

$$\begin{aligned} \Upsilon_{\tilde{\mathbf{z}}_{xy}} &= (\tilde{R} \Sigma_{\mathbf{z}_{xy}} \tilde{R}^T)^{-1}, & E &= (\mathcal{A}^a \otimes \mathbf{I}_2)^T L^{-1} (\mathcal{A}^a \otimes \mathbf{I}_2) \Upsilon_{\tilde{\mathbf{z}}_{xy}} - \mathbf{I}_{2m}, \\ \Sigma_{\hat{\theta}^a_{\mathcal{V}^a}} &= ((\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}})^{-1} - J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} E J)^{-1}, & L &= (\mathcal{A}^a \otimes \mathbf{I}_2) \Upsilon_{\tilde{\mathbf{z}}_{xy}} (\mathcal{A}^a \otimes \mathbf{I}_2)^T, \end{aligned} \tag{6.26}$$

and $\hat{\mathbf{p}}^a_{\mathcal{V}}$ is obtained from the previous expressions as in eq. (6.24). A full development of these expressions can be found in the following section. From (6.25), it can be seen that the computation of $\hat{\mathbf{x}}^a_{\mathcal{V}}$ and $\hat{\theta}^a_{\mathcal{V}}$ involves matrix inversions and other operations that require the knowledge of the whole system. Although a priori the proposed strategy would require a centralized implementation, in the next sections we show a proposal to carry out the computations in a distributed way.

Development of the expressions of the localization algorithm

During the first phase, $\tilde{\theta}^a_{\mathcal{V}^a}$ and its covariance $\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}}$ are

$$\tilde{\theta}^a_{\mathcal{V}^a} = \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} \mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} \mathbf{z}_\theta, \qquad \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} = (\mathcal{A}^a \Sigma^{-1}_{\mathbf{z}_\theta} (\mathcal{A}^a)^T)^{-1}. \tag{6.27}$$

In the second phase, the updated measurements $\mathbf{w}$ and a first order propagation of the uncertainty $\Sigma_{\mathbf{w}}$ are

$$\mathbf{w} = \begin{bmatrix} \tilde{\mathbf{z}}_{xy} \\ \tilde{\theta}^a_{\mathcal{V}^a} \end{bmatrix} = \begin{bmatrix} \tilde{R}\mathbf{z}_{xy} \\ \tilde{\theta}^a_{\mathcal{V}^a} \end{bmatrix}, \qquad \Sigma_{\mathbf{w}} = \begin{bmatrix} \tilde{R}\Sigma_{\mathbf{z}_{xy}}\tilde{R}^T + J\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} J^T & J\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} \\ \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} J^T & \Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}} \end{bmatrix}^T. \tag{6.28}$$

The estimates in the third phase are the solution of the linear system

$$\hat{\mathbf{p}}^a_{\mathcal{V}^a} = \begin{bmatrix} \hat{\mathbf{x}}^a_{\mathcal{V}^a} \\ \hat{\theta}^a_{\mathcal{V}^a} \end{bmatrix} = (B \Sigma_{\mathbf{w}}^{-1} B^T)^{-1} B \Sigma_{\mathbf{w}}^{-1} \mathbf{w}. \tag{6.29}$$

To write in explicit form $\hat{\mathbf{x}}^a_{\mathcal{V}^a}$ and $\hat{\theta}^a_{\mathcal{V}^a}$ we first compute the information matrix $\Upsilon_{\mathbf{w}} = \Sigma_{\mathbf{w}}^{-1}$,

$$\Upsilon_{\mathbf{w}} = \begin{bmatrix} \Upsilon_{\tilde{\mathbf{z}}_{xy}} & -\Upsilon_{\tilde{\mathbf{z}}_{xy}} J \\ -J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} & \Sigma^{-1}_{\tilde{\theta}^a_{\mathcal{V}^a}} + J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} J \end{bmatrix}, \tag{6.30}$$

where $\Upsilon_{\tilde{\mathbf{z}}_{xy}}$ is as in eq. (6.26), $\Upsilon_{\tilde{\mathbf{z}}_{xy}} = (\tilde{R}\Sigma_{\mathbf{z}_{xy}}\tilde{R}^T)^{-1}$, and where we have used the following blockwise inversion relations $\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$, with

$$\begin{aligned} E &= A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1} = \left(A - BD^{-1}C\right)^{-1}, \\ F &= -A^{-1}B\left(D - CA^{-1}B\right)^{-1} = -\left(A - BD^{-1}C\right)^{-1}BD^{-1}, \\ G &= -\left(D - CA^{-1}B\right)^{-1}CA^{-1} = -D^{-1}C\left(A - BD^{-1}C\right)^{-1}, \\ H &= \left(D - CA^{-1}B\right)^{-1} = D^{-1} + D^{-1}C\left(A - BD^{-1}C\right)^{-1}BD^{-1}. \end{aligned} \tag{6.31}$$

The information matrix $\Upsilon_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}} = (B\Sigma_{\mathbf{w}}^{-1}B^T)$ and its inverse $\Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}}$ are

$$\begin{aligned} \Upsilon_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}} &= \begin{bmatrix} (\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}}(\mathcal{A}^a \otimes \mathbf{I}_2)^T & -(\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}} J \\ -J^T\Upsilon_{\tilde{\mathbf{z}}_{xy}}(\mathcal{A}^a \otimes \mathbf{I}_2)^T & \Sigma^{-1}_{\tilde{\theta}_{\mathcal{V}^a}} + J^T\Upsilon_{\tilde{\mathbf{z}}_{xy}}J \end{bmatrix}, \\ \Sigma_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}} &= \begin{bmatrix} \Sigma_{\hat{\mathbf{x}}} & \Sigma_{\hat{\mathbf{x}},\hat{\theta}} \\ \Sigma^T_{\hat{\mathbf{x}},\hat{\theta}} & \Sigma_{\hat{\theta}} \end{bmatrix}, \text{ with} \end{aligned} \tag{6.32}$$

$$\begin{aligned} \Sigma_{\hat{\theta}} &= ((\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}})^{-1} - J^T\Upsilon_{\tilde{\mathbf{z}}_{xy}}EJ)^{-1}, \\ \Sigma_{\hat{\mathbf{x}}} &= L^{-1} + L^{-1}(\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}}J\Sigma_{\hat{\theta}}J^T\Upsilon_{\tilde{\mathbf{z}}_{xy}}(\mathcal{A}^a \otimes \mathbf{I}_2)^T L^{-1}, \\ \Sigma_{\hat{\mathbf{x}},\hat{\theta}} &= L^{-1}(\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}}J\Sigma_{\hat{\theta}}, \\ E &= (\mathcal{A}^a \otimes \mathbf{I}_2)^T L^{-1}(\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}} - \mathbf{I}, \\ L &= (\mathcal{A}^a \otimes \mathbf{I}_2)\Upsilon_{\tilde{\mathbf{z}}_{xy}}(\mathcal{A}^a \otimes \mathbf{I}_2)^T. \end{aligned} \tag{6.33}$$

6.4.2 Distributed algorithm

Phase 1

The initial orientation $\tilde{\theta}^a_{\mathcal{V}}$ in the first phase of the algorithm can be computed in a distributed fashion using the following Jacobi algorithm [18]. Let each robot $i \in \mathcal{V}$ maintain a variable $\tilde{\theta}^a_i(t) \in \mathbb{R}$. The anchor $i = a$ keeps its variable equal to zero for all time steps $t \in \mathbb{N}$,

$$\tilde{\theta}^a_i(0) = 0, \qquad \tilde{\theta}^a_i(t+1) = \tilde{\theta}^a_i(t), \qquad \text{for } i = a. \tag{6.34}$$

Each non-anchor robot $i \in \mathcal{V}^a$ initializes its variable at $t = 0$ with any value $\tilde{\theta}^a_i(0)$, and updates it at each time step $t \in \mathbb{N}$ by

$$\tilde{\theta}^a_i(t+1) = C_i^{-1}c_i + C_i^{-1}\sum_{e=(i,j)\in\mathcal{E}}(\Sigma_{\mathbf{z}^\theta_e})^{-1}\tilde{\theta}^a_j(t) + C_i^{-1}\sum_{e=(j,i)\in\mathcal{E}}(\Sigma_{\mathbf{z}^\theta_e})^{-1}\tilde{\theta}^a_j(t), \tag{6.35}$$

where

$$c_i = -\sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^\theta})^{-1}\mathbf{z}_e^\theta + \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^\theta})^{-1}\mathbf{z}_e^\theta,$$
$$C_i = \sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^\theta})^{-1} + \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^\theta})^{-1}. \tag{6.36}$$

The previous expressions are the Jacobi iterations associated to (6.17). Let $\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}$ and $\eta_{\tilde{\theta}^a_{\mathcal{V}^a}}$ be respectively the information matrix and vector of $\tilde{\theta}^a_{\mathcal{V}^a}$,

$$\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}} = (\Sigma_{\tilde{\theta}^a_{\mathcal{V}^a}})^{-1} = \mathcal{A}^a\Sigma^{-1}_{\mathbf{z}_\theta}(\mathcal{A}^a)^T, \qquad \eta_{\tilde{\theta}^a_{\mathcal{V}^a}} = \mathcal{A}^a\Sigma^{-1}_{\mathbf{z}_\theta}\mathbf{z}_\theta. \tag{6.37}$$

Let C contain the elements in the diagonal of $\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}$,

$$C = \mathrm{Diag}([\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}]_{2,2}, \ldots, [\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}]_{n,n}),$$

and D be $D = C - \Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}$. The first equation in (6.17) can be rewritten as

$$\Upsilon_{\tilde{\theta}^a_{\mathcal{V}^a}}\tilde{\theta}^a_{\mathcal{V}^a} = \eta_{\tilde{\theta}^a_{\mathcal{V}^a}}, \qquad \tilde{\theta}^a_{\mathcal{V}^a} = C^{-1}D\tilde{\theta}^a_{\mathcal{V}^a} + C^{-1}\eta_{\tilde{\theta}^a_{\mathcal{V}^a}}. \tag{6.38}$$

From here, we can write

$$\tilde{\theta}^a_{\mathcal{V}^a}(t+1) = C^{-1}D\tilde{\theta}^a_{\mathcal{V}^a}(t) + C^{-1}\eta_{\tilde{\theta}^a_{\mathcal{V}^a}}, \tag{6.39}$$

initialized at $t = 0$ with $\tilde{\theta}^a_{\mathcal{V}^a}(0)$. By operating with $\mathcal{A}^a\Sigma^{-1}_{\mathbf{z}_\theta}\mathbf{z}_\theta$ and $\mathcal{A}^a\Sigma^{-1}_{\mathbf{z}_\theta}(\mathcal{A}^a)^T$, it can be seen that (6.35) is the $i-th$ row of (6.39). The system (6.39) converges to $\tilde{\theta}^a_{\mathcal{V}^a}$ in eq. (6.17), and equivalently each $\tilde{\theta}^a_i(t)$ in (6.35) converges to $\tilde{\theta}^a_i$ for $i \in \mathcal{V}^a$, if the spectral radius of $C^{-1}D$ is less than 1,

$$\rho(C^{-1}D) < 1, \tag{6.40}$$

and the anchor variable, $\tilde{\theta}^a_i(t)$ with $i = a$, remains equal to 0 for all the iterations t. The value $\rho(C^{-1}D)$ gives the convergence speed of the system, converging faster for $\rho(C^{-1}D)$ closer to 0. Recalling that $\Sigma_{\mathbf{z}_\theta}$ is a diagonal matrix, then each variable $\tilde{\theta}^a_i(t)$ asymptotically converges to the i-th entry $\tilde{\theta}^a_i$ of the vector $\tilde{\theta}^a_{\mathcal{V}^a}$ in (6.17) [17,18] that would be computed by a centralized system.

Observe that the computations are fully distributed and they exclusively rely on local information. The constants C_i and c_i are computed by each robot $i \in \mathcal{V}^a$ using exclusively the measurements $\mathbf{z}_e^\theta$ and covariances $\Sigma_{\mathbf{z}_e^\theta}$ of its incoming $e = (j,i)$ or outgoing edges $e = (i,j)$. Also the variables $\tilde{\theta}^a_j(t)$ used to update its own $\tilde{\theta}^a_i(t+1)$ belong to neighboring robots $j \in \mathcal{N}_i$.

Phase 2

Let us assume that the robots have executed $t_{\max}$ iterations of the previous algorithm, and let $\bar{\theta}_i^a$ be their orientation at iteration $t_{\max}$, $\bar{\theta}_i^a = \tilde{\theta}_i^a(t_{\max})$. Then, the second phase of the algorithm is executed to transform the locally expressed measurements $\mathbf{z}_{xy}$ into the measurements expressed in the reference frame of the anchor node $\tilde{\mathbf{z}}_{xy}$. As previously stated, the estimated orientations $\bar{\theta}_i^a$ do not change during this phase (6.21). Let $\bar{R} = \mathcal{R}(\bar{\theta}_{\mathcal{V}^a}^a)$ be defined by using the orientations $\bar{\theta}_i^a$ instead of $\tilde{\theta}_i^a$ in (6.20). Since the matrix $\bar{R}$ is block diagonal, each robot $i \in \mathcal{V}$ can locally transform its own local measurements,

$$\bar{\mathbf{z}}_e^{xy} = \bar{R}_e \mathbf{z}_e^{xy}, \text{ for all } e = (i,j) \in \mathcal{E}. \tag{6.41}$$

Since the robots use $\bar{\theta}$ instead of $\tilde{\theta}$, also the updated measurements obtained during the second phase are $\bar{\mathbf{z}}_{xy}$ instead of $\tilde{\mathbf{z}}_{xy}$. This second phase is local and it is executed in a single iteration.

Phase 3

In order to obtain the final estimate $\hat{\mathbf{p}}_{\mathcal{V}^a}^a$, the third step of the algorithm (6.23) apparently requires the knowledge of the covariance matrix $\Sigma_{\mathbf{w}}$, which at the same time, requires the knowledge of $\Sigma_{\tilde{\theta}_{\mathcal{V}^a}^a}$. However, a distributed computation of these matrices cannot be carried out in an efficient way. Here we present a distributed algorithm for computing $\hat{\mathbf{p}}_{\mathcal{V}}^a$.

Let each robot $i \in \mathcal{V}$ maintain a variable $\hat{\mathbf{p}}_i^a(t) \in \mathbb{R}^3$, composed of its estimated position $\hat{\mathbf{x}}_i^a(t) \in \mathbb{R}^2$ and orientation $\hat{\theta}_i^a(t) \in \mathbb{R}$, and let $\hat{\mathbf{p}}_{\mathcal{V}}^a(t)$ be the result of putting together the $\mathbf{p}_i^a(t)$ variables for all $i \in \mathcal{V}$. The anchor robot keeps its variable equal to zero for all the iterations,

$$\hat{\mathbf{p}}_i^a(0) = \mathbf{0}_{3\times 1}, \qquad \hat{\mathbf{p}}_i^a(t+1) = \hat{\mathbf{p}}_i^a(t), \qquad \text{for } i = a. \tag{6.42}$$

Each non-anchor robot $i \in \mathcal{V}^a$ initializes its variable at $t = 0$ with any value $\hat{\mathbf{p}}_i^a(0)$ and updates $\hat{\mathbf{p}}_i(t)$ at each time step $t \in \mathbb{N}$ by

$$\hat{\mathbf{p}}_i^a(t+1) = \begin{bmatrix} \hat{\mathbf{x}}_i^a(t+1) \\ \hat{\theta}_i^a(t+1) \end{bmatrix} = M_i^{-1} \left(\mathbf{f}_i(\hat{\mathbf{p}}_{\mathcal{V}}^a(t)) + \mathbf{m}_i \right), \tag{6.43}$$

where

$$M_i = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}, \ \mathbf{f}_i(\mathbf{p}_{\mathcal{V}}^a(t)) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \ \mathbf{m}_i = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}. \tag{6.44}$$

Let $\Upsilon_{\tilde{\mathbf{z}}_e^{xy}}$ be the block within the matrix $\Upsilon_{\tilde{\mathbf{z}}_{xy}}$ in (6.26) associated to an edge $e = (i,j) \in \mathcal{E}$,

$$\Upsilon_{\tilde{\mathbf{z}}_e^{xy}} = \tilde{R}_e (\Sigma_{\mathbf{z}_e^{xy}})^{-1} (\tilde{R}_e)^T. \tag{6.45}$$

The elements within M_i are

$$\begin{aligned}
M_1 &= \sum_{e=(i,j)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} + \sum_{e=(j,i)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}}, \\
M_2 &= \sum_{e=(i,j)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{S}_e \ \mathbf{z}_e^{xy}, \\
M_3 &= \sum_{e=(i,j)\in\mathcal{E}} (\mathbf{z}_e^{xy})^T (\tilde{S}_e)^T \Upsilon_{\tilde{\mathbf{z}}_e^{xy}}, \\
M_4 &= \sum_{e=(i,j)\in\mathcal{E}} (\mathbf{z}_e^{xy})^T (\tilde{S}_e)^T \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{S}_e \ \mathbf{z}_e^{xy} + \sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{xy}})^{-1} + \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{xy}})^{-1}.
\end{aligned} \tag{6.46}$$

The elements within $\mathbf{f}_i(\hat{\mathbf{p}}_{\mathcal{V}}^a(t))$, which is the term depending on the previous estimates $\hat{\mathbf{p}}_{\mathcal{V}}^a(t) = (\hat{\mathbf{x}}_{\mathcal{V}}^a(t)^T, \hat{\theta(t)}_{\mathcal{V}}^a)^T$, are

$$\begin{aligned}
f_1 &= \sum_{e=(i,j)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \ \hat{\mathbf{x}}_j^a(t) + \sum_{e=(j,i)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \ \hat{\mathbf{x}}_j^a(t) + \sum_{e=(j,i)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \ \tilde{S}_e \ \mathbf{z}_e^{xy} \ \hat{\theta}_j^a(t), \\
f_2 &= \sum_{e=(i,j)\in\mathcal{E}} (\mathbf{z}_e^{xy})^T (\tilde{S}_e)^T \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \hat{\mathbf{x}}_j^a(t) - \sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{xy}})^{-1} \hat{\theta}_j^a(t) - \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{xy}})^{-1} \hat{\theta}_j^a(t).
\end{aligned} \tag{6.47}$$

Finally, the terms within $\mathbf{m}_i$ are

$$\begin{aligned}
m_1 &= -\sum_{e=(i,j)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{\mathbf{z}}_e^{xy} + \sum_{e=(j,i)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{\mathbf{z}}_e^{xy} + \sum_{e=(i,j)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{S}_e \mathbf{z}_e^{xy} \tilde{\theta}_i^a - \sum_{e=(j,i)\in\mathcal{E}} \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{S}_e \mathbf{z}_e^{xy} \tilde{\theta}_j^a, \\
m_2 &= -\sum_{e=(i,j)\in\mathcal{E}} (\mathbf{z}_e^{xy})^T (\tilde{S}_e)^T \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{\mathbf{z}}_e^{xy} + \sum_{e=(i,j)\in\mathcal{E}} (\mathbf{z}_e^{xy})^T (\tilde{S}_e)^T \Upsilon_{\tilde{\mathbf{z}}_e^{xy}} \tilde{S}_e \mathbf{z}_e^{xy} \tilde{\theta}_i^a \\
&\quad - \sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{\theta}})^{-1} \tilde{\theta}_j^a - \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{\theta}})^{-1} \tilde{\theta}_j^a + \sum_{e=(i,j)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{\theta}})^{-1} \tilde{\theta}_i^a + \sum_{e=(j,i)\in\mathcal{E}} (\Sigma_{\mathbf{z}_e^{\theta}})^{-1} \tilde{\theta}_i^a.
\end{aligned} \tag{6.48}$$

Theorem 6.4.1. *The estimates $\hat{\mathbf{p}}_i(t)$ computed by each robot $i \in \mathcal{V}$ by the distributed algorithm* (6.42)-(6.43) *converge to $\hat{\mathbf{p}}_i^a = [(\hat{\mathbf{x}}_i^a)^T \hat{\theta}_i^a]^T$ for connected measurement graphs $\mathcal{G}$ with ring or string structure.*

Proof. For the anchor $i = a$, it is true since $\hat{\mathbf{p}}_i^a(t) = \mathbf{0}$ for all the time steps. Now we focus on the non-anchor nodes in $\mathcal{V}^a$. First of all, we show that $\hat{\mathbf{p}}_i^a$ is an equilibrium point of the algorithm (6.43) for all $i \in \mathcal{V}^a$. Let $\Upsilon_{\hat{\mathbf{p}}_{\mathcal{V}^a}^a}$ be the information matrix associated to $\hat{\mathbf{p}}_{\mathcal{V}^a}^a$, i.e., $\Upsilon_{\hat{\mathbf{p}}_{\mathcal{V}^a}^a} = (\Sigma_{\hat{\mathbf{p}}_{\mathcal{V}^a}^a})^{-1}$,

$$\Upsilon_{\hat{\mathbf{p}}_{\mathcal{V}^a}^a} = \begin{bmatrix} L & -\mathcal{A}^a \Upsilon_{\tilde{\mathbf{z}}_{xy}} J \\ -J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} (\mathcal{A}^a \otimes \mathbf{I}_2)^T & \mathcal{A}^a \Sigma_{\mathbf{z}_\theta}^{-1} (\mathcal{A}^a)^T + J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} J \end{bmatrix}, \tag{6.49}$$

where L and $\Upsilon_{\tilde{\mathbf{z}}_{xy}}$ are given by (6.26). Analyzing the term $B\Sigma_{\mathbf{w}}^{-1}$ in (6.23), it can be seen that it is

$$B\Sigma_{\mathbf{w}}^{-1} = \begin{bmatrix} (\mathcal{A}^a \otimes \mathbf{I}_2) \Upsilon_{\tilde{\mathbf{z}}_{xy}} & -(\mathcal{A}^a \otimes \mathbf{I}_2) \Upsilon_{\tilde{\mathbf{z}}_{xy}} J \\ -J^T Y_{\tilde{\mathbf{z}}_{xy}} & \mathcal{A}^a \Upsilon_{\mathbf{z}_\theta}^{-1} (\mathcal{A}^a \otimes \mathbf{I}_2)^T + J^T \Upsilon_{\tilde{\mathbf{z}}_{xy}} J \end{bmatrix}. \tag{6.50}$$

If we express the third phase in the following way

$$\Upsilon_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}}\ \hat{\mathbf{p}}^a_{\mathcal{V}^a} = B\ \Sigma^{-1}_{\mathbf{w}}\ \mathbf{w}, \tag{6.51}$$

and then we consider the rows associated to robot i, we get

$$\hat{\mathbf{p}}^a_i = \begin{bmatrix} \hat{\mathbf{x}}^a_i \\ \hat{\theta}^a_i \end{bmatrix} = M_i^{-1}\left(\mathbf{f}_i(\hat{\mathbf{p}}^a_{\mathcal{V}}) + \mathbf{m}_i\right), \tag{6.52}$$

with M_i, $\mathbf{f}_i(\mathbf{p}^a_{\mathcal{V}}(t))$ and $\mathbf{m}_i$ as in (6.44)-(6.48).

Now we prove that the system is convergent. Let $M = \text{blkDiag}(M_2, \ldots, M_n)$ and $\hat{\mathbf{q}}^a_{\mathcal{V}^a}$ be a permutation of $\hat{\mathbf{p}}^a_{\mathcal{V}^a}$ so that the estimates of each robot appear together, $\hat{\mathbf{q}}^a_{\mathcal{V}^a} = \left[(\hat{\mathbf{x}}^a_2)^T\hat{\theta}^a_2, \ldots, (\hat{\mathbf{x}}^a_n)^T\hat{\theta}^a_n\right]^T$. Equivalently, the permuted version of the information matrix $\Upsilon_{\hat{\mathbf{p}}^a_{\mathcal{V}^a}}$ is $\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}$. The estimates $\hat{\mathbf{p}}^a_i(t)$ computed by each robot $i \in \mathcal{V}^a$ with the distributed algorithm (6.43) converge to $\hat{\mathbf{p}}^a_i = [(\hat{\mathbf{x}}^a_i)^T\hat{\theta}^a_i]^T$ if $\rho(M^{-1}(M - \Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})) < 1$, or equivalently if

$$\rho(\mathbf{I} - M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) < 1. \tag{6.53}$$

Since $\lambda(\mathbf{I} - M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) = 1 - \lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})$, then (6.43) converges if $0 < \lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) < 2$. The first part $0 < \lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})$ can be easily checked taking into account that both M^{-1} and $\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}$ are nonsingular, symmetric, positive definite, and that $\lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) \geq \frac{\lambda_{\min}(M^{-1})}{\lambda_{\max}(\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})}$ [77, Lemma 1]. Since $0 < \frac{\lambda_{\min}(M^{-1})}{\lambda_{\max}(\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})}$, then $0 < \lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})$.

In order to prove the second part, $\lambda(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) < 2$, let us first focus on the structure of the information matrix $\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}$. This matrix has zeros for the elements associated to non neighboring robots, and thus it is compatible with $\text{adj}(\mathcal{G}) \otimes \mathbf{I}_3$, where $\text{adj}(\mathcal{G})$ is the adjacency matrix of the graph, and $\mathbf{I}_3$ is the 3×3 identity matrix. For ring or string graphs, the adjacency matrix can be reordered grouping the elements around the main diagonal resulting in a matrix that has semi bandwidth $s = 1$, i.e.,

$$\text{adj}(\mathcal{G})_{ij} = 0 \text{ for } |i - j| > s.$$

As a consequence, the information matrix $\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}$ has block semi bandwidth $s' = 1$, and as stated by [77, Theorem 1],

$$\lambda_{\max}(M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}}) < 2^{s'} = 2.$$

□

Due to the structure of the information matrices, the third phase of the algorithm can be expressed in terms of local information (6.43)-(6.48) and interactions with neighbors, and thus it can be implemented in a distributed fashion. It is observed that the robots actually use $\bar{\theta}^a_{\mathcal{V}}$ instead of $\tilde{\theta}^a_{\mathcal{V}}$ and as a result, the solution obtained is slightly different from the one in the centralized case. We experimentally analyze the effects of these differences.

6.5 Centroid-based Noisy Position Localization

This section addresses the problem of estimating the positions of the robots from noisy measurements of the relative positions of neighbors. We propose a distributed method for the estimation of the centroid of the network of robots. The localization of all robots in the network is then obtained relative to the estimated centroid. The usual approach to multi-robot localization assumes instead that one anchor robot exists in the network, and the other robots positions are estimated with respect to the anchor. We show that the proposed centroid-based algorithm converges to the optimal solution, and that such a centroid-based representation produces results that are more accurate than anchor-based ones, irrespective of the selected anchor. In previous sections we denoted $\mathbf{p}_i$ the pose of a robot i. Since in this section we exclusively consider robot positions, for clarity we use a different symbol $\mathbf{x}_i$ for the robots variables.

Consider that each robot $i \in \{1, \dots, n\}$ has a $p-$dimensional state $\mathbf{x}_i \in \mathbb{R}^p$ and it observes the states of a subset of the robots relative to its own state, $\mathbf{x}_j - \mathbf{x}_i$. These states can be, for instance, positions in cartesian coordinates, orientations, speeds, accelerations, or current times. Each edge $e = (i, j) \in \mathcal{E}$ in the relative measurements graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents that robot i has a noisy relative measurement $\mathbf{z}_e \in \mathbb{R}^p$ of the state of robot j,

$$\mathbf{z}_e = \mathbf{x}_j - \mathbf{x}_i + \mathbf{v}_e, \tag{6.54}$$

where $\mathbf{v}_e \sim N(\mathbf{0}_{p\times p}, \Sigma_{\mathbf{z}_e})$ is a Gaussian additive noise. We let $\mathbf{z} \in \mathbb{R}^{mp}$ and $\Sigma_{\mathbf{z}} \in \mathbb{R}^{mp\times mp}$ contain the information of the m measurements,

$$\mathbf{z} = (\mathbf{z}_1^T, \dots, \mathbf{z}_m^T)^T, \qquad \Sigma_{\mathbf{z}} = \text{blkDiag}(\Sigma_{\mathbf{z}_1}, \dots, \Sigma_{\mathbf{z}_m}), \tag{6.55}$$

We assume that the measurement graph $\mathcal{G}$ is directed and weakly connected, and that an robot i can exchange data with both its in and out neighbors $\mathcal{N}_i$ so that the associated communication graph is undirected. The estimation from relative measurements problem consists of estimating the states of the n robots from the relative measurements $\mathbf{z}$. Any solution can be determined only up to an additive constant. Conventionally [18] one of the robots $a \in \mathcal{V}$, e.g., the first one $a = 1$, is established as an anchor with state $\hat{\mathbf{x}}_a^a = \mathbf{0}_p$. We call such approaches anchor-based and add the superscript a to their associated variables. The Best Linear Unbiased Estimator of the states $\hat{\mathbf{x}}_{\mathcal{V}^a}^a \in \mathbb{R}^{(n-1)p}$, $\hat{\mathbf{x}}_{\mathcal{V}^a}^a = ((\hat{\mathbf{x}}_2^a)^T, \dots, (\hat{\mathbf{x}}_n^a)^T)^T$, of the non-anchor robots $\mathcal{V}^a = \mathcal{V}\backslash\{a\}$ relative to a are obtained as follows [18],

$$\hat{\mathbf{x}}_{\mathcal{V}^a}^a = \Sigma_{\hat{\mathbf{x}}_{\mathcal{V}^a}^a} (\mathcal{A}^a \otimes \mathbf{I}_p) \Sigma_{\mathbf{z}}^{-1} \mathbf{z}, \qquad \Sigma_{\hat{\mathbf{x}}_{\mathcal{V}^a}^a} = \left((\mathcal{A}^a \otimes \mathbf{I}_p)\Sigma_{\mathbf{z}}^{-1}(\mathcal{A}^a \otimes \mathbf{I}_p)^T\right)^{-1}, \tag{6.56}$$

where $\mathcal{A}^a \in \mathbb{R}^{(n-1)\times m}$ is the incidence matrix of $\mathcal{G}$ as in eq. (6.1), but without the row associated to the anchor a. From now on, both $\hat{\mathbf{x}}_{\mathcal{V}}^a = (\mathbf{0}_p^T, (\hat{\mathbf{x}}_{\mathcal{V}^a}^a)^T)^T$ and $\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^a} = \text{blkDiag}\left(\mathbf{0}_{p\times p}, \Sigma_{\hat{\mathbf{x}}_{\mathcal{V}^a}^a}\right)$, include the estimated state of the anchor a as well.

6.5.1 Distributed estimation relative to an anchor

We are interested in distributed strategies where each robot i iteratively estimates its own state in eq. (6.56) through local interactions with its neighbors $\mathcal{N}_i$. Among the different

existing methods for estimating the states $\hat{\mathbf{x}}^a_{\mathcal{V}}$ relative to an anchor, we use the Jacobi algorithm [18], although other distributed methods such as the Jacobi Overrelaxation [26], or the Overlapping Subgraph Estimator [19] could alternatively be applied. The approach in [119], based on the cycle structure of the graph, could be used as well, although it requires multi-hop communication.

Considering eq. (6.56), it can be seen that computing $\hat{\mathbf{x}}^a_{\mathcal{V}^a}$ is equivalent to finding a solution to the system $\Upsilon\hat{\mathbf{x}}^a_{\mathcal{V}^a} = \eta$, being η and Υ the information vector and matrix associated to $\hat{\mathbf{x}}^a_{\mathcal{V}^a}$ and $\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}^a}}$,

$$\eta = (\mathcal{A}^a \otimes \mathbf{I}_p)\,\Sigma_{\mathbf{z}}^{-1}\mathbf{z}, \qquad \Upsilon = (\mathcal{A}^a \otimes \mathbf{I}_p)\,\Sigma_{\mathbf{z}}^{-1}\,(\mathcal{A}^a \otimes \mathbf{I}_p)^T. \tag{6.57}$$

This can be iteratively solved with the Jacobi method [26], where the variable $\hat{\mathbf{x}}^a_{\mathcal{V}^a}(t) \in \mathbb{R}^{(n-1)p}$ is initialized with an arbitrary value $\hat{\mathbf{x}}^a_{\mathcal{V}^a}(0)$ and it is updated at each step t with the following rule,

$$\hat{\mathbf{x}}^a_{\mathcal{V}^a}(t+1) = D^{-1}N\hat{\mathbf{x}}^a_{\mathcal{V}^a}(t) + D^{-1}\eta, \tag{6.58}$$

being D, N the following decomposition of $\Upsilon = [\Upsilon_{ij}]$:

$$D = \text{blkDiag}(\Upsilon_{22}, \ldots, \Upsilon_{nn}), \qquad N = D - \Upsilon. \tag{6.59}$$

The previous variable $\hat{\mathbf{x}}^a_{\mathcal{V}^a}(t)$ converges to $\hat{\mathbf{x}}^a_{\mathcal{V}^a}$ if the Jacobi matrix $J = D^{-1}N$ has spectral radius less than or equal to one, $\rho(J) = \rho(D^{-1}N) < 1$. The interest of the Jacobi method is that it can be executed in a distributed fashion when the information matrix Υ is compatible with the graph (if $j \notin \mathcal{N}_i$ then $\Upsilon_{ij} = \Upsilon_{ji} = \mathbf{0}_{p\times p}$), and when in addition the rows of Υ and of η associated to each robot $i \in \mathcal{V}^a$ only depend on data which is local to robot i. Next, the general anchor-based estimation algorithm [18] base on the Jacobi method is presented. It allows each robot $i \in \mathcal{V}$ to iteratively estimates its own $\hat{\mathbf{x}}^a_i$ within $\hat{\mathbf{x}}^a_{\mathcal{V}^a} = ((\hat{\mathbf{x}}^a_2)^T, \ldots, (\hat{\mathbf{x}}^a_n)^T)^T$ in a distributed fashion.

Algorithm 6.5.1. *Let each robot $i \in \mathcal{V}$ have a variable $\hat{\mathbf{x}}^a_i(t) \in \mathbb{R}^p$ initialized at $t = 0$ with $\hat{\mathbf{x}}^a_i(0) = \mathbf{0}_p$. At each time step t, each robot $i \in \mathcal{V}$ updates $\hat{\mathbf{x}}^a_i(t)$ with*

$$\hat{\mathbf{x}}^a_i(t+1) = \sum_{j\in\mathcal{N}_i} M_i\mathcal{B}_{ij}\hat{\mathbf{x}}^a_j(t) + \sum_{e=(j,i)\in\mathcal{E}} M_i\Sigma^{-1}_{\mathbf{z}_e}\mathbf{z}_e - \sum_{e=(i,j)\in\mathcal{E}} M_i\Sigma^{-1}_{\mathbf{z}_e}\mathbf{z}_e, \tag{6.60}$$

where M_i and $\mathcal{B}_{ij}$ are $p\times p$ matrices with $M_i = \mathbf{0}$ for $i = a$, $M_i = (\sum_{j\in\mathcal{N}_i}\mathcal{B}_{ij})^{-1}$ for $i \neq a$, and

$$\mathcal{B}_{ij} = \begin{cases} \Sigma^{-1}_{\mathbf{z}_e} + \Sigma^{-1}_{\mathbf{z}_{e'}} & \text{if } e = (i,j), e' = (j,i) \in \mathcal{E} \\ \Sigma^{-1}_{\mathbf{z}_e} & \text{if } e = (i,j) \in \mathcal{E}, (j,i) \notin \mathcal{E} \\ \Sigma^{-1}_{\mathbf{z}_e} & \text{if } e = (j,i) \in \mathcal{E}, (i,j) \notin \mathcal{E} \end{cases}. \tag{6.61}$$

The convergence of this estimation algorithm has been proved [18, Theorem 1] for connected measurement graphs with independent relative measurements, under the assumption that either

(i) The covariance matrices of the measurements are exactly diagonal; or

(ii) All measurements have exactly the same covariance matrix.

However, we would like our algorithm to be applicable to a wider case of relative noises, in particular to independent noises, with not necessarily diagonal or equal covariance matrices. Next we use results on block matrices [51], see Section 6.5.4, to prove the convergence of the Jacobi algorithm for this more general case.

Theorem 6.5.2. *Let the measurement graph $\mathcal{G}$ be weakly connected, $\Sigma_{\mathbf{z}_1}, \ldots, \Sigma_{\mathbf{z}_m}$ be the covariance matrices, not necessarily equal or diagonal, associated to m independent $p-$dimensional measurements, and $\Sigma_{\mathbf{z}}$ be their associated block-diagonal covariance matrix as in eq.* (6.55)*. Then, the spectral radius of $D^{-1}N$, with D and N computed as in eqs.* (6.57)-(6.59)*, is less than 1,*

$$\rho(D^{-1}N) < 1. \tag{6.62}$$

Proof. In order to prove (6.62) we use the definitions and results in Section 6.5.4. We first analyze the contents of Υ and show that Υ is of class Z^p_{n-1} according to Definition 6.5.7 in Section 6.5.4. Then, we use Lemma 6.5.8 and Theorem 6.5.9 to show that Υ is of class M^p_{n-1} as in Definition 6.5.7. Finally, we show that $\Upsilon+\Upsilon^T \in M^p_{n-1}$ and use Theorem 6.5.10 to prove (6.62). Note that the subscript $n-1$ used in this proof instead of n comes from the fact that $\Upsilon = [\Upsilon_{ij}]$, with $i, j \in \mathcal{V}^a$ and $|\mathcal{V}^a| = n-1$.

We first analyze the contents of the information matrix Υ given by eq. (6.57). Each block Υ_{ij} of the information matrix Υ is given by

$$\Upsilon_{ij} = \begin{cases} -\mathcal{B}_{ij} & \text{if } j \in \mathcal{N}_i, j \neq i \\ \mathbf{0} & \text{if } j \notin \mathcal{N}_i, j \neq i \end{cases}, \qquad \text{and } \Upsilon_{ii} = \sum_{j \in \mathcal{N}_i} \mathcal{B}_{ij}, \tag{6.63}$$

for $i, j \in \mathcal{V}^a$, where $\mathcal{B}_{ij}$ is given by eq. (6.61). Note that $\mathcal{B}_{ij}$ is symmetric and that $\mathcal{B}_{ij} \succ \mathbf{0}$[1] and thus $-\mathcal{B}_{ij} \prec \mathbf{0}$ and symmetric. Therefore, matrix Υ is of class Z^p_{n-1} according to Definition 6.5.7.

Now we focus on Lemma 6.5.8. We are interested in showing that, given any subset of robots $\mathcal{J} \subset \mathcal{V}^a$, there exists $i \in \mathcal{J}$ such that $\sum_{j \in \mathcal{J}} \Upsilon_{ij} \succ \mathbf{0}$. First we analyze the case $\mathcal{J} = \mathcal{V}^a$. Observe that Υ does not have rows or columns associated to the anchor robot a, i.e., $\Upsilon = [\Upsilon_{ij}]$ with $i, j \in \mathcal{V}^a$. On the other hand, for each robot i that has the anchor a as a neighbor, $a \in \mathcal{N}_i$, the block Υ_{ii} includes $\mathcal{B}_{ia}$. Therefore, $\sum_{j \in \mathcal{V}^a} \Upsilon_{ij} \succeq \mathbf{0}$ for all $i \in \mathcal{V}^a$, specifically

$$\sum_{j \in \mathcal{V}^a} \Upsilon_{ij} = \mathbf{0} \text{ if } a \notin \mathcal{N}_i, \text{ and} \qquad \sum_{j \in \mathcal{V}^a} \Upsilon_{ij} = \mathcal{B}_{ia} \succ \mathbf{0}, \text{ when } a \in \mathcal{N}_i. \tag{6.64}$$

[1] $A \succ B$ ($A \succeq B$) represent that matrix $A - B$ is positive-definite (positive-semidefinite). Equivalently, $\prec$, $\preceq$ are used for negative-definite and negative-semidefinite matrices.

Since $\mathcal{G}$ is connected, $a \in \mathcal{N}_i$ for at least one robot $i \in \mathcal{V}^a$. Now consider a proper subset $\mathcal{J} \subsetneq \mathcal{V}^a$. Note that for each $i \in \mathcal{J} \subsetneq \mathcal{V}^a$,

$$\sum_{j\in\mathcal{J}} \Upsilon_{ij} = \mathbf{0} \text{ if } \mathcal{N}_i \subseteq \mathcal{J}, \text{ and} \qquad \sum_{j\in\mathcal{J}} \Upsilon_{ij} = \sum_{j\in\mathcal{N}_i\setminus\mathcal{J}} \mathcal{B}_{ij} \succ \mathbf{0}, \text{ otherwise.} \tag{6.65}$$

Since $\mathcal{G}$ is connected, given any proper subset $\mathcal{J} \subsetneq \mathcal{V}^a$ of robots, there is always an robot $i \in \mathcal{J}$ that has at least one neighbor outside $\mathcal{J}$ or that has the anchor a as a neighbor, for which $\sum_{j\in\mathcal{J}} \Upsilon_{ij} \succ \mathbf{0}$. Therefore Lemma 6.5.8 holds, and by applying Theorem 6.5.9 taking $u_2, \ldots, u_n = 1$ we conclude that matrix $\Upsilon \in M^p_{n-1}$. Since Υ is symmetric, then $\Upsilon + \Upsilon^T \in M^p_{n-1}$, and by [51, Theorem 4.7] we conclude that $\rho(D^{-1}N) < 1$. □

Corollary 6.5.3. *Let $\mathcal{G}$ be connected, $\Sigma_{\mathbf{z}_1}, \ldots, \Sigma_{\mathbf{z}_m}$ be the covariance matrices associated to m independent $p-$dimensional measurements, and $\Sigma_{\mathbf{z}}$ be their associated block-diagonal covariance matrix as in eq. (6.55). Consider that each robot $i \in \mathcal{V}$ executes the Algorithm 6.5.1 to update its variable $\hat{\mathbf{x}}^a_i(t)$. Then, for all $i \in \mathcal{V}$,*

$$\lim_{t\to\infty} \hat{\mathbf{x}}^a_i(t) = \hat{\mathbf{x}}^a_i, \tag{6.66}$$

converges to the anchor-based centralized solution $\hat{\mathbf{x}}^a_i$ given by eq. (6.56).

6.5.2 Centroid estimation

The accuracy of the estimated states $\hat{\mathbf{x}}^a_{\mathcal{V}}$, $\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}$ in anchor-based approaches depends on the selected anchor a. Instead, we compute the states of the robots $\hat{\mathbf{x}}^{cen}_{\mathcal{V}}$, $\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}$ relative to the *centroid* given by the average of the states,

$$\hat{\mathbf{x}}^{cen}_{\mathcal{V}} = (\mathbf{I} - H_{cen})\,\hat{\mathbf{x}}^a_{\mathcal{V}}, \quad \Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}} = (\mathbf{I} - H_{cen})\,\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}\,(\mathbf{I} - H_{cen})^T, \tag{6.67}$$
$$\text{where } H_{cen} = (\mathbf{1}_n \otimes \mathbf{I}_p)\,(\mathbf{1}_n \otimes \mathbf{I}_p)^T/n.$$

The interest of this representation is that the states of the robots $\hat{\mathbf{x}}^{cen}_{\mathcal{V}}$, $\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}$ with respect to the centroid are the same regardless of the anchor robot, i.e., the centroid solution is unique. Additionally, as the following result shows, it produces more accurate estimates than the ones provided by any anchor selection. We compare the block-traces[2] blkTr of their covariance matrices [20].

Proposition 6.5.4. *The covariance matrices of the centroid-based $\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}$ and anchor-based $\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}$ estimates satisfy, for all anchors $a \in \mathcal{V}$,*

$$\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}\right) \preceq \text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}\right), \qquad \text{Tr}\left(\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}\right) \leq \text{Tr}\left(\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}\right). \tag{6.68}$$

Proof. Let P_{ij} and Q_{ij} be the $p \times p$ blocks of, respectively, the anchor and the centroid-based covariances, $\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}} = [P_{ij}]$, $\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}} = [Q_{ij}]$ with $i, j \in \mathcal{V}$. The block-trace of the anchor-based covariance matrix is

$$\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}^a_{\mathcal{V}}}\right) = \sum_{i=1}^{n} P_{ii}. \tag{6.69}$$

[2]The block-trace of a matrix defined by blocks $P = [P_{ij}]$ with $i, j \in \{1, \ldots, n\}$ is the sum of its diagonal blocks, $\text{blkTr}(P) = \sum_{i=1}^{n} P_{ii}$

Considering eq. (6.67), each block in the main diagonal of the centroid-based $\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}$ covariance matrix is given by

$$Q_{ii} = P_{ii} - \frac{1}{n}\sum_{j=1}^{n}(P_{ij} + P_{ji}) + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{j'=1}^{n} P_{jj'}, \tag{6.70}$$

for $i \in \mathcal{V}$, and thus its block-trace is

$$\begin{aligned}\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}\right) &= \sum_{i=1}^{n} Q_{ii} = \sum_{i=1}^{n} P_{ii} - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij} \\ &= \text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}\right) - (\mathbf{1}_n \otimes \mathbf{I}_p)^T \Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}} (\mathbf{1}_n \otimes \mathbf{I}_p)/n. \end{aligned} \tag{6.71}$$

Since $\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}$ is symmetric and positive-semidefinite, then $(\mathbf{1}_n \otimes \mathbf{I}_p)^T \Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}} (\mathbf{1}_n \otimes \mathbf{I}_p) \succeq \mathbf{0}$, and thus $\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}\right) - \text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}\right) \preceq \mathbf{0}$, as in eq. (6.68). Observe that the trace of the block-trace of a matrix A is equal to its trace, $\text{Tr}(\text{blkTr}(A)) = \text{Tr}(A)$. Since $\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}\right) - \text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}\right) \preceq \mathbf{0}$, the elements in the main diagonal of $\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}\right)$ are smaller than or equal to the ones in the main diagonal of $\text{blkTr}\left(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}\right)$ so that

$$\text{Tr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}) = \text{Tr}(\text{blkTr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}})) \leq \text{Tr}(\text{blkTr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}})) = \text{Tr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}).$$

□

In particular, from eq. (6.71), $\text{Tr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}) - \text{Tr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\text{Tr}(P_{ij})$. Note that the previous result holds when the anchor state $\hat{\mathbf{x}}_a^a$ is set to a general value, not necessarily $\mathbf{0}$. It also holds when there is more than one anchor. Consider that the first k robots are anchors. In this case, matrix $\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}} = [P_{ij}]$ has its blocks $P_{ij} = \mathbf{0}$ for $i, j \in \{1, \ldots, k\}$, and eq. (6.71) gives $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{cen}}) = \text{blkTr}(\Sigma_{\hat{\mathbf{x}}_{\mathcal{V}}^{a}}) - \sum_{i=k+1}^{n}\sum_{j=k+1}^{n} P_{ij}/n$, where $\sum_{i=k+1}^{n}\sum_{j=k+1}^{n} P_{ij}/n \succeq \mathbf{0}$.

We propose an algorithm that allows each robot $i \in \mathcal{V}$ to compute its state $\hat{\mathbf{x}}_i^{cen}$ with respect to the centroid in a distributed fashion, where $\hat{\mathbf{x}}_{\mathcal{V}}^{cen} = ((\hat{\mathbf{x}}_1^{cen})^T, \ldots, (\hat{\mathbf{x}}_n^{cen})^T)^T$ is given in eq. (6.67). These states sum up to zero, $\hat{\mathbf{x}}_1^{cen} + \cdots + \hat{\mathbf{x}}_n^{cen} = \mathbf{0}$, since $(\mathbf{1}_n \otimes \mathbf{I}_p)(\mathbf{I} - H_{cen}) = \mathbf{0}$, and for neighboring robots i and j satisfy $\hat{\mathbf{x}}_i^{cen} = \hat{\mathbf{x}}_j^{cen} - \hat{\mathbf{x}}_j^a + \hat{\mathbf{x}}_i^a$, where $\hat{\mathbf{x}}_{\mathcal{V}}^a = ((\hat{\mathbf{x}}_1^a)^T, \ldots, (\hat{\mathbf{x}}_n^a)^T)^T$. Thus, a straightforward solution would consist of firstly computing the anchor-based states of the robots $\hat{\mathbf{x}}_{\mathcal{V}}^a$, and in a second phase initializing the robots' variables so that they sum up to zero, $\hat{\mathbf{x}}_i^{cen}(0) = \mathbf{0}$, for $i \in \mathcal{V}$, and updating them at each step t with an averaging algorithm that conserves the sum:

$$\hat{\mathbf{x}}_i^{cen}(t+1) = \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j}(\hat{\mathbf{x}}_j^{cen}(t) - \hat{\mathbf{x}}_j^a + \hat{\mathbf{x}}_i^a) \tag{6.72}$$

for $i \in \mathcal{V}$, where $\mathcal{W} = [\mathcal{W}_{i,j}]$ is a doubly stochastic weight matrix such that $\mathcal{W}_{i,j} > 0$ if $(i,j) \in \mathcal{E}$ and $\mathcal{W}_{i,j} = 0$ when $j \notin \mathcal{N}_i$. Besides, $\mathcal{W}_{i,i} \in [\alpha, 1]$, $\mathcal{W}_{i,j} \in \{0\} \cup [\alpha, 1]$ for all $i, j \in \mathcal{V}$, for some $\alpha \in (0, 1]$. More information about averaging algorithms can be found at [27, 113, 152] and in Appendix A. The term $-\hat{\mathbf{x}}_j^a + \hat{\mathbf{x}}_i^a$ is the relative measurement $\mathbf{z}_e$ with $e = (j, i)$ for noise free scenarios, and the optimal or corrected measurement [119] $\hat{\mathbf{z}}_e$ for the noisy case, $\hat{\mathbf{z}} = (\mathcal{A} \otimes \mathbf{I}_p)^T \hat{\mathbf{x}}_{\mathcal{V}}^a$, with $\hat{\mathbf{z}} = ((\hat{\mathbf{z}}_1)^T, \ldots, (\hat{\mathbf{z}}_m)^T)^T$. In what follows we propose an algorithm where, at each iteration t, (6.72) is executed not on the exact $\hat{\mathbf{x}}_i^a, \hat{\mathbf{x}}_j^a$, but on the most recent estimates $\hat{\mathbf{x}}_i^a(t), \hat{\mathbf{x}}_j^a(t)$ obtained with Algorithm 6.5.1.

6.5.3 Distributed centroid estimation algorithm

Now we are ready to present the distributed algorithm for estimating the states of the robots relative to the centroid.

Algorithm 6.5.5. *Let each robot $i \in \mathcal{V}$ have an estimate of its own state relative to the centroid, $\hat{\mathbf{x}}_i^{cen}(t) \in \mathbb{R}^p$, initialized at $t = 0$ with $\hat{\mathbf{x}}_i^{cen}(0) = \mathbf{0}$. At each time step t, each robot $i \in \mathcal{V}$ updates $\hat{\mathbf{x}}_i^{cen}(t)$ with*

$$\hat{\mathbf{x}}_i^{cen}(t+1) = \sum_{j \in \mathcal{N}_i \cup \{i\}} \mathcal{W}_{i,j}(\hat{\mathbf{x}}_j^{cen}(t) + \hat{\mathbf{x}}_i^a(t) - \hat{\mathbf{x}}_j^a(t)), \tag{6.73}$$

where $\hat{\mathbf{x}}_i^a(t), \hat{\mathbf{x}}_j^a(t)$ are the most recent estimates that robots i and j have at iteration t of the variables in Algorithm 6.5.1 and $\mathcal{W}_{i,j}$ are the Metropolis weights as defined in eq. (A.3) *in Appendix A.*

Theorem 6.5.6. *Let all the robots $i \in \mathcal{V}$ execute the Algorithm 6.5.5 and let $\mathcal{G}$ be connected. Then, the estimated states $\hat{\mathbf{x}}_i^{cen}(t)$ at each robot $i \in \mathcal{V}$ asymptotically converge to the state of i relative to the centroid $\hat{\mathbf{x}}_i^{cen}$ given by eq.* (6.67)*,*

$$\lim_{t \to \infty} \hat{\mathbf{x}}_i^{cen}(t) = \hat{\mathbf{x}}_i^{cen}. \tag{6.74}$$

Let $\mathbf{e}_{cen}(t) = \left[(\hat{\mathbf{x}}_1^{cen}(t) - \hat{\mathbf{x}}_1^{cen})^T, \ldots, (\hat{\mathbf{x}}_n^{cen}(t) - \hat{\mathbf{x}}_n^{cen})^T\right]^T$ be the error vector containing the estimation errors of the n robots at iteration t. For fixed communication graphs $\mathcal{G}$, the norm of the error vector after t iterations of Algorithm 6.5.5 satisfies

$$||\mathbf{e}_{cen}(t)||_2 \leq \lambda_{\text{eff}}^t(\mathcal{W})||\mathbf{e}_{cen}(0)||_2 + 2p(n-1)\sigma_J \lambda_{\text{eff}}^t(\mathcal{W}) \sum_{k=1}^{t} \left(\frac{\rho(J)}{\lambda_{\text{eff}}(\mathcal{W})}\right)^k, \tag{6.75}$$

where J is the Jacobi matrix $J = D^{-1}N$, with D and N computed as in eqs. (6.57)-(6.59)*, σ_J is a constant that depends on the initial Jacobi error and on J. $\mathcal{W}$ is the Metropolis weight matrix as defined in eq.* (A.3) *in Appendix A, and $\mathbf{e}_{cen}(0)$ is the initial error at $t = 0$.*

Proof. First of all, we derive the expression for the convergence rate in eq. (6.75). We express Algorithm 6.5.5 in terms of the error vectors associated to the centroid $\mathbf{e}_{cen}(t)$ and the anchor-based $\mathbf{e}_a(t) \in \mathbb{R}^{(n-1)p}$ estimation methods (Algorithms 6.5.1 and 6.5.5), $\mathbf{e}_{cen}(t) = \left[(\hat{\mathbf{x}}_1^{cen}(t))^T, \ldots, (\hat{\mathbf{x}}_n^{cen}(t))^T\right]^T - \hat{\mathbf{x}}_{\mathcal{V}}^{cen}$, with $\hat{\mathbf{x}}_{\mathcal{V}}^{cen} = \left[(\hat{\mathbf{x}}_1^{cen})^T, \ldots, (\hat{\mathbf{x}}_n^{cen})^T\right]^T$ given by eq. (6.67), and $\tilde{\mathbf{e}}_a(t) = \left[(\hat{\mathbf{x}}_2^a(t)^T, \ldots, \hat{\mathbf{x}}_n^a(t)^T\right]^T - \hat{\mathbf{x}}_{\mathcal{V}^a}^a$, with $\hat{\mathbf{x}}_{\mathcal{V}^a}^a = \left[(\hat{\mathbf{x}}_2^a)^T, \ldots, (\hat{\mathbf{x}}_n^a)^T\right]^T$ given by eq. (6.56), where for simplicity we let the robot $i = 1$ be the anchor a. We let $\mathbf{e}_a(t)$ be $(\mathbf{0}_p^T, \tilde{\mathbf{e}}_a(t)^T)^T$. Recall that $\sum_{j \in \mathcal{N}_i \cup \{i\}} \hat{\mathbf{x}}_i^a(t) = \hat{\mathbf{x}}_i^a(t)$ and that the estimated states relative to the centroid $\hat{\mathbf{x}}_{\mathcal{V}}^{cen}$ are $\hat{\mathbf{x}}_{\mathcal{V}}^{cen} = (\mathbf{I} - H_{cen})\hat{\mathbf{x}}_{\mathcal{V}}^a$ as in eq. (6.67). Algorithm 6.5.5 becomes

$$\mathbf{e}_{cen}(t) = (\mathcal{W} \otimes \mathbf{I}_p)\mathbf{e}_{cen}(t-1) + ((\mathbf{I}_n - \mathcal{W}) \otimes \mathbf{I}_p)\mathbf{e}_a(t-1) + P\hat{\mathbf{x}}_{\mathcal{V}}^a, \tag{6.76}$$

where the term P that is multiplying $\hat{\mathbf{x}}_{\mathcal{V}}^a$ is

$$P = \mathbf{I} - (\mathcal{W} \otimes \mathbf{I}_p) - (\mathbf{I} - (\mathcal{W} \otimes \mathbf{I}_p))(\mathbf{I} - H_{cen}) = (\mathbf{I} - (\mathcal{W} \otimes \mathbf{I}_p))H_{cen}. \tag{6.77}$$

We use the fact that $(\mathcal{W} \otimes \mathbf{I}_p)H_{cen} = H_{cen}$, and the previous expression gives $P = \mathbf{0}$ and eq. (6.76) becomes

$$\begin{aligned}\mathbf{e}_{cen}(t) &= (\mathcal{W} \otimes \mathbf{I}_p)\mathbf{e}_{cen}(t-1) + ((\mathbf{I}_n - \mathcal{W}) \otimes \mathbf{I}_p)\mathbf{e}_a(t-1) = \\ &= (\mathcal{W} \otimes \mathbf{I}_p)^t \mathbf{e}_{cen}(0) + \sum_{k=0}^{t-1} (\mathcal{W} \otimes \mathbf{I}_p)^{t-k-1} \left((\mathbf{I} - \mathcal{W}) \otimes \mathbf{I}_p\right) \mathbf{e}_a(k). \end{aligned} \tag{6.78}$$

Then, the norm of the error $\mathbf{e}_{cen}(t)$ satisfies

$$\|\mathbf{e}_{cen}(t)\|_2 \leq \lambda_{\text{eff}}^t(\mathcal{W})\|\mathbf{e}_{cen}(0)\|_2 + 2\sum_{k=0}^{t-1} \lambda_{\text{eff}}^{t-k-1}(\mathcal{W})\|\mathbf{e}_a(k)\|_2, \tag{6.79}$$

where we have used the fact that $\| \left((\mathcal{W} - \mathbf{I}) \otimes \mathbf{I}_p\right) \|_2 \leq 2$ since $\mathcal{W}$ is the Metropolis weight matrix given by eq. (A.3) in Appendix A.

We analyze now the norm of error $\|\mathbf{e}_a(t)\|_2$, which is related to the error vector of the Jacobi algorithm $\tilde{\mathbf{e}}_a(t) \in \mathbb{R}^{(n-1)p}$ by $\mathbf{e}_a(t) = (\mathbf{0}, \tilde{\mathbf{e}}_a^T(t))^T$. Let J be the Jacobi matrix, and $V_J = [\mathbf{v}_{p+1}(J), \ldots, \mathbf{v}_{np}(J)]$ and $\lambda_J = \text{Diag}\left(\lambda_{p+1}(J), \ldots, \lambda_{np}(J)\right)$ be its associated eigenvectors and eigenvalues so that $J = V_J\, \lambda_J\, V_J^{-1}$, and $||\mathbf{v}_i(J)||_2 = 1$. The error vector $\tilde{\mathbf{e}}_a(t)$ evolves according to

$$\tilde{\mathbf{e}}_a(t) = J\tilde{\mathbf{e}}_a(t-1) = J^t\tilde{\mathbf{e}}_a(0). \tag{6.80}$$

For each initial error vector $\tilde{\mathbf{e}}_a(0)$ there exist $\sigma_{p+1}, \ldots, \sigma_{np}$ such that

$$\tilde{\mathbf{e}}_a(0) = \sum_{i=p+1}^{np} \sigma_i \mathbf{v}_i(J),$$

and then the error vector $\tilde{\mathbf{e}}_a(t)$ after t iterations of the Jacobi algorithm given by eq. (6.80) can be expressed as

$$\tilde{\mathbf{e}}_a(t) = V_J \lambda_J^t V_J^{-1} V_J \left[\sigma_{p+1}, \ldots, \sigma_{np}\right]^T = \sum_{i=p+1}^{np} \sigma_i \mathbf{v}_i(J) \lambda_i^t(J).$$

Let $\sigma_J = \max_{i=p+1}^{np} |\sigma_i|$, and $\rho(J) = \max_{i=p+1}^{np} |\lambda_i(J)|$. For all $t \geq 0$, the norm of the error vector $||\tilde{\mathbf{e}}_a(t)||_2$ satisfies

$$||\mathbf{e}_a(t)||_2 = ||\tilde{\mathbf{e}}_a(t)||_2 \leq p(n-1)\sigma_J \rho^t(J). \tag{6.81}$$

Linking this with eq. (6.79) gives that the convergence rate is

$$\|\mathbf{e}_{cen}(t)\|_2 \leq \lambda_{\text{eff}}^t(\mathcal{W})\|\mathbf{e}_{cen}(0)\|_2 + 2p(n-1)\sigma_J \sum_{k=0}^{t-1} \lambda_{\text{eff}}^{t-k-1}(\mathcal{W})\rho^k(J), \tag{6.82}$$

as in eq. (6.75).

Now we prove the asymptotical convergence to the centroid (6.74). If both the Jacobi and the general algorithm have the same convergence rate, $\rho(J) = \lambda_{\text{eff}}(\mathcal{W})$, then eq. (6.82) gives

$$||\mathbf{e}_{cen}(t)||_2 \leq \lambda^t_{\text{eff}}(\mathcal{W})||\mathbf{e}_{cen}(0)||_2 + 2p(n-1)\sigma_J \lambda^{t-1}_{\text{eff}}(\mathcal{W})t, \tag{6.83}$$

whereas for $\rho(J) \neq \lambda_{\text{eff}}(\mathcal{W})$, it gives

$$||\mathbf{e}_{cen}(t)||_2 \leq \lambda^t_{\text{eff}}(\mathcal{W})||\mathbf{e}_{cen}(0)||_2 + \frac{2p(n-1)\sigma_J}{\rho(J) - \lambda_{\text{eff}}(\mathcal{W})}(\rho^t(J) - \lambda^t_{\text{eff}}(\mathcal{W})). \tag{6.84}$$

Note that $\lambda_{\text{eff}}(\mathcal{W}) < 1$ for connected graphs $\mathcal{G}$. Then, the term $\lambda^t_{\text{eff}}(\mathcal{W})||\mathbf{e}_{cen}(0)||_2$ in eqs. (6.83) and (6.84) exponentially tends to zero as $t \to \infty$ regardless of the initial error $\mathbf{e}_{cen}(0)$. For the case $\rho(J) = \lambda_{\text{eff}}(\mathcal{W})$, the term $\lambda^t_{\text{eff}}(\mathcal{W})t$ in eq. (6.83) is decreasing for $t \geq \frac{\lambda_{\text{eff}}(\mathcal{W})}{1-\lambda_{\text{eff}}(\mathcal{W})}$ and thus it tends to zero as $t \to \infty$. For $\rho(J) \neq \lambda_{\text{eff}}(\mathcal{W})$, the term $(\rho^t(J) - \lambda^t_{\text{eff}}(\mathcal{W}))$ in eq. (6.84) asymptotically tends to zero since $\lambda_{\text{eff}}(\mathcal{W})$ is less than 1, and as stated by Theorem 6.5.2, $\rho(J) < 1$. Therefore, $\lim_{t\to\infty} ||\mathbf{e}_{cen}(t)||_2 = 0$, where $||\mathbf{e}_{cen}(t)||_2 = 0$ iff $\mathbf{e}_{cen}(t) = 0$, what concludes the proof. □

6.5.4 Z_n^p and M_n^p matrices defined by blocks

In [51] a classification of matrices defined by blocks and a study of their properties is given. Here we show a brief summary of some of these properties. We use the notation $A = [A_{ij}]$ for a real matrix $A \in \mathbb{R}^{np\times np}$ defined by blocks, where each block A_{ij} is a $p \times p$ matrix, for all $i, j \in \{1, \ldots, n\}$.

Definition 6.5.7. [51] *Matrix A is of class Z_n^p if A_{ij} is symmetric for all $i, j \in \{1, \ldots, n\}$ and $A_{ij} \preceq \mathbf{0}$ for all $i, j \in \{1, \ldots, n\}, j \neq i$. In addition, it is of class $\hat{Z}_n^p$ if $A \in Z_n^p$ and $A_{ii} \succ \mathbf{0}$ for all $i \in \{1, \ldots, n\}$. Matrix A is of class M_n^p if $A \in \hat{Z}_n^p$ and there exist positive scalars $u_1, \ldots, u_n > 0$ such that*

$$\sum_{j=1}^{n} u_j A_{ij} \succ \mathbf{0} \text{ for all } i \in \{1, \ldots, n\}.$$

Lemma 6.5.8. [51, Lemma 3.8] *Let $A \in Z_n^p$ and assume that $\forall \mathcal{J} \subset \{1, \ldots, n\}$ there exists $i \in \mathcal{J}$ such that $\sum_{j\in\mathcal{J}} A_{ij} \succ \mathbf{0}$. Then, there exists a permutation π such that $\sum_{j\geq i} A_{\pi(i),\pi(j)} \succ \mathbf{0}$, for all $i \in \{1, \ldots, n\}$.*

Theorem 6.5.9. [51, Theorem 3.11] *Let $A \in Z_n^p$, let $u_1, \ldots, u_n > 0$ and let*

$$\sum_{j=1}^{n} A_{ij} u_j \succeq \mathbf{0}, \text{ for all } i \in \{1, \ldots, n\}. \tag{6.85}$$

Assume that there exists a permutation π of $\{1, \ldots, n\}$ such that

$$\sum_{j\geq i} A_{\pi(i),\pi(j)} u_{\pi j} \succ \mathbf{0}, \text{ for all } i \in \{1, \ldots, n\}. \tag{6.86}$$

Then, $A \in M_n^p$.

Theorem 6.5.10. [51, Theorem 4.7] *Let*

$$A + A^T \in M_n^p, \qquad D = \text{blkDiag}\left(A_{11}, \ldots, A_{nn}\right), \qquad \text{and } A = D - N.$$

Then $\rho\left(D^{-1}N\right) < 1$.

6.6 Discussion

Noise-free pose localization

In order to show the performance of the noise-free pose localization algorithm, we have carried out several simulations with a team composed by 7 robots exploring an environment of 20×20 m with 300 features, see Fig. 6.1.

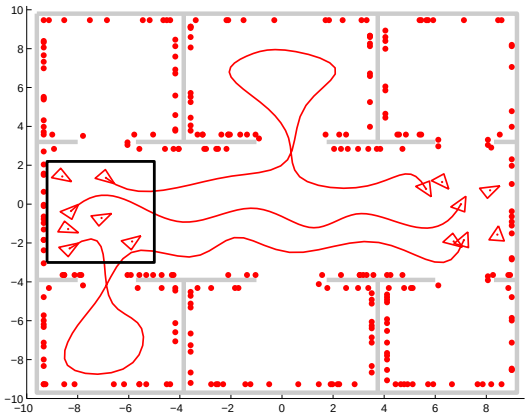


Figure 6.1: A team of 7 robots explore an environment of 20×20 m. Gray areas are walls and red dots are the ground-truth location of landmarks. Initially, the robots are placed in the black box region. From these initial poses, they reach consensus on the global reference frame. After that, they explore the environment and build their maps according this global reference frame. We display the trajectories followed by robots 2, 3, and 5, together with the final poses of the 7 robots.

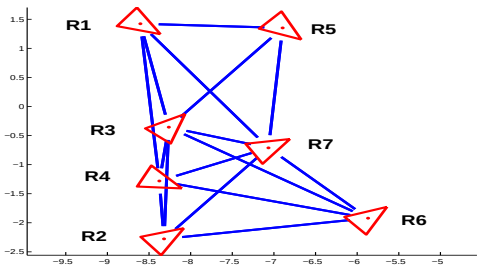


Figure 6.2: Communication graphs associated to the initial robot poses in Fig. 6.1. There is a link (blue solid line) between any pair of robot poses (red triangles) that are within a distance of 3 m.

Initially, the robots are placed within the black rectangle in Fig. 6.1. We assume that a pair of robots can exchange information and compute their relative poses if they are within a distance of 3 m. The robots execute the consensus on the global reference frame algorithm (Section 6.3) using the initial communication graph $\mathcal{G}$ (Fig. 6.2). Based on local interactions with its neighbors, each robot i computes the position and orientation of the global frame relative to its own pose (Fig. 6.3). The estimates $(T_G^i(t), R_G^i(t))$ of each robot (gray triangles) converge very fast to the correct centroid and Karcher mean of the team $(T_G^{\mathtt{w}}, R_G^{\mathtt{w}})$ (red triangle). Recall that, from this estimates $(T_G^i(t), R_G^i(t))$, each robot i locally computes its pose $\mathbf{p}_i^G$ relative to the global frame G as in eq. (6.5). After 10 iterations, the errors in the $x-$ and $y-$coordinates (Fig. 6.4 (a), (b)) are less than 0.11 cm and 2.12 cm respectively, and the error in the estimated orientation (Fig. 6.4 (c)) is less than 0.43 degrees.

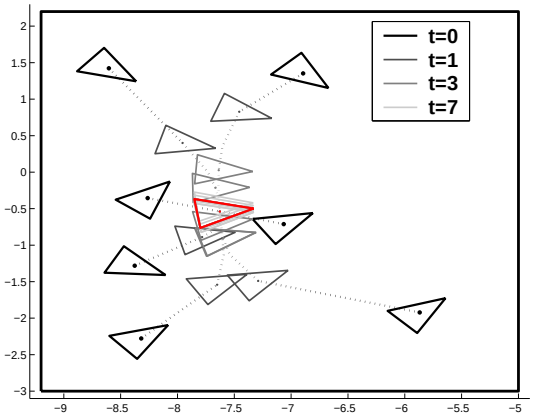
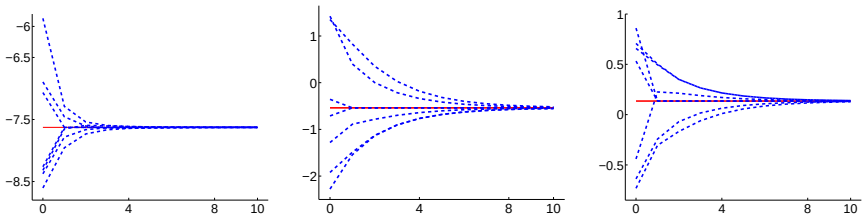


Figure 6.3: From the initial poses within the black box in Fig. 6.1, the robots reach a consensus on the global reference frame. Initially ($t = 0$), each robot i estimates the global frame $(T_G^i(t), R_G^i(t))$ as its own pose (black triangles). During successive iterations, they update $(T_G^i(t), R_G^i(t))$ based on their neighbors estimates and their relative poses $(T_j^i(t), R_j^i(t))$. We display the centroid and Karcher mean estimated by the 7 robots (gray triangles), transformed into a common reference frame, for iterations $t = 1, 3, 7$. Their estimates after 7 iterations are very close to the ground truth centroid and Karcher mean $(T_G^{\mathtt{w}}, R_G^{\mathtt{w}})$ (red triangle).

Noisy pose localization

A set of simulations have been carried out to show the performance of the noisy pose localization algorithm and to compare the results of the distributed (Section 6.4.2) and the centralized (Section 6.4.1) approaches.

In the first set of experiments, a team of $n = 20$ robots are placed along a ring of radius 4 m with their orientations randomly generated within $\pm\frac{\pi}{2}$ (Fig. 6.5). Each robot measures the relative pose of the next robot and these measurements are corrupted with noises in the x- and y- coordinates of 6 cm standard deviation, and of 1 degree for the orientation. The robots execute the proposed method to compute their pose with



(a) $x-$**coordinate** **(b)** $y-$**coordinate** **(c) orientation**

Figure 6.4: Position of the centroid and Karcher mean orientation estimated by the 7 robots (blue dashed) compared to the true centroid and Karcher mean (red solid), along 10 iterations.

respect to the anchor node $R1$. The experiment is repeated 100 times and the average

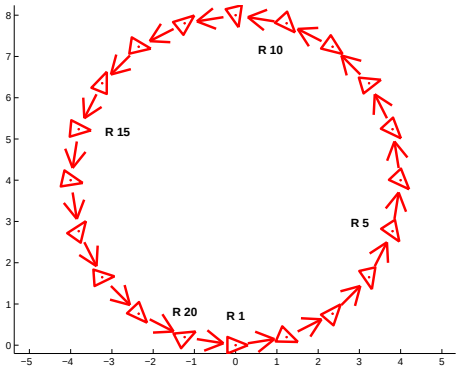


Figure 6.5: A team of 20 robots are placed along a circle of $4m$ radius. Each robot (red triangles) measures the relative pose of its next robot (outgoing arrow), and exchanges data with both its previous and next robots.

results can be seen in Table 6.1. The solution computed by the localization algorithm in Section 6.4.1 presents a high accuracy. The greatest difference between the ground truth and the obtained positions are of around 25 cm for the $x-$ and $y-$ coordinates, with around 13 cm standard deviation, and of around 4 degrees for the orientation, with approximately 1.7 degrees of standard deviation.

The distributed implementation of the algorithm (Section 6.4.2) has also been tested for the scenario in Fig.6.5 and its results have been compared to the ones that would be obtained by the centralized algorithm in Section 6.4.1. We use the flagged initialization [18] that is known to produce fast convergence results. The convergence speeds during the first and the third phases depend on the values of respectively $\rho(C^{-1}D)$ in (6.40) and $\rho(\mathbf{I} - M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})$ in (6.53). The closer to one the values are, the slower the system converges. Observe that the second phase is always executed in a single iteration and for this reason it does not have any convergence speed associated. In this experiment, both $\rho(C^{-1}D)$ and $\rho(\mathbf{I}-M^{-1}\Upsilon_{\hat{\mathbf{q}}^a_{\mathcal{V}^a}})$ are close to one (0.99), and thus we expect the algorithm to converge slowly. After executing the first phase during $t = 50$ iterations, the obtained $\bar{\theta}^a_{\mathcal{V}}$ still differs from the centralized solution $\tilde{\theta}^a_{\mathcal{V}}$ by around 0.16°. If we increase the number of iterations we obtain better approximations that differ only by 0.01 ($t = 100$) and $8.5e-05$

Table 6.1: Results for the scenario in Fig. 6.5

LOCALIZATION RESULTS VS. GROUND TRUTH			
	Max error	Avg standard deviation	
Orientation phase 1	3.38°	1.87°	
x-coordinate phase 3	27.85 cm	13.45 cm	
y-coordinate phase 3	24.33 cm	12.31 cm	
Orientation phase 3	4.03°	1.66°	
DISTRIBUTED IMPLEMENTATION (flagged-initialization)			
$\rho(C^{-1}D)$	0.99		
$\rho(\mathbf{I} - M^{-1}\Upsilon_{\tilde{\mathbf{q}}^a_{\mathcal{V}^a}})$	0.99		
Max error	$t = 50$	$t = 100$	$t = 200$
Orientation phase 1	0.16°	0.01°	$8.5e-05$°
x-coordinate phase 3	1.74 cm	1.64 cm	1.64 cm
y-coordinate phase 3	0.84 cm	0.49 cm	0.48 cm
Orientation phase 3	0.29°	0.12°	0.11°

($t = 200$) degrees. The next three rows show the results after executing the second phase followed by 200 iterations of the third phase. Since the second and third phases have been executed using $\bar{\theta}^a_{\mathcal{V}}$ instead of $\tilde{\theta}^a_{\mathcal{V}}$, the final results also differ. For the case $t = 200$ (third column), the difference between the pose estimated by the distributed and centralized approaches is small (1.64 cm and 0.48 cm for the $x-$ and $y-$ coordinates, and 0.11 degrees for the orientation), and similar results are obtained for $t = 100$. However, for $t = 50$ the final errors are larger (1.74 cm and 0.84 cm for the $x-$ and $y-$ coordinates, and 0.29 degrees for the orientation), which is due to the use of a less accurate approximation of $\tilde{\theta}^a_{\mathcal{V}}$.

A simulation with 10 robots placed as in Fig. 6.6 has been carried out. A robot i measures the relative position and orientation of a second robot j if there is an arrow from i into j. The measured positions are corrupted with additive noises of standard deviation proportional to the distance d between the robots, 5% d for the x-coordinate, and 0.7% d for the y-coordinate. And the measured orientations are corrupted with additive noises of 2.5 degrees of standard deviation. The robots execute the distributed algorithm (Fig. 6.8) during the phase 1 to compute their orientations with respect to the anchor node R1. At $t = 0$ they initialize their orientations to 0 (black). Along the iterations the robots update their estimates (gray), which successively approach the orientations that would be obtained by a centralized system (blue). After the first phase (Fig. 6.7(a)), the orientations computed by the robots (blue) are very close to the ground truth data (red). After that, they execute the second phase to transform their local measurements. The relative positions measured by the robots were initially expressed into the reference frame of the observer. After the phase 2, they are expressed in the reference frame of the anchor node (Fig. 6.7(b)). And finally, they execute the last phase to obtain both, their positions and orientations relative to the anchor node (Fig. 6.7(c)). As an example, a detail of the evolution of the estimates at R10 can be observed in Fig. 6.9. Although the convergence was previously proved only for graphs with low connectivity (ring or string graphs), in

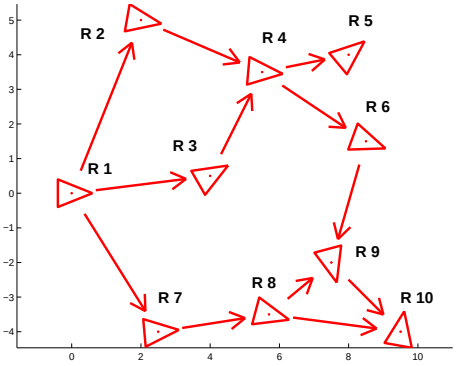
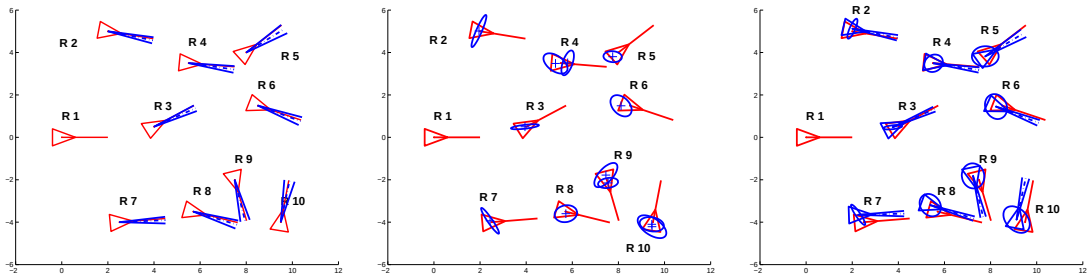


Figure 6.6: A team of 10 robots are placed at unknown poses in a planar environment. Each robot can measure the relative pose (position and orientation) of a subset of the other robots (outgoing arrows).



(a) Phase 1 (b) Phase 2 (c) Phase 3

Figure 6.7: The robots execute the proposed strategy to obtain their poses relative to the anchor R1 for the experiment in Fig. 6.6. The estimates (blue dashed) are compared to the ground truth data (red solid). The covariances computed by the centralized approach are also displayed (blue solid). (a) After the first phase, the robots obtain their orientations relative to the anchor R1 (blue dashed). These orientation are very similar to the ground data (red solid). (b) The robots transform their local position measurements according to the previously obtained global orientation. (c) Finally, the robots compute their position relative to the anchor (blue triangles). We are also displaying the covariances obtained by the centralized algorithm (blue ellipses).

the experiments with general communication graphs the algorithm has been found to converge as well.

Centroid-based noisy position localization

We study the performance of the algorithm presented in Section 6.5 in a planar multi-robot localization scenario (Fig. 6.10) with $n = 20$ robots (black circles) that get noisy measurements (gray crosses and ellipses) of the position of robots which are closer than 4 meters.

Each robot $i \in \mathcal{V}$ is used as an anchor and its covariance matrix $\Sigma_{\hat{\mathbf{x}}^i_{\mathcal{V}}}$ is computed. The eigenvalues of the block-traces $\mathrm{blkTr}(\Sigma_{\hat{\mathbf{x}}^i_{\mathcal{V}}})$ of their covariance matrices are depicted in

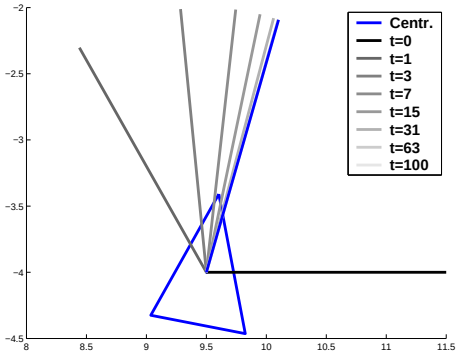


Figure 6.8: Phase 1 of the proposed strategy. The orientations estimated by the robot R10 along different iterations of the distributed algorithm (gray) successively approach the centralized solution (blue).

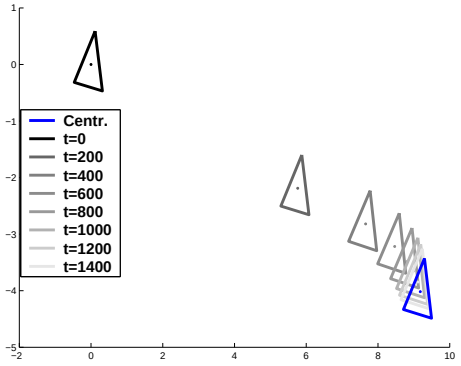
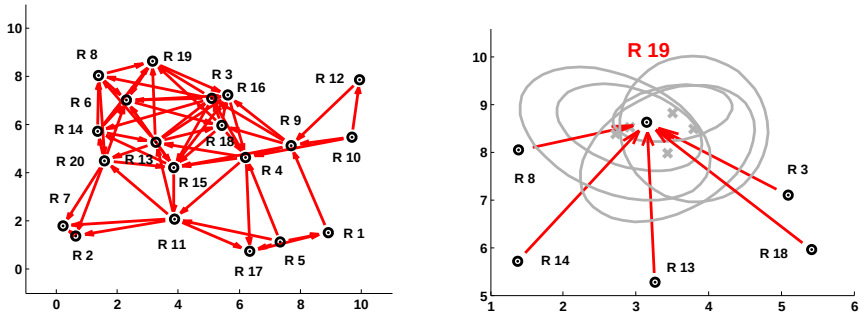


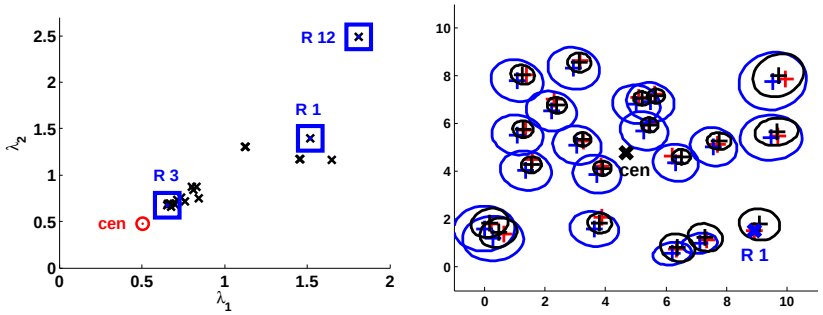
Figure 6.9: At $t = 0$ the estimated pose of R10 in Fig. 6.6 is initialized in the origin (black). Along the iterations, its estimated orientation almost does not vary, while its pose (gray) successively approaches the centralized solution (blue).



(a) Ground truth scenario **(b) Measurements of $R19$'s position**

Figure 6.10: **(a)** 20 robots (black circles) are placed randomly in a region of 10 meters $\times$10 meters. There is an ingoing or outgoing edge $e = (i, j) \in \mathcal{E}$ (red arrows) between pairs of robots that are closer than 4 meters. **(b)** Each robot i has a noisy measurement $\mathbf{z}_e$ (gray crosses and ellipses) of the relative position of its out-neighbors j, with $e = (i, j)$. The noises are proportional to the distance between the robots.

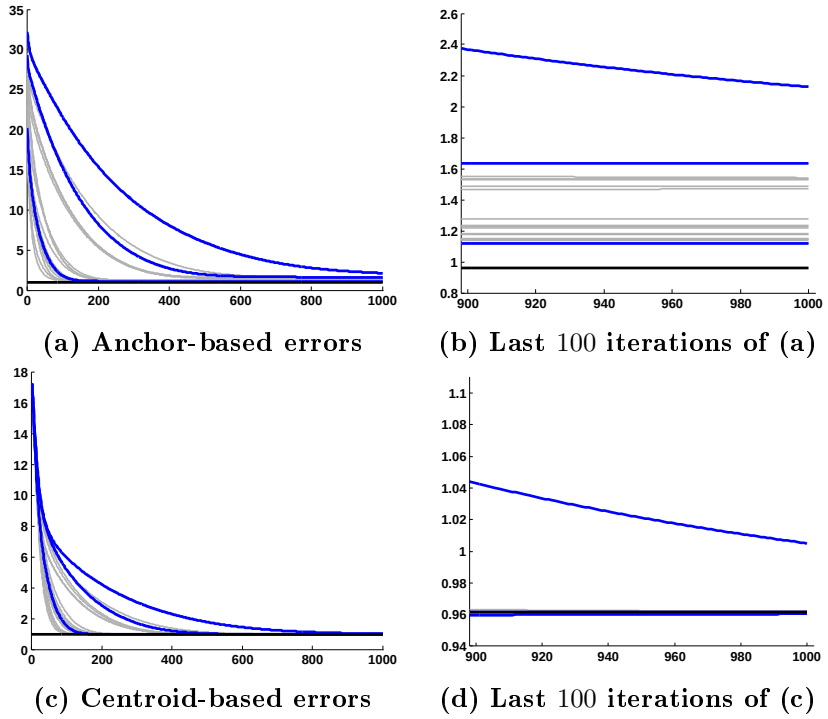
Fig. 6.11 **(a)** (black crosses). For all the possible anchors $i \in \mathcal{V}$, the eigenvalues associated to the anchor-based covariance matrices $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}^i_{\mathcal{V}}})$ are larger (more uncertain) than the ones associated to the centroid-based covariance matrix $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}})$ (red circle). Robots $R3$ and $R12$ which produce respectively the most and the least precise anchor-based results, and robot $R1$ which is conventionally used as the anchor (in blue) are studied in detail. Fig. 6.11 **(b)** shows the estimated states relative to the centroid $\hat{\mathbf{x}}^{cen}_{\mathcal{V}}$, $\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}}$ (black crosses and ellipses) and the anchor $a = R1$, $\hat{\mathbf{x}}^{a}_{\mathcal{V}}$, $\Sigma_{\hat{\mathbf{x}}^{a}_{\mathcal{V}}}$ (blue crosses and ellipses) compared with the ground-truth positions of the robots (red crosses). The ground-truth position of the centroid is marked with a black 'x'. The errors and covariances associated to the centroid-based estimates (black) are in general smaller than the ones obtained by using $R1$ as the anchor (blue).



(a) Eigenvalues of $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}^i_{\mathcal{V}}})$ **(b) Estimated positions**

Figure 6.11: **(a)** The eigenvalues of $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}^i_{\mathcal{V}}})$ when each robot $i \in \mathcal{V}$ is taken as the anchor (black crosses) are always greater than the ones of the centroid-based $\text{blkTr}(\Sigma_{\hat{\mathbf{x}}^{cen}_{\mathcal{V}}})$ representation (red circle). The values when the anchor is $R1$ which is the classical situation, and when it is $R3$ and $R12$ which are the extreme values are shown in blue. **(b)** The positions estimated relative to the centroid (black crosses and ellipses) are in general more accurate and closer to the ground-truth (red crosses) than the ones estimated by using $R1$ as the anchor (blue crosses and ellipses).

We analyze the states estimated by the n robots along 1000 iterations of the proposed algorithm (Fig. 6.12). Robots initialize their states $\hat{\mathbf{x}}^a_i(t)$, $\hat{\mathbf{x}}^{cen}_i(t)$ with zeros and execute Algorithms 6.5.1 and 6.5.5. We generate specific noises as the ones in Fig. 6.10 for 100 different samples. For each of them, we record the norm of the error vector containing the difference between the estimates at the n robots and the ground-truth positions at each iteration t. In Fig. 6.12 **(a)** we show the results of Algorithm 6.5.1 when each robot $i \in \mathcal{V}$ is used as the anchor (gray lines). The special cases that the anchor is $R1$, $R3$ and $R12$ are displayed in blue. The black line is the asymptotic error reached with the centroid-based estimation method. As it can be seen, the errors reached with the anchor-based solutions are greater than the ones associated to the centroid. This is even more evident in Fig. 6.12 **(b)**, which shows the last 100 iterations in Fig. 6.12 **(a)**. In Fig. 6.12 **(c)** we show the equivalent errors for the centroid-based estimation algorithm (Algorithm 6.5.5), using all the possible anchors for Algorithm 6.5.1. Here, in all cases the error estimates (gray lines) converge to the asymptotic error of the centroid-based estimation method (black line).



(a) Anchor-based errors **(b) Last 100 iterations of (a)**

(c) Centroid-based errors **(d) Last 100 iterations of (c)**

Figure 6.12: The experiment in Fig. 6.10 is generated 100 times with the same noise levels but different noise values. We display the average norm of the error with the difference between the estimates and the ground truth for the 100 different experiments. **(a)** Results of Algorithm 6.5.1 when each robot $i \in \mathcal{V}$ is used as the anchor (gray lines). The special cases that the anchor is $R1$, $R3$ and $R12$ are shown in blue. The black line is the asymptotic error reached with the centroid-based estimation. **(b)** Detail of iterations 900-1000 in **(a)**. The errors of the anchor-based solutions are bigger than the ones associated to the centroid. **(c)** Results of Algorithm 6.5.5 using all the possible anchors. **(d)** Detail of iterations 900-1000 in **(c)**. The anchor choice affects to the convergence speed of Algorithm 6.5.5. However, the final accuracy is independent on the selected anchor, and in all cases the errors converge to the asymptotic centroid error (black line).

6.7 Conclusions

Along this chapter, the network localization problem has been addressed for three different scenarios: the pose network localization from noise-free relative measurements, the pose network localization from noisy relative measurements, and the position network computation from noisy measurements. We have proposed distributed strategies that allow the robots to agree on a common global frame, and to compute their poses relative to the global frame. The presented algorithms exclusively rely on local computations and data exchange with direct neighbors. Besides, they only require the robots to maintain their own estimated poses relative to the common frame. Thus, the memory load of the algorithm is low compared to methods where each robot must also estimate the positions or poses of any other robot. The noise-free pose localization algorithm has been demonstrated to perform properly in noise-free scenarios and to have a fast convergence speed. For the noisy case, the pose localization algorithm has been shown to correctly compute the poses of the robots relative to an anchor node, for ring or string topologies. The centroid-based position localization method has been proved to produce more accurate results than any anchor-based solution. Besides, in the experiments we have observed that it converges faster than the anchor-based solutions.

Chapter 7
Strategies for Improved Multi Robot Perception

This chapter discusses strategies for improving the multi-robot perception. First, we propose a method that allows the robots to decide their next motions so that the global map is more precise. We define a cost function measuring the perception improvement when a robot moves to a new position. Each robot selects a finite set of candidate next positions within its local landmark-based stochastic map and evaluates the cost function on the candidate positions. Then, based on this information, robots in the team coordinate their next motions. We present a solution designed for omnidirectional cameras, although the results can be extended to conventional cameras. Second, we propose a method that allows the robots to compute the *algebraic connectivity*, which is very important in the context of network topology. This parameter characterizes the speed of convergence of most of the distributed algorithm presented in this thesis, i.e., it establishes the rate at which the variables maintained by the robots converge to the desired values. Thus, we can have a deeper control of the quality of our methods by controlling the network topology so that the algebraic connectivity remains within some specific values.

7.1 Introduction

The convenience of having groups of robots working cooperatively has been stressed along this document. An important issue in these scenarios consists of placing the robots at positions where they are more useful. Extensive research has been focused on the optimization of facilities [98,99,132], and on coverage problems [45,89], where a group of robots is optimally placed in an environment of interest to achieve maximum coverage. Here we focus on controlling the robot motions so that their perception of the environment is improved. In the previous chapters, we have proposed several methods that allow the robots to cooperatively perceive the environment and to acquire a global map of their surroundings. In this chapter, we discuss strategies to optimize individual robot motions to maximize the information collected about the scene, and to optimize the convergence speed of the distributed map merging algorithms.

First, we discuss the cooperative selection of next robot motions. We focus on strategies that improve the local maps taking into account the proposed improvements of other

robots, giving rise to an improvement of the global map. This problem is highly related to exploration guided by information and active sensing, which has been investigated for both single robot [131,136] and multi-robot systems [28]. The previous works are based on grid maps, where frontier cells dividing between already explored and unknown sections can be easily detected. Robots evaluate a cost function on this small subset of destinations and make decisions propagating small pieces of information with the other robots. The exploration [115, 116] and feature tracking [155] problems turn out to be more complicated for landmark-based representations, since the number of candidate destinations is infinite. It is common the use of global optimization methods, where robots search for the best position to reduce the whole map uncertainty. Every robot makes decisions based on its current local estimate of the global map and propagates its observations to the other robots so that they can update their maps. These approaches result in weak robot coordination, because without a common global map estimate, different robots may end up exploring exactly the same regions. In addition, many of these solutions use gradient methods to find minima on the cost function. Gradient algorithms are computationally expensive since the gradient must be reevaluated at every step. Besides, they may find local minima, and the step size adjustment is complicated. Alternatively, clustering methods can be used to select a finite subset of candidate positions [150]. Our work belongs to this latter class, although instead of choosing frontier positions as in [150], our robots look for already explored places which present big uncertainties. We focus on one step strategies instead of considering path planning or trajectory optimization methods [84, 85]. These methods use a larger time horizon and consider the cost function for multiple successive robot motions, and thus they present important scaling problems for the multi-robot case.

We define a method for landmark-based maps which lets the robots select a finite set of destinations based on their local data. Each robot associates to these positions cost values that can be sent to other robots in the team with the aim of negotiating their next motions. This solution presents multiple appealing features due to its low computation complexity and to the fact that the robots do not need to wait for having a good global map estimate. Instead, they can negotiate on small pieces of information, ensuring that the resulting global map will be improved.

In the second part of this chapter, we discuss the problem of improving the convergence speed of the distributed map merging algorithms. The algebraic connectivity is an important network property for several multi-robot systems to reach convergence and it characterizes the convergence rate. In this chapter, we address the problem of computing this algebraic connectivity in a distributed fashion. We provide a successive solution where, at every iteration, each robot obtains a more accurate estimate of the algebraic connectivity. As an application, we use this algebraic connectivity estimation algorithm to derive an adaptive version of the event-triggered consensus protocol in [124], where at each iteration, the robots adjust their behavior according to their most recent estimate of the algebraic connectivity. Connectivity control methods establish robot motions that preserve or maximize some network connectivity property. The $k-$connectivity matrix of the graph is computed in a centralized fashion in [156]. There are several distributed methods that compute spanning subgraphs [157], the left Laplacian eigenvector with eigenvalue 1 for directed unbalanced graphs [112], or the first four moments (mean, variance, skewness

and kurtosis) of the Laplacian eigenvalue spectrum [111]. In [126], the motion control strategy maximizes the algebraic connectivity without actually computing it. Although the previous control methods improve the network connectivity, they do not characterize any particular eigenvalue of the Laplacian as required in our case. A method that computes and tracks the eigenvalues of the network topology is given in [58]. Robots execute a local interaction rule that makes their states oscillate at frequencies corresponding to these eigenvalues, and use the Fast Fourier Transform (FFT) on their states to identify these eigenvalues. The main limitation of this work is that the proper adjustment of the FFT, so that the eigenvalues can be correctly identified, is nontrivial. In addition, some robots may observe only a subset of the eigenvalues and thus they need to execute additional coordination rules for propagating their data. Several solutions to the computation of the Laplacian spectra rely on the power iteration method or variations [63, 74, 154]. Power iteration [68] selects an initial vector and then repeatedly multiplies it by a matrix and normalizes it. This vector converges to the eigenvector associated to the leading eigenvalue (the one with the greatest absolute value) of the matrix. The original matrix can be previously deflated so that a particular eigenvalue becomes the leading one. A continuous-time version of the power iteration is proposed in [154] for computing the Fiedler eigenvector, which is the one associated to the algebraic connectivity of the graph. In [74] the orthogonal iteration method is used for simultaneously computing the k leading eigenvectors of a matrix. This algorithm can be seen as a generalization of the power iteration where, at each step, the k vectors are multiplied by the matrix and ortonormalized. The previous method is used in [63] for computing the Fiedler eigenvector. The interest of power iteration methods is that each robot only needs to maintain its own component within the estimate of the eigenvector. Then, the product of this vector by the Laplacian can be executed locally by the robots. However, the main limitation of these approaches consists on the normalization and orthonormalization of the vectors at each step. For [74], it involves a gossip-based information aggregation algorithm [73] and for [154] a distributed averaging method [59]. Therefore, several iterations of the previous algorithms must be executed by the robots between consecutive steps of the power method in order to ensure that they have achieved the required accuracy in the vector normalization. Besides, the previous methods only ensure convergence but they do not give any upper or lower bound relating the true algebraic connectivity and the estimates at each iteration.

We propose a method for computing the algebraic connectivity where, at each step k, the robots compute the $k-$th power of a deflated Laplacian in a distributed fashion. The computation of the powers of the adjacency matrix was first discussed in [10]. Whenever the robots want to obtain an estimate of the algebraic connectivity, they run a max$-$consensus method [134]. The robots do not need to wait for the max$-$consensus to finish before starting the next step $k+1$. Instead, they can continue executing the matrix power algorithm in parallel. An additional benefit of our method is that it automatically provides the total number of robots n. We provide proofs of convergence of the algebraic connectivity computation algorithm, we characterize its convergence speed, and give upper and lower bounds for the true algebraic connectivity at each iteration. Then, we combine the previous ideas with [124] and present an adaptive triggered consensus

algorithm where the most recent estimate of the algebraic connectivity is used at each step. We show that both processes can be executed simultaneously due to the upper and lower bounds of the true algebraic connectivity.

7.2 Improving the Map Precision with Motion Control

In this section, we propose a method that allows the robots to decide their next motions so that the global map is more precise. We define a cost function measuring the perception improvement when a robot moves to a new position. Each robot selects a finite set of candidate next positions within its local landmark-based stochastic map and evaluates the cost function on the candidate positions. Then, based on this information, robots in the team coordinate their next motions. We present a solution designed for omnidirectional cameras, although the results can be extended to conventional cameras.

We consider n robots exploring an unknown environment. Robots move on the plane and estimate their motions (translations and rotations) using odometry information. They are equipped with an omnidirectional camera sensor that gives bearing-only measurements of some features in the environment. Every robot has a local map of the portion of the environment observed by himself, expressed as a set of estimates of the position of static features in the environment. Those estimates have an associated covariance that represents how precise the estimate is, and the quality of the map can be measured in terms of these covariances. Robots can improve their local estimates of features via robot redeployment that let them reduce the feature uncertainty. The n robots have communication capabilities for exchanging information between them.

7.2.1 Vantage locations

In order to manage a finite set of vantage locations for every robot, we adopt a one-feature improvement strategy. Every robot in the team will attempt to improve, at least, the observation of one of the landmarks. As a side effect, the observations of other landmarks are also improved. We compute vantage locations where the observation of the landmark produces a high uncertainty reduction. In [125] authors discuss uncertainty minimization within the Extended Kalman Filter (EKF) framework. They show that the minimization of the map uncertainty $\mathbf{P}$ is highly related to the maximization of the covariance of the innovation $\mathbf{S}$. In addition, they show that if the system is driven to the optimal position to obtain maximum information gain, it results in numerically unstable update steps for the EKF. The system becomes more unstable as the robot moves closer to a landmark. To derive the vantage locations to observe the landmarks, we analyze the robot locations that lead to the maximization of $\mathbf{S}$.

In this section, we discuss the best poses for observing a landmark. We frequently refer to Section 7.2.2, where a full development of the Extended Kalman Filter (EKF) is given. For a landmark i, the covariance of the innovation S_i is (7.8)

$$S_i = \mathbf{H}_i \mathbf{P}(k+1|k)\mathbf{H}_i^T + \sigma_z^2.$$

If we take $\mathbf{P}(k+1|k) = \mathbf{I}$ and apply (7.5), (7.6), we can express S_i as

$$S_i = 1 + \sigma_z^2 + 2/r^2,$$

where $r = \sqrt{(x_{k+1} - x_i)^2 + (y_{k+1} - y_i)^2}$ is the distance between the robot pose and the landmark. As it can be observed, the maximization of S_i is equivalent to the minimization of r. Now we introduce into the study the landmark covariance, taking[1]

$$\mathbf{P}(k+1|k) = \text{blkDiag}\left(\mathbf{I}, \begin{bmatrix} P_{xx} & 0 \\ 0 & P_{yy} \end{bmatrix}, \mathbf{I}\right),$$

and expressing $P_{yy} = kP_{xx}$, with $k > 1$. This models an uncertainty ellipse with its mayor axis perpendicular to the y-axis. For robot poses at a constant distance r, $x_{k+1} = r\cos\alpha$, $y_{k+1} = r\sin\alpha$, the value of S_i is

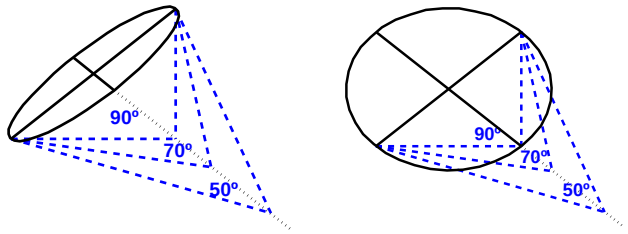
$$S_i = 1 + \sigma_z^2 + \frac{1 + P_{xx}}{r^2} + (k-1)\frac{P_{xx}}{r^2}\cos^2(\alpha),$$

Computing the first and the second derivative of S_i, the critical points of S_i are $\alpha = 0 + n\pi, n \in \mathbb{Z}$ and $\alpha = \frac{\pi}{2} + n\pi, n \in \mathbb{Z}$. Besides we have that

$$\frac{\partial^2 S_i}{\partial \alpha^2} = 2(k-1)\frac{P_{xx}}{r^2}(1 - 2\cos^2(\alpha)).$$

Since $k > 1$, S_i reaches a maximum for $\alpha = 0 + n\pi, n \in \mathbb{Z}$ and a minimum for $\alpha = \frac{\pi}{2} + n\pi, n \in \mathbb{Z}$. Then we can conclude that S_i is maximized when the distance between the robot and the feature is minimized, and that for constant distances, S_i is maximized when the landmark is observed from a position in a line perpendicular to its mayor axis.

However, minimizing r may make the system unstable [125]. To avoid this situation, we compute the optimal position for observing a feature so that it lays outside its uncertainty ellipse. It is at a distance from the center so that the angle α between the lines from the vantage point to the extremes of the ellipse must be less than $\frac{\pi}{2}$ (Fig. 7.1).



(a) ellipse **(b) circle**

Figure 7.1: Optimal positions for observing a feature with $\alpha = 90, 70$ and 50 degrees when its associated uncertainty is an ellipse (a) or a circle (b). As long as $\alpha < \pi/2$ the optimal position remains outside the uncertainty ellipse associated to the feature.

[1] $A = \text{blkDiag}(B_1, \ldots, B_r)$ returns a matrix A defined by blocks with $A_{ii} = B_i$ and $A_{ij} = \mathbf{0}$ for $i \neq j$.

7.2.2 Expected maps for the vantage locations

When a robot moves to a vantage location and takes new measurements of the environment, its local map estimate may become more precise. We compute this expected map applying the same algorithm used by the robots to build their local maps, using the same measurement and odometry models, and assuming that i) the measurements are exactly the expected bearings to the features from the new robot pose, and ii) the odometry estimate is exactly the new robot pose. In the following, we provide the expressions associated to the Extended Kalman Filter (EKF) executed by the robots.

Relative motion computation

We let $\hat{\mathbf{x}}_r(k) = (\hat{x}_k, \hat{y}_k, \hat{\theta}_k)^T$ be the robot pose at time k, $\hat{\mathbf{x}}_i(k) = (\hat{x}_i, \hat{y}_i)^T$ be the feature estimates at time k for $i \in \{1, \ldots, m\}$, and $\hat{\mathbf{x}}(k) = (\hat{\mathbf{x}}_r(k)^T, \hat{\mathbf{x}}_1(k)^T \ldots, \hat{\mathbf{x}}_m(k)^T)^T$ be the local map estimate at time k, with associated covariance $\mathbf{P}(k)$. Let $\mathbf{x}_g = (x_g, y_g, \theta_g)^T$ be the goal vantage location where the robot plans to move to. The relative translation and rotation $\mathbf{x}_{k+1}^k$ between $\hat{\mathbf{x}}_r(k)$ and $\mathbf{x}_g$ can be computed as:

$$\mathbf{x}_{k+1}^k = (\ominus \hat{\mathbf{x}}_r(k)) \oplus \mathbf{x}_g,$$

where the operator $\ominus$ is the inverse location vector

$$\ominus \hat{\mathbf{x}}_r(k) = \begin{bmatrix} -\hat{x}_k \cos\hat{\theta}_k - \hat{y}_k \sin\hat{\theta}_k \\ \hat{x}_k \sin\hat{\theta}_k - \hat{y}_k \cos\hat{\theta}_k \\ -\hat{\theta}_k \end{bmatrix}.$$

The operator $\oplus$ is the composition of two location vectors. It returns a location vector that transforms coordinates between the reference frames $\hat{\mathbf{x}}_r(k)$ and $\mathbf{x}_g$. Then, the expression for the relative transformation $\mathbf{x}_{k+1}^k$ between the robot pose at time k and the goal pose at $k+1$ is:

$$\mathbf{x}_{k+1}^k = \begin{bmatrix} (x_g - \hat{x}_k)\cos\hat{\theta}_k + (y_g - \hat{y}_k)\sin\hat{\theta}_k \\ -(x_g - \hat{x}_k)\sin\hat{\theta}_k + (y_g - \hat{y}_k)\cos\hat{\theta}_k \\ \theta_g - \hat{\theta}_k \end{bmatrix}. \tag{7.1}$$

EKF prediction

The prediction step of the localization and mapping algorithm computes the map $\bar{\mathbf{x}}(k+1) = \mathbf{x}(k+1|k) = (\bar{\mathbf{x}}_r(k+1)^T, \bar{\mathbf{x}}_1(k+1)^T \ldots \bar{\mathbf{x}}_m(k+1)^T)^T$, with associated covariance $\mathbf{P}(k+1|k)$, based on the previous state $\hat{\mathbf{x}}(k), \mathbf{P}(k)$, and on the odometry measurements $\mathbf{x}_{k+1}^k = (x_{odom}, y_{odom}, \theta_{odom})$ with covariance matrix $\mathbf{P}_{odom}$. Here, $\bar{\mathbf{x}}_r(k+1) = (\bar{x}_{k+1}, \bar{y}_{k+1}, \bar{\theta}_{k+1})^T$ and $\bar{\mathbf{x}}_i(k+1) = (\bar{x}_i, \bar{y}_i)^T$ for $i \in \{1, \ldots, m\}$. The odometry measurements $\mathbf{x}_{k+1}^k$ are given by (7.1), and the odometry noise is modeled as three independent noises in the perpendicular and parallel translations and rotation, $\mathbf{P}_{odom} = \text{Diag}\left(\sigma_x^2, \sigma_y^2, \sigma_\theta^2\right)$, where $\sigma_x = K_x d$ and $\sigma_y = K_y d$ are proportional to the translation distance $d = \sqrt{(x_g - \hat{x}_k)^2 + (y_g - \hat{y}_k)^2}$. The

equations used to predict the new state are

$$\begin{aligned}\bar{\mathbf{x}}(k+1) &= ((\bar{\mathbf{x}}_r(k+1))^T, \bar{x}_1, \bar{y}_1, \ldots, \bar{x}_m, \hat{y}_m)^T = ((\hat{\mathbf{x}}_r(k) \oplus \mathbf{x}_{k+1}^k)^T, \hat{x}_1, \hat{y}_1, \ldots, \hat{x}_m, \hat{y}_m)^T, \\ \mathbf{P}(k+1|k) &= \mathbf{J}_1 \mathbf{P}(k) \mathbf{J}_1^T + \mathbf{J}_2 \mathbf{P}_{odom} \mathbf{J}_2^T, \end{aligned} \tag{7.2}$$

where the operator $\oplus$ is the composition of the location vectors $\hat{\mathbf{x}}_r(k)$ and $\mathbf{x}_{k+1}^k$

$$\hat{\mathbf{x}}_r(k) \oplus \mathbf{x}_{k+1}^k = \begin{bmatrix} \hat{x}_k + x_{odom} \cos\hat{\theta}_k - y_{odom} \sin\hat{\theta}_k \\ \hat{y}_k + x_{odom} \sin\hat{\theta}_k + y_{odom} \cos\hat{\theta}_k \\ \hat{\theta}_k + \theta_{odom} \end{bmatrix},$$

and $\mathbf{J}_1$, $\mathbf{J}_2$ are the Jacobians of the prediction operation relative to, respectively, the map and the odometry measurement:

$$\begin{aligned} \mathbf{J}_1 &= \text{blkDiag}(\mathbf{j}_1, \mathbf{I}), & \mathbf{J}_2 &= ((\mathbf{j}_2)^T, \mathbf{0})^T, \\ \mathbf{j}_1 &= \begin{bmatrix} 1 & 0 & -x_{odom} \sin\hat{\theta}_k - y_{odom} \cos\hat{\theta}_k \\ 0 & 1 & x_{odom} \cos\hat{\theta}_k - y_{odom} \sin\hat{\theta}_k \\ 0 & 0 & 1 \end{bmatrix}, & \mathbf{j}_2 &= \begin{bmatrix} \cos\hat{\theta}_k & -\sin\hat{\theta}_k & 0 \\ \sin\hat{\theta}_k & \cos\hat{\theta}_k & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{7.3}$$

Measurement prediction

For every feature with coordinates $\bar{\mathbf{x}}_i(k+1) = (\bar{x}_i, \bar{y}_i)^T$ in the map, the expected bearing measurement with respect to the robot that should be obtained from the predicted robot location $\bar{\mathbf{x}}_r(k+1) = (\bar{x}_{k+1}, \bar{y}_{k+1}, \bar{\theta}_{k+1})^T$ is

$$h_i(\bar{\mathbf{x}}_r(k+1), \bar{\mathbf{x}}_i(k+1)) = \text{atan2}\left(\frac{-(\bar{\text{x}}_\text{i} - \bar{\text{x}}_{\text{k+1}}) \sin\bar{\theta}_{\text{k+1}} + (\bar{\text{y}}_\text{i} - \bar{\text{y}}_{\text{k+1}}) \cos\bar{\theta}_{\text{k+1}}}{(\bar{\text{x}}_\text{i} - \bar{\text{x}}_{\text{k+1}}) \cos\bar{\theta}_{\text{k+1}} + (\bar{\text{y}}_\text{i} - \bar{\text{y}}_{\text{k+1}}) \sin\bar{\theta}_{\text{k+1}}} \right). \tag{7.4}$$

The Jacobian of the observation model is

$$\begin{aligned} \mathbf{H}_i &= \begin{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{x}}_r(k+1)} & 0 \cdots 0 & \frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{x}}_i(k+1)} & 0 \cdots 0 \end{bmatrix}, \\ \frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{x}}_r(k+1)} &= \begin{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \bar{x}_{k+1}} & \frac{\partial \mathbf{h}_i}{\partial \bar{y}_{k+1}} & \frac{\partial \mathbf{h}_i}{\partial \bar{\theta}_{k+1}} \end{bmatrix}, \qquad \frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{x}}_i(k+1)} = \begin{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \bar{x}_i} & \frac{\partial \mathbf{h}_i}{\partial \bar{y}_i} \end{bmatrix}, \end{aligned} \tag{7.5}$$

where $\partial \mathbf{h}_i / \partial \bar{\theta}_{k+1} = -1$ and

$$\begin{aligned} \frac{\partial \mathbf{h}_i}{\partial \bar{x}_{k+1}} &= -\frac{\bar{y}_{k+1} - \bar{y}_i}{(\bar{x}_{k+1} - \bar{x}_i)^2 + (\bar{y}_{k+1} - \bar{y}_i)^2}, & \frac{\partial \mathbf{h}_i}{\partial \bar{y}_{k+1}} &= \frac{\bar{x}_{k+1} - x_i}{(\bar{x}_{k+1} - \bar{x}_i)^2 + (\bar{y}_{k+1} - \bar{y}_i)^2}, \\ \frac{\partial \mathbf{h}_i}{\partial \bar{x}_i} &= \frac{\bar{y}_{k+1} - \bar{y}_i}{(\bar{x}_{k+1} - \bar{x}_i)^2 + (\bar{y}_{k+1} - \bar{y}_i)^2}, & \frac{\partial \mathbf{h}_i}{\partial \bar{y}_i} &= -\frac{\bar{x}_{k+1} - \bar{x}_i}{(\bar{x}_{k+1} - \bar{x}_i)^2 + (\bar{y}_{k+1} - \bar{y}_i)^2}, \end{aligned} \tag{7.6}$$

and $\mathbf{h}$ and $\mathbf{H}$ collect the information from all the features in the map,

$$\mathbf{h} = (h_1^T, \ldots, h_n^T)^T, \qquad \mathbf{H} = (\mathbf{H}_1^T, \ldots, \mathbf{H}_n^T)^T. \tag{7.7}$$

EKF update

The final map estimate $\hat{\mathbf{x}}(k+1) = \mathbf{x}(k+1|k+1)$ and covariance $\mathbf{P}(k+1|k+1)$ are obtained as

$$\begin{aligned} \hat{\mathbf{x}}(k+1) &= \bar{\mathbf{x}}(k+1) + \mathbf{K}(\mathbf{z}-\mathbf{h}), \qquad & \mathbf{P}(k+1|k+1) &= (\mathbf{I}-\mathbf{KH})\mathbf{P}(k+1|k), \\ \mathbf{K} &= \mathbf{P}(k+1|k)\mathbf{H}^T\mathbf{S}^{-1}, & \mathbf{S} &= \mathbf{HP}(k+1|k)\mathbf{H}^T + \mathbf{R}, \end{aligned} \tag{7.8}$$

where $\nu = \mathbf{z} - \mathbf{h}$ is the innovation, $\mathbf{S}$ is the innovation covariance and $\mathbf{K}$ is the Kalman gain. $\mathbf{z}$ is the vector with all the measurements of the features. Since the observations are taken equal to the predicted measurements, then $\mathbf{z} = \mathbf{h}$ and the innovation ν is zero. Therefore the state vector does not change, $\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1)$. The matrix $\mathbf{R}$ is the covariance of the observation noise and is equal to $\sigma_z^2\mathbf{I}$, where σ_z is the standard deviation of the visual sensor noise.

7.2.3 Cost minimization approach

Before proceeding with the explanation of the proposed strategy, we briefly discuss here alternative methods based on cost minimization. For each of the positions where the robot can move $\mathbf{x} = (x_{k+1}, y_{k+1})$, we define the following cost function $f(\mathbf{x})$,

$$f(\mathbf{x}) = \mathrm{Tr}(\mathbf{P}(k+1|k+1)),$$

where $\mathbf{P}(k+1|k+1)$ is the covariance associated to the expected map given by eq. (7.8). Note that from the analysis performed in the previous section, the robot orientation has no effect on the cost function. Now we use the following gradient descendant algorithm [24] that looks for (global or local) minima of the cost function,

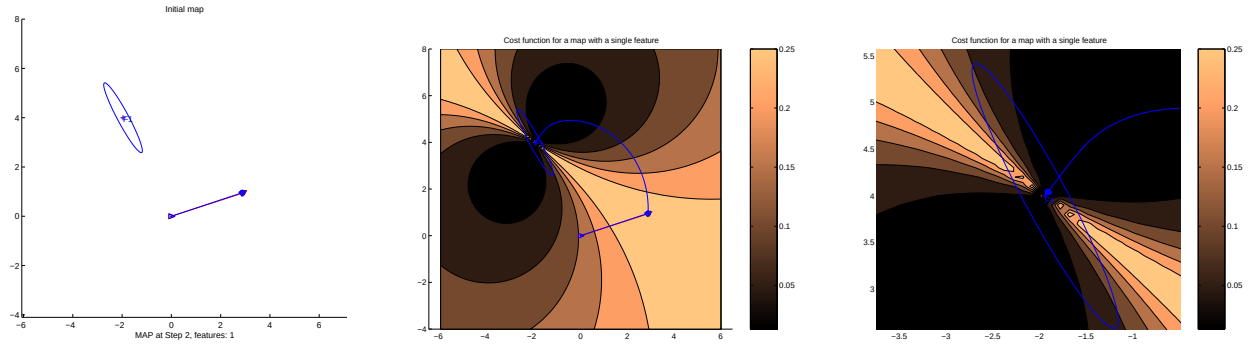
$$\mathbf{x}(0) = (x_k, y_k), \qquad \dot{\mathbf{x}} = -\nabla f(\mathbf{x}),$$

whose discrete version is

$$\mathbf{x}(0) = (x_k, y_k), \qquad \mathbf{x}(t+1) = \mathbf{x}(t) - h\nabla f(\mathbf{x}(t)), \tag{7.9}$$

where (x_k, y_k) is the current robot position, $\mathbf{x}(t)$ is the value of $\mathbf{x}$ at time t, $\nabla f(\mathbf{x}(t)$ is the gradient of f evaluated in $\mathbf{x}(t)$ and h is the time step. In the continuous formulation, a local minimum point is found when $\dot{\mathbf{x}} = 0$, and in the discrete one when $\mathbf{x}(t+1) - \mathbf{x}(t)$ is small enough. We execute this algorithm using a map with a single feature (Fig. 7.2 **(a)**, blue ellipse) and show the cost values for a region around the feature (Figs. 7.2 **(b)** and **(c)**). Darker areas represent lower values of the cost function. The trajectory followed by the robot when it executes the gradient descendant algorithm is shown in blue. As it can be seen, the lowest cost robot positions correspond to the locations we identified as vantage in Section 7.2.1. Moreover, the last part of the robot trajectory follows a direction perpendicular to the major axis of the ellipse, as we did in Section 7.2.1 for selecting vantage locations. If no additional mechanisms are used, the gradient algorithm drives the robot to a final location inside the uncertainty ellipse of the feature (Figs. 7.2 **(b)** and **(c)**), which may lead to ill-conditioned situations which are prone to filter divergence [125].

Therefore, the selection of vantage points discussed in Section 7.2.1 is capable of choosing locations similar to the ones from a gradient descendant, with a much lower computational cost, and with the additional benefit that we can explicitly impose the robot to remain outside the uncertainty region associated to the feature. In Fig. 7.3 we perform a similar analysis for features parameterized in inverse-depth, and conclude that it seems that it would be possible to compute vantage locations also for these representations.



(a) **(b)** **(c)**

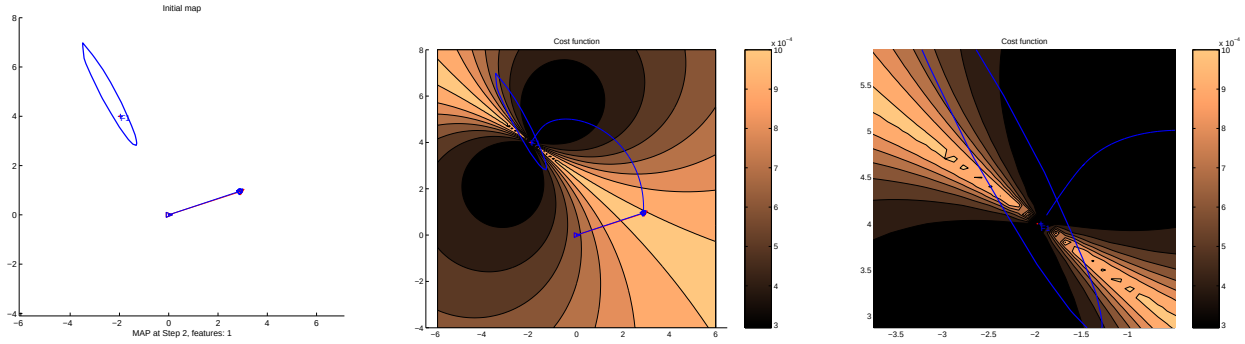
Figure 7.2: Cost function, cartesian parametrization. **(a)**: The initial map with a single feature. The blue ellipse represents the feature estimate, and the blue triangles are the estimates of the robot poses. The ground truth positions for the feature and the robot are displayed in red. **(b)**: The cost function resulting of placing the robot in different positions in the environment, computing the predicted map. Darker regions represent lower costs. The blue arc begining at the last robot pose represent the (x, y) robot positions used by the gradient algorithm to find minima in the cost function. **(c)**: Detail of the final robot position found by the gradient algorithm.

7.2.4 Strategy for improved perception

As previously discussed, we are interested in motion control strategies where robots make decisions based on their local maps, that produce improvements in the global map. The global map of the robot team is as explained in Chapters 3 and 4. Thus, the strategy we present here does not necessarily have to be executed after the map merging process. Instead, it can be executed after the localization (Chapter 6) data association processes (Chapter 5), in parallel to the execution of the map fusion. Recall that given the n independent local maps with mean $\hat{\mathbf{x}}_i$ and a covariance matrix $\Sigma_i = \mathbf{P}_i(k+1|k+1)$, for $i \in \{1, \ldots, n\}$, the mean and covariance matrix of the global map of the robot team is given by (Section 3, eq. (3.4))

$$\hat{\mathbf{x}}_G = \left(\sum_{i=1}^{n} H_i^T \Sigma_i^{-1} H_i \right)^{-1} \sum_{i=1}^{n} H_i^T \Sigma_i^{-1} \hat{\mathbf{x}}_i, \qquad \Sigma_G = \left(\sum_{i=1}^{n} H_i^T \Sigma_i^{-1} H_i \right)^{-1}, \tag{7.10}$$

where H_i are the observations matrices relating the features observed by the robots. From this expression, we can see that feature estimates with smaller covariances have



(a) **(b)** **(c)**

Figure 7.3: Cost function, inverse-depth parametrization. **(a)**: The initial map with a single feature. The blue ellipse represents the feature estimate, and the blue triangles are the estimates of the robot poses. The ground truth positions for the feature and the robot are displayed in red. **(b)**: The cost function resulting of placing the robot in different positions in the environment, computing the predicted map. Darker regions represent lower costs. The blue arc beginning at the last robot pose represent the (x, y) robot positions used by the gradient algorithm to find minima in the cost function. **(c)**: Detail of the final robot position found by the gradient algorithm.

larger information matrices and thus they greatly influence their estimates in the global map. Therefore, a precise estimate of a feature can be obtained if, at least, one robot has observed it with high precision. This observation motivates our strategy.

Aggregate objective function

The global cost function F measures the best contributions for the estimate of every feature

$$F(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_{j=1}^{m} \min_i f_{ij}(\mathbf{x}_i)), \tag{7.11}$$

where $\mathbf{x}_i$ is the next position of the robot i at time $k+1$, for $i \in \{1, \ldots, n\}$ and f_{ij} is the individual cost for the feature j in the local map of the robot i when the next position of the robot i is $\mathbf{x}_i$.

The individual cost functions $f_{ij}(\mathbf{x}_i)$ evaluate the covariance matrix $\mathbf{P}_i(\mathbf{x}_i)$ of the expected map of robot i when it moves to the location $\mathbf{x}_i$. They measure the uncertainty of the sub-matrix $[\mathbf{P}_i(\mathbf{x}_i)]_{jj}$ within $\mathbf{P}_i(\mathbf{x}_i)$ associated to the feature j. There exist many metrics for measuring the uncertainty in a covariance matrix. [148] compares metrics based on the determinant, eigenvalues and trace, concluding that all of them perform properly. Here, we select the trace. The individual cost f_{ij} is

$$f_{ij}(\mathbf{x}_i)) = \mathrm{Tr}([\mathbf{P}_i(\mathbf{x}_i)]_{jj}). \tag{7.12}$$

Table 7.1: Predicted costs for the candidate next motions for robot i

	feat 1	$\cdots$	feat m
$\mathbf{x}_i^1$	$f_{i1}(\mathbf{x}_i^1)$	$\cdots$	$f_{im}(\mathbf{x}_i^1)$
$\vdots$	$\vdots$		$\vdots$
$\mathbf{x}_i^K$	$f_{i1}(\mathbf{x}_i^K)$	$\cdots$	$f_{im}(\mathbf{x}_i^K)$

Table 7.2: Global cost for a specific selection of next robot poses

	feat 1	$\cdots$	feat m
$\mathbf{x}_1^{l_1}$	$f_{11}(\mathbf{x}_1^{l_1})$	$\cdots$	$f_{1m}(\mathbf{x}_1^{l_1})$
$\vdots$	$\vdots$		$\vdots$
$\mathbf{x}_n^{l_n}$	$f_{n1}(\mathbf{x}_n^{l_n})$	$\cdots$	$f_{nm}(\mathbf{x}_n^{l_n})$
min	$\min_i f_{i1}(\mathbf{x}_i^{l_i})$	$\cdots$	$\min_i f_{im}(\mathbf{x}_i^{l_i})$

Coordination strategy

The proposed strategy for motion coordination consists of an iterative algorithm where, at every time step, the team of robot performs the following actions:

1. **Optimal position for feature observation:** Every robot i computes the best position for observing the K features with higher covariances in its local map, where $K \leq m_i$ for $i \in \{1, \ldots, n\}$ is adjusted depending on the performance requirements. As a result, robot $i \in \{1, \ldots, n\}$, obtains K next position candidates which we express as $\mathbf{x}_i^1, \cdots, \mathbf{x}_i^K$.

2. **Map prediction:** For all the candidate next positions $\mathbf{x}_i^1, \cdots, \mathbf{x}_i^K$, robot i computes the predicted map and evaluates the local cost function $f_{ij}(\mathbf{x}_i^l)$ for all the features, $l \in \{1, \ldots, K\}$, $j \in \{1, \ldots, m\}$, $i \in \{1, \ldots, n\}$. If an estimate of the feature j cannot be found in the local map of robot i, then we set $f_{ij}(\mathbf{x}_i^l) = \infty$ for all $l \in \{1, \ldots, K\}$. Every robot i can construct a table with the values of f_{ij} (see Table 7.1).

3. **Minimization of the global cost:** Given a selected combination of next robot poses $\mathbf{x}_1^{l_1}, \cdots, \mathbf{x}_n^{l_n}$, its associated global cost is computed as $\sum_{j=1}^{m} \min_i f_{ij}(\mathbf{x}_i^{l_i}))$. This is equivalent to sum the values in the last row (min) in Table 7.2. The best robot-vantage location assignment is the one minimizing the global cost F (see equation (7.11)). Every robot can compute this value, based on the information received from the other robots, and on its own data.

4. **Motion and observation:** Once the best motions have been decided for the team, the robots move to the new positions, they observe the environment and update their local maps.

In the coordination strategy presented so far, the computationally expensive operations, consisting of finding vantage locations and computing the cost function, are carried out locally by each robot, which it very interesting. On the other hand, the process of exchanging data and negotiating on the next robot poses, implicitly assumes all-to-all communications. A possible solution would consist of clustering together features that simultaneously benefit from a robot being placed nearby them. Then, a problem of robot-to-cluster assignment (assign different robots to different clusters of features while minimizing the total cost) would be addressed, instead of the robot-to-feature strategy presented along this section. The robot-to-cluster assignment problems can be solved by the use of auction algorithms [23, 25], for which exist distributed implementations for robot networks [90]. Future extensions of the work presented in this section can be in the line of these discussed ideas about clustering of features.

7.3 Improving the Convergence Speed

In multi-robot control problems, the convergence properties and the convergence speed of the system depend on the algebraic connectivity of the graph. Several examples appear in the distributed methods proposed in this thesis. The convergence speed of the static map merging algorithm presented in Chapter 3 depends on the second eigenvalue of W with the largest absolute value $\lambda_2(W)$ associated to the Metropolis weight matrix W (Section 3.7, eq. (3.20)). The localization methods in Chapter 6, Sections 6.3 and 6.5 also present a convergence speed depending on $\lambda_2(W)$. The rate of convergence of the dynamic map merging algorithm in Chapter 4 depends on the second smallest $\lambda_2(L_W)$ and the largest $\lambda_n(L_W)$ eigenvalues of the Laplacian matrix L_W (Section 4.3). In this section, we present a novel distributed algorithm where the robots estimate the leading eigenvalue of a weight matrix or a Laplacian, obtaining a more accurate estimate at each iteration. We extend this method to allow the computation of the algebraic connectivity of the Laplacian $\lambda_2(L_W)$. Note that for the Metropolis weight matrix W, robots can compute the algebraic connectivity $\lambda_2(L_W)$ of the associated Laplacian $L_W = I - W$, and then get $\lambda_2(W) = 1 - \lambda_2(L_W)$. This algorithm relies on the distributed computation of the powers of the adjacency matrix. We provide proofs of convergence and convergence rate of the algebraic connectivity estimation algorithm. In addition, we give upper and lower bounds at each iteration of the estimated algebraic connectivity. We apply this method to an event-triggered consensus scenario, where the most recent estimate of the algebraic connectivity is used for adapting the behavior of the average consensus algorithm. We show that both processes can be executed in parallel, i.e., the robots do not need to wait for obtaining a good estimate of the algebraic connectivity before starting the averaging algorithm. We use the notation defined in Table 7.3.

Consider a set of $n \in \mathbb{N}$ robots with indices $i \in \{1, \dots, n\}$. The robots can exchange information with nearby robots. This information can be represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, n\}$ are the robots, and $\mathcal{E}$ are the edges. There is an edge $(i, j) \in \mathcal{E}$ between robots i and j if they can exchange data. We say a $n \times n$ matrix C is *compatible* with $\mathcal{G}$ if $C_{ij} = 0$ iff $(i, j) \notin \mathcal{E}$ for $j \neq i$; note that we let the elements in the diagonal C_{ii} be either equal or different than 0. The adjacency matrix $\mathcal{A} \in \{0, 1\}^{n \times n}$

Table 7.3: Notation.

Indices			
n	Number of robots.	k	Iteration number, $k \in \mathbb{N}$.
i, j	Robot indices.	t	Time, $t \in \mathbb{R}_{t \geq 0}$.

Matrix operations, eigenvalues and eigenvectors	
A_{ij}, $[A]_{ij}$	(i,j) entry of matrix A.
$\text{diag}(b_1, \ldots, b_r)$	matrix A with $A_{ii} = b_i$ and $A_{ij} = 0$.
$\lambda_i(A)$, $\mathbf{v}_i(A)$	i^{th} eigenvalue and eigenvector of A.
λ_A	$\text{diag}(\lambda_1(A), \ldots, \lambda_r(A))$.
V_A	$[\mathbf{v}_1(A), \ldots, \mathbf{v}_r(A)]$.
$\|A\|_\infty$	Induced $\infty-$norm of A, $\max_i \sum_{j=1}^{n} \lvert A_{ij} \rvert$.
$\|A\|_2$	Spectral norm of A, $\max_i \sqrt{\lambda_i(A^T A)}$.
$\rho(A)$	Spectral radius of A, $\max_i \lvert \lambda_i(A) \rvert$.

Special matrices	
$\mathbf{I}_r$	$r \times r$ identity matrix.
$\mathbf{0}_r$, $\mathbf{1}_r$	Column vectors with the r entries equal to 0 and 1.
$\mathcal{A}$	Adjacency matrix of the graph.
$\mathcal{L}$	Laplacian matrix of the graph, $\mathcal{L} = \text{diag}(\mathcal{A}\mathbf{1}) - \mathcal{A}$.
$\lambda_\star(\mathcal{L})$	Algebraic connectivity, the second-smallest $\lambda_i(\mathcal{L})$.

of $\mathcal{G}$ is defined by

$$\mathcal{A}_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}, \qquad \text{for } i, j \in \mathcal{V}. \tag{7.13}$$

We assume that the undirected communication graph $\mathcal{G}$ is connected. We use $\mathcal{N}_i$ for the set of neighbors of a robot i with whom i can exchange data, $\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E}\}$, and we let d_i be the *degree* of node i defined as the cardinality of $\mathcal{N}_i$, and $d_{\max} = \max_{i \in \mathcal{V}} d_i$.

The Laplacian matrix $\mathcal{L} \in \mathbb{R}^{n \times n}$ of $\mathcal{G}$ is the positive-semidefinite matrix

$$\mathcal{L} = \text{diag}(\mathcal{A}\mathbf{1}) - \mathcal{A}. \tag{7.14}$$

Note that both $\mathcal{A}$ and $\mathcal{L}$ are compatible with the graph. In this chapter we sort the eigenvalues of $\mathcal{L}$ as follows,

$$\lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L}). \tag{7.15}$$

The Laplacian matrix $\mathcal{L}$ has the following well known properties, see e.g., [102]:

(i) It has an eigenvector $\mathbf{v}_1(\mathcal{L}) = \mathbf{1}/\sqrt{n}$ with associated eigenvalue $\lambda_1(\mathcal{L}) = 0$, $\mathcal{L}\mathbf{1}/\sqrt{n} = \mathbf{0}$;

(ii) When the graph $\mathcal{G}$ is connected, then all the other eigenvalues are strictly greater than 0,

$$0 = \lambda_1(\mathcal{L}) < \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L}); \text{ and}$$

(iii) Its eigenvalues are upper-bounded by $\lambda_n(\mathcal{L}) \leq 2d_{\max}$.

The *algebraic connectivity* of $\mathcal{G}$ denoted by $\lambda_\star(\mathcal{L})$ is the second-smallest eigenvalue $\lambda_2(\mathcal{L})$ of the Laplacian $\mathcal{L}$.

7.3.1 Consensus protocol and event-based control

Consider each robot $i \in \mathcal{V}$ has single-integrator dynamics

$$\dot{x}_i(t) = u_i(t), \text{ with } x_i(t), u_i(t) \in \mathbb{R}, \tag{7.16}$$

where u_i denotes the control input at robot i given by

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)). \tag{7.17}$$

With stack vectors $x = [x_1, \ldots, x_n]^T$, $u = [u_1, \ldots, u_n]$,

$$\dot{x}(t) = -\mathcal{L}x(t), \qquad x(0) = x_0. \tag{7.18}$$

A well known result [103] is that if $\mathcal{G}$ is undirected and connected, then the previous algorithm globally asymptotically solves the average consensus problem, i.e.,

$$\lim_{t \to \infty} x_i(t) = \sum_{i=1}^{n} x_i(0)/n. \tag{7.19}$$

However in general it is necessary to implement the continuous-time law on a digital platform. This has motivated the research on event-based and time-scheduled control. It is of special interest the triggered-based control method in [124], where each robot monitors its own state $x_i(t)$ continuously. However, the robots do not have continuous communication but instead propagate their most recent states at some time instances. Also the control law is piecewise constant. The following rule given in [124, Theorem 4] allows each robot i to locally determine when to trigger and transmit its new state. Let $e_i(t)$ be the measurement error at robot i containing the difference between its actual current state $x_i(t)$ at time t and the last transmitted one $\hat{x}_i(t)$,

$$e_i(t) = \hat{x}_i(t) - x_i(t), \qquad \text{for } i \in \mathcal{V}, \tag{7.20}$$

with stack vector $e(t) = [e_1(t), \ldots, e_n(t)]^T$. Let c_1, α be constants satisfying

$$c_1 > 0 \text{ and } 0 < \alpha < \lambda_\star(\mathcal{L}). \tag{7.21}$$

The authors define the time-dependent trigger function

$$f_i(t, e_i(t)) = |e_i(t)| - c_1 e^{-\alpha t}, \tag{7.22}$$

where an event for robot i is triggered as soon as $f_i(t, e_i(t)) > 0$ resulting in robot i sending its most recent state $\hat{x}_i$ to its neighbors. Whenever an event is triggered, i.e.,

robot i transmitting its new state $x_i(t)$ or receiving an updated state $x_j(t)$ from one of its neighbors, it (i) updates its control-law immediately, and (ii) reevaluates the trigger function and computes the next triggering time. As stated by [124, Theorem 4], for connected graphs if robots execute the previous procedure, then for all initial conditions x_0 the overall system converges to average consensus asymptotically. Furthermore the closed-loop system does not exhibit Zeno-behavior.

Note however that condition (7.21) requires all the robots to know $\lambda_\star(\mathcal{L})$ or a lower bound of it.

Problem 7.3.1. *Our goal is to design distributed algorithms to allow the robots to compute $\lambda_\star(\mathcal{L})$ (or a lower bound of) in a distributed fashion. Then, we modify the trigger function so that instead of using a fixed α, robots adapt this value depending on the most recent and accurate estimate of the algebraic connectivity $\lambda_\star(\mathcal{L})$.*

7.3.2 Distributed computation of the algebraic connectivity

First of all, we present a novel algorithm for computing the algebraic connectivity $\lambda_\star(\mathcal{L})$ of the graph in a distributed fashion. This algorithm relies on the observation that (i) the induced infinite norm of a matrix $\|C\|_\infty$ can be easily computed in a distributed fashion with a $max-$consensus method, provided that each robot knows a row of this matrix; and that (ii) $\|C^k\|_\infty^{\frac{1}{k}}$ successively approaches the spectral radius $\rho(C)$ of matrix C. This allows us to propose a method for estimating the algebraic connectivity that is suited for distributed implementation.

We start this section presenting some results about the computation of the powers of matrices which are *compatible* with the graph.

Distributed computation of power of matrices

We show that if matrix C is *compatible* with the graph structure, the computation of its powers can be done in a distributed fashion. Our discussion refers to fixed undirected communication graphs, although the method can be easily extended to time-varying graphs. The presented algorithm was originally proposed in [10] for adjacency matrices defined by blocks. Here we propose an improved version that does not require the knowledge of n.

Algorithm 7.3.2 (Basic Distributed Power Computation)**.** *Let each robot $i \in \mathcal{V}$ maintain an estimate $\hat{C}_{ij}(k)$ of the (i,j) entries of the $k-$th power of C, $[C^k]_{ij}$, for all $j \in \mathcal{V}$. At $k = 0$, robot i initializes its variables $\hat{C}_{ij}(k)$ with*

$$\hat{C}_{ii}(0) = 1, \text{ and } \hat{C}_{ij}(0) = 0 \text{ for } j \in \mathcal{V} \setminus \{i\}. \tag{7.23}$$

At each $k \geq 1$, robot i updates these variables as follows,

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j}(k), \text{ for } j \in \mathcal{V}. \tag{7.24}$$

Note that this algorithm is distributed and each robot i updates its variable using exclusively information from its neighbors $j \in \mathcal{N}_i$ and from itself. It requires each robot i to store n scalars and to exchange n scalars at each iteration k. We discuss now the outcomes of this algorithm.

Proposition 7.3.3. *When C is compatible with the graph, the outcomes of algorithm* (7.24) *at each step $k \geq 0$ are exactly the entries of the $k-$th power of C, C^k.*

Proof. Each robot i maintains exactly the $i-$th row of C^k. For $k = 0$, it is straightforward to see that eq. (7.23) gives the identity matrix $\mathbf{I}$ which is exactly C^0. For $k \geq 1$, we consider the explicit expression for $C^{k+1} = CC^k$, where each (i, j) entry is given by

$$[C^{k+1}]_{ij} = \sum_{j'=1}^{n} C_{ij'}[C^k]_{j'j}. \tag{7.25}$$

Since C is compatible with the graph, then $C_{ij'} = 0$ for $j' \notin \mathcal{N}_i \cup \{i\}$ and the previous expression gives

$$[C^{k+1}]_{ij} = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'}[C^k]_{j'j}, \tag{7.26}$$

which is exactly what algorithm (7.24) does. □

Observe that the previous algorithm does not compute an estimate of the centralized one. Instead, it exactly computes the same solution as a centralized system, but distributing the operations over the robots. Observe that it remains valid if the communication graph is time-varying, in which case each robot i computes

$$[\hat{C}(k)]_{ij} = [C(k)C(k-1)\dots C(0)]_{ij}, \text{ for } j \in \mathcal{V}.$$

The main limitation of the previous procedure is that it assumes that in the initial phase (eq. (7.23)) each robot knows the total amount of other robots in the network n, and that at each step k (eq. (7.24)) it knows the identities j, j' associated to each piece of data $\hat{C}_{j'j}(k)$. We show here that the algorithm can be slightly modified so that the previous requirement is not necessary. We only impose the assumption that each robot i has assigned a unique identifier $ID(i)$, e.g., its IP address.

Algorithm 7.3.4 (Distributed Power Computation)**.** *Let each robot $i \in \mathcal{V}$ maintain a set $l_i(k)$ with the identifiers $ID(j)$ of the robots j it has discovered, and an estimate $\tilde{C}_{ij}(k)$ of the (i, j) entries of the $k-$th power of C, $[C^k]_{ij}$, associated to these robots j, $ID(j) \in l_i(k)$.*

At $k = 0$, each robot $i \in \mathcal{V}$ initializes a single variable $\tilde{C}_{ii}(k)$ and a single identifier,

$$\tilde{C}_{ii}(0) = 1, \qquad l_i(0) = \{ID(i)\}, \tag{7.27}$$

and sends this data to its neighbors $\mathcal{N}_i$.

At each step $k \geq 1$, robot i first looks for new robots in the information $l_j(k)$ received from its neighbors, and updates its identifiers $l_i(k)$ accordingly,

$$l_i(k+1) = \bigcup_{j \in \mathcal{N}_i \cup \{i\}} l_j(k). \tag{7.28}$$

Then, robot i creates a new variable $\tilde{C}_{ij}(k)$ initialized with 0, $\tilde{C}_{ij}(k) = 0$, for each recently discovered robot j,

$$ID(j) \in l_i(k+1) \text{ and } ID(j) \notin l_i(k).$$

Finally, robot i and updates all its variables $\tilde{C}_{ij}(k)$, for $ID(j) \in {}_i(k+1)$, by

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in_{j'}(k)} C_{ij'} \tilde{C}_{j'j}(k), \tag{7.29}$$

and sends these variables $\tilde{C}_{ij}(k)$, for $ID(j) \in l_i(k)$, to its neighbors as well as its discovered identifiers $l_i(k)$.

Proposition 7.3.5. *Let us define for each robot $i \in \mathcal{V}$ and each step $k \geq 0$ variables $\tilde{C}_{ij}(k) = 0$ for all $ID(j) \notin l_i(k)$. Then, when C is compatible with the graph, the outcomes of algorithm (7.29) are exactly the outcomes of (7.24). As a result, they are exactly the entries of the $k-$th power of C, C^k.*

Proof. We first consider Algorithm 7.3.2 together with the robot identifier management rule, with initialization

$$l_i(0) = \{ID(i)\}, \qquad \hat{C}_{ii}(0) = 1, \qquad \hat{C}_{ij}(0) = 0, \text{ for } j \neq i, \tag{7.30}$$

and update rule

$$l_i(k+1) = \bigcup_{j \in \mathcal{N}_i \cup \{i\}} \{l_j(k)\}, \quad \text{and } \hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j}, \text{ for all } j \in \mathcal{V}. \tag{7.31}$$

We want to show that if $j \notin l_i(k)$ then the element $\hat{C}_{ij}$ is zero. This is proved by induction. It is true for $k = 0$, see eq. (7.30). Let us assume that for k it is true that, for all $i \in \mathcal{V}$, if $j \notin l_i(k)$ then $\hat{C}_{ij} = 0$. Consider a j which at $k+1$ satisfies $j \notin l_i(k+1)$. By eq. (7.31) it means that $j \notin \bigcup_{j' \in \mathcal{N}_i \cup \{i\}} \{l_{j'}(k)\}$ and therefore for all $j' \in \mathcal{N}_i \cup \{i\}$ the element $\hat{C}_{j'j} = 0$. Then, the update rule (7.31) gives

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} 0 = 0. \tag{7.32}$$

Now we prove that the outcomes $\tilde{C}_{ij}(k)$ of algorithm (7.3.4) are exactly equal to $\hat{C}_{ij}(k)$ for all $k \geq 0$, $i \in \mathcal{V}$ and all $j \in l_i(k)$. Note that for all $k \geq 0$ all $i \in \mathcal{V}$ and all $j \notin l_i(k)$ the elements $\tilde{C}_{ij}(k)$ do not exist whereas $\hat{C}_{ij}(k) = 0$ as shown above. This can be shown by induction by taking into account the following issue. We pay attention to eq. (7.29) for $ID(j) \in {}_i(k+1)$,

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in_{j'}(k)} C_{ij'} \tilde{C}_{j'j}(k). \tag{7.33}$$

For $j \notin l_{j'}(k)$ we have $\hat{C}_{j'j}(k) = 0$. If $\hat{C}_{j'j}(k) = \tilde{C}_{j'j}(k)$ at k, then at $k+1$ we have

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in_{j'}(k)} C_{ij'} \hat{C}_{j'j}(k) + \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \notin_{j'}(k)} C_{ij'} 0, \tag{7.34}$$

which is exactly the update rule for $\hat{C}_{ij}(k+1)$. □

Therefore Algorithm 7.3.4 provides each robot i with all the elements $[C^k]_{ij}$ for $j \in \mathcal{V}$ of the $i-th$ row of the power matrix C^k. It is fully distributed and only requires each robot i to have a unique identifier $ID(i)$.

The results presented so far hold for both fixed and time-varying graphs. Now we show that in addition, for fixed graphs, the previous method can be used for obtaining the number of robots n.

Proposition 7.3.6. *For each robot $i \in \mathcal{V}$, let k_i be the first instant for which $l_i(k) = l_i(k-1)$,*

$$k_i = \min\{k \mid l_i(k) = l_i(k-1)\}. \tag{7.35}$$

Then, $n = |l_i(k_i)|$.

Proof. Note that $l_i(k-1)$ contains the identifiers of the $(k-1)-$hop neighbors of i. By the definition of a path, if there are no new robots at distance k, then there cannot be new robots at distances greater than k. Therefore $l_i(k-1)$ already contains the identifiers of all the robots that are connected with i. Since the graph is connected, these robots are all the robots in the network and $n = |l_i(k_i)|$. □

Distributed computation of the spectral radius

Now we present a distributed algorithm that allows the computation of the spectral radius of a symmetric matrix C compatible with the graph. It relies on the observation that, for any induced norm $\|.\|$, [67, Chapter 5.6]

$$\rho(C) \leq \|C\|, \text{ and } \rho(C) = \lim_{k\to\infty} \|C^k\|^{\frac{1}{k}}. \tag{7.36}$$

We propose to use the induced $\infty-$norm $\|.\|_\infty$, which is the maximum absolute row sum of the matrix,

$$\|C^k\|_\infty = \max_{i\in\mathcal{V}} \sum_{j=1}^{n} |[C^k]_{ij}|, \tag{7.37}$$

since it can be easily computed by the robots using a distributed $max-$consensus algorithm provided that each robot i knows the $i-th$ row of C^k.

Algorithm 7.3.7 (Distributed Spectral Radius)**.** *Consider the robots executing Algorithm 7.3.4 for a symmetric matrix C compatible with the graph. At each iteration k we define for each robot the sum $c_i(k)$ of the absolute values of its variables $\tilde{C}_{ij}(k)$,*

$$c_i(k) = \sum_{ID(j)\in l_i(k)} |\tilde{C}_{ij}(k)|. \tag{7.38}$$

Robots execute the following $\max-$consensus [134] on their variables $c_i(k)$,

$$\beta_i(k) = c_i(k), \qquad \beta_i(k+\tau+1) = \max_{j\in\mathcal{N}_i\cup\{i\}} \beta_j(k+\tau), \tag{7.39}$$

which finishes after $T = \text{diam}(\mathcal{G})$ *communication rounds with variables* $\beta_i(T)$ *at all the robots* $i \in \mathcal{V}$ *containing the maximum of the inputs* $c_i(k)$ *over all the network,*

$$\beta_1(k+T) = \cdots = \beta_n(k+T) = \max_{i \in \mathcal{V}} c_i(k). \tag{7.40}$$

The spectral radius $\beta_i^*(k)$ *estimated by each robot* i *at step* $k \geq 1$ *is given by*

$$\beta_i^*(k) = (\beta_i(k+T))^{\frac{1}{k}} = (\max_{j \in \mathcal{V}} c_j(k))^{\frac{1}{k}}. \tag{7.41}$$

Observe that this computation of $c_i(k)$ in eq. (7.38) is local to each robot i, since it maintains the $i-th$ row of C^k. Note that the estimated spectral radius $\beta_i^*(k)$ associated to step k is available at the robots T iterations later (at iteration $k+T$). However, the max$-$consensus iterations (7.39) are executed independently (in parallel) to the Algorithm 7.3.4. This means that robots do not have to wait T iterations for the max$-$consensus to converge before executing a new iteration of Algorithm 7.3.4.

Now we present a result that establishes the convergence of the previous algorithm to the spectral radius of the matrix C.

Theorem 7.3.8. *Consider each robot* i *executing Algorithm 7.3.7 with a symmetric matrix* C *compatible with the graph. Then, as* $k \to \infty$ *all the variables* $\beta_i^*(k)$ *converge to the spectral radius* $\rho(C)$ *of matrix* C*,*

$$\lim_{k \to \infty} \beta_i^*(k) = \rho(C), \text{ for all } i \in \mathcal{V}, \tag{7.42}$$

and for all $k \geq 1$*,*

$$(\sqrt{n})^{\frac{-1}{k}} \beta_i^*(k) \leq \rho(C) \leq \beta_i^*(k). \tag{7.43}$$

Proof. First note that Algorithm 7.3.7 computes the $k-$th root of the induced infinite norm of C^k. Since we showed that $\tilde{C}_{ij}(k) = 0$ for $ID(j) \notin l_i(k)$, then $c_i(k)$ in eq. (7.38) is exactly the absolute sum of the $i-th$ row of C^k. The max$-$consensus (7.39) provides each robot with the induced infinite norm of C^k. Therefore, $\beta_i^*(k)$ in eq. (7.41) equals $\beta_i^*(k) = (\|C^k\|_\infty)^{\frac{1}{k}}$, which combined with eq. (7.36) gives

$$\rho(C) = \lim_{k \to \infty} \|C^k\|_\infty^{\frac{1}{k}} = \lim_{k \to \infty} \beta_i^*(k), \tag{7.44}$$

as stated in eq. (7.42).

Now we focus on the inequalities in eq. (7.43). From (7.36),

$$\rho(C) = (\rho(C^k))^{\frac{1}{k}} \leq \|C^k\|_\infty^{\frac{1}{k}} = \beta_i^*(k), \tag{7.45}$$

which gives the right part in eq. (7.43). Since matrix C is symmetric, then its spectral norm $\|C\|_2 = \max_i \sqrt{\lambda_i(C^2)}$ equals its spectral radius $\rho(C) = \max_i |\lambda_i(C)|$,

$$\rho(C) = \|C\|_2 = \|C^k\|_2^{\frac{1}{k}}. \tag{7.46}$$

The spectral $\|C^k\|_2$ and induced infinite $\|C^k\|_\infty$ norms of a matrix C^k are related by $(\sqrt{n})^{-1}\|C^k\|_\infty \leq \|C^k\|_2$, [67, Chapter 5.6], which combined with eq. (7.46) gives

$$(\sqrt{n})^{-\frac{1}{k}}\beta_i^*(k) = (\sqrt{n})^{-\frac{1}{k}}\|C^k\|_\infty^{\frac{1}{k}} \leq \|C^k\|_2^{\frac{1}{k}} = \rho(C), \tag{7.47}$$

as stated in eq. (7.43) and the proof is completed. □

Observe that in the previous process the robots are not required to know n. Now we are ready to show how the algebraic connectivity of the graph $\lambda_\star(\mathcal{L})$ can be computed by using the previous algorithm.

Distributed computation of the algebraic connectivity

The method roughly consists of transforming the Laplacian $\mathcal{L}$ matrix associated to the undirected and connected graph $\mathcal{G}$ into a new matrix C where the algebraic connectivity $\lambda_\star(\mathcal{L})$ becomes the leading eigenvalue, i.e., the spectral radius $\rho(C)$ is an expression that depends on $\lambda_\star(\mathcal{L})$. Then, the procedure presented in the previous sections is used for computing $\rho(C)$ in a distributed fashion. The estimated algebraic connectivity is finally obtained by reversing the transformation.

Proposition 7.3.9. *Consider the following deflated version of the Perron matrix [102, 151, 154] of the Laplacian,*

$$C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n. \tag{7.48}$$

The relationship between the eigenvalues of C and $\mathcal{L}$ is

$$\lambda_1(C) = 0, \qquad \lambda_i(C) = 1 - \beta\lambda_i(\mathcal{L}), \text{ for } i \in \{2, \ldots, n\}. \tag{7.49}$$

$\rho(C)$ is associated to the algebraic connectivity $\lambda_\star(\mathcal{L})$ by

$$\lambda_\star(\mathcal{L}) = (1 - \rho(C))/\beta, \qquad \text{if } 0 < \beta < 1/\lambda_n(\mathcal{L}). \tag{7.50}$$

The previous result can be easily proven as follows. Note that similar results can be found at [102, 151, 154].

Proof. First of all, we show that $\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n$ has the same eigenvalues as $\mathcal{L}$ for $i \in \{2, \ldots, n\}$, but the first one that is equal to r,

$$\lambda_1(\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n) = r, \qquad \text{and } \lambda_i(\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n) = \lambda_i(\mathcal{L}) \text{ for } i \in \{2, \ldots, n\}. \tag{7.51}$$

Consider the following orthogonal matrix $V_\mathcal{L}$ composed of eigenvectors of $\mathcal{L}$,

$$V_\mathcal{L} = \left[\mathbf{1}/\sqrt{n}, \mathbf{v}_2(\mathcal{L}), \ldots, \mathbf{v}_n(\mathcal{L})\right] = \left[\mathbf{1}/\sqrt{n}, \tilde{V}_\mathcal{L}\right], \tag{7.52}$$

which exists since $\mathcal{L}$ is symmetric. Then,

$$V_{\mathcal{L}}^T \mathcal{L} V_{\mathcal{L}} = \lambda_{\mathcal{L}} = \text{diag}(0, \lambda_2(\mathcal{L}), \ldots, \lambda_n(\mathcal{L})), \tag{7.53}$$

We apply the same operation to $\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n$ and get

$$\begin{aligned} V_{\mathcal{L}}^T \left(\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n\right) V_{\mathcal{L}} &= \lambda_{\mathcal{L}} + r \begin{bmatrix} \mathbf{1}^T\mathbf{1}/n & \mathbf{1}^T\tilde{V}_{\mathcal{L}}/\sqrt{n} \\ \tilde{V}_{\mathcal{L}}^T\mathbf{1}/\sqrt{n} & \tilde{V}_{\mathcal{L}}^T\mathbf{1}\mathbf{1}^T\tilde{V}_{\mathcal{L}} \end{bmatrix} = \lambda_{\mathcal{L}} + r \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \text{diag}(r, \lambda_2(\mathcal{L}), \ldots, \lambda_n(\mathcal{L})), \end{aligned} \tag{7.54}$$

since $\mathbf{1}^T\mathbf{1} = n$ and $\mathbf{1}^T\tilde{V}_{\mathcal{L}} = \mathbf{0}$, what yields eq. (7.51). Now note that the eigenvalues of the Laplacian $\mathcal{L}$ and of the deflated Perron matrix C in eq. (7.48) satisfy

$$\lambda_i(C) = 1 - \beta\lambda_i\left(\mathcal{L} + (1/\beta)\mathbf{1}\mathbf{1}^T/n\right), \text{ for } i \in \mathcal{V}, \tag{7.55}$$

which combined with eq. (7.51), with $r = 1/\beta$, gives the relationship in eq. (7.49).

Now let us define β as follows

$$\beta = \varepsilon/\lambda_n(\mathcal{L}), \quad \text{for some } \varepsilon \in (0, 1), \tag{7.56}$$

and express the eigenvalues of C accordingly,

$$\begin{aligned} &\lambda_1(C) = 0, \text{ and for } i \in \{2, \ldots, n\}, \\ &\lambda_i(C) = 1 - \varepsilon\lambda_i(\mathcal{L})/\lambda_n(\mathcal{L}). \end{aligned} \tag{7.57}$$

Recall that $\lambda_n(\mathcal{L}) \geq \lambda_i(\mathcal{L}) > 0$ for all $i \in \{2, \ldots, n\}$. Then,

$$1 > \lambda_2(C) \geq \cdots \geq \lambda_n(C) > \lambda_1(C) = 0, \tag{7.58}$$

and the eigenvalue of C with largest absolute value is $\lambda_2(C)$, what concludes the proof. □

The previous deflated Perron matrix $C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n$ is not compatible with the graph and thus Algorithm 7.3.7 cannot be immediately applied in a distributed fashion. Note however that, since $\mathbf{1}/\sqrt{n}$ is the eigenvector $\mathbf{v}_1(C)$ of C associated to the eigenvalue $\lambda_1(C) = 0$, then, for all $k \geq 1$,

$$C^k = (\mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n)^k = (\mathbf{I} - \beta\mathcal{L})^k - \mathbf{1}\mathbf{1}^T/n, \tag{7.59}$$

where matrix $\mathbf{I} - \beta\mathcal{L}$ is compatible with the graph. We propose to use a variation (Algorithm 7.3.10) of Algorithm 7.3.7.

Before presenting Algorithm 7.3.10, we discuss some issues regarding the number of robots n. Note that the number of robots n is used in the computation of β. In case the robots do not know n from the beginning, they can compute $\beta = \varepsilon/(2d_{\max})$, which satisfies $\beta < 1/\lambda_n(\mathcal{L})$ as in Proposition 7.3.9 by executing a max−consensus algorithm on the robots degrees in an initial phase. Once β has been computed, robots can start Algorithm 7.3.10. At each step k of Algorithm 7.3.10, robots can always execute Algorithm 7.3.4 for computing the powers of matrix $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. However, they can only execute eqs. (7.61)-(7.63) for getting the output $\hat{\lambda}_i(k)$ when they know n. At each step k robots use Proposition 7.3.6 to find out if they have already found n and thus if they can proceed with eqs. (7.61)-(7.63). Alternatively, n can be computed in an initial phase as, for instance, in [94].

Algorithm 7.3.10 (Distributed Algebraic Connectivity). *Let $\varepsilon \in (0,1)$ be known by all the robots and $\beta = \varepsilon/(2n)$. Consider the robots execute Algorithm 7.3.4 for computing the powers of matrix $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. At each step $k \geq 1$, each robot $i \in \mathcal{V}$ has variables $\hat{C}_{ij}(k)$ with the (i,j) entries of the $k-$th power of matrix $\hat{C}$, $[\hat{C}^k]_{ij}$, for all the robots it has discovered so far, $ID(j) \in l_i(k)$. Note that*

$$[\hat{C}^k]_{ij} = [C^k]_{ij} + 1/n, \tag{7.60}$$

with C being the deflated Perron matrix in eq. (7.48).

At each step k, each robot i computes

$$\hat{c}_i(k) = \sum_{ID(j)\in l_i(k)} |\hat{C}_{ij}(k) - 1/n| + (n - |l_i(k)|)/n, \tag{7.61}$$

it executes a $\max-$consensus algorithm to obtain $\max_{j\in\mathcal{V}} \hat{c}_j(k)$ and lets $\hat{\beta}_i^*(k)$ be

$$\hat{\beta}_i^*(k) = (\max_{j\in\mathcal{V}} \hat{c}_j(k))^{\frac{1}{k}}. \tag{7.62}$$

The algebraic connectivity estimated by each robot $i \in \mathcal{V}$ at step $k \geq 1$ is given by

$$\hat{\lambda}_i(k) = \left(1 - \hat{\beta}_i^{\star}(k)\right)/\beta. \tag{7.63}$$

Theorem 7.3.11. *Consider each robot i executes Algorithm 7.3.10 for a connected graph. Then, as $k \to \infty$ all the variables $\hat{\lambda}_i(k)$ asymptotically converge to the algebraic connectivity $\lambda_*(\mathcal{L})$,*

$$\lim_{k\to\infty} \hat{\lambda}_i(k) = \lambda_*(\mathcal{L}), \quad \text{for } i \in \mathcal{V}, \tag{7.64}$$

and for each step $k \geq 1$ we have the following lower- and upper-bounds for $\lambda_\star(\mathcal{L})$:

$$\hat{\lambda}_i(k) \leq \lambda_\star(\mathcal{L}) \leq (\sqrt{n})^{\frac{-1}{k}} \hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta. \tag{7.65}$$

Proof. First note that $\beta = \varepsilon/(2n)$ satisfies $0 < \beta < 1/\lambda_n(\mathcal{L})$ since $\varepsilon \in (0,1)$ and $\lambda_n(\mathcal{L}) \leq 2d_{\max} < 2n$, where $d_{\max}$ is the maximum degree in the graph. Therefore, as stated in Proposition 7.3.9, the algebraic connectivity is $\lambda_*(\mathcal{L}) = (1 - \rho(C))/\beta$, where C is the deflated Perron matrix $C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n = \hat{C} - \mathbf{1}\mathbf{1}^T/n$. From Proposition 7.3.5, for all $i \in \mathcal{V}$, the variables $\hat{C}_{ij}(k)$ are equal to $[\hat{C}^k]_{ij}$ for $ID(j) \in l_i(k)$, whereas $[\hat{C}^k]_{ij} = 0$ for $ID(j) \notin l_i(k)$. Linking this with eqs. (7.59), (7.60) yields

$$\begin{aligned} [C^k]_{ij} &= [\hat{C}^k]_{ij} - 1/n, \text{ for } ID(j) \in l_i(k), \\ [C^k]_{ij} &= -1/n, \text{ for } ID(j) \notin l_i(k), \end{aligned} \tag{7.66}$$

for all $i \in \mathcal{V}$, $k \geq 1$. Therefore $\hat{c}_i(k)$ in eq. (7.61) is the absolute row sum of the $i-$th row of C^k, and $\hat{\beta}_i^*(k)$ in eq. (7.62) is $\hat{\beta}_i^*(k) = \|C^k\|_\infty^{\frac{1}{k}}$. As stated from eqs. (7.44)-(7.47),

$$(\sqrt{n})^{\frac{-1}{k}} \|C^k\|_\infty^{\frac{1}{k}} \leq \rho(C) \leq \|C^k\|_\infty^{\frac{1}{k}}, \tag{7.67}$$

since C is symmetric. Combining this with eqs. (7.63) and (7.50) we get eqs. (7.64) and (7.65) and the proof is completed. □

7.3.3 Distributed adaptive triggered average consensus

In this section we apply the Distributed Algebraic Connectivity algorithm previously presented to an event-triggered consensus scenario. Event-triggered control strategies [49, 124, 133] are appropriate for scenarios where the state variables evolve in continuous time but where the robots may exchange data only at specific time instances. They have the advantage that they save communication costs and that they keep the control law piecewise constant. The event-triggered control algorithm proposed in [124] requires the knowledge of the algebraic connectivity for properly adjusting the algorithm parameters and ensuring convergence and absence of Zeno-behavior.

The adaptive triggered average consensus algorithm is as the one presented in Section 7.3.1,

$$\dot{\mathbf{x}}(t) = -\mathcal{L}\hat{\mathbf{x}}(t) = \mathbf{u}(t), \qquad e_i(t) = \hat{x}_i(t) - x_i(t), \tag{7.68}$$

except for the trigger function, which now is

$$|e_i(t)| \leq c_1 e^{-\alpha(t)t}, \tag{7.69}$$

where now α is not fixed but is adapted depending on the previously estimated algebraic connectivity as follows:

$$\hat{\alpha}(k) = \hat{\lambda}(k), \text{ for } k \in \mathbb{N}, \qquad \text{and } \alpha(t) = \gamma\hat{\alpha}(k), \text{ for } t \in [k, k+1), \tag{7.70}$$

where $0 < \gamma < 1$ is a constant and $\hat{\lambda}(k)$ is the outcome of the algorithm in the previous section, satisfying

$$\hat{\alpha}(k) = \hat{\lambda}(k) \leq \lambda_\star(\mathcal{L}). \tag{7.71}$$

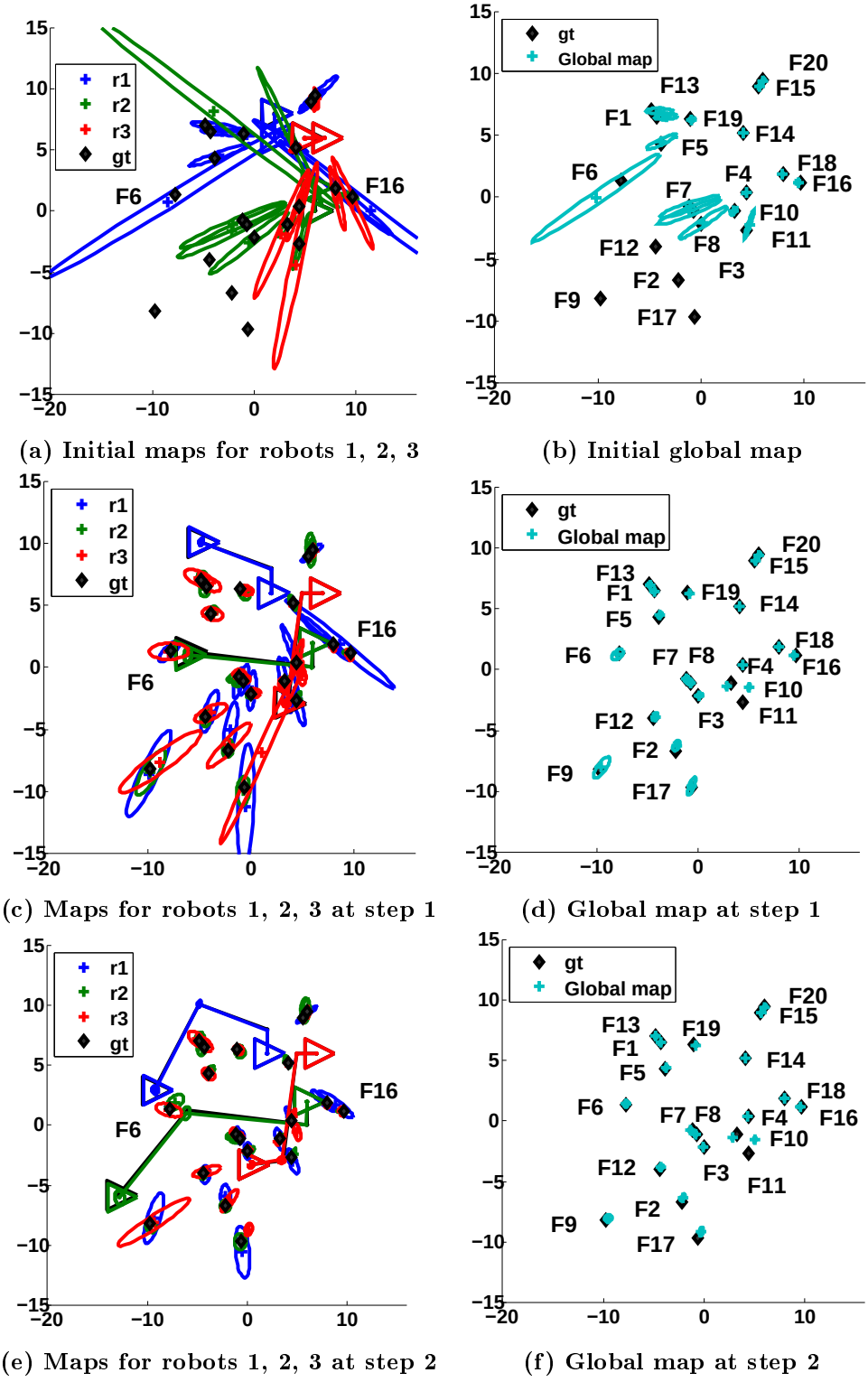
Theorem 7.3.12. *Algorithm* (7.68)-(7.70) *asymptotically converges to the average of the initial states and does not exhibit Zeno-behavior.*

Proof. The result holds based on similar analysis as in [124]. □

7.4 Discussion

Map precision

In order to show the performance of the strategy for improving the map precision, a simulation has been carried out where a team composed by three robots explore an obstacle-free environment. They estimate their motions based on odometry information and sense the environment using an omnidirectional camera that provides bearing to the landmarks. The observation noise is $\sigma_z = 1$ degree whereas the odometry noises are $\sigma_x = 0.01d$, $\sigma_y = 0.01d$, $\sigma_\theta = 2.5$ degrees, where the translation noise is proportional to the traveled distance d. Robots process the odometry data and the measurements and build their local maps (Fig. 7.4 (a)). The ground-truth data is displayed in black, and points, lines, and triangles represent respectively landmark positions, robot motions, and robot poses. The



(a) Initial maps for robots 1, 2, 3

(b) Initial global map

(c) Maps for robots 1, 2, 3 at step 1

(d) Global map at step 1

(e) Maps for robots 1, 2, 3 at step 2

(f) Global map at step 2

Figure 7.4: Some steps of the strategy for improving the map precision are displayed. Black dots are the ground-truth landmark positions. Local map estimates from different robots are shown in different colors (a, c, e). Figures at the right column (b, d, f) show the global map $\hat{\mathbf{x}}_G, \Sigma_G$ associated to the local maps, whose uncertainty is reduced as an indirect result of the improvement in the local maps.

maps and trajectories estimated by the robots are shown in different colors. The global map (Fig. 7.4 (b)), given by (7.10), associated to the initial local maps (Fig. 7.4 (a)) has several features with large uncertainties.

Robots compute their candidate next positions, and evaluate the cost function at these vantage locations. Then, they solve the robot-to-feature assignment as explained in Section 7.2.4 and move to the selected locations (Fig. 7.4 (c)). Observe for instance the feature $F16$, which has been observed by the three robots; robot $r1$ (blue) possess a very uncertain estimate, r3 (red) has a better estimate, and the estimate at r2 (green) is very precise. Since at least one of the robots that has a precise estimate of $F16$, the global map already has a precise estimate of $F16$ (Figs. 7.4 (b) and (d)). Robots do not move to better observe $F16$ but instead move to more interesting locations. This illustrates the proposed strategy, where robots attempt to reduce the uncertainty of the global map, instead of focusing on their local features estimates. As a result of the robot motions (Figs. 7.4 (c)), the precision of the most uncertain features ($F6, F7, F8, F3$) in the initial global map (Figs. 7.4 (b)) are greatly reduced (Figs. 7.4 (d)). Robots execute the proposed strategy again (Figs. 7.4 (e)), improving the precision of their local maps and of the global map (Figs. 7.4 (f)). Since this global map has reached a high precision, next iterations of the algorithm add no significant improvements.

Convergence speed

In order to show the performance of the algorithm to compute the algebraic connectivity, we have performed a set of simulations with $n = 20$ robots randomly placed as in Fig. 7.5. The algebraic connectivity $\hat{\lambda}_i(k)$ estimated by the robots $i \in \mathcal{V}$ at each step k using the proposed method is depicted in Fig. 7.6. $\hat{\lambda}_i(k)$ (light gray solid) is the same for all of them at each step k. It is a lower-bound for the true algebraic connectivity $\lambda_*(\mathcal{L})$ (dark gray solid), and asymptotically converges to $\lambda_*(\mathcal{L})$. The expression $(\sqrt{n})^{\frac{-1}{k}}\hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed) is an upper-bound for $\lambda_*(\mathcal{L})$ for each step k and it asymptotically converges to $\lambda_*(\mathcal{L})$.

We have studied the performance of our algorithm compared to the distributed power iteration algorithm for the matrix $\mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n$ (Figs. 7.7, 7.8),

$$\begin{aligned}\mathbf{y}(k) &= \mathbf{w}(k)/\text{normalization cons.}(\mathbf{w}(k)),\\ \mathbf{w}(k+1) &= (\mathbf{I} - \beta\mathcal{L})\mathbf{y}(k) - \text{deflation cons.}(\mathbf{y}(k)),\end{aligned}$$

After each power iteration step, robots execute $T_{\text{cons}} = 10, 25, 50$, and 100 consensus iterations for deflating $\mathbf{I} - \beta\mathcal{L}$ and normalizing $\mathbf{w}(k)$. The consensus iterations are computed with a classical discrete-time averaging rule, $\mathbf{z}(t+1) = \mathcal{W}\mathbf{z}(t)$, and using the Metropolis weight matrix $\mathcal{W}$ [153] given by eq. (A.3) in Appendix A.

In Fig. 7.7 we show the estimates produced by our algorithm, $\hat{\lambda}_i(k)$ (light gray solid), which are the lower-bound, and the upper-bound $(\sqrt{n})^{\frac{-1}{k}}\hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed). Recall that in our algorithm, these estimates are equal for all the robots $i \in \mathcal{V}$. For the power iteration estimates, we are displaying the Rayleigh quotient $\mathbf{w}(k+1)^T\mathbf{y}(k)/\mathbf{y}^T(k)\mathbf{y}(k)$, that considers simultaneously the estimates at all the robots. The estimates of our algorithm converge much faster than the ones produced by the power

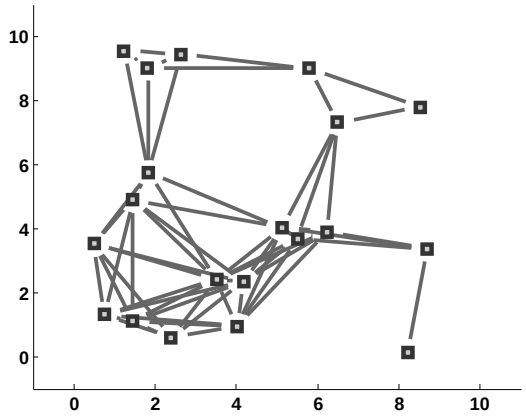


Figure 7.5: 20 robots (black squares) are placed randomly in a region of 10×10 meters. There is an edge $e = (i, j) \in \mathcal{E}$ (gray lines) between pairs of robots that are closer than 4 meters.

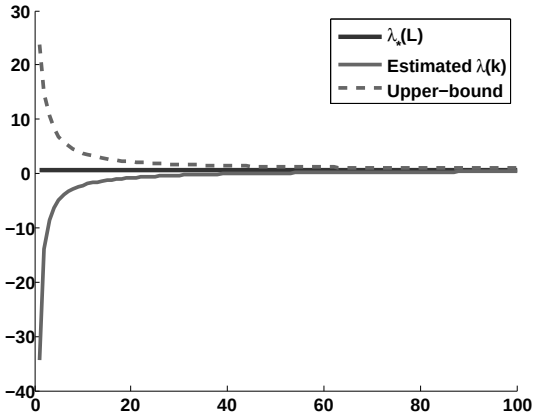


Figure 7.6: The algebraic connectivity $\hat{\lambda}_i(k)$ estimated by the robots $i \in \mathcal{V}$ (light gray solid) using the proposed method is the same for all of them at each step k. It is a lower-bound for the true algebraic connectivity $\lambda_*(\mathcal{L})$ (dark gray solid), and asymptotically converges to $\lambda_*(\mathcal{L})$. The expression $(\sqrt{n})^{\frac{-1}{k}}\hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed) is an upper-bound for $\lambda_*(\mathcal{L})$ for each step k and it asymptotically converges to $\lambda_*(\mathcal{L})$.

iteration algorithms, due to the fact that we do not need to wait for a consensus process to finish before starting a new step. In addition, our estimates asymptotically converge to the true algebraic connectivity, whereas the power iteration methods only converge to a neighborhood which is further from the true $\lambda_*(\mathcal{L})$ as the number of consensus iterations T_{cons} for normalizing and deflating decreases.

A benefit of power iteration methods compared to our proposal is that, at each iteration, the robots send constant size messages, whereas in our case the messages have size n. We have made an analysis of the evolution of the estimates of the algorithms versus the total size of messages sent per robot (Fig. 7.8). As it can be observed, also in this case the estimates produced by our method are more accurate than for the power iteration

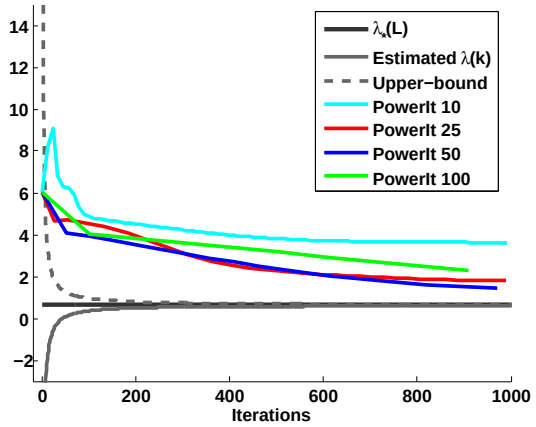


Figure 7.7: The estimates obtained with the proposed method (light gray dashed and solid lines) converge very fast to $\lambda_*(\mathcal{L})$ (dark gray solid). The estimates produced with the Power iteration algorithm (colored solid lines), with $T_{\text{cons}} = 10, 20, 50$ and 100 consensus iterations for normalizing and deflating, need much more iterations to converge. Besides, they do no converge exactly to $\lambda_*(\mathcal{L})$, but to a value which is closer to $\lambda_*(\mathcal{L})$ as the number of consensus iterations T_{cons} increases.

methods.

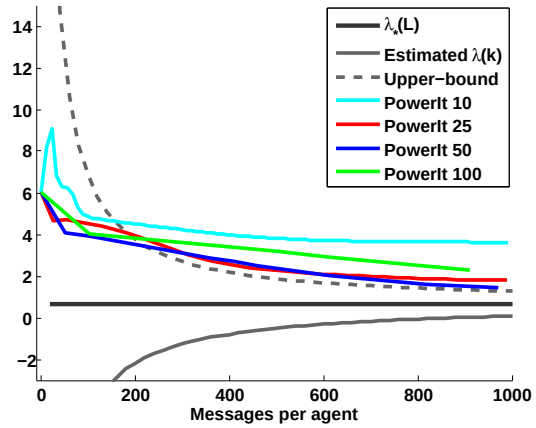


Figure 7.8: With the proposed algorithm, each robot i sends messages of size n per iteration, whereas for power iteration methods, messages have constant size. However, also in this case our method behaves better than the power iteration approaches. For the same communication usage, the estimates produced by our algorithm (light gray dashed and solid lines) are more precise than the ones obtained with the power iteration algorithms (colored solid lines).

7.5 Conclusions

In this chapter we have discussed strategies for improving the multi-robot perception. First, we have proposed a motion control strategy for improving the precision of the local feature-based maps, and as a result, of the global merged map. The described strategy selects a finite set of candidate motions to the robots, and computes its associated cost in the form of the individual contributions of every feature. Therefore, this cost presents a space complexity linear on the map size. This information is used by the team members to negotiate their next motions, presenting the benefit that robots do not need to wait for having a good global map estimate when they coordinate. Second, we have presented a distributed method to compute the algebraic connectivity for networked robot systems with limited communication. The algebraic connectivity establishes the convergence speed of the map merging algorithm. At each iteration, the algorithm produces both an upper and a lower bound estimates of the algebraic connectivity. We have proved theoretically and experimentally that both estimates asymptotically converge to the true algebraic connectivity. We have shown that our method outperforms the classical distributed power iteration, providing more accurate estimates, and improving the network communication usage. The ability to give upper and lower bounds of the algebraic connectivity has been demonstrated to have a great importance for combining this method with higher level algorithms for adaptive consensus in a parallel fashion, i.e., where both processes are executed simultaneously. We have combined this algorithm with an event-triggered adaptive consensus scenario where, at each iteration, the most recent estimate of the algebraic connectivity is used by the consensus method.

Chapter 8
Real Experiments

We have tested the methods proposed in this document under real data and different communication schemes. We have conducted several experiments using a data set from [60] with bearing-only data described in Appendix B for studying the performance of the static and dynamic map merging algorithms and the data association method. We have also tested the data association algorithm using real images acquired with cameras. Additionally, we have analyzed our algorithms under real data acquired with an RGB-D sensor, which may perform better in these kind of scenarios.

Static Map Merging

We use a data set from [60] described in Appendix B with bearing information obtained with vision in an environment of $60m \times 45m$ performing 3297 steps. It is an indoor scenario where the robot moves along corridors and rooms. The data set contains real odometry data and images captured at every step (Fig. 8.1). The images are processed and measurements to natural landmarks are provided. The natural landmarks are vertical lines extracted from the images and processed in the form of bearing-only data. The observations in the dataset are labeled so that we have the ground-truth data association. This dataset is very challenging for a conventional visual map building algorithm due to the limited field of view of the camera (Sony EVI-371DG). Furthermore, the camera is pointing forward in the same direction of robot motion and the robot traverses rooms and corridors with few features in common. Notice that this situation is much more complex than situations where the camera can achieve big parallax, or systems with ominidirectional cameras, where features within 360 degrees around the robot are observed.

We have carried out these experiments with 9 robots. The total area covered by the robots is a square of 30 m x 30 m (Fig. 8.2). We run a separate SLAM in each robot and obtain 9 maps. We use a bearing-only SLAM algorithm with features parameterized in inverse-depth [92] followed by a transform to Cartesian coordinates before the merging process. The reader is referred to Chapter 2 to find a detailed explanation of the SLAM algorithm we used. We express the local maps in global coordinates according to the relative robot poses seen in Fig. 8.2, obtaining the results shown in Fig. 8.3. Note that this is the result of putting the maps together, without applying a merging method. The team of robots execute the fusion algorithm presented in Chapter 3 to merge the local

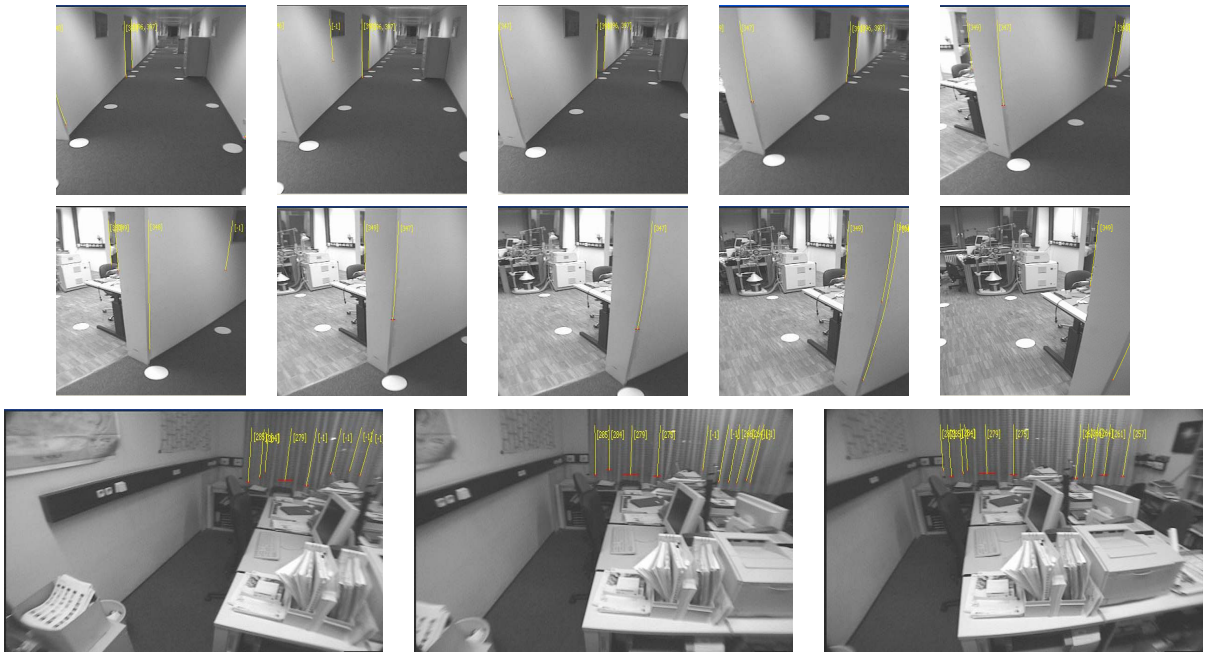


Figure 8.1: An example of the images used by the 9 robots during the navigation to test the proposed method [60]. Although the data set also provides artificial landmarks (white circles on the floor), we do not use them, and instead we test the algorithm using the lines extracted from natural landmarks (in yellow).

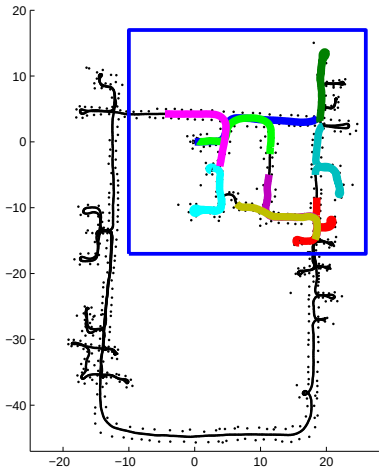


Figure 8.2: Trajectories followed by the 9 robots. They cover a region of 30m x 30m of the whole dataset map. In order to give an idea of the scene structure, we display in black the path in the dataset and a set of artificial landmarks (black dots) placed on both sides of the trajectory, which are not used in the experiment. Here, the rooms can be identified since robots enter and leave them describing short trajectories. The long, straight motions correspond to corridors.

maps. We study the behavior of the map merging method under three different scenarios:

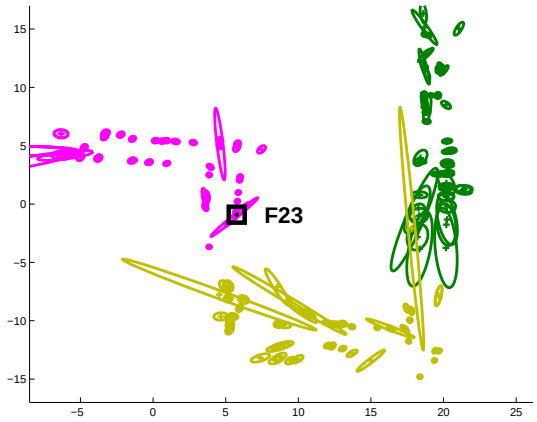
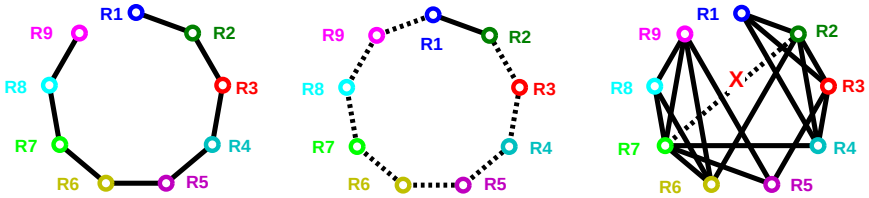


Figure 8.3: Local maps obtained by robots 2 (green), 6 (yellow), and 9 (pink) after following their trajectories in Fig. 8.2. The feature F23 within the black box will be used for testing purposes within this section.

a fixed communication graph, a graph with switching topology and a graph with link failures (Fig. 8.4). We illustrate the performance of our algorithm by comparing the



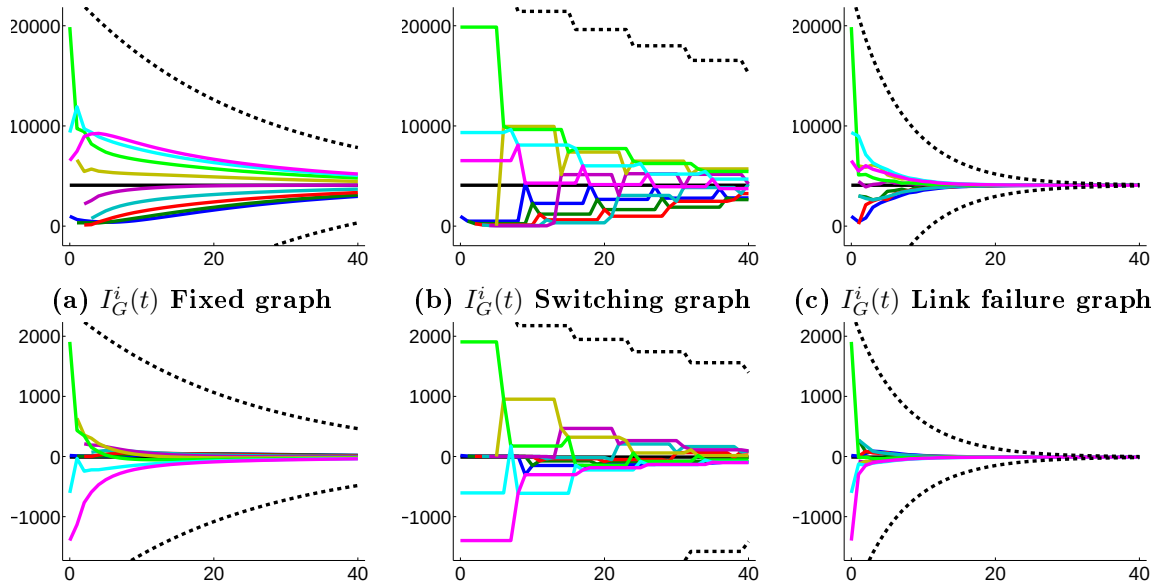
(a) Fixed graph **(b) Switching graph** **(c) Link failure graph**

Figure 8.4: Communication graphs. (a) The topology is a string, with robots 1 and 9 in the extremes. Each robot has two neighbors, except the extreme robots, that have a single neighbor. This graph remains fixed for all the iterations of the algorithm. (b) For each iteration t, there exists a single edge linking robots $((t-1) \mod 9)+1$ and $(t \mod 9)+1$. (c) A connected communication graph where at each iteration one of its link fails.

global map estimated by the robots along the iterations with the actual global map. We use two features to do this: F23, which has been observed by several robots, and F368, which belongs to a room visited exclusively by robot 4.

In Figs. 8.5, 8.6 we show the estimated information matrices $I_G^i(t)$ and vectors $\mathbf{i}_G^i(t)$ (colored lines) during 40 iterations, compared to the global map I_G, $\mathbf{i}_G$ (black line). We display the subcomponent associated to the x-coordinate of features F23 and F368. As can be observed, in all cases the estimates converge to the average value very fast. However, for F368, the consensus is reached faster than for F23. This happens because only robot 4 possess an initial value for F368 (Fig. 8.6, iteration 0). As the other robots receive information of F368, their estimates are displayed in colors. We can see that, since this initial value is the unique source of information for F368, the other robots do not disagree with this information and they just incorporate it into their estimates. In a few iterations,

all the robots possess an estimate for F368 very close to the average value. However, for F23 there exist many initial values at iteration 0 (Fig. 8.5) from robots 1,7,8,9. The robots receive different information for F23 from different sources and, therefore, they must reach an agreement. As a result, there is a higher discrepancy in their estimates.



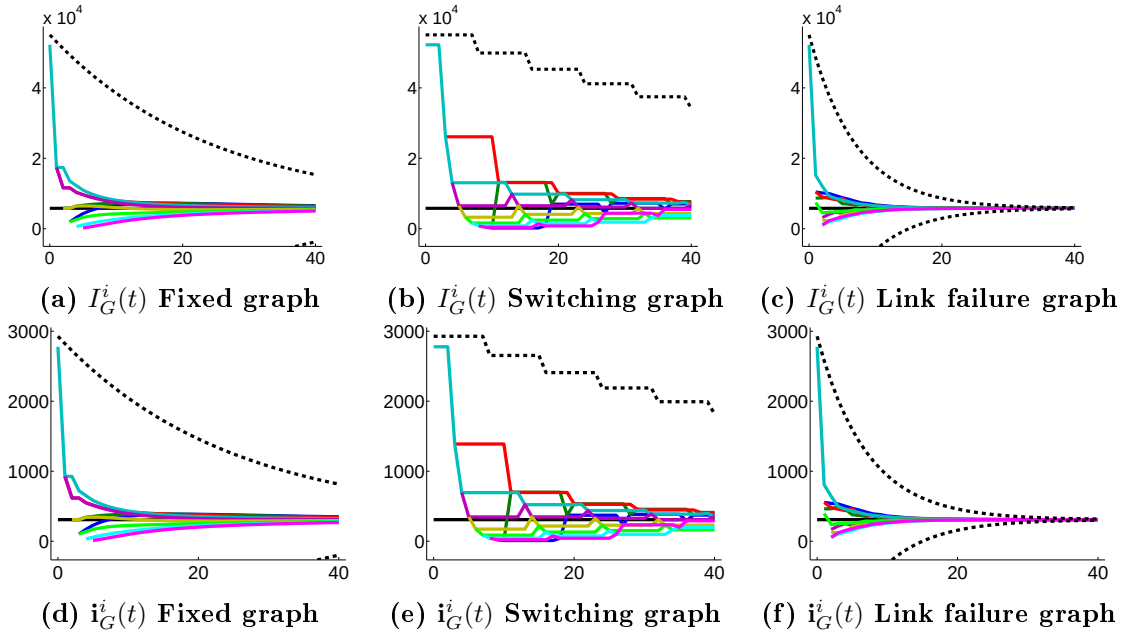
(a) $I^i_G(t)$ **Fixed graph** **(b)** $I^i_G(t)$ **Switching graph** **(c)** $I^i_G(t)$ **Link failure graph**

(d) $\mathbf{i}^i_G(t)$ **Fixed graph** **(e)** $\mathbf{i}^i_G(t)$ **Switching graph** **(f)** $\mathbf{i}^i_G(t)$ **Link failure graph**

Figure 8.5: Estimated position ($x-$coordinate) of F23 at each robot along 40 iterations. We display its associated components within the information matrices $I^i_G(t)$ (first row) and vectors $\mathbf{i}^i_G(t)$ (second row). We analyze the results for the fixed (first column), the switching (second column) and the link failure (third column) communication graphs.

We analyze the evolution of the mean $\mathbf{x}^i_G(t)$ and covariance $\Sigma^i_G(t)$ (3.8) for F23 and F368 estimated by each robot i (Figs. 8.7, 8.8). We compare them with the mean $\mathbf{x}_G$ and covariance Σ_G of the global map, and with the numerical covariance $Q^i_G(t)$ at each iteration. It can be seen that $\mathbf{x}^i_G(t)$ converges to $\mathbf{x}_G$, and $\Sigma^i_G(t)$ converges to $n\Sigma_G$. Besides, it can also be seen that the numerical covariance $Q^i_G(t)$ remains bounded by $\Sigma^i_G(t)$ for all the iterations (Figs. 8.7, 8.8). When we analyzed the estimates in the information form, we observed that only robot 4 was providing information for F368. However, when we study the evolution of the estimates in the mean and covariance form (Fig. 8.7, 8.8), we can see that robots which did not observe F368 are providing estimates which disagree with those from robot 4. This is due to the effect of the correlations between F368 and the other features. Therefore, modifications in the estimates of features correlated with F368 produce modifications in the estimate of F368.

We analyze the effects of the communication topology on the performance of the algorithm. In the fixed and the switching communication graphs (Fig. 8.5, first and second column), the convergence is slower than for the link failure graph (Fig. 8.5, third column). In this fixed graph (Fig. 8.4 (a)) the topology is a string, with robots 1 and 9 in the extremes. This is a specially bad configuration since the time needed to propagate information from the extreme robots to the whole network is maximal. The per step



(a) $I^i_G(t)$ **Fixed graph** (b) $I^i_G(t)$ **Switching graph** (c) $I^i_G(t)$ **Link failure graph**

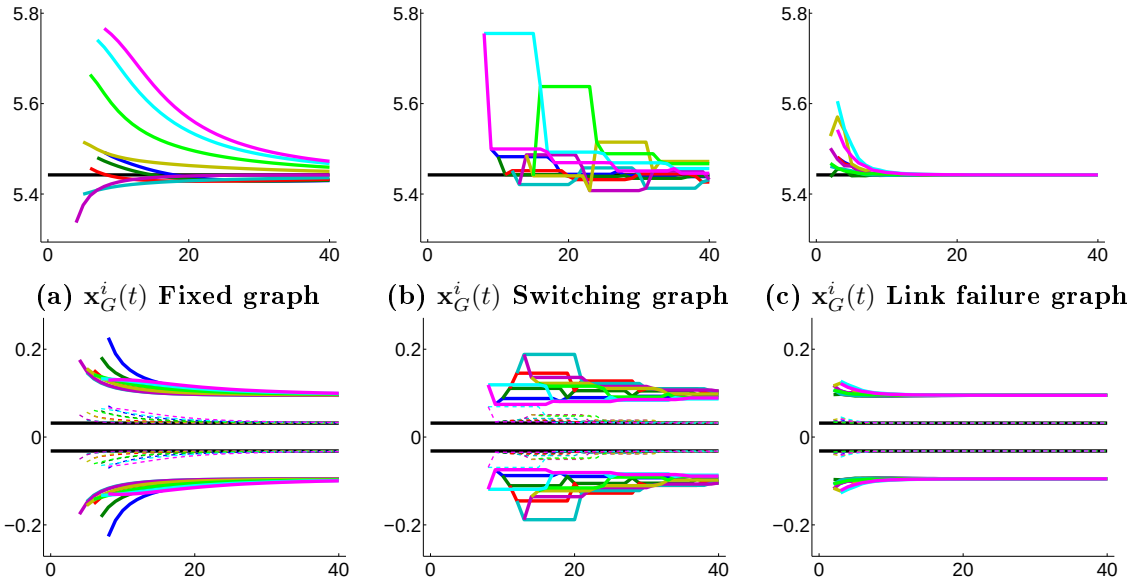
(d) $\mathbf{i}^i_G(t)$ **Fixed graph** (e) $\mathbf{i}^i_G(t)$ **Switching graph** (f) $\mathbf{i}^i_G(t)$ **Link failure graph**

Figure 8.6: Feature F368, information matrices $I^i_G(t)$ and vectors $\mathbf{i}^i_G(t)$. We display the entry within the information matrix $I^i_G(t)$ associated to the $x-$coordinate of F368, along 40 iterations. For the three network topologies, the estimates at each robot (color solid lines) remain within the theoretical bounds (black dashed lines) while they asymptotically approach the global map I_G (black solid line).

convergence factor $\gamma = |\lambda_2(W)|$ (3.20) depends on the Metropolis weights matrix, which is

$$W = \frac{1}{3}\begin{bmatrix} 2 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \mathbf{0} & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \mathbf{0} & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}.$$

We obtain a value for $\gamma = 0.96$ close to 1. This produces a slow convergence. The convergence bounds are displayed in black dashed lines (Figs. 8.5, 8.6, first column). In the switching graph case (Fig. 8.4 (b)), at every time instant, only one communication link exists in the graph and this sequence takes place in a circular fashion. This is a very extreme communication scheme where, although the conditions for convergence are satisfied, the converge speed is expected to be slow. We can see that (Figs. 8.5, 8.6, second column) for each robot, estimates remain unchanged during long periods of time, then they experiment two consecutive changes, and then they remain unchanged again. Each robot remains isolated during 7 iterations, maintaining its estimates unchanged. Then, it exchanges information with its previous neighbor and, in the next iteration, with its next neighbor. In our case, at each iteration t, there exists a single edge linking robots $((t-1) \mod 9) + 1$ and $(t \mod 9) + 1$. The index of joint connectivity is $\tau = 8$ since every



(a) $\mathbf{x}_G^i(t)$ **Fixed graph** **(b)** $\mathbf{x}_G^i(t)$ **Switching graph** **(c)** $\mathbf{x}_G^i(t)$ **Link failure graph**

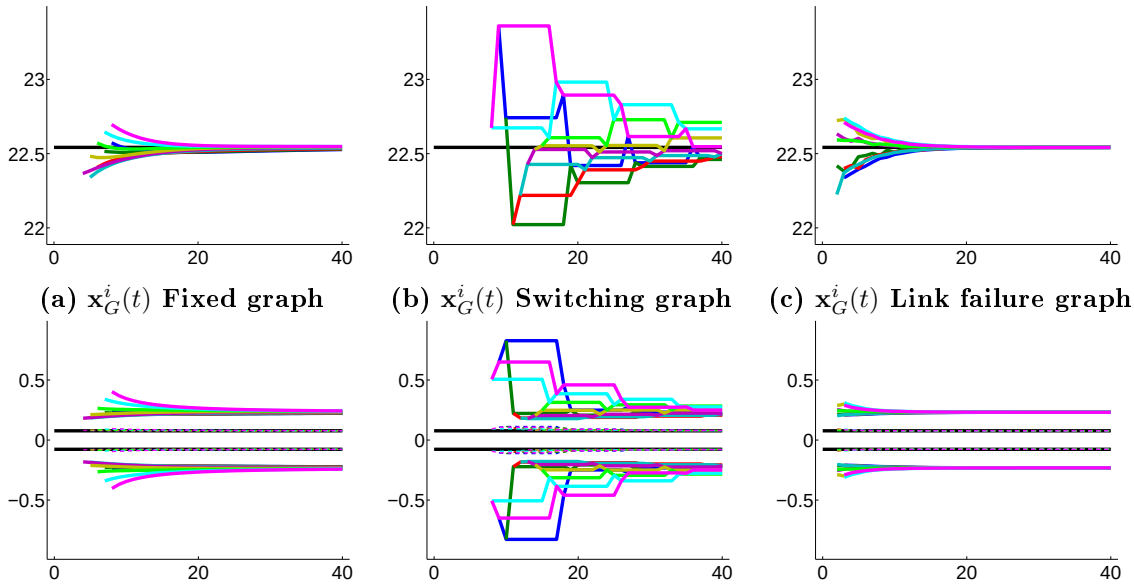
(d) $\Sigma_G^i(t)$ **Fixed graph** **(e)** $\Sigma_G^i(t)$ **Switching graph** **(f)** $\Sigma_G^i(t)$ **Link failure graph**

Figure 8.7: Estimated position ($x-$coordinate) of F23 at each robot along 40 iterations. We display its associated components within the mean $\mathbf{x}_G^i(t)$ (first row) and covariance $\Sigma_G^i(t)$ (second row) estimated by each robot i. We analyze the results for the fixed (first column), the switching (second column) and the link failure (third column) communication graphs. All the mean estimates $\mathbf{x}_G^i(t)$ (color solid lines) approach the average value (black solid line). The numerical covariance matrix $Q_G^i(t)$ (color dashed lines) asymptotically approaches the covariance matrix Σ_G (black solid line) of the global map. The numerical covariance $Q_G^i(t)$, which cannot be computed by the robots using local information, is bounded by the locally computed covariance matrix $\Sigma_G^i(t)$ (color solid lines). This matrix converges to $n\Sigma_G$.

8 iterations the joint graph is connected. There are only 9 different Metropolis weight matrices $W(t)$, depending on the linked robots at time t, that are repeated successively. We obtained a value for $\delta = 0.89$ using (3.21). We draw the bounds using black dashed lines (Figs. 8.5, 8.6, second column).

In the link-failure graph (Fig. 8.4 (c)), at each iteration one of the links in the graph fails although the graph remains connected. Thus, we obtain an index of joint connectivity of $\tau = 1$. Evaluating all the possible Metropolis weight matrices in this graph, we obtain $\delta = 0.80$. We show the convergence speed bounds (Figs. 8.5, 8.6, third column) using black dashed lines. This communication scheme exhibits the fastest convergence speed, since $0.80^t \leq 0.96^t \leq 0.89^{\lfloor t/8 \rfloor}$ for all $t = 0, 1, \ldots$. This faster convergence can also be observed in the estimated mean and covariance (Figs. 8.7, 8.8), where the estimates approach the global map faster for the link failure graph (third column). It is noted that regardless of the presence of link failures or changes in the communication topology, the numerical covariance remains bounded by the locally computed covariance matrix (Figs. 8.7, 8.8).

In addition, we display (Fig. 8.9) the global map estimated by robot 1 after 5, and 20 iterations (colored lines) of the merging algorithm, and under the fixed communication



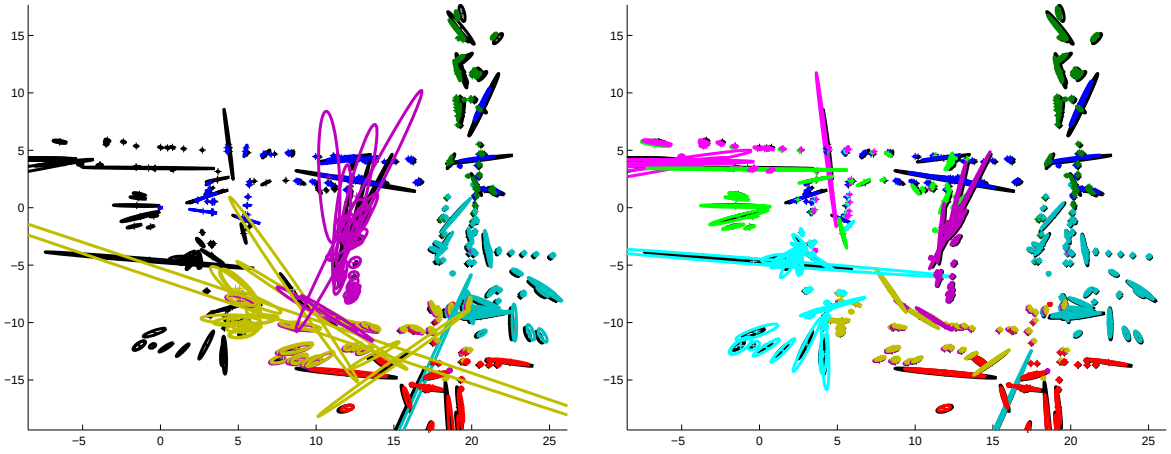
(a) $\mathbf{x}_G^i(t)$ **Fixed graph** (b) $\mathbf{x}_G^i(t)$ **Switching graph** (c) $\mathbf{x}_G^i(t)$ **Link failure graph**

(d) $\Sigma_G^i(t)$ **Fixed graph** (e) $\Sigma_G^i(t)$ **Switching graph** (f) $\Sigma_G^i(t)$ **Link failure graph**

Figure 8.8: Estimated position ($x-$coordinate) of F368 at each robot along 40 iterations. We display its associated components within the mean $\mathbf{x}_G^i(t)$ (first row) and covariance $\Sigma_G^i(t)$ (second row) estimated by each robot i. We analyze the results for the fixed (first column), the switching (second column) and the link failure (third column) communication graphs. All the mean estimates $\mathbf{x}_G^i(t)$ (color solid lines) approach the average value (black solid line). The numerical covariance matrix $Q_G^i(t)$ (color dashed lines) asymptotically approaches the covariance matrix Σ_G (black solid line) of the global map. The numerical covariance $Q_G^i(t)$, which cannot be computed by the robots using local information, is bounded by the locally computed covariance matrix $\Sigma_G^i(t)$ (color solid lines). This matrix converges to $n\Sigma_G$.

graph (Fig. 8.4 (a)). The maps estimated by the 9 robots are similar. We compare the estimates at robot 1 to the global map in (3.4) (black lines). Due to the network configuration, after 5 iterations robot 1 has received information from the initial local maps of robots 1 to 6. However, it still knows nothing of the local maps of robots 7 to 9 (Fig. 8.9 (a)). As previously stated, this fixed communication graph has a slow convergence speed. However, after 20 iterations the map estimated by robot 1 is very close to the global map (Fig. 8.9 (b)). In addition, it is observed that the information fusion leads to a great improvement in the map quality, where not only the uncertainty is greatly decreased, but also the local maps are corrected.

Finally, we have studied the performance of the map merging algorithm in terms of execution times (Fig. 8.10). During the first iterations, the peaks on the execution times are due to the expansion and arrangement of the information matrices and vectors $I_G^i(t)$ and $\mathbf{i}_G^i(t)$ that are performed by the robots whenever they discover new features in its neighbors' information. These memory allocation operations, which are computationally expensive, give rise to this behavior. In the fixed graph case (in blue solid) this situation continues until iteration 5, when robot 5, the robot in the central position within the



(a) 5 iterations **(b) 20 iterations**

Figure 8.9: Global map estimated by robot 1 after 5 **(a)** and 20 **(b)** iterations of the merging algorithm, and under the fixed communication graph (Fig. 8.4 (a)). Different colors identify the source local map. Although the global map contains a single estimate per feature, the features observed by more than one robot are displayed by multiple colored ellipses. The global map $\mathbf{x}_G$, Σ_G is displayed in black. Although there is a slow convergence speed associated to the fixed communication graph used here, the estimated global map after 20 iterations is very close to the global map.

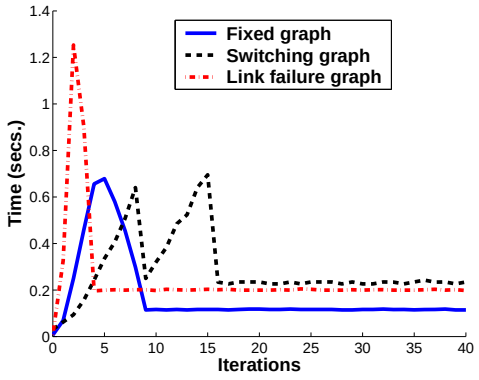


Figure 8.10: Execution times (per iteration and robot) exhibited by the merging algorithm under the three tested communication graphs.

string graph, has received information from all the robots. Its information matrix $I^i_G(5)$ reaches its maximum size and, from here to the end of the experiment, its global map estimate changes but its size remains unchanged. The execution times reach a peak at iteration 5 and from here on, it decreases. From iterations 5 to 9 other robots achieve the maximal size of their matrices $I^i_G(t)$, and finally, from iteration 9 to the end of the experiment, the global map size remains unchanged for all the robots. For this reason, we can see that the execution times are drastically reduced from iteration 9 to the end of the experiment. For the switching graph (in black dashed), the first robots that receive information from all the others are 8 and 9, at iteration 8, and from iterations 9 to 15

robots 1 to 7 successively expand their maps to the maximum size. Finally, from iteration 16 to the end, all the robot's global map estimates present the maximal size and only its contents change. For this reason, the execution times decrease. Finally, for the link failure graph (in dotted red), due to its higher connectivity, all the expansion operations are carried out during the first 4 iterations, giving rise to the larger peak observed in the plot. After these expansion operations, the execution times are similar for the three communication graphs.

Dynamic Map Merging

In the second set of experiments, we also use real data from the data set [60] described in Appendix B with bearing information obtained with vision (Sony EVI-371DG). The landmarks are vertical lines extracted from the images (Fig. 8.11).

Figure 8.11: An example of the images that the robot team uses during the navigation to test the proposed method [60]. We test the algorithm using the lines extracted from natural landmarks (in yellow).

We select 8 subsections of the whole path for the operation of 8 different robots (Fig. 8.12). The robots execute the proposed algorithm for merging their local maps communicating through range-limited graphs as in Fig. 8.13, with the Metropolis weights (eq. (A.3) in Appendix A) and its Laplacian matrix (eq. (4.9)), and with the parameters $\gamma = 1.8$ and $h = 0.8$. In this experiment, we get $\lambda_\star = 0.97$. They execute a total of $L = 500$ consensus iterations. The robots run a total of $K = 5$ map update steps. Between consecutive map update steps k, $k+1$, each robot performs 10 steps of a bearing-only SLAM algorithm (Fig. 8.14).

The algorithm is executed for 3 different configurations. In the first one, the robots execute a small number of consensus iterations $l = 25$ after each map update step $k = 1, \ldots, 4$, and the remaining $L-(K-1)l = 400$ iterations after the last one. In the second case, they use $l = 50$ and execute the remaining 300 at $k = K$. And in the last one, they use an equal number of iterations per step $l = (L/K) = 100$. The obtained scaled estimation errors for the information matrices $|[I_i^k(t)]_{r,s} - [I_{avg}^k]_{r,s}|/\sigma_I$ and information vectors $|[\mathbf{i}_i^k(t)]_r - [\mathbf{i}_{avg}^k]_r|/\sigma_{\mathbf{i}}$ are displayed in Fig. 8.15 (a) and (b) along the L consensus iterations. During the intermediate steps $k = 1, \ldots, K-1$, the configuration $l = 100$ (red solid) exhibits the fastest convergence, whereas $l = 50$ (green dashed) also produces good results. The configuration $l = 25$ (blue dashed-dotted) however is less precise and

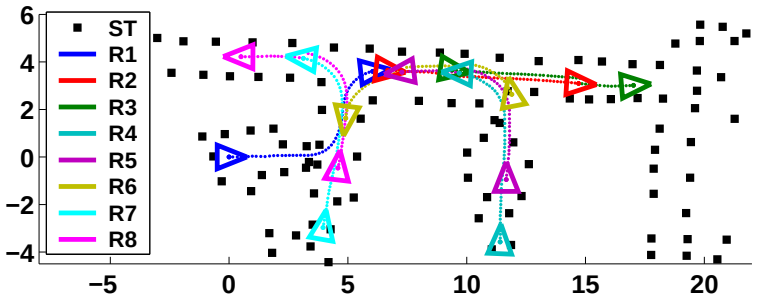


Figure 8.12: Approximate trajectories followed by the 8 robots (dots). Since there is no ground truth information available, a set of artificial landmarks (black squares) are displayed to give an idea of the scene.

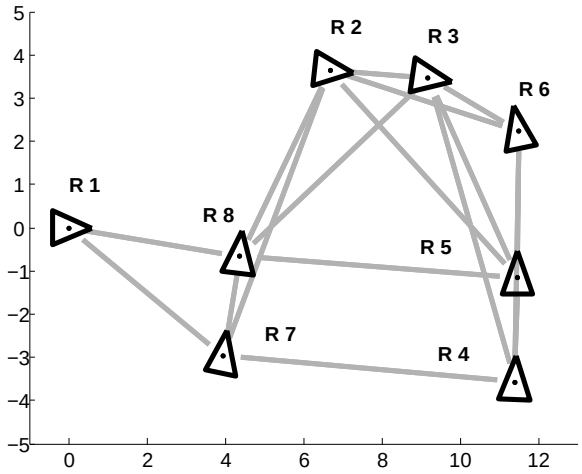


Figure 8.13: Sample communication graph for the 8 robots. There is an edge (gray lines) between two robots (black triangles) if their distance is smaller than 7.5 m.

its estimates are further from the average value. During the last step $k = K$ both $l = 25$ and $l = 50$ configurations reach a small final error. However, the configuration $l = 100$ which was reaching the best results during the previous steps, finishes with the worst final error. The configuration $l = 50$ produces interesting results since the intermediate errors are almost as good as for $l = 100$, whereas the final error is similar to the obtained by $l = 25$. After the L iterations, the final global maps $\hat{\mathbf{x}}_i^k(t)$, $\Sigma_i^k(t)$, computed by the dynamic map merging algorithm are very close to the global map $\hat{\mathbf{x}}_G^k$, Σ_G^k (4.5) at step $k = K$ that would be obtained by a centralized system. We show the global map at robot 1, for the $l = 100$ configuration (Fig. 8.16) after L iterations, which is very similar to the maps computed by the other robots (they are equal in the limit). Similar results have been obtained using the other configurations.

We compare the behavior of the dynamic consensus algorithm with a zero-initialization strategy (Fig. 8.17 (a)). The errors associated to the information vectors for even iteration numbers t are showed for both the dynamic consensus algorithm with $l = 100$ (black solid) and the zero-initialization strategy with $l = 100$ (red solid). For $k = 1$ both errors are

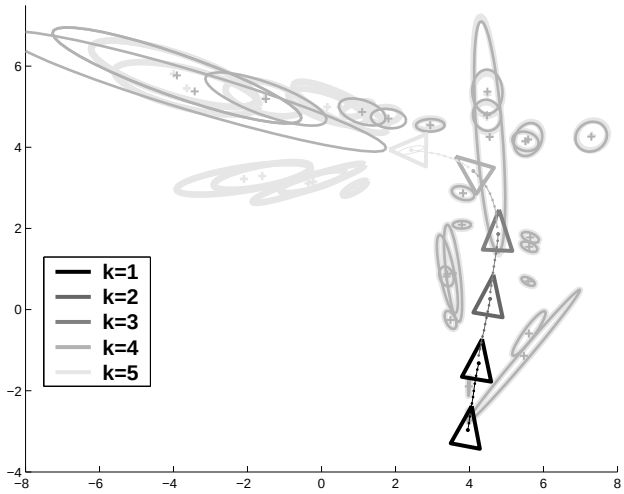
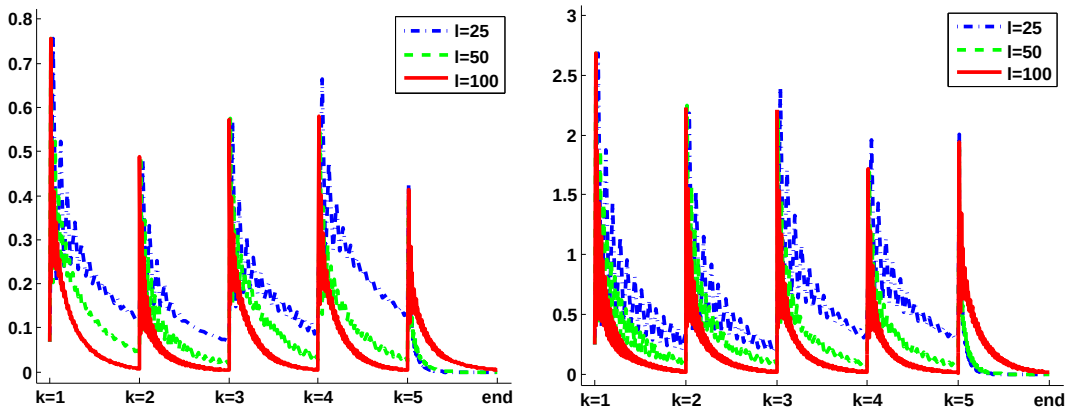


Figure 8.14: Local map estimated by robot R7. Gray triangles represent the pose of R7 for the different map update steps $k = 1, \ldots, 5$. R7 initiates a new map update step after executing 10 motions. Its local map at steps $k = 4$ (gray) and $k = 5$ (light gray) is also displayed. Between steps $k = 4$ and $k = 5$, R7 introduces new features into its local map and also improves the previous estimates at $k = 4$.



(a) $\max_{i,r,s} |[I_i^k(t)]_{r,s} - [I_{avg}^k]_{r,s}|/\sigma_I$. (b) $\max_{i,r} |[\mathbf{i}_i^k(t)]_r - [\mathbf{i}_{avg}^k]_r|/\sigma_{\mathbf{i}}$.

Figure 8.15: Estimation errors along the L consensus iterations. (a) $\max_{i,r,s} |[I_i^k(t)]_{r,s} - [I_{avg}^k]_{r,s}|/\sigma_I$, (b) $\max_{i,r} |[\mathbf{i}_i^k(t)]_r - [\mathbf{i}_{avg}^k]_r|/\sigma_{\mathbf{i}}$. The configuration $l = 25$ (blue dashed-dotted) maintains a high error along the intermediate steps $k = 1, \ldots, K-1$, but at the last step, it gets a high precision. For $l = 100$ (red solid) the precision at the end of each intermediate step $k = 1, \ldots, K-1$ is very high, but finishes with the worst final error. The configuration $l = 50$ (green dashed) produces accurate results during both the last and the intermediate steps.

equal since the dynamic consensus algorithm performs a zero-initialization. For the other steps $k = 2, \ldots, K$, the errors of our proposed algorithm are smaller than the ones obtained with the zero-initialization strategy. They are upper bounded by the theoretical rate of convergence in eq. (4.52) (gray dashed). We analyze the behavior of the algorithm under

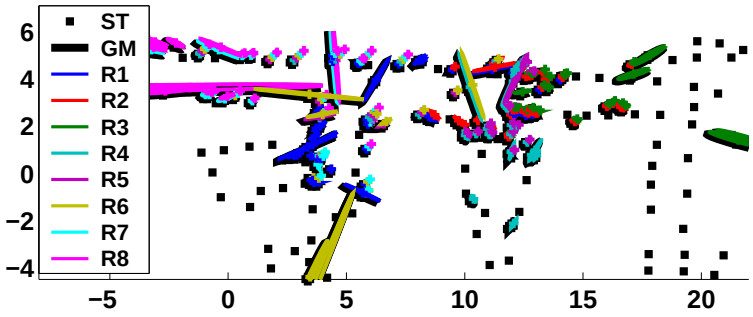
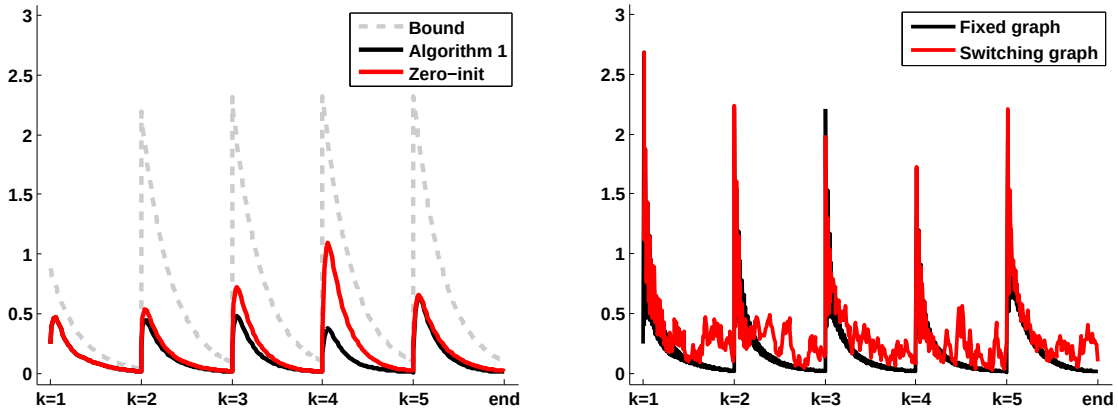


Figure 8.16: Global map $\hat{\mathbf{x}}_i^k(t)$, $\Sigma_i^k(t)$, estimated by robot $i = 1$ at the last consensus iteration L for the configuration $l = 100$. The sections associated to the inputs of the different robots are displayed in different colors. As it can be seen, it is very similar to the global map $\hat{\mathbf{x}}_G^k$, Σ_G^k at step $k = K$, displayed in black. The black squares are the same artificial landmarks than in Fig. 8.12 and they are displayed to give an idea of the scene.



(a) $\max_{i,r} |[\mathbf{i}_i^k(t)]_r - [\mathbf{i}_{avg}^k]_r|/\sigma_{\mathbf{i}}$ **versus** $f_k(t)\alpha_{\mathbf{i}}/\sigma_{\mathbf{i}} + g_k(t)$ (b) $\max_{i,r} |[\mathbf{i}_i^k(t)]_r - [\mathbf{i}_{avg}^k]_r|/\sigma_{\mathbf{i}}$

Figure 8.17: Estimation errors along the L consensus iterations for $l = 100$. (a) Comparison with a zero-initialization strategy and with the bounds for even iteration numbers t. The errors obtained with the configuration $l = 100$ (black solid) are always within the theoretical bounds (gray dashed) and they are smaller than the errors associated to the zero-initialization strategy (red solid). (b) Comparison with a switching graph. The robots execute Algorithm 4.4.1 using the communication graph $\mathcal{G}$ in Fig. 8.13 where, at each iteration t and step k, one of the links is selected randomly and erased from $\mathcal{G}$. The behavior of the algorithm under the fixed (black solid) and the switching (red solid) graphs is compared for $l = 100$. The estimation errors for both the fixed (black solid) and switching (red solid) graphs converge to zero.

time-varying communication graphs (Fig. 8.17 (b)). Robots exchange data according to the communication graph $\mathcal{G}$ in Fig. 8.13. At each iteration $t =$ and step k, one of the links $\mathcal{G}$ fails and it is erased from $\mathcal{G}$. We display the estimation when the robots execute the proposed algorithm with with $l = 100$ under the fixed graph in Fig. 8.13 without (black solid) and with (red solid) link failures. Here, although the variations in the graph

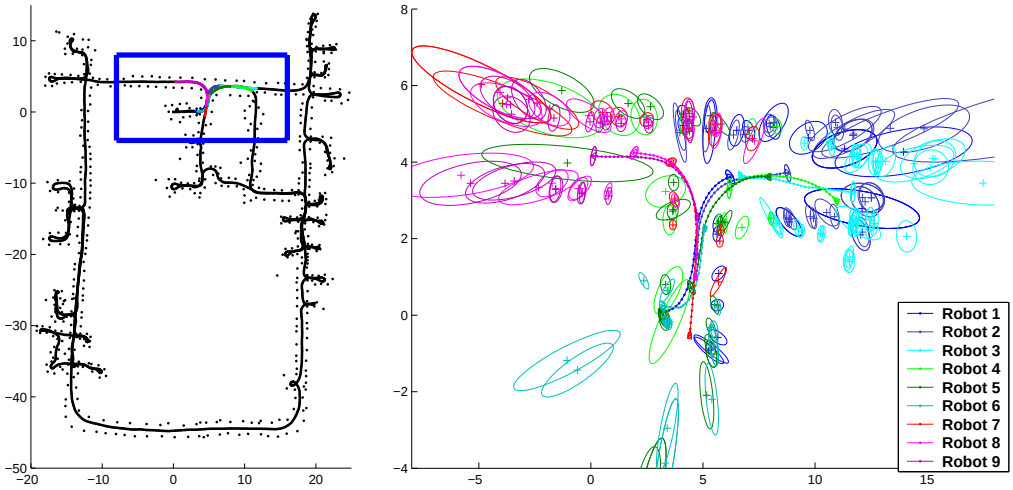
topology take place very often (at each iteration), these variations are small. Therefore, as discussed in Section 4.4, the estimates of the proposed algorithm correctly track the average of the inputs (red solid). Obviously, this convergence is slower than for the fixed graph case (black solid).

Data Association of Stochastic Maps

The behavior of the data association algorithm is analyzed also with the data set [60] described in Appendix B with bearing information obtained with vision (Sony EVI-371DG). The landmarks are vertical lines extracted from the images (Fig. 8.18). The measurements are labeled so that we can compare our results with the ground-truth data association. We select 9 subsections of the whole path for the operation of 9 different robots (Fig. 8.19 (a)). A separate SLAM is executed on each subsection, producing the 9 local maps (Fig. 8.19 (b)). The local data associations are computed using the JCBB [97] since it is very convenient for clutter situations like the considered scenario (Fig. 8.19 (b)). The JCBB is applied to the local maps of any pair of neighboring robots. We analyze the performance of the algorithm under 3 communication graphs (Fig. 8.20).

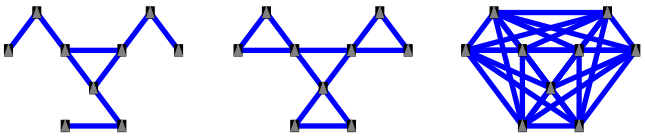
Figure 8.18: An example of the images used by the 9 robots during the navigation to test the proposed method [60]. Although the data set also provides artificial landmarks (white circles on the floor), we test the algorithm using the lines extracted from natural landmarks (in yellow).

Table 8.1 gives statistics about the number of inconsistencies found considering the different network topologies in Fig. 8.20. We show the obtained associations compared to the ground truth results. The number of association sets is the number of connected components of $\mathbf{A}^t$. The number of good links (true positives) are obtained associations between 2 features which are true (ground truth). The missing links (false negatives) are associations that are in the ground truth information, but have been not detected. And spurious links (false positives) are associations found between features that are different according to the ground truth. The sixth row, C, is the number of conflictive sets. The next row in the table shows the total number of features which have been associated to any other feature from other local map. The last row gives information about how many of those features are conflictive. The amount of missing and spurious associations obtained is very high for the three network topologies. This is the expected result for many real scenarios, where the landmarks are close to each other, and where the only available information are their cartesian coordinates. As a result, the conflictive features are more than a 10% of the total. In communication graphs with more cycles (Fig. 8.20 (b)(c)),



(a) (b)

Figure 8.19: (a) Section of the dataset used in the experiments. (b) Local maps acquired by 9 robots exploring the region in (a).



(a) One cycle (b) Four cycles (c) Complete

Figure 8.20: Communication graphs between the 9 robots after the exploration.

there are more conflictive features. In the three cases, after a single execution of the detection and the resolution algorithms, all the inconsistencies are solved (Table 8.2, 1st row). An interesting result is that, although our algorithm cannot distinguish between good and spurious edges, in practice a high number of the deleted edges (last row) are spurious.

Data Association of Images

We have also tested the performance of our proposal with a set of images. A team of robots equipped with cameras and limited communication capabilities is a typical situation in which not all the images are available to execute a global matching. A solid set of matches is required independently of the task the team is performing (visual SLAM, formation control, etc.). In the proposed experiment 6 robots moving in formation are considered (Fig. 8.21).

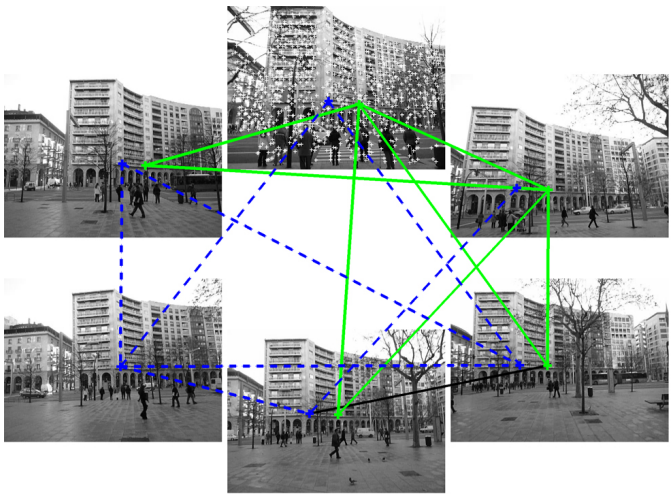
Each robot acquires one image with its onboard camera and extracts SURF features [21]. The local matching is only applied to pairs of images which are connected in the communication graph. For the local matching the epipolar constraint combined with RANSAC is imposed [66]. The detection and resolution of inconsistencies is analyzed for four different typical communication graphs (Fig. 8.22).

Table 8.1: Initial associations between the 9 local maps

Comm. graph	(a)	(b)	(c)
Association sets (ground truth)	242	284	400
Association sets	182	218	290
Good links (true positives)	160	190	228
Missing links (false negatives)	82	94	172
Spurious links (false positives)	22	28	62
Conflictive sets (C)	3	5	8
Number of features m_{sum}	138	144	154
Conflictive features	16	24	51

Table 8.2: Management of the inconsistencies

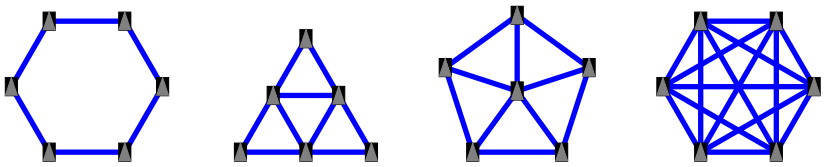
Comm. graph	(a)	(b)	(c)
Iterations	1	1	1
Initial conflictive sets	3	5	8
Deleted links	6	10	34
Good deleted links (true positives)	2	2	12
Spurious deleted links (false positives)	4	8	22



;

Figure 8.21: Images acquired by a team of 6 robots moving in formation. We illustrate as well an example of one inconsistency found by the algorithm. The inconsistency is represented by the whole set of depicted links (green, blue and black). It is observed that if all the links are considered, features of the same image are matched. After executing Algorithm 5.5.1 (Chapter 5) the inconsistency is solved. In this example the root of the inconsistency is the top-middle image. The black line is the link deleted by the algorithm. Solid green lines represent one of the conflict free components and dashed blue lines the second. For clarity, the rest of the SURF features are only shown in the top middle image.

Although the epipolar constraint discards most part of the wrong matches, some spurious associations are still found. Figure 8.21 shows an example of an inconsistency found



(a) Ring (b) Pyramidal (c) Star-Ring (d) Complete

Figure 8.22: Formations used in the experiments

by our algorithm in the case of the formation (d). The figure also shows how the Algorithm 5.5.1 (Chapter 5) solves such inconsistency; the result is a subset of the initial matches with more connected components than before but without any conflict.

More general results about the experiment can be seen in Table 8.3. Since for this experiment there is no ground truth available, the number of missing and spurious links found in the local matching cannot be provided. Some interesting conclusions are extracted with the obtained results. First of all, the number of association sets and the number of features involved m_{sum} are increased with the number of edges in $\mathcal{G}_{com}$. The more local associations there are, the more matches the algorithm will have available. With respect to the inconsistencies, they grow with the number of cycles in $\mathcal{G}_{com}$ because each cycle can generate inconsistencies independently of the rest of the communication network. The size of the cycles also influence on the conflicts, cycles of smaller length will cause more inconsistencies because the number of local associations required to find a conflict is also smaller. We define a full match as an association set in which the n cameras

Table 8.3: Initial associations (before propagation)

Formation	(a)	(b)	(c)	(d)
Association sets	528	590	605	632
Conflictive sets (c)	2	22	31	72
Number of features m_{sum}	1335	1643	1743	2069
Conflictive features	14	144	204	521
Full Matches	5	11	14	11
Partial Matches	214	271	302	391

of the network match the same feature. The optimal solution is found when the network finds the m full matches. Although ground truth is not available in this examples, by looking at the correspondences we have counted the amount of full matches. This number is very small due to missing matches and occlusions caused by the trees. For that reason we also define a partial match when 3 or more cameras correctly match the same feature, because in such case the propagation is required for the association.

Cameras execute the resolution algorithm. Table 8.4 shows the statistics on how are the solutions to the conflicts provided by our algorithm. After executing the resolution algorithm, all the inconsistencies have been solved (second and third rows in Table 8.4), and both the number of full and partial matches are increased (forth and fifth rows). The spurious and the good links that the algorithm deletes, have been manually counted (last two rows). It can be seen that in general, the algorithm tends to delete more spurious

than good links.

Table 8.4: Management of the inconsistencies

Formation	(a)	(b)	(c)	(d)
Initial conflictive sets	2	22	31	72
Resulting conflictive sets	0	0	0	0
Resulting incons. feats.	0	0	0	0
Full Matches	6	11	14	16
Partial Matches	215	286	326	447
Deleted links	2	26	46	131
Good deleted links (true positives)	1	2	19	43
Spurious deleted links (false positives)	1	24	27	88

Data Association and Map Merging with RGB-D Data

We have performed a set of experiments using RGB-D cameras (Fig. 8.23), which provides both regular RGB (Fig. 8.24, first row) and depth image information (Fig. 8.24, second row). Thus, it is possible to compute the cloud of points in 3D from a single image (Fig. 8.24, third row).

Figure 8.23: RGB-D camera used.

We consider a scenario with 9 robots. Initially, robots are placed at unknown poses in the environment. From their initial pose, robots take an image of the scene (Fig. 8.25). They extract SIFT or SURF features from their RGB images and they use the depth information and the camera parameters to compute the 3D position of these features, as in Fig. 8.24, third row.

Given the sets of features of two robots, it is possible to establish matches based on the SIFT/SURF descriptor of the features. Then, robots compute their relative pose (rotation and translation) using the matches and the 3D position of the features in a robust way (RANSAC) and discard matches that disagree with the most supported relative pose candidate (Figs. 8.26).

Typically, the number of matches between overlapping images is high, and the noise in the image points is small. Therefore, the previous method provides highly accurate

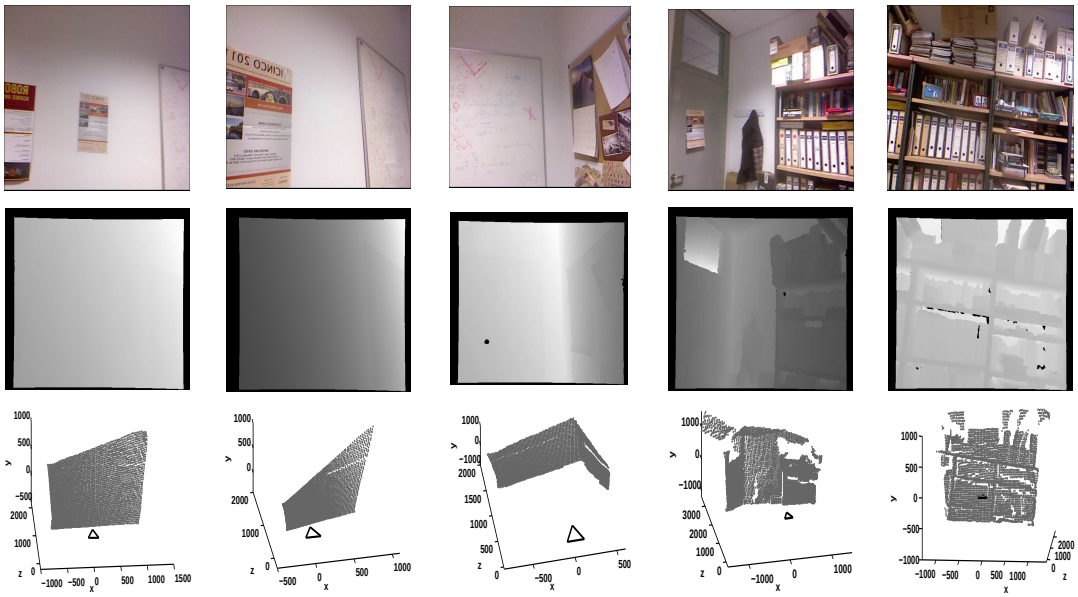


Figure 8.24: An example of the images obtained by the 9 robots with the RGB-D sensor.

Figure 8.25: Images taken from the first robot poses. We are depicting the matches (red lines) between some of the images.

results. In Fig. 8.27 we are showing the result of rotating and translating the 3D points of images in Fig. 8.26 according to the relative pose estimated.

Robots use their initial images to compute the relative poses of their nearby robots (Fig. 8.25) and based on this information, they compute their pose in the common reference frame as explained in Chapter 6 (Fig. 8.28). This method to obtain the robot poses in the common frame is not only restricted to RGB-D images taken from the first robot poses. It can be equivalently applied to images acquired during the exploration, or to the local maps of the robots.

Each robot explores a region and builds a map of the environment. We let each robot execute a SLAM algorithm with SIFT/SURF features parameterized in 3D cartesian coordinates. Robots are represented by their 3D position and orientation, and robot motions are predicted by computing the relative rotation and translation between successive images. Fig. 8.29 shows the resulting map (red points and ellipses) obtained by robot 3 (dark gray triangle) along its trajectory (dashed line). We also show the 3D RGB-D

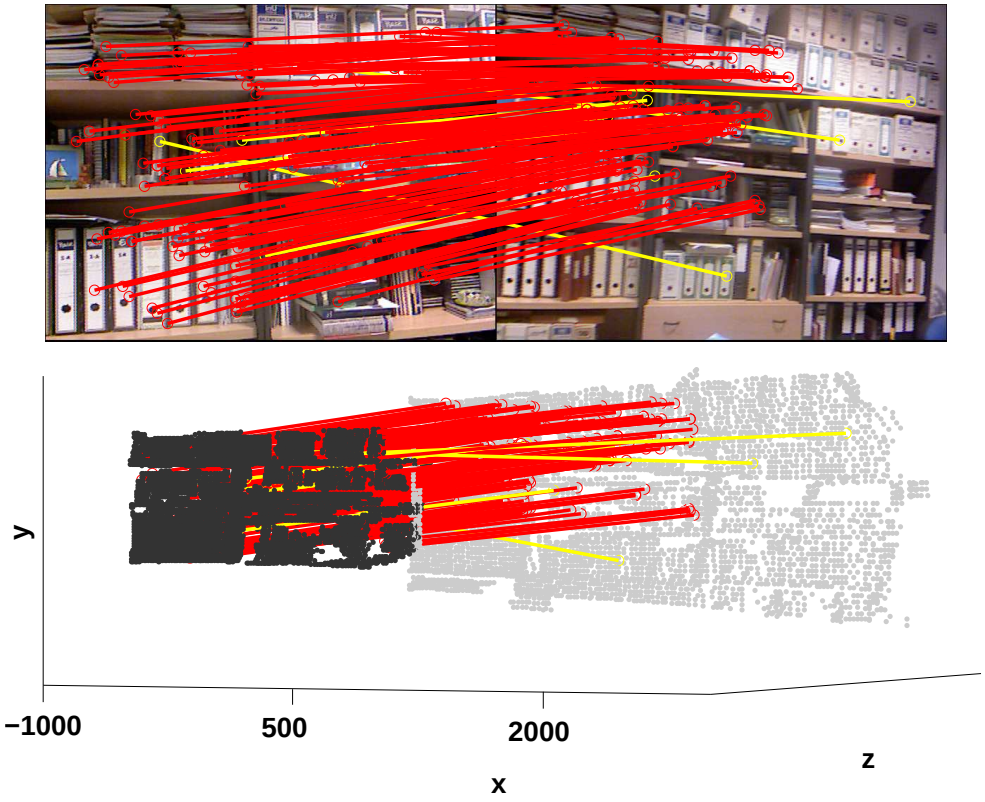


Figure 8.26: Surf / Sift matches (red) that satisfy the relative rotation and translation restriction. The ones that are rejected are depicted in yellow.

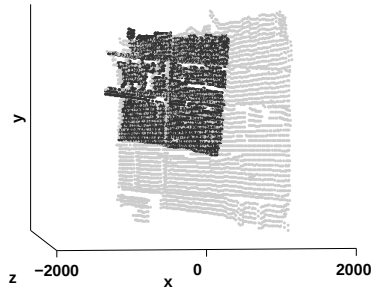


Figure 8.27: Points rotated and translated according to the previously computed relative poses.

points observed from some of the steps of the robot trajectory (light gray points) to give an idea of the scene.

After exploring, nearby robots compute the local data association between their maps based on the SIFT/SURF descriptors and the position of their features. Then, they propagate the local associations and find and solve inconsistencies as explained in Chapter 5 (Fig. 8.30).

Finally, they merge their local maps and build a global map of the environment using the communication graph in Fig. 8.31. In Fig. 8.32 we explain how this communication

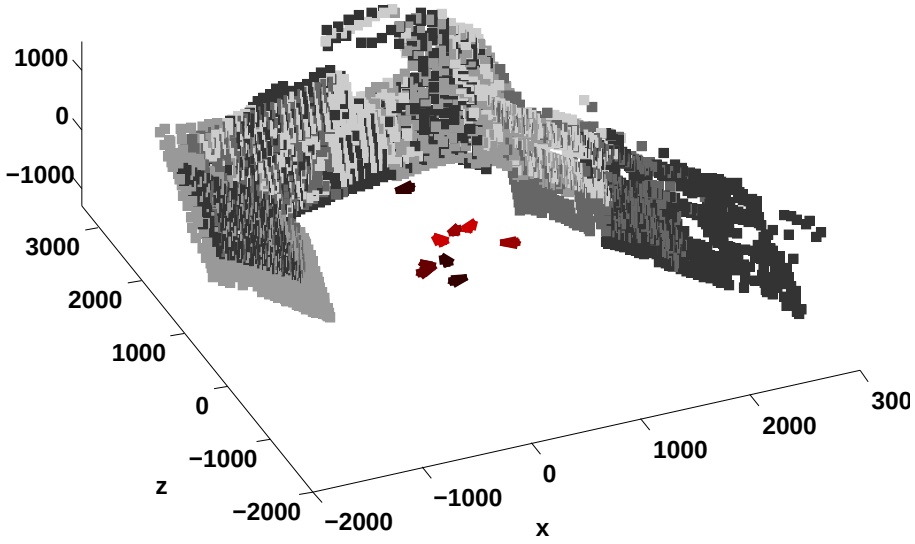


Figure 8.28: Initial robot poses in the global frame (red triangles). The RBG-D points observed from these initial poses are displayed to give an idea of the scene and of the accuracy of the obtained poses.

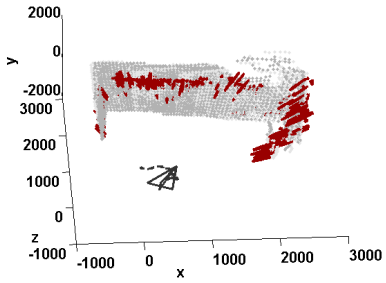


Figure 8.29: Robot 3 (dark gray triangle) explores the environment and builds a map (red points and ellipses). The RBG-D points observed from some steps of the robot trajectory displayed (light gray points) to give an idea of the scene.

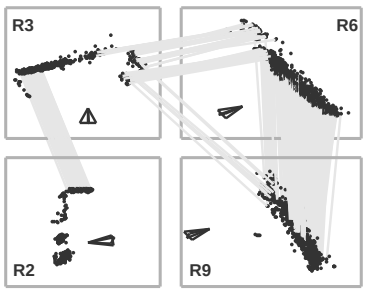


Figure 8.30: Robots (dark triangles) compute the data association (light lines) between their local maps (dark dots). Here we display the associations between robots $R2, R3, R6$, and $R9$.

graph affect the information exchange. We are displaying a link (blue line) between pairs of robots that have received information from each other during the any previous iteration. Initially (Fig. 8.32 **(a)**), each robot only has its local information and thus there are no lines in the graph. During the first iteration (Fig. 8.32 **(b)**), robots exchange information with their one-hop neighbors (blue lines in Fig. 8.31). At iteration 2 robots exchange data with their neighbors again according to the graph in Fig. 8.31, and thus they have access to two-hop neighbors data (lines in Fig. 8.32 **(c)**). The process is repeated during the next iteration, having access to three-hop neighbors data (Fig. 8.32 **(d)**), and so on. After iteration 4 (Fig. 8.32 **(e)-(f)**), each robot has received information from all the other robots. Additional iterations allow the robots to obtain a more accurate estimate of the global map.

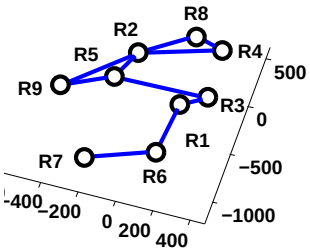


Figure 8.31: Robots (circles) exchange data with their neighbors in the communication graph (linked trough lines).

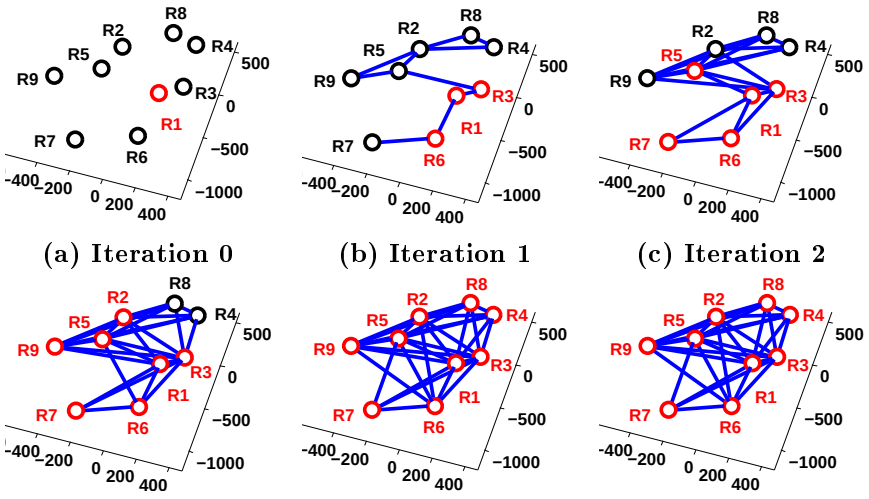
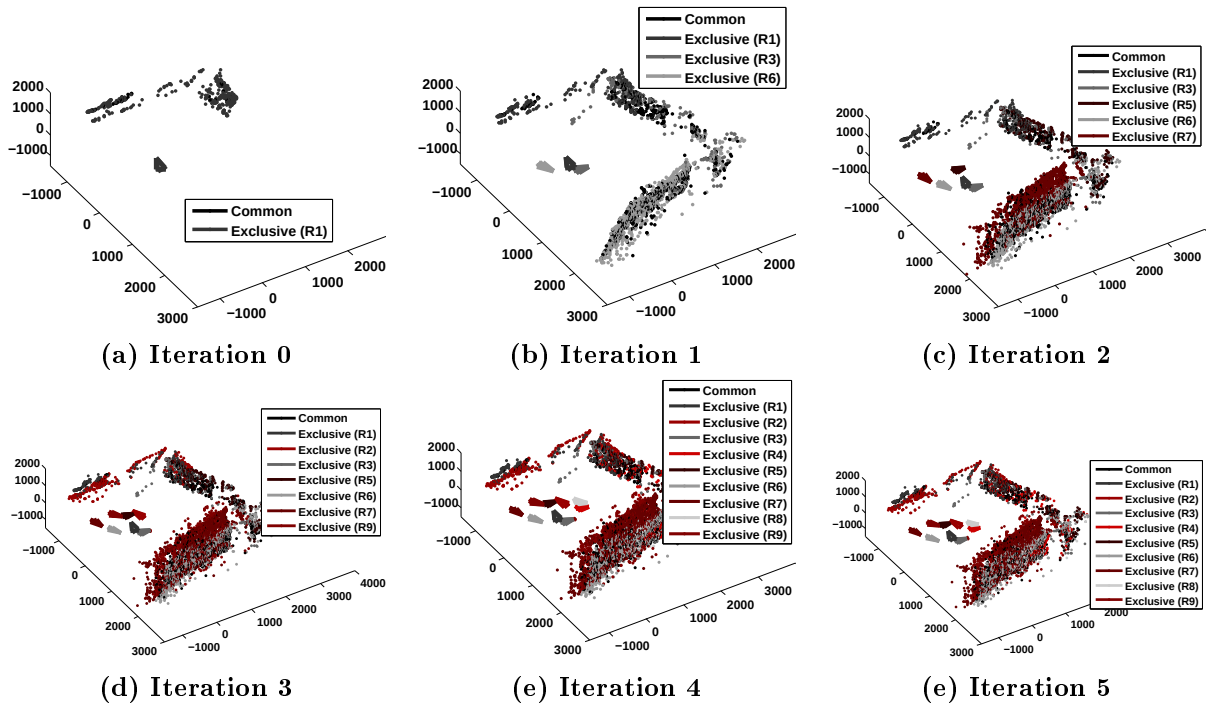


Figure 8.32: Initially, robots only know their local information **(a)**. At iteration 1, robots (circles) exchange data with their neighbors in the communication graph (linked trough lines) depicted in **(b)**. Robots keep on exchanging data using the communication graph in **(b)** during several iterations. Blue lines in figures **(c)-(f)** indicate that the two robots have received information from each other during the previous iterations. Red circles are robots that have received data from $R1$; equivalently, $R1$ has received data of these robots as well.

Robots fuse their maps as explained in Chapter 3. Fig. 8.33 shows the global map estimated by robot $R1$ along iterations 0 to 4, with contains features observed exclusively by a single robot (exclusive), as well as features observed by several robots (common). After 4 iterations, robot $R1$ has received information from all the other robots and thus its map already contains estimates for all the features observed by the team. Successive iterations of the map merging algorithm produce more accurate estimates of the features (Fig. 8.34).



(a) Iteration 0 **(b) Iteration 1** **(c) Iteration 2**

(d) Iteration 3 **(e) Iteration 4** **(e) Iteration 5**

Figure 8.33: Global map estimated by robot $R1$ at iterations $0-5$. Common features observed by several robots and exclusive areas observed by a single robot are depicted in different colors.

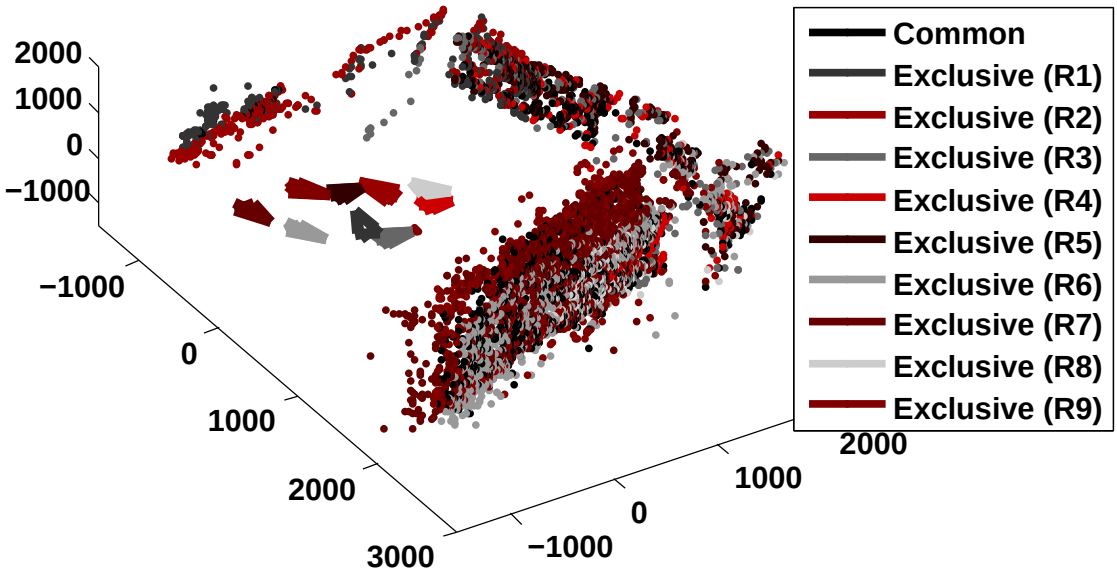


Figure 8.34: Global map estimated by robot $R1$ after 20 iterations of the map merging algorithm.

Chapter 9
Conclusions

Along this document, the problem of perception in robotic networks has been investigated. We have presented a new method for merging stochastic feature-based maps acquired by a team of robots for scenarios with limited communication. The robots explore an environment and build their local maps. When they finish the exploration, they fuse their information and build a global map. The whole method is fully decentralized, relying exclusively on local interactions between neighboring robots. Under fixed connected communication graphs, or time-varying jointly connected topologies, the estimates at each robot asymptotically converge to the global map. Moreover, the intermediate estimates at each robot present interesting properties that allow their use at any time: (i) the mean of the global map estimated by each robot is unbiased at each iteration; (ii) the numerical covariance of the global map estimated by each robot, which cannot be locally computed, is bounded by the locally computed covariance. Experimental results have shown the performance of the method for robots equipped with omnidirectional and conventional cameras. The robustness of the map fusion algorithm under link failures and changes in the communication topology has been demonstrated theoretically and tested experimentally.

We have investigated the dynamic map merging case. Along its operation, each robot observes the environment and builds and maintains its local map. Simultaneously, the robots communicate and build a global map of the environment. We have presented an algorithm for dynamically merging visual maps in a robot network with limited communication. This algorithm allows the robots to have a better map of the environment containing the features observed by any other robot in the team. Thus, it helps the coordination of the team in several multi-robot tasks such as exploration or rescue. The algorithm correctly propagates the new information added by the robots to their local maps. We have shown that, with the proposed strategy, the robots correctly track the global map. At the final step, they obtain the last global map, which contains the last updated information at all the robots. In addition, we have formally characterized the convergence speed of this algorithm and given proofs of its convergence.

We have addressed two additional issues that appear in multi-robot perception scenarios: the establishment of a common reference frame, and the association of the features observed by the different robots. We have presented a new technique to match several sets of features observed by a team of robots in a consistent way under limited communications. Local associations are found only within robots that are neighbors in the

communication graph. After that a fully decentralized method to compute all the paths between local associations is carried out, allowing the robots to detect all the inconsistencies related with their observations. For every conflictive set detected, in a second step the method is able to delete local associations to break the conflict using only local communications. The whole method is proved to finish in a finite amount of time finding and solving all the inconsistent associations. Experimental results show the performance of the method in map merging scenarios, and also show its applicability for multiple view problems.

One important problem in this context is the establishment of a common reference frame in the robot network, which has been deeply investigated as well. Usually, robots start at unknown poses and do not share any reference frame. The localization problem consists of establishing this common frame and computing the robots' poses relative to this frame. Each robot is capable of measuring the relative pose of its neighboring robots. However, it does not know the poses of far robots, and it can only exchange data using the range-limited communication network. The network localization problem has been addressed for three different scenarios: the pose network localization from noise-free relative measurements, the pose network localization from noisy relative measurements, and the position network computation from noisy measurements. We have proposed distributed strategies that allow the robots to agree on a common global frame, and to compute their poses or positions relative to the global frame. The presented algorithms exclusively rely on local computations and data exchange with direct neighbors and have been proved to converge under mild conditions on the communication graph. Besides, they only require each robot to maintain an estimate of its own position or pose. Thus, the memory load of the algorithm is low compared to methods where each robot must also estimate the positions or poses of any other robot

In addition, we have investigated strategies for driving the robots to positions where the global map of the environment is more precise, and for speeding up the information fusion algorithms. First, we have proposed a motion control strategy for improving the precision of the local feature-based maps, and as a result, of the global merged map. The described strategy selects a finite set of candidate motions to the robots, and computes its associated cost in the form of the individual contributions of every feature. Therefore, this cost presents a space complexity linear on the map size. This information is used by the team members to negotiate their next motions, presenting the benefit that robots do not need to wait for having a good global map estimate when they coordinate. Second, we have presented a distributed method to compute the algebraic connectivity for networked robot systems with limited communication. The algebraic connectivity establishes the convergence speed of the map merging algorithm. At each iteration, the algorithm produces both an upper and a lower bound estimates of the algebraic connectivity. We have proved theoretically and experimentally that both estimates asymptotically converge to the true algebraic connectivity. We have shown that our method outperforms the classical distributed power iteration, providing more accurate estimates, and improving the network communication usage. The ability to give upper and lower bounds of the algebraic connectivity has been demonstrated to have a great importance for combining this method with higher level algorithms for adaptive consensus in a parallel fashion, i.e.,

where both processes are executed simultaneously.

For all the previous described problems, we have proposed novel solutions which go beyond the current state of the art. We have given theoretical proofs of correctness and tests of performance. Several of the results presented in this thesis have already been published in international journals and conferences with high impact in the robotics community, whereas the most recent results have been submitted and are currently under review.

Bibliography

[1] Alriksson, P., and Rantzer, A. Distributed Kalman filtering using weighted averaging. In *Int. Symposium on Mathematical Theory of Networks and Systems* (Kyoto, Japan, July 2006).

[2] Anderson, B. D. O., Shames, I., Mao, G., and Fidan, B. Formal theory of noisy sensor network localization. *SIAM Journal on Discrete Mathematics 24*, 2 (2010), 684–698.

[3] Aragues, R., Carlone, L., Calafiore, G., and Sagues, C. Multi agent localization from noisy relative pose measurements. In *IEEE Int. Conf. on Robotics and Automation* (Shanghai, China, May 2011), pp. 364–369.

[4] Aragues, R., Carlone, L., Sagues, C., and Calafiore, G. Distributed centroid estimation from noisy relative measurements. *Systems & Control Letters* (2011). Submitted.

[5] Aragues, R., Cortes, J., and Sagues, C. Distributed map merging in a robotic network. In *Workshop on Network Robot Systems, IEEE/RSJ Int. Conf. on Intelligent Robots & Systems* (Nice, France, Sept. 2008).

[6] Aragues, R., Cortes, J., and Sagues, C. Motion control strategies for improved multi robot perception. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (St. Louis, USA, Oct. 2009), pp. 1065 –1070.

[7] Aragues, R., Cortes, J., and Sagues, C. Dynamic consensus for merging visual maps under limited communications. In *IEEE Int. Conf. on Robotics and Automation* (Anchorage, AK, May 2010), pp. 3032–3037.

[8] Aragues, R., Cortes, J., and Sagues, C. Distributed consensus algorithms for merging feature-based maps with limited communication. *Robotics and Autonomous Systems 59*, 3-4 (2011), 163 – 180.

[9] Aragues, R., Cortes, J., and Sagues, C. Distributed consensus on robot networks for dynamically merging feature-based maps. *IEEE Transactions on Robotics* (2011). Submitted.

[10] Aragues, R., Montijano, E., and Sagues, C. Consistent data association in multi-robot systems with limited communications. In *Robotics: Science and Systems* (Zaragoza, Spain, June 2010).

[11] Aragues, R., and Sagues, C. Parameterization and initialization of bearing-only information: a discussion. In *Int. Conf. on Informatics in Control, Automation and Robotics* (Funchal, Portugal, May 2008), vol. RA-1, pp. 252–261.

[12] Aragues, R., Shi, G., Dimarogonas, D. V., Sagues, C., and Johansson, K. H. Distributed algebraic connectivity estimation for adaptive event-triggered consensus. In *American Control Conference* (2012). Submitted.

[13] Avidan, S., Moses, Y., and Moses, Y. Centralized and distributed multi-view correspondence. *International Journal of Computer Vision 71*, 1 (2007), 49–69.

[14] Bailey, T. Constrained initialisation for bearing-only SLAM. In *IEEE Int. Conf. on Robotics and Automation* (Taipei, Taiwan, Sept. 2003), pp. 1996–1971.

[15] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping: part II. *IEEE Robotics & Automation Magazine 13*, 3 (2006), 108–117.

[16] Bailey, T., Nebot, E. M., Rosenblatt, J. K., and Durrant-Whyte, H. Data association for mobile robot navigation: a graph theoretic approach. In *IEEE Int. Conf. on Robotics and Automation* (San Francisco, USA, Apr. 2000), pp. 2512–2517.

[17] Barooah, P., da Silva, N., and Hespanha, J. Distributed optimal estimation from relative measurements: Applications to localization and time synchronization. Tech. rep., University of California, Santa Barbara, 2006.

[18] Barooah, P., and Hespanha, J. Distributed estimation from relative measurements in sensor networks. In *Int. Conf. on Intelligent Sensing and Information Processing* (Chennai, India, Jan. 2005), pp. 88–93.

[19] Barooah, P., and Hespanha, J. Estimation on graphs from relative measurements. *IEEE Control Systems Magazine 27*, 4 (2007), 57–74.

[20] Barooah, P., and Hespanha, J. Error scaling laws for linear optimal estimation from relative measurements. *IEEE Transactions on Information Theory 55*, 12 (2009), 5661–5673.

[21] Bay, H., Tuytelaars, T., and Gool, L. V. SURF: Speeded up robust features. In *European Conference on Computer Vision* (2006), pp. 404–417.

[22] Bennet, D. J., and McInnes, C. R. Distributed control of multi-robot systems using bifurcating potential fields. *Robotics and Autonomous Systems 58*, 3 (2010), 256–264.

[23] Bertsekas, D. P. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications 1* (1992), 7–66.

[24] Bertsekas, D. P. *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, MA, 1999.

[25] Bertsekas, D. P., and Castañón, D. A. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing 17* (1991), 707–732.

[26] Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[27] Bullo, F., Cortes, J., and Martinez, S. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[28] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. Coordinated multi-robot exploration. *IEEE Transactions on Robotics 21*, 3 (2005), 376–386.

[29] Cadena, C., Ramos, F., and Neira, J. Efficient large scale SLAM including data association using the Combined Filter. In *European Conference on Mobile Robotics* (Mlini/Dubrovnik, Croatia, Sept. 2009), pp. 217–222.

[30] Calafiore, G. Distributed randomized algorithms for probabilistic performance analysis. *Systems & Control Letters 58*, 3 (2009), 202–212.

[31] Calafiore, G., and Abrate, F. Distributed linear estimation over sensor networks. *International Journal of Control 82*, 5 (2009), 868–882.

[32] Calafiore, G., Carlone, L., and Wei, M. A distributed gauss-newton approach for range-based localization of multi agent formations. In *IEEE Multi-Conf. on Systems and Control* (Yokohama, Japan, Sept. 2010), pp. 1152 – 1157.

[33] Calafiore, G., Carlone, L., and Wei, M. A distributed gradient method for localization of formations using relative range measurements. In *IEEE Multi-Conf. on Systems and Control* (Yokohama, Japan, Sept. 2010), pp. 1146 – 1151.

[34] Carli, R., Chiuso, A., Schenato, L., and Zampieri, S. Distributed Kalman filtering based on consensus strategies. *IEEE Journal on Selected Areas in Communications 26* (2008), 622–633.

[35] Carlone, L., Aragues, R., Castellanos, J. A., and Bona, B. A first-order solution to simultaneous localization and mapping with graphical models. In *IEEE Int. Conf. on Robotics and Automation* (Shanghai, China, May 2011), pp. 1764–1771.

[36] Carlone, L., Aragues, R., Castellanos, J. A., and Bona, B. A linear approximation for graph-based simultaneous localization and mapping. In *Robotics: Science and Systems* (Los Angeles, CA, USA, June 2011).

[37] Carlone, L., Aragues, R., Castellanos, J. A., and Bona, B. A linear approximation for nonlinear pose graph optimization. *IEEE Transactions on Robotics* (2011). Submitted.

[38] Carpin, S. Fast and accurate map merging for multi-robot systems. *Autonomous Robots 25*, 3 (2008), 305–316.

[39] Carpin, S., Birk, A., and Jucikas, V. On map merging. *Robotics and Autonomous Systems 53*, 1 (2005), 1–14.

[40] Casbeer, D. W., and Beard, R. Distributed information filtering using consensus filters. In *American Control Conference* (St. Louis, USA, June 2009), pp. 1882 –1887.

[41] Censi, A. An accurate closed-form estimate of ICP's covariance. In *IEEE Int. Conf. on Robotics and Automation* (Roma, Italy, Apr. 2007), pp. 3167–3172.

[42] Chang, H. J., Lee, C. S. G., Hu, Y. C., and Lu, Y.-H. Multi-robot SLAM with topological/metric maps. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (San Diego, USA, Oct. 2007), pp. 1467–1472.

[43] Cortes, J. Global and robust formation-shape stabilization of relative sensing networks. *Automatica 45*, 12 (2009), 2754 – 2762.

[44] Cortés, J., Martínez, S., and Bullo, F. Robust rendezvous for mobile networks via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control 51*, 8 (2006), 1289–1298.

[45] Cortés, J., Martínez, S., Karatas, T., and Bullo, F. Coverage control for mobile sensing networks. In *IEEE Int. Conf. on Robotics and Automation* (Washington, USA, 2002), pp. 1327–1332.

[46] Costa, A., Kantor, G., and Choset, H. Bearing-only landmark initialisation with unknown data association. In *IEEE Int. Conf. on Robotics and Automation* (New Orleans, USA, Apr. 2004), pp. 1164–1770.

[47] Davison, A. J. Real-time simultaneous localisation and mapping with a single camera. In *IEEE Int. Conf. on Computer Vision* (Nice, France, Oct. 2003), pp. 1403–1410.

[48] Deming, R. W., and Perlovsky, L. I. Concurrent multi-target localization, data association, and navigation for a swarm of flying sensors. *Information Fusion 8*, 3 (2007), 316 – 330.

[49] Dimarogonas, D. V., and Johansson, K. H. Event-triggered control for multi-agent systems. In *IEEE Conf. on Decision and Control* (Shanghai, China, Dec. 2009), pp. 7131 –7136.

[50] Dunbar, W. B., and Murray, R. M. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica 42*, 4 (2006), 549–558.

[51] Elsner, L., and Mehrmann, V. Convergence of block iterative methods for linear systems arising in the numerical solution of Euler equations. *Numerische Mathematik 59*, 1 (1991), 541–559.

[52] Fax, J. A., and Murray, R. M. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control 49*, 9 (2004), 1465–1476.

[53] Ferrari, V., Tuytelaars, T., and Gool, L. V. Wide-baseline multiple-view correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Madison, USA, June 2003), pp. 718–725.

[54] Ferrari, V., Tuytelaars, T., and Gool, L. V. Wide-baseline multiple-view correspondences. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2003), pp. 718–725.

[55] Fischler, M. A., and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[56] Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Stewart, B. Distributed multirobot exploration and mapping. *IEEE Proceedings 94*, 7 (2006), 1325–1339.

[57] Franceschelli, M., and Gasparri, A. On agreement problems with gossip algorithms in absence of common reference frames. In *IEEE Int. Conf. on Robotics and Automation* (Anchorage, USA, May 2010), pp. 4481–4486.

[58] Franceschelli, M., Gasparri, A., Giua, A., and Seatzu, C. Decentralized Laplacian eigenvalues estimation for networked multi-agent systems. In *IEEE Conf. on Decision and Control* (Shanghai, P. R. China, Dec. 2009), pp. 2717 –2722.

[59] Freeman, R. A., Yang, P., and Lynch, K. M. Stability and convergence properties of dynamic average consensus estimators. In *IEEE Conf. on Decision and Control* (San Diego, CA, Dec. 2006), pp. 398–403.

[60] Frese, U., and Kurlbaum, J. A data set for data association, June 2008. Electronically available at http://www.sfbtr8.spatial-cognition.de/ insidedataassociation/.

[61] Ganguli, A., Cortés, J., and Bullo, F. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics 25*, 2 (2009), 340–352.

[62] Gava, C. C., Vassallo, R. F., Roberti, F., Carelli, R., and Filho, T. F. B. Nonlinear control techniques and omnidirectional vision for team formation on cooperative robotics. In *IEEE Int. Conf. on Robotics and Automation* (Roma, Italy, Apr. 2007), pp. 2409–2414.

[63] Gennaro, M. C. D., and Jadbabaie, A. Decentralized control of connectivity for multi-agent systems. In *IEEE Conf. on Decision and Control* (San Diego, CA, Dec. 2006), pp. 3628 –3633.

[64] Gil, A., Reinoso, O., Ballesta, M., and Julia, M. Multi-robot visual SLAM using a rao-blackwellized particle filter. *Robotics and Autonomous Systems 58*, 1 (2009), 68–80.

[65] Grime, S., and Durrant-Whyte, H. Data fusion in decentralized sensor networks. *Control Engineering Practice 2*, 5 (1994), 849 – 863.

[66] Hartley, R., and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[67] Horn, R. A., and Johnson, C. R. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[68] Householder, A. *The Theory of Matrices in Numerical Analysis*. Dover Publications, New York, 1964.

[69] Howard, A. Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research 25*, 12 (2006), 1243–1256.

[70] Ji, M., and Egerstedt, M. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics 23*, 4 (2007), 693–703.

[71] Julier, S., and Uhlmann, J. K. General decentralised data fusion with covariance intersection (CI). In *Handbook of Multisensor Data Fusion*, D. L. Hall and J. Llinas, Eds. CRC Press, 2001.

[72] Kaess, M., and Dellaert, F. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems 57*, 12 (2009), 1198 – 1210.

[73] Kempe, D., Dobra, A., and Gehrke, J. Gossip-based computation of aggregate information. In *IEEE Symposium on Foundations of Computer Science* (Cambridge, MA, USA, Oct. 2003), pp. 482–491.

[74] Kempe, D., and McSherry, F. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences 74*, 1 (2008), 70 – 83.

[75] Klippenstein, J., Zhang, H., and Wang, X. Feature initialization for bearing-only visual SLAM using triangulation and the unscented transform. In *IEEE Int. Conf. on Mechatronics and Automation* (Harbin, China, Aug. 2007), pp. 1599–1604.

[76] Knuth, J., and Barooah, P. Distributed collaborative localization of multiple vehicles from relative pose measurements. In *Allerton Conference on Communications, Control and Computing* (Urbana-Champaign, USA, Oct. 2009), pp. 314–321.

[77] Kolotilina, L. Bounds for eigenvalues of symmetric block jacobi scaled matrices. *Journal of Mathematical Sciences 79*, 3 (1996), 1043–1047.

[78] Konolige, K., Gutmann, J., and Limketkai, B. Distributed map-making. In *Workshop on Reasoning with Uncertainty in Robotics, Int. Joint Conf. on Artificial Intelligence* (Acapulco, Mexico, Aug. 2003).

[79] Krick, L. Application of graph rigidity in formation control of multi-robot networks. Master's thesis, University of Toronto, Canada, 2007.

[80] Kwok, N. M., and Dissanayake, G. An efficient multiple hypothesis filter for bearing-only SLAM. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Sendai, Japan, Sept. 2004), pp. 736–741.

[81] Kwok, N. M., Ha, Q. P., Huang, S., Dissanayake, G., and Fang, G. Mobile robot localization and mapping using a gaussian sum filter. *International Journal of Control, Automation, and Systems 5*, 3 (2007), 251–268.

[82] Lafferriere, G., Williams, A., Caughman, J., and Veerman, J. J. P. Decentralized control of vehicle formations. *Systems & Control Letters 54*, 9 (2005), 899–910.

[83] Lee, H. S., and Lee, K. M. Multi-robot SLAM using ceiling vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (St. Louis, USA, Oct. 2009), pp. 912–917.

[84] Leung, C., Shoudong, H., and Dissanayake, G. Active SLAM using model predictive control and attractor based exploration. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Beijing, China, Oct. 2006), pp. 5026–5031.

[85] Leung, C., Shoudong, H., and Dissanayake, G. Active SLAM in structured environments. In *IEEE Int. Conf. on Robotics and Automation* (Pasadena, CA, USA, May 2008), pp. 1898–1903.

[86] Leung, K. Y. K., Barfoot, T. D., and Liu, H. Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach. *IEEE Transactions on Robotics 26*, 1 (2010), 62–77.

[87] Leung, K. Y. K., Barfoot, T. D., and Liu, H. H. T. Decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Taipei, Taiwan, Oct. 2010), pp. 3554 –3561.

[88] Lynch, K. M., Schwartz, I. B., Yang, P., and Freeman, R. A. Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics 24*, 3 (2008), 710–724.

[89] Martínez, S., Bullo, F., Cortés, J., and Frazzoli, E. On synchronous robotic networks – Part II: Time complexity of rendezvous and deployment algorithms. In *IEEE Conf. on Decision and Control* (Seville, Spain, Dec. 2005), pp. 8313–8318.

[90] Michael, N., Zavlanos, M. M., Kumar, V., and Pappas, G. J. Distributed multi-robot task assignment and formation control. In *IEEE Int. Conf. on Robotics and Automation* (Pasadena, CA, May 2008), pp. 128–133.

[91] Moakher, M. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications 24*, 1 (2002), 1–16.

[92] Montiel, J. M. M., Civera, J., and Davison, J. Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems* (Philadelphia, USA, Aug. 2006).

[93] Montijano, E., Aragues, R., and Sagues, C. Distributed multi-view matching in networks with limited communications. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2011). Submitted.

[94] Montijano, E., Martínez, S., and Sagues, C. De-RANSAC: Robust distributed consensus in sensor networks. Submitted, May 2010.

[95] Mostagh, N., and Jadbabaie, A. Distributed geodesic control laws for flocking of nonholonomic agents. *IEEE Transactions on Automatic Control 52*, 4 (2007), 681 – 686.

[96] Nebot, E. M., Bozorg, M., and Durrant-Whyte, H. F. Decentralized architecture for asynchronous sensors. *Autonomous Robots 6*, 2 (1999), 147–164.

[97] Neira, J., and Tardós, J. D. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation 17*, 6 (2001), 890–897.

[98] Okabe, A., Boots, B., and Sugihara, K. Nearest neighbourhood operations with generalized Voronoi diagrams: A review. *International Journal of Geographical Information Systems 8*, 1 (1994), 43–71.

[99] Okabe, A., and Suzuki, A. Locational optimization problems solved through Voronoi diagrams. *European Journal of Operational Research 98*, 3 (1997), 445–56.

[100] Olfati-Saber, R. Distributed Kalman filter with embedded consensus filters. In *IEEE Conf. on Decision and Control* (Seville, Spain, 2005), pp. 8179–8184.

[101] Olfati-Saber, R. Distributed Kalman filtering for sensor networks. In *IEEE Conf. on Decision and Control* (New Orleans, LA, Dec. 2007), pp. 5492–5498.

[102] Olfati-Saber, R., Fax, J. A., and Murray, R. M. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE 95*, 1 (2007), 215–233.

[103] Olfati-Saber, R., and Murray, R. M. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control 49*, 9 (2004), 1520–1533.

[104] Ouyang, W., Tombari, F., Mattoccia, S., Stefano, L. D., and Cham, W. Performance evaluation of full search equivalent pattern matching algorithms. To appear.

[105] Papadimitriou, C. H., and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998, ch. 6.1 The Max-Flow, Min-Cut Theorem, pp. 120–128.

[106] Parker, L. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents 2*, 1 (2008), 5–14.

[107] Parker, L. E. Current state of the art in distributed robotic systems. In *Distributed Autonomous Robotic Systems 4*, L. E. Parker, G. Bekey, and J. Barhen, Eds. Springer Verlag, 2000, pp. 3–12.

[108] Paz, L. M., Tardos, J. D., and Neira, J. Divide and conquer: EKF SLAM in $O(n)$. *IEEE Transactions on Robotics 24*, 5 (2008), 1107–1120.

[109] Peleg, D. *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 2000.

[110] Pfingsthorn, M., Slamet, B., and Visser, A. A scalable hybrid multi-robot SLAM method for highly detailed maps. In *Lecture Notes in Artificial Intelligence*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds., vol. 5001. 2008, pp. 457–464.

[111] Preciado, V. M., Zavlanos, M. M., Jadbabaie, A., and Pappas, G. J. Distributed control of the laplacian spectral moments of a network. In *American Control Conference* (Baltimore, Maryland, USA, July 2010), pp. 4462 –4467.

[112] Qu, Z., Li, C., and Lewis, F. Cooperative control based on distributed estimation of network connectivity. In *American Control Conference* (San Francisco, CA, USA, July 2011), pp. 3441 – 2446.

[113] Ren, W., and Beard, R. W. *Distributed Consensus in Multi-vehicle Cooperative Control*. Communications and Control Engineering. Springer Verlag, London, 2008.

[114] Rizzini, D. L., and Caselli, S. Metric-topological maps from laser scans adjusted with incremental tree network optimizer. *Robotics and Autonomous Systems 57*, 10 (2009), 1036–1041.

[115] Rocha, R., Dias, J., and Carvalho, A. Cooperative multi-robot systems: : A study of vision-based 3-D mapping using information theory. *Robotics and Autonomous Systems 53*, 3-4 (2005), 282–311.

[116] Rocha, R., Dias, J., and Carvalho, A. Cooperative multi-robot systems a study of vision-based 3-D mapping using information theory. In *IEEE Int. Conf. on Robotics and Automation* (Barcelona, Spain, Apr. 2005), pp. 384–389.

[117] Rodríguez-Losada, D., Matía, F., and Jiménez, A. Local maps fusion for real time multirobot indoor simultaneous localization and mapping. In *IEEE Int. Conf. on Robotics and Automation* (New Orleans, USA, Apr. 2004), pp. 1308–1313.

[118] Roumeliotis, S., and Bekey, G. Distributed multirobot localization. *IEEE Transactions on Robotics and Automation 18*, 5 (2002), 781–795.

[119] Russell, W. J., Klein, D., and Hespanha, J. P. Optimal estimation on the graph cycle space. In *American Control Conference* (Baltimore, USA, June 2010), pp. 1918–1924.

[120] Sagues, C., Murillo, A. C., Guerrero, J. J., Goedemé, T., Tuytelaars, T., and Gool, L. V. Localization with omnidirectional images using the 1D radial trifocal tensor. In *IEEE Int. Conf. on Robotics and Automation* (Orlando, USA, May 2006), pp. 551–556.

[121] Sarlette, A., Sepulchre, R., and Leonard, N. E. Autonomous rigid body attitude synchronization. *Automatica 45*, 2 (2008), 572–577.

[122] Savvides, A., Garber, W. L., Moses, R., and Srivastava, M. B. An analysis of error inducing parameters in multihop sensor node localization. *IEEE Transactions on Mobile Computing 4*, 6 (2005), 567–577.

[123] Schultz, A. C., and Parker, L. E., Eds. *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[124] Seyboth, G. S., Dimarogonas, D. V., and Johansson, K. H. Control of multi-agent systems via event-based communication. In *IFAC World Congress* (Milano, Italy, Aug. 2011). Accepted.

[125] Sim, R. Stable exploration for bearings-only SLAM. In *IEEE Int. Conf. on Robotics and Automation* (Barcelona, Spain, Apr. 2005), pp. 2411–2416.

[126] Simonetto, A., Keviczky, T., and Babuska, R. On distributed maximization of algebraic connectivity in robotic networks. In *American Control Conference* (San Francisco, CA, USA, July 2011), pp. 2180 – 2185.

[127] Skrjanc, I., and Klancar, G. Optimal cooperative collision avoidance between multiple robots based on bernstein-bézier curves. *Robotics and Autonomous Systems 58*, 1 (2010), 1–9.

[128] Solá, J., Monin, A., Devy, M., and Lemaire, T. Undelayed initialization in bearing only SLAM. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Edmonton, Canada, Aug. 2005), pp. 2499–2504.

[129] Spanos, D. P., Olfati-Saber, R., and Murray, R. M. Approximate distributed Kalman filtering in sensor networks with quantifiable performance. In *Symposium on Information Processing of Sensor Networks (IPSN)* (Los Angeles, CA, Apr. 2005), pp. 133–139.

[130] Spanos, D. P., Olfati-Saber, R., and Murray, R. M. Distributed sensor fusion using dynamic consensus. In *IFAC World Congress* (Prague, CZ, July 2005).

[131] Stachniss, C., and Burgard, W. Mapping and exploration with mobile robots using coverage maps. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Las Vegas, USA, Oct. 2003), pp. 27–31.

[132] Suzuki, A., and Drezner, Z. The p-center location problem in an area. *Location Science 4*, 1/2 (1996), 69–82.

[133] Tabuada, P. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control 52*, 9 (2007), 1680 –1685.

[134] Tahbaz-Salehi, A., and Jadbabaie, A. A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times. In *IEEE Conf. on Decision and Control* (San Diego, CA, USA, Dec. 2006), pp. 4664–4669.

[135] Tanner, H. G., Pappas, G. J., and Kumar, V. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation 20*, 3 (2004), 443–455.

[136] Tao, T., Huang, Y., Sun, F., and Wang, T. Motion planning for SLAM based on frontier exploration. In *IEEE Int. Conf. on Mechatronics and Automation* (Harbin, China, Aug. 2007), pp. 2120–2125.

[137] Thrun, S., and Liu, Y. Multi-robot SLAM with sparse extended information filters. In *Int. Symposium of Robotics Research* (Sienna, Italy, Oct. 2003), pp. 254–266.

[138] Thrun, S., Liu, Y., Koller, D., Ng, A., and Durrant-Whyte, H. Simultaneous localisation and mapping with sparse extended information filters. *International Journal of Robotics Research 23*, 7-8 (2004), 693–716.

[139] Trawny, N., and Roumeliotis, S. A unified framework for nearby and distant landmarks in bearing-only SLAM. In *IEEE Int. Conf. on Robotics and Automation* (Orlando, USA, May 2006), pp. 1923–1929.

[140] Trawny, N., Roumeliotis, S. I., and Giannakis, G. B. Cooperative multi-robot localization under communication constraints. In *IEEE Int. Conf. on Robotics and Automation* (Kobe, Japan, May 2009), pp. 4394–4400.

[141] Trawny, N., Zhou, X. S., Zhou, K. X., and Roumeliotis, S. I. Inter-robot transformations in 3-d. *IEEE Transactions on Robotics 26*, 2 (2010), 226–243.

[142] Tron, R., Vidal, R., and Terzis, A. Distributed pose averaging in camera networks via consensus on SE(3). In *ACM/IEEE International Conference on Distributed Smart Cameras* (Stanford University, USA, Sept. 2008).

[143] Utete, S., and Durrant-Whyte, H. F. Routing for reliability in decentralised sensing networks. In *American Control Conference* (June 1994), vol. 2, pp. 2268 – 2272.

[144] Varghese, B., and McKee, G. A mathematical model, implementation and study of a swarm system. *Robotics and Autonomous Systems 58*, 3 (2010), 287–294.

[145] Vidal, R., Shakernia, O., and Sastry, S. Omnidirectional vision-based formation control. In *Allerton Conference on Communications, Control and Computing* (Oct. 2002), pp. 1625–1634.

[146] Vincent, R., Fox, D., Ko, J., Konolige, K., Limketkai, B., Morisset, B., Ortiz, C., Schulz, D., and Stewart, B. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence 52*, 1 (2008), 229–255.

[147] Wei, M., Aragues, R., Sagues, C., and Calafiore, G. Distributed noisy range network localization. In *American Control Conference* (2012). Submitted.

[148] Wenhardt, S., Deutsch, B., Angelopoulou, E., and Niemann, H. Active visual object reconstruction using d-, e-, and t-optimal next best views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Minnesota, USA, June 2007), pp. 1–7.

[149] Williams, S. B., and Durrant-Whyte, H. Towards multi-vehicle simultaneous localisation and mapping. In *IEEE Int. Conf. on Robotics and Automation* (Washington, DC, USA, May 2002), pp. 2743–2748.

[150] Xi, B., Guo, R., Sun, F., and Huang, Y. Simulation research for active simultaneous localization and mapping based on Extended Kalman Filter. In *IEEE Int. Conf. on Automation and Logistics* (Qingdao, China, Sept. 2008), pp. 2443–2448.

[151] Xiao, L., and Boyd, S. Fast linear iterations for distributed averaging. *Systems & Control Letters 53* (2004), 65–78.

[152] Xiao, L., Boyd, S., and Lall, S. A scheme for robust distributed sensor fusion based on average consensus. In *Symposium on Information Processing of Sensor Networks (IPSN)* (Los Angeles, CA, Apr. 2005), pp. 63–70.

[153] Xiao, L., Boyd, S., and Lall, S. A space-time diffusion scheme for peer-to-peer least-square estimation. In *Symposium on Information Processing of Sensor Networks (IPSN)* (Nashville, TN, Apr. 2006), pp. 168–176.

[154] Yang, P., Freeman, R., Gordon, G., Lynch, K., Srinivasa, S., and Sukthankar, R. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica 46*, 2 (2010), 390 – 396.

[155] Yang, P., Freeman, R. A., and Lynch, K. M. Distributed cooperative active sensing using consensus filters. In *IEEE Int. Conf. on Robotics and Automation* (Roma, Italy, Apr. 2007), pp. 405–410.

[156] Zavlanos, M. M., and Pappas, G. J. Controlling connectivity of dynamic graphs. In *IEEE Conf. on Decision and Control* (Seville, Spain, Dec. 2005), pp. 6388 –6393.

[157] Zavlanos, M. M., and Pappas, G. J. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics 24*, 6 (Dec. 2008), 1416 –1428.

[158] Zhou, X., and Roumeliotis, S. Robot-to-robot relative pose estimation from range measurements. *IEEE Transactions on Robotics 24*, 6 (2008), 1379–1393.

[159] Zhou, X. S., and Roumeliotis, S. I. Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Beijing, China, Oct. 2006), pp. 1785–1792.

Appendix A
Averaging Algorithms and Metropolis Weights

Throughout this document, we frequently refer to averaging algorithms. They have become very popular in sensor networks due to their capability to reach agreement in a distributed way. Let us assume that each robot $i \in \mathcal{V}$ has initially a scalar value $z_i(0) \in \mathbb{R}$. Let $W \in \mathbb{R}_{\geq 0}^{n \times n}$ be a doubly stochastic matrix such that $W_{i,j} > 0$ if $(i,j) \in \mathcal{E}$ and $W_{i,j} = 0$ when $j \notin \mathcal{N}_i$. This matrix is such that $W_{i,i} \in [\alpha, 1]$, $W_{i,j} \in \{0\} \cup [\alpha, 1]$ for all $i, j \in \mathcal{V}$, for some $\alpha \in (0, 1]$. Assume the communication graph $\mathcal{G}$ is connected. If each robot $i \in \mathcal{V}$ updates $z_i(t)$ at each time step $t \geq 0$ with the following averaging algorithm,

$$z_i(t+1) = \sum_{j=1}^{n} W_{i,j}\ z_j(t), \tag{A.1}$$

then, as $t \to \infty$, the variables $z_i(t)$ reach the same value for all $i \in \mathcal{V}$, i.e., they reach a consensus. Moreover, the consensus value is the average of the initial values,

$$\lim_{t \to \infty} z_i(t) = z_\star = \frac{1}{n} \sum_{j=1}^{n} z_j(0), \tag{A.2}$$

for all $i \in \mathcal{V}$ [27, 113]. Observe that each robot i updates its variables $z_i(t)$ using local information since the weight matrix has zero entries for non-neighboring robots, $W_{i,j} = 0$ when $j \notin \mathcal{N}_i$. Let $\mathbf{e}(t) = (z_1(t), \ldots, z_n(t))^T - (z_\star, \ldots, z_\star)^T$ be the error vector at iteration t. The number of iterations t necessary for reaching $||\mathbf{e}(t)||_2 / ||\mathbf{e}(0)||_2 < \epsilon$ ranges between a single iteration for complete graphs, and order $n^2 \log(\epsilon^{-1})$ iterations for networks with lower connectivity like string and circular graphs [27, Theorems 1.79 and 1.80].

A common choice for the matrix $W \in \mathbb{R}^{n \times n}$ is given by the Metropolis weights given by [152],

$$W_{i,j} = \begin{cases} \frac{1}{1+\max\{|\mathcal{N}_i|, |\mathcal{N}_j|\}} & if\ j \in \mathcal{N}_i, j \neq i, \\ 0 & if\ j \notin \mathcal{N}_i, j \neq i, \\ 1 - \sum_{j \in \mathcal{N}_i} W_{i,j}, & if\ j = i, \end{cases} \tag{A.3}$$

for $i, j \in \mathcal{V}$, $j \neq i$, where $|\mathcal{N}_i|$, $|\mathcal{N}_j|$ are the number of neighbors of robots i, j. Note that each robot can compute the weights that affect its evolution using only local information. The algorithm (A.1) using the Metropolis weights W converges to the average of the inputs.

Appendix B
Dataset Visual Data

Throughout this document, we have analyzed the performance of several of our algorithms under real data. Often, we use a data set from [60] with bearing information obtained with vision in an environment of $60 \times 45m$ performing 3297 steps. It is an indoor scenario where the robot moves along corridors and rooms. The data set contains real odometry data and images captured at every step (Figs. B.1 and B.2). The images are processed and measurements to natural landmarks are provided. The natural landmarks are vertical lines extracted from the images and processed in the form of bearing-only data. The observations in the dataset are labeled so that we have the ground-truth data association. This dataset is very challenging for a conventional visual map building algorithm due to the limited field of view of the camera (Sony EVI-371DG). Furthermore, the camera is pointing forward in the same direction of robot motion and the robot traverses rooms (Fig. B.2) and corridors (Fig. B.1) with few features in common. Notice that this situation is much more complex than situations where the camera can achieve big parallax, or systems with ominidirectional cameras, where features within 360 degrees around the robot are observed.

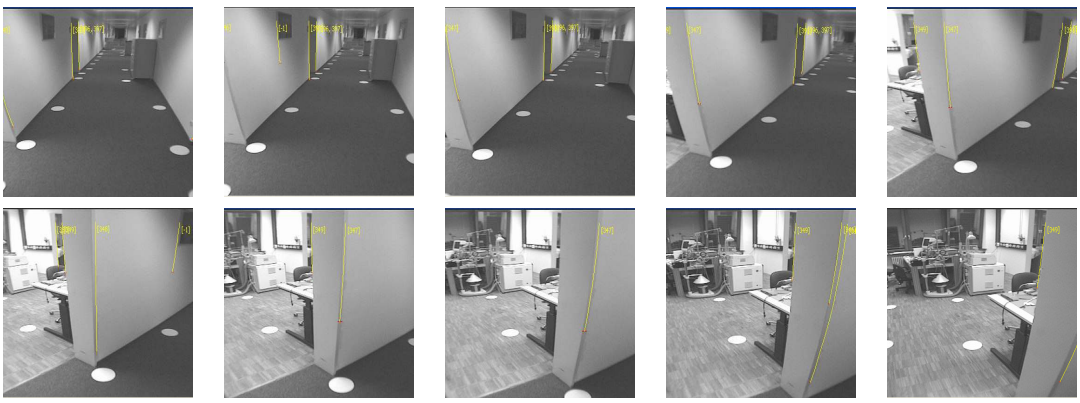


Figure B.1: An example of the images used by the robot team during the navigation to test the proposed algorithms [60]. Although the data set also provides artificial landmarks (white circles on the floor), we test the algorithm using the lines extracted from natural landmarks (in yellow). In the dataset, robots traverse long corridors.

We usually select a section of this dataset and let operate a team of robots in this region. As an example, in Fig. B.3 we display the trajectories followed by 9 robots (colored

Figure B.2: An example of the images used by the robot team during the navigation to test the proposed algorithms [60]. We test the algorithm using the lines extracted from natural landmarks (in yellow). Part of the robots trajectories consists of entering into rooms.

lines). In order to give an idea of the scene structure, we display in black the path in the dataset and a set of artificial landmarks (black dots) placed on both sides of the trajectory, which are not used in the experiment.

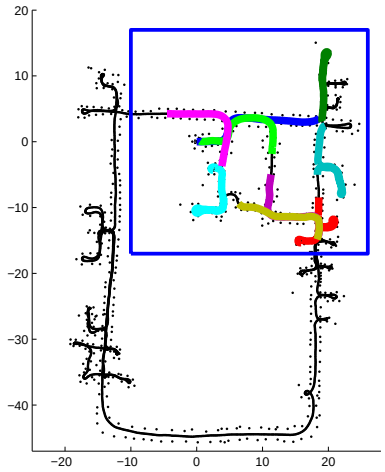


Figure B.3: Example of trajectories followed by 9 robots. They cover a region of 30m x 30m of the whole dataset map. In order to give an idea of the scene structure, we display in black the path in the dataset and a set of artificial landmarks (black dots) placed on both sides of the trajectory, which are not used in the experiment. Here, the rooms can be identified since robots enter and leave them describing short trajectories. The long, straight motions correspond to corridors.