Eduardo Montijano Muñoz

Distributed consensus in multi-robot systems with visual perception

Departamento

Informática e Ingeniería de Sistemas

Director/es

Sagüés Blázquiz, Carlos

ISSN 2254-7606

DISTRIBUTED CONSENSUS IN MULTI-ROBOT SYSTEMS WITH VISUAL PERCEPTION

Autor

Eduardo Montijano Muñoz

Director/es

Sagüés Blázquiz, Carlos

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2012

Universidad Zaragoza

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

PhD Thesis

DISTRIBUTED CONSENSUS IN MULTI-ROBOT SYSTEMS WITH VISUAL PERCEPTION

Eduardo Montijano Muñoz

Director: Carlos Sagüés Blázquiz

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Acknowledgments

This Thesis represents the end of the hard work done during several years. It wouldn't be fair if I didn't spend at least one page thanking all the people that have contributed somehow in its development.

Firstly, I would like to thank Carlos for advising me. I'm aware that I'm not an easy person to work with and yet he has been there the whole time supporting and helping me with everything. I hope this Thesis is not the end of our work together, it is a pleasure to work with you and I believe we still have many things to say together.

This work has been done in collaboration with several people that deserve all my gratitude. Rosario Aragüés and Juan I. Montijano from the University of Zaragoza, Sonia Martínez from the University of California San Diego and Xiaoming Hu and Johan Thunberg from the Royal Institute of Technology in Stockholm. I also had the chance to spend some time in U.C. Berkeley, thanks to Mac Schwager and Claire Tomlin. Ana Cris Murillo provided insightful comments regarding Chapter 6 and without the help of Luis Riazuelo and Danilo Tardioli, the real experiments in Chapter 7 wouldn't have been possible. Finally, thanks to Davinia Vera for all her work and for standing me as her advisor in her PFC.

Family has been a constant source of support during these years. I cannot count the times that my parents, Juan and Teresa, have helped me with crazy last minute stuff. They have seen the best and the worst of me, and they still love me with all their heart. The whole Montijanada, always caring for me. I would also like to mention Pedro and Martha, the political family that anybody would like to have.

Friends have played an important role in the Thesis as well, keeping me sane and taking me out to party when I needed it the most. The list of people that I should mention here is soooooooooo long that I'm going to... mention everybody, because all of them deserve it! Starting with "los Orcos", Gordo, Gollum, Huevo, Chorizo and Carachochi, always destroying the Lair. Miguel and Natalia (friki-fotos), the Mata's family (Quenopoo) and Elena and Raúl, for all the amazing moments we've spent together. Javier and Juan Luis, visiting me all over the globe. The Bohemians, always losing at trivia, and Alex, bitting the dusk playing Soul Calibur. Rosario, Ana and Darío, for all the adventures we lived in California and all the Diisasters, conference mode and coffee breaks are not the same without all of you! Nikos (always fighting with the monsters), Marco (german Bro!), Andrew and all the awesome friends I met while I was in San Diego. Corentin, Enrico and Johan, the three Swedish musketeers. Mac, Simone, Pan, Pg, Ryo and Mike, next time we will eat 100$ bucks in sushi each!

Lastly, I would like to specially thank the most amazing person I've ever met. Without her, this Thesis could have not been possible and she deserves as much credit about it as me. I'm talking about Inés, my girlfriend, who has been there all the time, cheering me up in the dark moments and celebrating with me the good ones. I love you ♡.

Resumen

La idea de equipos de robots actuando con autonomía y de manera cooperativa está cada día más cerca de convertirse en realidad. Los sistemas multi robot pueden ejecutar tareas de gran complejidad con mayor robustez y en menos tiempo que un robot trabajando solo. Por otra parte, la coordinación de un equipo de robots introduce complicaciones que los ingenieros encargados de diseñar estos sistemas deben afrontar.

Conseguir que la percepción del entorno sea consistente en todos los robots es uno de los aspectos más importantes requeridos en cualquier tarea cooperativa, lo que implica que las observaciones de cada robot del equipo deben ser transmitidas a todos los otros miembros. Cuando dos o más robots poseen información común del entorno, el equipo debe alcanzar un consenso usando toda la información disponible. Esto se debe hacer considerando las limitaciones de cada robot, teniendo en cuenta que no todos los robots se pueden comunicar unos con otros.

Con este objetivo, se aborda la tarea de diseñar algoritmos distribuidos que consigan que un equipo de robots llegue a un consenso acerca de la información percibida por todos los miembros. Específicamente, nos centramos en resolver este problema cuando los robots usan la visión como sensor para percibir el entorno. Las cámaras convencionales son muy útiles a la hora de ejecutar tareas como la navegación y la construcción de mapas, esenciales en el ámbito de la robótica, gracias a la gran cantidad de información que contiene cada imagen. Sin embargo, el uso de estos sensores en un marco distribuido introduce una gran cantidad de complicaciones adicionales que deben ser abordadas si se quiere cumplir el objetivo propuesto.

En esta Tesis presentamos un estudio profundo de los algoritmos distribuidos de consenso y cómo estos pueden ser usados por un equipo de robots equipados con cámaras convencionales, resolviendo los aspectos más importantes relacionados con el uso de estos sensores. En la primera parte de la Tesis nos centramos en encontrar correspondencias globales entre las observaciones de todos los robots. De esta manera, los robots son capaces de detectar que observaciones deben ser combinadas para el cálculo del consenso. También lidiamos con el problema de la robustez y la detección distribuida de espurios durante el cálculo del consenso. Para contrarrestar el incremento del tamaño de los mensajes intercambiados por los robots en las etapas anteriores, usamos las propiedades de los polinomios de Chebyshev, reduciendo el número de iteraciones que se requieren para alcanzar el consenso.

En la segunda parte de la Tesis, centramos nuestra atención en los problemas de crear un mapa y controlar el movimiento del equipo de robots. Presentamos soluciones para alcanzar un consenso en estos escenarios mediante el uso de técnicas de visión por computador ampliamente conocidas. El uso de algoritmos de estructura y movimiento nos permite obviar restricciones tales como que los robots tengan que observarse unos a otros directamente durante el control o la necesidad de especificar un marco de referencia común. Adicionalmente, nuestros algoritmos tienen un comportamiento robusto cuando la calibración de las cámaras no se conoce. Finalmente, la evaluación de las propuestas se realiza utilizando un data set de un entorno urbano y robots reales con restricciones de movimiento no holónomas.

Todos los algoritmos que se presentan en esta Tesis han sido diseñados para ser ejecutados de manera distribuida. En la Tesis demostramos de manera teórica las principales propiedades de los algoritmos que se proponen y evaluamos la calidad de los mismos con datos simulados e imágenes reales. En resumen, las principales contribuciones de esta Tesis son:

- Un conjunto de algoritmos distribuidos que permiten a un equipo de robots equipados con cámaras convencionales alcanzar un consenso acerca de la información que perciben. En particular, proponemos tres algoritmos distribuidos con el objetivo de resolver los problemas de encontrar correspondencias globales entre la información de todos los robots, detectar y descartar información espuria, y reducir el número de veces que los robots tienen que comunicarse entre ellos antes de alcanzar el consenso.
- La combinación de técnicas de consenso distribuido y estructura y movimiento en tareas de control y percepción. Se ha diseñado un algoritmo para construir un mapa topológico de manera cooperativa usando planos como características del mapa y restricciones de homografía como elementos para relacionar las observaciones de los robots. También se ha propuesto una ley de control distribuida utilizando la geometría epipolar con el objetivo de hacer que el equipo de robots alcance una orientación común sin la necesidad de observarse directamente unos a otros.

Abstract

The idea of a team of robots executing cooperative tasks in an autonomous manner is everyday closer to become a reality. Multi-robot systems can perform complex tasks with more robustness or in less time than one robot working alone. On the other hand, the coordination of a team of robots introduces new challenges that the designers of these systems must face.

A globally consistent perception of the environment is a key component for the proper cooperation of the team of robots, which requires for the robots to communicate their observations to all the other members. When two or more robots have common observations the team needs to reach a consensus combining all of them. This must be done considering the limitations that each robot has, taking into account that not all the robots can communicate with each other.

To this end, we consider as the main objective of this work the development of distributed algorithms that make a team of robots reach an agreement about the information they perceive. We focus our work in solving this problem when the robots perceive the world using vision sensors. These sensors are very useful in many essential robotic tasks like autonomous navigation and mapping, due to the big amount of information a single image contains. However, in a distributed setup the use of these kind of sensors brings up many complications that need to be addressed.

In this Thesis we present a deep study of distributed consensus algorithms and how they can be used by a team of robots equipped with monocular cameras, solving the most important issues that appear because of the use of these sensors. In the first part of the Thesis we address the problem of finding global correspondences between the observations of the different robots. In this way, the robots know which observations must be combined in the computation of the consensus. We also deal with the problem of robustness and distributed outlier detection, giving a solution to discard erroneous measurements. To counteract the increase in the size of the messages caused by the previous steps, we use the properties of Chebyshev polynomials, reducing the number of iterations required to achieve the consensus.

In the second part of the Thesis, we focus on the problems of mapping the environment and controlling the motion of the team of robots. We apply well known computer vision algorithms to reach the consensus in these two scenarios. We show that using structure from motion, requirements such as the direct observation of the other robots during the control loop or the knowledge of a common frame are avoided. In addition, the lack of calibration information is not a major issue using our algorithms. The evaluation of the solutions is done using a large urban data-set and real non-holonomic robots.

All the algorithms presented in this Thesis are well designed to be executed in a distributed fashion by a team of robots with limited communication capabilities. We theoretically prove the main properties of all the proposed algorithms and test their quality using simulated and real data. Specifically, the main contributions of the Thesis are:

- A set of distributed algorithms that make possible for a team of robots equipped with cameras to reach a consensus about the information they perceive. In particular, we propose three distributed algorithms that solve the problem of finding global correspondences between the robots, detect possible outliers and reduce the total number of communication rounds required by the network to achieve the consensus.

- The combination of distributed consensus and structure from motion techniques in multi-robot perception and control tasks. We design an algorithm to cooperatively build a topological map of the environment considering planes as features and using homography constraints to relate the observations of different robots. We also propose a distributed control law using the epipolar constraint to make the team of robots to reach an agreement in their orientations without the necessity of directly observing each other.

Contents

List of Figures

List of Tables

List of Algorithms

List of symbols

Table 1: List of the most important symbols used in the Thesis

Symbol	Description
$\mathbf{A}$	Association matrix (Chapter 3, Equation (3.3))
$\mathbf{C}_i$	Adjacency matrix relating the planes of robot i (Chapter 6, Section 6.2)
C	Number of conflictive sets (Chapter 3, Definition 3.4.1)
$\mathcal{C}$	One conflictive set (Chapter 3, Definition 3.4.1)
c, d	Parameters of the scaled Chebyshev polynomial (Chapter 5, Equation (5.7))
d_{ij}	Difference between the epipoles i and j (Chapter 7, Equation (7.5))
d_f	Diameter of $\mathcal{G}_{dis}$ (Chapter 3, Equation (3.2))
d_v	Diameter of $\mathcal{G}$ (Chapter 2, Definition 2.2.4)
$\mathbf{E} = [e_{rs}]$	Weighted association matrix (Chapter 3, Equation (3.10))
$\mathcal{E}$	Edges in the communication graph (Chapter 2, Section 2.2.2)
$\mathcal{E}_{dis}$	Edges in the association graph (Chapter 3, Equation 3.2)
e_{ij}, e_{ji}	Epipoles of the images i and j (Chapter 7, Equation (7.3))
$\mathcal{F}_{dis}$	Nodes of the association graph (Chapter 3, Section 3.2.3)
f_r^i	r^{th} feature observed by the i^{th} robot (Chapter 3, Section 3.2)
$\mathcal{G}$	Communication graph between all the robots (Chapter 2, Section 2.2.2)
$\mathcal{G}_{dis}$	Association graph with limited communications (Chapter 3, Section 3.2.3)
$\mathcal{G}_i^\pi$	Topological map of robot i (Chapter 6, Section 6.2)
$\mathbf{H}_{k,k+1}^m$	Homography related plane m in images k and $k+1$ (Chapter 6, Equation (6.1))
h	One hypothesis (Chapter 4, Equation (4.10))
h^*	Hypothesis with the most votes (Chapter 4, Equation (4.11))
$\mathbf{K}$	Calibration matrix (Chapter 6, Section 6.2)
L	Set of the eigenvalues of the time-varying weight matrices (Chapter 5, Equation (5.18))
m_i	Number of features observed by robot i (Chapter 3, Equation 3.2)
$\tilde{m}_i$	Number of features of robot i in the conflict (Chapter 3, Definition 3.4.1)
m_{sum}	Total number of features (Chapter 3, Equation (3.2))
N	Number of robots (Chapter 2, Section 2.2.2)
$\mathcal{N}_i$	Neighbors of robot i (Chapter 2, Definition 2.2.2)
n	Degree of the polynomial and time index (Chapter 5, Equation (5.6)))
$\mathbf{P}_i$	Variable used to compute ML in a distributed manner (Chapter 2, Equation (2.14))
$\mathbf{P}_{rs}$	Generic permutation matrix of rows r and s (Chapter 3, Equation (3.11))
$\mathcal{P}_i$	Set of planes in the map of robot i (Chapter 6, Section 6.2)
$P_n(x)$	Shifted scaled Chebyshev polynomial (Chapter 5, Equation (5.7))
p_{in}	Probability of one datum of being inlier (Chapter 4, Assumption 4.3.1)
	Continued on next page

Table 1 – continued from previous page

Symbol	Description
p_{suc}	Probability of success in RANSAC (Chapter 4, Assumption 4.3.1)
$\mathbf{q}_i$	Variable used to compute ML in a distributed manner (Chapter 2, Equation (2.14))
$\mathcal{S}^i$	Set of features observed by robot i (Chapter 3, Equation 3.2)
$T_n(x)$	Chebyshev polynomial of degree n evaluated in x (Chapter 5, Section 5.2)
$\mathbf{u}_i^{\mathbf{P}}$	Dynamic input for the dynamic voting (Chapter 4, Equation (4.25))
$\mathbf{u}_i^{\mathbf{q}}$	Dynamic input for the dynamic voting (Chapter 4, Equation (4.25))
$\mathbf{u}_i^{v}$	Dynamic input for the dynamic voting (Chapter 4, Equation (4.25))
$u_i(t)$	Dynamic input to compute the number of robots (Chapter 4, Equation (4.7))
$\mathcal{V}$	The set of all robots (Chapter 2, Section 2.2.2)
$\mathcal{V}_{con}$	Set of robots that end up voting for the hypothesis (Chapter 4, Proposition 4.5.2)
$\mathcal{V}_h$	Set of robots that use their information in one hypotheses (Chapter 4, Equation (4.13))
$\mathcal{V}_{in}$	Set of robots with inlier information (Chapter 4, Section 4.4.1)
$\mathcal{V}_{inc}$	Set of robots that participate in a conflict (Chapter 3, Definition 3.4.1)
$\mathbf{v}_i$	i^{th} eigenvector of **W** (Chapter 2, Equation (2.7))
$\mathbf{W}(t) = [w_{ij}]$	Weight matrix at time t (Chapter 2, Equation (2.3))
w_{ij}	Geodesic distance of the epipoles (only in Chapter 7, Equation (7.4))
$\mathbf{X}_{ij}$	Variable to compute the powers of **A** (Chapter 3, Equation (3.5))
$\mathbf{x}(0)$	Initial conditions stacked $[x_1(0), \ldots, x_N(0)]$ (Chapter 2, Equation (2.3))
$\mathbf{x}_i^e$	Homogeneous coordinates (Chapter 4, Equation (4.1))
$\mathbf{x}^h$	Average of the observations in $\mathcal{V}_h$ (Chapter 4, Equation (4.13))
$\bar{x}$	Average of the initial conditions, $\bar{x} = 1/N \sum_{i \in \mathcal{V}} x_i(0)$ (Chapter 2, Equation (2.6))
$x_i(0)$	Initial conditions of robot i (Chapter 2, Definition 2.3.1)
x_i^h	Scale coordinate when using homogeneous coordinates (Chapter 4, Section 4.2)
$\mathbf{y}_r^i$	Variable to compute the improved powers of **A** (Chapter 3, Algorithm 1)
$\mathbf{z}_r$	Variables used in *Maximum Error Cut* to break inconsistencies (Chapter 3, Algorithm 2)
α	Focal length of the camera (Chapter 7, Section 7.2.2)
β	Calibration estimation (Chapter 7, Equation (7.5))
$\boldsymbol{\beta}_i(t)$	Vector to compute the number of robots (Chapter 4, Equation (4.4))
γ_i	i^{th} coefficient in the base defined by the eigenvectors of **W** (Chapter 2, Equation (2.7))
$\boldsymbol{\theta}_i(t)$	Maximum likelihood (ML) estimation of robot i (Chapter 4, Equation (4.20))
$\boldsymbol{\theta}_{ML}$	Maximum likelihood (ML) of the initial conditions (Chapter 2, Equation (2.13))
θ_{ij}	Relative orientation between robots i and j (Chapter 7, Equation (7.2))
$\kappa_n(c)$	Indicator of the eigenvalues position (Chapter 5, Equation (5.28))
$\boldsymbol{\Lambda}_i$	Covariance associated to $x_i(0)$ (Chapter 2, Equation (2.13))
$\boldsymbol{\Lambda}_{ML}$	Covariance of the Maximum likelihood (Chapter 4, Equation (4.20))
λ_i	i^{th} eigenvalue of **W** (Chapter 2, Assumption 2.3.2)
λ_m, λ_M	Parameters of the sifted scaled Chebyshev polynomial (Chapter 5, Equation (5.7))
$\lambda_{\min}, \lambda_{\max}$	Lower and upper bound in the estimation of λ_2 (Chapter 5, Equation (5.30))
$\boldsymbol{\Pi}_i$	Features composing the map of robot i (Chapter 6, Section 6.2)
$\tau(x)$	Function that defines the direct expression the $T_n(x)$ (Chapter 5, Equation (5.4))
$\boldsymbol{\upsilon}_i$	Voting vector (Chapter 4, Section 4.4.2)
Φ	A discrete partition of $\mathbb{R}$ (Chapter 4, Definition 4.2)
$\chi_i(t)$	Chi-square test to vote one hypothesis (Chapter 4, Equation (4.23))
ψ_{ij}	Bearing angle between robots i and j (Chapter 7, Equation (7.2))

Chapter 1

Introduction

1.1 Multi-Robot Systems, Distributed Consensus and Vision Sensors

The idea of autonomous robots helping humans in the execution of different tasks is nowadays a reality. During the last decade, there has been an enormous advance in technology that has made possible for individual robots to perform hard tasks with high precision [135]. From "simple" repetitive tasks like cleaning a house to more sophisticated operations such as mine clearance, or cleanup and recovery of oil spills, robots have become an important element in our lives, see Figure 1.1. However, as happens with humans, cooperative work seems more attractive and produces better results than individual work.

Figure 1.1: Robots have become an important element in our lives. A vacuum robot cleaning a house (left) and a robot handling an explosive (right). Right figure reproduced with permission of the authors [119]

Let us consider for example a brigade of firemen entering a house on fire, Fig. 1.2. It is obvious that a team of firemen will perform all the tasks faster and better than a single fireman. To success, the team requires a common knowledge about the house, for example, where are the exits of the building, the state of the fire in the different rooms or where are located possible survivors.

Figure 1.2: Brigade of firemen acting cooperatively. Figure reproduced with permission of the authors [119]

If instead of a firemen brigade the team is integrated by robots (Fig. 1.3) the problem of cooperation remains similar. Multi-robot approaches present several advantages over mono-robot solutions [18]. A team of mobile robots, possibly combined with static agents such as surveillance cameras, even cooperating with humans, may allow to perform complex tasks that for a single robot would be very difficult to complete, or not achievable at all. Cooperative work allows to perform the tasks in less time or with the least cost. On the other hand, the coordination of a team of robots (Fig. 1.3) introduces new challenges that designers of these systems must face.

Figure 1.3: Team of robots.

One way to deal with multiple robots is by using a central computer that receives and processes all the information of all the robots. Centralized approaches use a specific computer, usually called server, which optimally takes the decisions and coordinates the team. These approaches are easier to design and to implement, however, they are hardly scalable and become unsuitable when the number of robots is large enough. Moreover, if the server has a failure, the task will not be accomplished, which makes the system have too little reliability.

Distributed approaches [78], on the other hand, are harder to design but in general perform better. These solutions assume that each robot has its own computer onboard and sends the information it has from the scene to a subset of the robots, its neighbors. In this way, the team of robots is able to handle changes in the communication network, usually caused by changes in their positions or failures in their systems. As a consequence, the system is more reliable and independent of individual members, Figure 1.4.

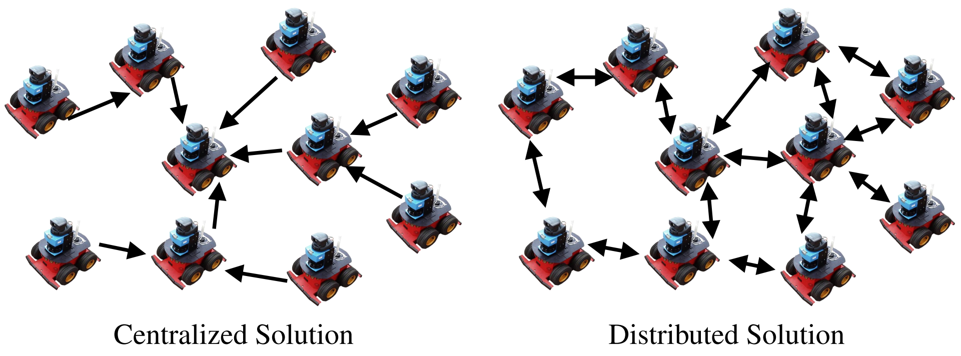


Figure 1.4: Centralized (left) versus distributed (right) solution. In a centralized setup, one robot gathers the information of the whole network and makes the decision. In a distributed scenario all the robots play the same role and exchange messages with all their neighbors. As a consequence, the system is more reliable and independent of individual members.

Researchers of multi-robot systems have turn to nature in many occasions to find ideas to solve different coordination problems in a distributed way. This occurs because nature contains perfect examples of distributed cooperation that can be replicated by a team of robots. Each animal can be seen as an autonomous element interacting with the environment. But at the same time, there is cooperation and interaction with other animals of the same species. In some cases each member of the team is assigned a different role, like ant colonies and swarms of bees, where explorers look for food, builders construct new passages in the colony and the queen is in charge of breeding new offspring.

Other species do not distinguish between different members, every individual behaves equally to the others, following a set of simple rules that lead to emergent behaviors. This is the case, for example, of a fish shoal or a flock of birds moving in formation. There is no leader conducting everybody, each animal individually makes decisions about the direction to move by just looking at near mates. However, the whole group of animals end up moving in the same direction, forming a specific pattern, Figure 1.5.

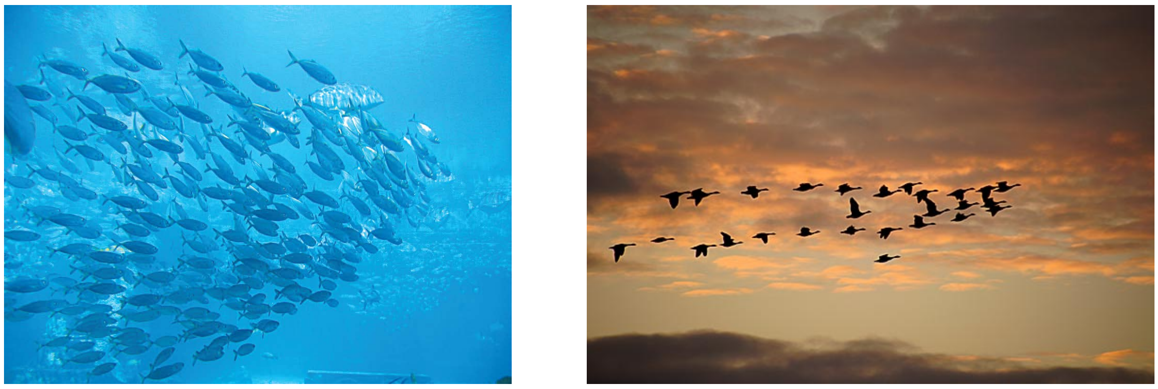


Figure 1.5: A fish shoal (left) and a flock of birds (right) moving in formation. Each individual moves only considering the motion of those animals close to it. However, the full group of animals presents an emergent behavior, moving all in the same direction. Left figure reproduced with permission of the authors [82]. Right figure reproduced with Creative Commons License, copyright of © Ian Britton [16].

This second scenario is very appealing in multi-robot systems [61], to begin with, because all the robots play the same role in the task. This simplifies the task of designing the system because the same algorithm, which is in general quite simple, is executed by all the robots without differences. Secondly, because the robots do not require to have full knowledge about the information perceived by all the robots, it is just enough to exchange information with the robots nearby. Finally, because a global behavior is still achieved. Even when everything is done locally, the system is able to reach some global behavior, e.g., all the robots move in the same direction.

In this framework, one of the most important emergent behaviors that can be asked in a multi-robot system is the consensus between all the robots. The idea of reaching a consensus has a great interest in many cooperative tasks. Generally speaking, we say that the team of robots has reached a consensus when all the robots agree about some data value which is of interest for the team. The essence of these data will depend on the specific application context. For example, the data can be the orientations or the positions of all the robots, and the consensus can be used to make the team of robots move in the same direction, attitude consensus [61, 107, 125], like in Figure 1.6, or gather together at a specific point, rendezvous [32].

Consensus is also of great interest in perception tasks. A fundamental aspect in any robotic system is how to describe the environment it interacts with. The capacity of a multi-robot system to achieve any task depends crucially on its ability to consistently perceive the world. Global perception of the environment is one of the key components in these systems, which means that all the robots need to manage the same information in order to show a good global behavior. However, each robot has limitations in what it sees and who it can communicate with depending on its sensors. This implies that the robots must somehow communicate among themselves

Figure 1.6: Group of aeroplanes moving in the same direction.

their local observations in order to reach a consensus [106] on the information they have of the environment.

Continuing with the firemen example, if the robots want to find wether or not there is a fire going on in a determined room, they could measure the temperature of that room. If the temperature is above some value, then the robots can conclude that there is a fire happening in that room. However, the temperature measured by each robot might not necessarily be the same. Moreover, there might be robots without a temperature sensor. Therefore, proper mechanisms to fuse the information of all the robots in such a way that they reach a consensus are required [155].

Finally, in perception tasks, a key aspect in developing a good consensus algorithm is the information the robots need to exchange and therefore, the sensor used to measure this information. The use of vision sensors in many robotic tasks presents several advantages over other existing sensors. The amount of information that a single image contains is remarkable and current digital cameras have a very low price, specially if we compare it with the price of other sensing devices. In addition, vision sensors have been proved to be very powerful in some essential robotic tasks like exploration and mapping [70] or autonomous navigation and localization [128]. These reasons make vision sensors a really good candidate to be used by a multi-robot team.

From the research point of view, the design of distributed consensus algorithms for multi-robot systems equipped with monocular cameras is very challenging. During the last years, there has been an increasing interest in distributed vision systems [121]. However, there are few works that consider visual information and distributed consensus together, e.g., [60, 99, 145]. This is probably caused by the complications that appear when using these sensors. Communication and computation issues, data association problems and robustness are some of the most important factors that probably caused this void. The amount of information available in one image makes it hard to decide which are the best features to use in order to reach a consensus. Additionally, calibration issues and the lack of depth information are other factors that may affect as well. In this Thesis we will try to find solutions to some of these problems, developing a set of algorithms that allow a team of robots to reach a consensus about their visual information.

1.2 Objectives of this Thesis

In the context of multi-robot systems for perception and control tasks we consider as the main objective of this research the design of distributed consensus algorithms suitable for teams equipped with monocular cameras and with limited communication capabilities.

This main objective can be separated into two related subgoals:

- **Consensus with vision sensors:** As we have mentioned, the use of vision sensors complicates the problem of reaching an agreement by a robotic network. A very important objective of this thesis is to identify and solve the complications that may appear by using these sensors. Issues like the observation of different elements by the robots and the robustness to possible perception outliers and inconsistencies need to be addressed, otherwise, the robots will not achieve the consensus. The goal is to propose distributed algorithms well suited to reach a consensus about visual information. In addition, a priority of the proposed methods will be to keep the good properties of linear iterations to reach the consensus.

- **Visual consensus in perception and control tasks:** A fundamental task in robotics is the construction of a map of the environment where the robot is supposed to work. When this task is done simultaneously by several robots, additional mechanisms are required to ensure that all of them perceive the environment in the same way. Moreover, in many situations it is interesting for the robots to maintain some kind of formation or moving as a flock. In these cases, the use of monocular cameras will constrain the information they can perceive, forcing the robots either to see each other to keep the formation or to perceive the environment, at the risk of losing their place in the pattern. The second objective of this Thesis is to propose a distributed methods based on consensus to solve these problems.

1.3 Contributions of the Thesis and Publications

In this Thesis we present a deep study of the distributed consensus algorithms and how they can be used by a team of robots equipped with monocular cameras. Specifically, the main contributions of the Thesis are the following:

- A set of distributed algorithms that make possible for a team of robots equipped with cameras reach a consensus about the information they perceive. The proposed algorithms follow the scheme of a distributed linear iteration [61]. However, they overcome significant problems that appear using similar existing algorithms when the information to be exchanged comes from vision sensors.
  - The first problem that our algorithms solve is the data association among the observations of all the robots. So far, existing consensus algorithms only deal with the problem of exchanging information about a single element of arbitrary size. However, in real scenarios with vision sensors, each robot can observe many features of the environment. Without proper mechanisms that help the team of robots to distinguish which observations of each member correspond with the observations of the others there is no way of obtain a reliable consensus. In the Thesis we propose a distributed data association algorithm that provides the team of robots with mechanisms to find global correspondences. The algorithm is also able to detect and solve possible conflicts caused by local mistakes.
  - Our data association solution, as any other existing matching algorithm, is not hundred per cent reliable. A small mistake in the initial association can be disastrous for the team of robots in posterior stages. For example, if the location of two different exit doors observed by two robots is mixed, the resulting location of the exit door will be totally wrong and useless to the team. For that reason we also propose a distributed algorithm able to detect outliers when executing the consensus iteration. The algorithm is inspired by the RANSAC [43] algorithm. However, in our approach all the steps are executed in a distributed fashion.
  - Finally, the robustness in the system is achieved by increasing the size of the messages the robots need to exchange. Even for small messages, a main limitation of existing consensus algorithms is the number of iterations (communication rounds) they require before obtaining a good solution. The research community, aware of this problem, has successfully reduced the convergence speed using polynomial techniques [68, 138]. However, these solutions require the network topology to remain fixed and well known. In the Thesis we present a new update rule using Chebyshev polynomials that overcomes these limitations and significantly speeds up the convergence rate to the consensus.

- The use of the adequate visual information to achieve the consensus in perception and control tasks.
  - Distributed consensus in perception tasks using homographies and planar regions. In this Thesis we show that a representation of the scene using planar regions simplifies the problem of reaching a consensus about the environment. The homography constraint between sets of coplanar points is a robust mechanism to detect and match areas of interest in images. It also contains the necessary information to transform the observations of different robots without the necessity of knowing the geometry of the scene, the intrinsic parameters of their cameras or their relative position. We propose an algorithm to cooperatively build a topological map of the environment considering planes as the features to represent it.
  - Distributed consensus in control tasks using the epipolar constraint. We propose a distributed control law that makes use of the epipoles to reach a common attitude. In the Thesis we show that from the cooperative control perspective, the use of epipoles to make the team of robots to reach an agreement in their orientations is very interesting. They can be computed from common features of the environment without the necessity of directly observe the other robots, they contain all the necessary information to compute the misalignment in the orientation of neighboring robots and the lack of calibration information is not a major issue. The proposed control law is robust to changes in the topology of the network and does not require to know the calibration of the cameras in order to achieve the desired configuration.

Despite the fact that most part of this research has been done in the Computer Science and Systems Engineering Department of the University of Zaragoza, some contributions of this Thesis have resulted from the stays at the University of California San Diego in the United States and the Royal Institute of Technology in Sweden.

In order to support the results and contributions obtained during this Thesis, most of the work has been submitted for publication to prestigious journals [96] and international peer reviewed conferences [7, 87, 88, 90, 91, 93–95, 97] in the fields of robotics, automatic control and computer vision. In particular, the following results have been obtained from this Thesis:

Journal Publications:

- Eduardo Montijano and Carlos Sagüés. “Distributed multi-camera visual mapping using topological maps of planar regions”. *Pattern Recognition.* Vol 44(7):1528-1539, July 2011
- Eduardo Montijano, Sonia Martínez and Carlos Sagüés. “De-RANSAC: Robust Distributed Consensus in Sensor Networks” *European Journal of Control.* Submitted.
- Eduardo Montijano, Rosario Aragüés and Carlos Sagüés “Distributed Multi-view Matching in Networks with Limited Communications” *IEEE Transactions on Robotics.* Submitted.
- Eduardo Montijano, Juan Ignacio Montijano and Carlos Sagüés “Chebyshev Polynomials in Distributed Consensus Applications” *IEEE Transactions on Signal Processing.* Submitted.

International Reviewed Conferences

- Eduardo Montijano, Johan Thunberg, Xiaoming Hu and Carlos Sagüés "Multi-Robot Distributed Visual Coordination using Epipoles" *50th IEEE Conference on Decision and Control and European Control Conference 2011*, pages 2750-2755, December 2011.
- Eduardo Montijano, Juan Ignacio Montijano and Carlos Sagüés "Adaptive Consensus and Algebraic Connectivity Estimation in Sensor Networks with Chebyshev Polynomials". *50th IEEE Conference on Decision and Control and European Control Conference 2011*, pages 4296-4301, December 2011.
- Eduardo Montijano, Juan Ignacio Montijano and Carlos Sagüés "Fast Distributed Consensus with Chebyshev Polynomials" *American Control Conference 2011*, pages 5450-5455, June 2011.
- Eduardo Montijano, Sonia Martínez and Carlos Sagüés "Distributed Robust Data Fusion Based on Dynamic Voting" *IEEE International Conference on Robotics and Automation 2011*, pages 5893-5898, May 2011.
- Rosario Aragüés, Eduardo Montijano and Carlos Sagüés "Consistent data association in multi-robot systems with limited communications" *Robotics: Science and Systems Conference 2010*, June 2010.
- Eduardo Montijano, Sonia Martínez and Carlos Sagüés "De-RANSAC: Robust Consensus for Robot Formations" *Network Science and Systems Issues in Multi-Robot Autonomy at IEEE International Conference on Robotics and Automation*, May 2010.
- Eduardo Montijano and Carlos Sagüés "Topological maps based on graphs of planar regions" *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1661-1666, October 2009.
- Eduardo Montijano and Carlos Sagüés "Fast Pose Estimation For Visual Navigation Using Homographies" *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2704-2709, October 2009.
- Eduardo Montijano and Carlos Sagüés "Position-Based Navigation Using Multiple Homographies" *IEEE Int. Conference on Emergent Technologies and Factory Automation*, pages 994-1001, September 2008.

1.4 Thesis Outline

The contents of this Thesis are organized as commented in the following:

- In Chapter 2 we introduce the consensus problem for robotic networks and how it can be achieved in a distributed manner using a linear iteration. We explain the main properties of this type of algorithms and discuss the most important works in this research topic. The chapter also serves as background for the rest of the Thesis, as most of the contributions make use of the different concepts explained in this chapter.
- In Chapter 3 we present a distributed solution to the data association problem. In the chapter we give a formal definition to the problem of finding global correspondences. We present a decentralized method for the propagation of matches and two algorithms to solve the possible inconsistencies that may appear. The chapter includes experiments with images considering different features. The contributions of this chapter were partially presented in [7] and are currently submitted [86].
- Chapter 4 deals with the problem of detecting and discarding outlier information during the execution of the linear iteration. The problem of outlier detection is introduced along with the tools to solve it in a distributed way. In the chapter a new algorithm, *De-RANSAC*, which was initially presented in [87] and improved in [88], is detailed. Additionally, in the chapter we propose a new distributed primitive to compute the number of connected robots in the network.

- The speed of convergence of the consensus method is treated in detail in Chapter 5. The chapter includes some background about Chebyshev polynomials and existing algorithms using polynomial methods to reach a distributed consensus. Then, a fast algorithm is proposed and studied in detail for different topologies. This algorithm was presented to the scientific community in [90, 91] and has been submitted for its publication in a journal [92].

- In Chapter 6 we make use of planar regions and homography constraints to build topological maps in a distributed fashion. The chapter explains how the individual maps of each robot are built in a first stage and how they are mixed using distributed consensus techniques. The experiments show how the whole method works and compare the proposed approach with a topological map construction based on images. Some of the contributions of this chapter were firstly published in [94, 95] and more recently in [96].

- The consensus from the control perspective is treated in Chapter 7. The chapter introduces the dynamics of the robots and the assumptions we made in the vision system for this particular problem. After that, a vision-based attitude consensus controller is proposed and discussed in detail. The chapter includes experiments in a simulated environment and experiments with real robots to demonstrate the performance of the controller. The controller was recently presented in [97].

- To finalize, in Chapter 8 we present the conclusions obtained during the development of this Thesis and possible lines of future work that derive from this work.

In order to simplify the reading of the Thesis, the proofs of all the theoretical results have been moved to the end of the chapters in which they appear.

Chapter 2

Robotic Networks and the Consensus Problem

This chapter introduces the necessary concepts to understand the proposals of the Thesis. We provide with the basic definitions to characterize a robotic network. First, we describe the robots we will use and their way to interact with each other and with the world. Then, we define the communications of the network using fixed and time-varying graphs and we give some definitions of interest. After that, in the chapter we review the distributed algorithms based on linear iterations and the application of this kind of algorithms to achieve the consensus using different weights. To conclude the chapter, we show some examples of different consensus applications solved using a linear iteration.

2.1 Introduction

The emergence of new sensing, computation and communication technologies has stimulated an intense research activity in multi-robot sensor systems. The integration of multiple robots in complex networks and information systems will enable users to close the loop in new applications for remote observation and actuation.

One of the first issues that need to be addressed in these systems is the configuration of the individual robots of the network and the way they interact with the world and with each other. It is clear that the actions of the team of robots will depend on the characteristics of each robot. For example, the behavior of the team will not be the same if the robots have omnidirectional range sensors than if they have cameras with limited field of view and no depth information. Similarly, the algorithms run by the network will be different if the robots have powerful communication devices that allow them to communicate with everybody than if they have limited communication capabilities. Therefore, the first step is to describe the main properties of the multi-robot systems we will study along the Thesis.

Once the robotic network is well defined and characterized, the second issue of importance is to give a good description of the problem we want to solve with the team of robots, that is, the consensus problem. Within the control and robotics communities, the problem of achieving consensus is a canonical problem that has received much attention. The goal is to devise a distributed algorithm that allows a group of robots to agree upon some specific information. Distributed linear algorithms and in particular, averaging iterations have a long and rich history, specially during the last decade. Several algorithms have been proposed to achieve consensus under different situations. To name a few, discrete time communications are treated in [61], and continuous time evolution of the system in [107]. Asynchronous communications are studied in [84] and time-varying signals and dynamic consensus in [163]. The convergence time of linear iterations and a finite time solution are proposed in [109, 138] respectively. Solutions to attitude synchronization and consensus on manifolds can be found in [124, 146]. Readers interested in a more comprehensive state of the art in this topic are referred to the books [18, 125], and the surveys [106, 126].

In this chapter we address these two issues, explaining the configuration of the robotic network that we will consider along the Thesis and formally describing the consensus problem and solutions to this problem using linear iterations.

2.2 Model of the Robotic Network

In this section we introduce the main properties of the multi-robot system we are considering.

2.2.1 Definition of a Robot

We start by giving a description of the robots we will consider along the Thesis. We define one robot as an autonomous machine with the following capacities:

- **Capacity to make its own decisions:** We consider that the robots have a computer onboard that allows them to make computations to carry out the desired task. Although we do not impose any restriction on the computation and storage capabilities of the robots, we will pay special attention to design algorithms that do not require complex computations and we will try to use as less storage space as possible.
- **Capacity to sense the world:** In order to perceive the world, we consider that each robot is equipped with a conventional camera. Along the Thesis we consider monocular cameras with limited field of view. In general, we assume that the cameras equipped by all the robots are the same but we do not require an exact knowledge about the intrinsic parameters of the cameras. Moreover, we will see that most of our algorithms are robust to the use of different cameras.
- **Capacity to move autonomously:** We assume that the robots are able to move freely within the environment in order to explore it. Leaving aside Chapter 7, that considers non holonomic motion constraints, in the rest of the thesis we do not impose any particular dynamics on the robots.
- **Capacity to communicate with other robots:** We assume that all the robots are equipped with a communication device that allows them to send and receive messages. As happens with any real communication device, the robots will have limitations to which they can communicate. We do not enter into the details about the specific causes why two robots cannot communicate, they can be, for example, the distance between the robots or interferences in the signal. A more specific definition of the communications is given later in the chapter.

An example of a robot is given in Figure 2.1. Since for our purposes there are not significant differences, we will use interchangeably the terms robot, agent and node.

2.2.2 Communication Model

We assume that the multi-robot system is composed by N robots, each one of them satisfying the properties above mentioned. The robots are labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$.

Assumption 2.2.1 (Identification of the robots)**.** *Each robot in the team $i \in \mathcal{V}$ can be univocally identified. This can be satisfied, for example, considering the IP addresses the robots use to communicate.*

The limited communication capabilities imply that not all the robots will be able to directly exchange information with each other. These limitations can be modeled using a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ contains the pairs of robots that can directly communicate. We say that there is a communication link between i and j when they can directly exchange messages, which happens if and only if $(i, j) \in \mathcal{E}$.

Figure 2.1: Example of a robot

Definition 2.2.2 (Neighbors of a robot)**.** *We define the neighbors of a robot $i \in \mathcal{V}$ as the set of robots that can directly communicate with i,*

$$\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}. \tag{2.1}$$

Unless otherwise specified, in the Thesis we will consider undirected communications. That is, $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$ and $j \in \mathcal{N}_i \Leftrightarrow i \in \mathcal{N}_j$. Initially, let us assume that the graph $\mathcal{G}$ is fixed over the time. When this is the case, we also require the graph to be connected.

Assumption 2.2.3 (Connectedness of the communication graph)**.** *The communication graph, $\mathcal{G}$, is connected, i.e., for every pair of robots i and j, there exists a path of communication links starting in i and ending in j.*

The length of these paths is another important concept in the Thesis.

Definition 2.2.4 (Diameter of a graph)**.** *Given a fixed communication graph, $\mathcal{G}$, we define its diameter as the maximum length of a path, in communication links, between any two robots in the graph. The diameter of $\mathcal{G}$ is denoted by d_v and for a fixed communication topology is always smaller than N.*

Time-Varying Communication Topology

As the robots move in the environment, their relative positions with the other robots change. As a consequence, it can happen that two robots that initially were able to communicate with each other lose their communication link. In this situation the communications between the robots are modeled with with a time-varying undirected graph $\mathcal{G}(t) = \{\mathcal{V}, \mathcal{E}(t)\}$. Now the edge set is time dependent and robots i and j can communicate at time t if and only if $(i, j) \in \mathcal{E}(t)$. The set of neighbors is also time-varying and denoted by $\mathcal{N}_i(t)$.

Under these circumstances, we make two assumptions about the communications between the robots.

Assumption 2.2.5 (Existence of a dwell time)**.** *There exists a lower bound, $\delta > 0$, on the time between two consecutive changes in the topology. Denoting t_k, $k \in \mathbb{N}$, the discrete time instants when the topology changes, then $t_{k+1} - t_k \geq \delta$, $\forall k$.*

Assumption 2.2.6 (Periodic joint connectivity)**.** *There exists a positive time period T such that, for any instant of time, t, the collection of communication topologies in the time interval $(t, t+T)$ is jointly connected. That is, the resulting graph from combining all the communication links that are available in the time interval $(t, t+T)$ is connected.*

Note that these assumptions do not impose any specific communication between two particular robots or the assumption of a central computer that manages all the information. The first assumption implies that the changes in the communication topology do not occur infinitely fast, which is reasonable to assume in a real robotic communication network. The second assumption is used to guarantee that the information of each robot is going to reach at some point, upper bounded in time, all the other robots directly or indirectly.

2.3 Distributed Consensus and Linear Iterations

We formally define the consensus problem and present solutions to solve it using a distributed linear iteration.

Let us consider that the robots are perceiving some quantity of interest with their sensors. The nature of this quantity can be, for example the position of the robot, its velocity or the descriptor of some feature of interest of the environment. For simplicity, to describe the problem we will consider that this quantity is a scalar number. Assume that each robot has some initial value of the quantity of interest, $x_i(0)$. Due to sensor noise or different initial configurations, the observations of each robot are most likely going to be different at the beginning. The consensus problem is formally defined as follows:

Definition 2.3.1 (The consensus problem)**.** *Given initial conditions $x_i(0),\ i = 1, \ldots, N$, we define the consensus problem as the problem of making the state of all the robots regarding the quantity of interest reach the same value, computed as a function, f, of the initial observations: $x_i = x_j = f(x_k(0)),\ k = 1, \ldots, N$, for all i and j in $\mathcal{V}$.*

This problem has a great importance in many robotic tasks such as sensor fusion and formation control. In the first case it is required for a proper representation of the environment whereas in the second is relevant to achieve a desired configuration, e.g., to make all the robots move in the same direction or meet at a fixed point.

The solutions we are interested in correspond with distributed algorithms that follow a linear iteration scheme. Linear iterations are very easy to implement, as they only require to compute linear combinations of different quantities. This simplicity makes them very interesting to be used in a distributed setup. In addition, they represent an important class of iterative algorithms that find applications in optimization, in the solution of systems of equations and in distributed decision making, see for instance [15, 18, 78].

A linear iteration computes weighted sums of the different values to achieve this objective. Specifically, each robot updates its value of the quantity of interest computing a weighted sum of its previous value and that of its direct neighbors,

$$x_i(t+1) = w_{ii}(t)x_i(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)x_j(t). \tag{2.2}$$

In the previous equation w_{ij} is the weight associated to the information given by the neighbor j. In the Thesis we will refer to iteration (2.2) as the standard discrete time distributed consensus algorithm. The extension to quantities of any dimension is straightforward, applying the same iteration rule for each one of the scalar components of the state vector independently.

If we consider the update rule of all the robots simultaneously, we can model the update as a N-dimensional discrete-time linear dynamical system with dynamics inherently related to the network structure,

$$\mathbf{x}(t+1) = \mathbf{W}(t)\mathbf{x}(t), \tag{2.3}$$

with $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))^T$ the values of the state of the different robots in vectorial form and $\mathbf{W}(t) = [w_{ij}(t)] \in \mathbb{R}^{N \times N}$, the weight matrix generated using all the individual weights.

The advantages of using a distributed linear iteration like (2.2) are the following:

- The algorithm is fully distributed because each robot is only using the information provided by its neighbors in the communication graph.

- The robots do not need to know the topology of the whole network to execute these algorithms, they only require the information about their direct neighbors. Moreover, the algorithm is robust to changes in the communication topology.
- The computational requirements of the robots are very low. Each robot is only computing sums and products of the available information. Additionally, the robots only require to store one datum in their computers, instead of as many data as robots in the network. This implies small storage space requirements as well.
- Finally, the communication requirements are also small. The robots only send messages containing their local value of the variable of interest.

All these reasons make the use of these algorithms very appealing in robotic networks.

On the other hand, the use of this kind of iterations assume that the communications between the robots are synchronous and uncorrupted. Nevertheless, some of the most standard issues in communication can be handled by a proper modeling of the communication graph. For example, packet drops and communication failures can be seen as regular changes in the communication topology and asynchronous communications can be modeled by considering a directed communication graph. Therefore, if we design our algorithms in such a way that they can handle these topologies, we can expect them to be robust to these communication issues.

In the following we deal with the problems of assigning the appropriate weights to the different elements so that different consensus objectives are achieved and we show examples of different applications where these algorithms can be used.

2.3.1 Max-Consensus and Average Consensus

In the Thesis we are mainly interested in two important consensus functions, the maximum (or minimum) and the average of the initial conditions.

Max-Consensus

The first consensus function we study is the maximum (or minimum) of the initial conditions. In this situation, the consensus is achieved when all the robots information is equal to the largest initial value,

$$x_i = \max_{j \in \mathcal{V}}(x_j(0)), \forall i \in \mathcal{V}. \tag{2.4}$$

The max-consensus objective has a great importance in leader-election routines. The following distributed iteration can be executed by all the robots to achieve this objective.

$$x_i(t+1) = \max(x_i(t), x_j(t)), \forall j \in \mathcal{N}_i(t). \tag{2.5}$$

Although this is not strictly a weighted combination of the states, it follows the principles of a linear iteration, where each robot assigns $w_{ij} = 1$ to the largest element of those it has access to and $w_{ij} = 0$ to all the rest. The update rule (2.5) is well known to converge in a finite number of rounds to the maximum of all the initial conditions [78].

Distributed Averaging

The second consensus function we study is the average of the initial conditions. This a good consensus function for situations in which the observations of all the robots have the same importance because the average weights all of them equally,

$$\bar{x} = \frac{1}{N} \sum_{i \in \mathcal{V}} x_i(0). \tag{2.6}$$

The average of the initial information of the robots can be achieved very easily using a linear iteration.

Let us consider that the communication graph remains fixed during the execution of the whole iteration, i.e., $\mathbf{W}(t) = \mathbf{W}$ for all t in equation (2.3). Then we impose following assumption on the weight matrix:

Assumption 2.3.2 (Fixed doubly stochastic weights)**.** *$\mathbf{W}$ is symmetric, row stochastic and compatible with the underlying graph, $\mathcal{G}$, i.e., it is such that $w_{ii} > 0,\ w_{ij} = 0$ if $(i,j) \notin \mathcal{E}$, $w_{ij} > 0$ only if $(i,j) \in \mathcal{E}$ and $\mathbf{W1} = \mathbf{1}$.*

Since the communication graph is connected, any matrix $\mathbf{W}$ that fulfills assumption 2.3.2, will have one eigenvalue $\lambda_1 = 1$, with associated right eigenvector $\mathbf{1}$, and algebraic multiplicity equal to one. The rest of the eigenvalues, sorted in decreasing order, satisfy $1 > \lambda_2 \geq \ldots \geq \lambda_N > -1$.

Without loss of generality, let us suppose that all the eigenvalues are simple. Any initial conditions $\mathbf{x}(0)$ can be expressed as a sum of eigenvectors of $\mathbf{W}$,

$$\mathbf{x}(0) = \gamma_1 \mathbf{v}_1 + \ldots + \gamma_N \mathbf{v}_N, \tag{2.7}$$

where $\mathbf{v}_i$ is the right eigenvector associated to the eigenvalue λ_i and γ_i a real coefficient, with the special case of γ_1, which in this case is equal to the average of the initial conditions, $\gamma_1 = \bar{x}$. It is clear that

$$\mathbf{x}(t) = \mathbf{W}^t \mathbf{x}(0) = \gamma_1 \mathbf{v}_1 + \lambda_2^t \gamma_2 \mathbf{v}_2 + \ldots + \lambda_N^t \gamma_N \mathbf{v}_N, \tag{2.8}$$

and since $|\lambda_i| < 1,\ i \neq 1$, then

$$\lim_{t \to \infty} \mathbf{x}(t) = \gamma_1 \mathbf{v}_1 = \bar{x}\mathbf{1}, \tag{2.9}$$

and the average consensus is asymptotically reached by all the robots in the network.

The asymptotic convergence implies that the exact consensus value will not be achieved in a finite number of iterations. In practice, the consensus is said to be achieved when $|x_i(t) - x_j(t) < tol|$ for all i and j, and a prefixed error tolerance *tol*. The convergence speed of (2.3) depends on $\max(|\lambda_2|, |\lambda_N|)$. This parameter is usually denoted as the algebraic connectivity of the network because it contains relevant information about the connectivity of the network. Without loss of generality, in the Thesis we assume that the algebraic connectivity of the network is characterized by λ_2, i.e., $|\lambda_2| \geq |\lambda_N|$.

When the communication topology varies at different iterations, we need to restrict a bit more the assumption on the weights.

Assumption 2.3.3 (Time varying non degenerate weights)**.** *$\mathbf{W}(t)$ has the property of being double stochastic and non degenerate $\forall t$. In other words:*

$$\begin{cases} w_{ij}(t) \in \{0\} \cup [\alpha, 1]\ \forall j, t \\ w_{ii}(t) = 1 - \sum_{j \neq i} w_{ij}(t) \geq \alpha, \\ \mathbf{1}^T \mathbf{W}(t) = \mathbf{1}^T,\ \mathbf{W}(t)\mathbf{1} = \mathbf{1}, \end{cases}$$

being α a positive constant and $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^N$.

Although this case is more tricky, convergence to the average can also be proved. For brevity, we refer the reader to [18] for the proof of convergence in this case.

2.3.2 Definition of the Matrix Weights

The only problem that remains to be solved is the specific selection of the weights that makes the matrix doubly stochastic and non degenerated. Both the convergence to the desired value and the speed which the consensus is achieved depend on these weights, which makes the selection of the weights one of the most important tasks to define a good linear iteration.

There are several to assign the weights in order to satisfy Assumption 2.3.3. A simple way to address this issue is by assigning the same weight, ϵ, to all the elements,

$$w_{ij} = \epsilon, \text{ and } w_{ii} = 1 - |\mathcal{N}_i|\epsilon. \tag{2.10}$$

In this case, the parameter α in Assumption 2.3.3 is equal to ϵ. For a fixed communication topology, the range of values that this parameter can take is directly related with the largest eigenvalue of the Laplacian matrix, $\mathbf{L}$, associated to the communication graph. The Laplacian matrix, $\mathbf{L}$, is defined in such a way that the diagonal elements are equal to the number of neighbors of each robot, $l_{ii} = |\mathcal{N}_i|$ and the rest of the elements, l_{ij}, are equal to 1 if robots i and j are direct neighbors and zero otherwise. The values that ϵ can take are in the interval $(0, 2/\lambda_1(\mathbf{L}))$ [153], being $\lambda_1(\mathbf{L})$ the largest eigenvalue of the Laplacian matrix. Among all these values, the fastest one to achieve the consensus is [153]

$$w_{ij}(t) = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{N-1}(\mathbf{L})}. \tag{2.11}$$

In the Thesis we will denote this weights by the *Best Constant Weights.*

Unfortunately, as we can see, assigning a constant value implicitly requires for the robots to have some global knowledge of the network. An easy way of satisfying Assumption 2.3.3 using only local information are the Metropolis Weights, where the different weights are computed as a function of information provided only by direct neighbors in the communication graph,

$$w_{ij}(t) = \begin{cases} \frac{1}{\max(|\mathcal{N}_i(t)|, |\mathcal{N}_j(t)|)+1}, & \text{if } (i,j) \in \mathcal{E}(t) \\ 1 - \sum\limits_{j \in \mathcal{N}_i(t)} w_{ij}(t), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} . \tag{2.12}$$

In this definition, the parameter α in Assumption 2.3.3 is in the worst case equal to $1/N$ and the rest of conditions are satisfied as well because the same of the weights in one row is always equal to one and the matrix is symmetric. We will also refer to these weights as the *Local Degree Weights* [153]. Note that in this case, in order to compute the weights, the robots only need to attach to their messages the information about the number of neighbors they have.

2.4 Application of Linear Iterations to Different Consensus Problems

To end this chapter, we present different robotic problems where the average consensus can be used as a distributed solution.

Sensor Fusion

One of the most interesting problems where the distributed averaging can be used is the sensor fusion of the measurements of all the robots. The problem of how to fuse several observations in a centralized manner has received considerable attention in the robotics literature. For example, [44], [53] study the integration of different sensor measurements taken by a single robot. In multi-robot systems, [55] investigates how to perform data fusion of several SLAM maps by a central unit.

Distributed averaging techniques allow to solve this problem in a distributed fashion in a simple way. For example, let us consider that the robots are measuring the temperature of the environment, which has a value around five Celsius degrees. Let us say that there are 25 robots deployed in the environment taking measurements. Each robot is equipped with a thermometer that gives an estimation of the temperature between 4.5 and 5.5 degrees. The communication network is depicted in the left image of Figure 2.2 inside a room. The right image of Figure 2.2 shows the evolution of all the robots' estates when executing (2.2). We can see that all the robots eventually achieve the same temperature value, i.e., the consensus, which is exactly the average of the initial values of all the robots.

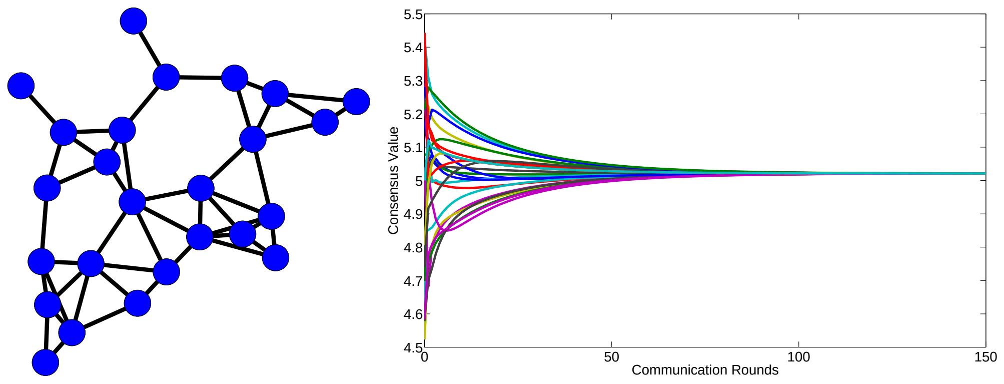


Figure 2.2: Example of distributed consensus for sensor fusion. The communication network is depicted on the left, where each blue dot represents one robot of the network and each black line represents a direct communication between a pair of robots. On the right we show the evolution of the consensus, with each line representing the different values of the state of each robot. We can see that as the robots execute the linear iteration, the value of all the robots tends to the same value, which is exactly the average of the initial conditions.

Sensor Fusion with Uncertainties

If the observations of the robots include a measure of their uncertainty, given by the covariance matrices $\boldsymbol{\Lambda}_i$, the problem changes from the computation of the average to the computation of the maximum likelihood (ML). The ML is estimated using a weighted least-squares approximation from the robot measurements as

$$\boldsymbol{\theta}_{\mathrm{ML}} = \left(\sum_{i=1}^{N} \boldsymbol{\Lambda}_i^{-1}\right)^{-1} \sum_{i=1}^{N} \boldsymbol{\Lambda}_i^{-1} \mathbf{x}_i(0). \tag{2.13}$$

An averaging linear iteration can be used to compute the ML [155]. At the beginning, each robot initializes two state variables, $\mathbf{P}_i(0) = \boldsymbol{\Lambda}_i^{-1}$ and $\mathbf{q}_i(0) = \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i(0)$. If the averaging linear iteration is applied to these variables, then

$$\lim_{t\to\infty} \mathbf{P}_i(t) = \frac{1}{N}\sum_{j\in\mathcal{V}} \mathbf{P}_j(0), \text{ and } \lim_{t\to\infty} \mathbf{q}_i(t) = \frac{1}{N}\sum_{j\in\mathcal{V}} \mathbf{q}_j(0), \tag{2.14}$$

and

$$\lim_{t\to\infty} \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t) = \boldsymbol{\theta}_{\mathrm{ML}}, \tag{2.15}$$

This distributed solution can be very helpful, for example, in the cooperative simultaneous localization and mapping of the environment by a team of robots, the SLAM problem [5].

Rendezvous of a team of robots

Another problem of high interest in multi-robot systems is the rendezvous of the team [32,37]. The rendezvous of a team of robots is achieved when all the robots gather at a fixed point in the environment. Solutions based on distributed averaging can also be of high interest for this problem. Figure 2.3 shows the trajectories of a team of 10 robots with linear dynamics and using averaging to control their motion. The individual control law of each robot is computed by averaging the positions of the robots that are neighbors in the communication graph. At the end, all the robots come together at the same point, which is the centroid of the initial positions of the ten robots.

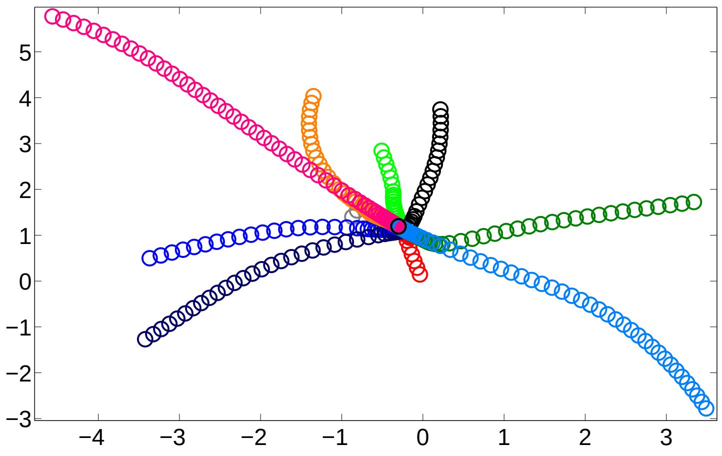


Figure 2.3: Example of a team of robots using a linear iteration to rendezvous in the centroid of their initial positions.

Flocking of a team of robots

Finally, a very important problem in cooperative control and formation control is the flocking of the team of robots [105]. This problem consists on making all the robots to move in the same direction and with the same velocity. The name comes from the analogy with the behavior of flocks of birds moving in formation (Fig. 1.5). As in the previous motivating examples, distributed averaging and linear iterations represent a powerful tool to achieve this behavior.

A simple example of this phenomenon is shown in Figure 2.4. This example assumes a team of robots moving on the plane with non holonomic motion constraints. At each step, the robots turn in the average direction of the headings of their neighbors. As a consequence, all the robots end up moving in the same direction. If the initial velocities of the robots were different, the same solution could be used to make them converge to the same value.

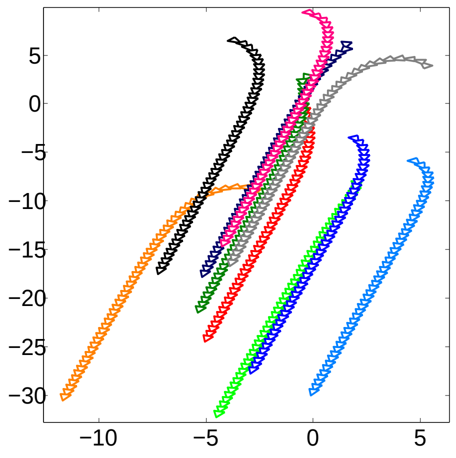


Figure 2.4: Example of a distributed control law that makes the team of robots move in the same direction by averaging their orientations.

2.5 Discussion

This chapter has introduced the robotic network that will be the focus of study along the Thesis. We have defined the main properties of the individual robots and their sensing and communication capabilities to interact with the environment. Graph theory presents itself as a very useful tool to formalize the interactions between the robots. Modeling the network with graphs we can describe the important concepts.

The chapter has also served to present the main problem we are interested in studying, the consensus problem. We have reviewed the solutions of this problem based on linear iterations and their interest in distributed systems with limited communication capabilities. Finally, we have introduced some of the most interesting multi-robot problems that can be solved using linear iterations and distributed averaging. We have presented some basic simulations to show how these problems find a simple solution with distributed averaging.

In the examples we have presented the most basic version of these problems, omitting important aspects of real life that should be taken into account. For example, in the sensor fusion scenario we have assumed that the robots were only measuring a good value of a single data, the temperature. However, in perception tasks with vision sensors each robot will observe many features, and some of them erroneously. Therefore, additional mechanisms are required to achieve a proper consensus. In the flocking scenario we have not considered visibility issues, what happens if the robots do not see each other or do not know how to measure their relative positions. In the rest of the Thesis we present our contributions in this field, considering more realistic versions of some of these problems.

Chapter 3

The Data Association Problem

"It is said that one image is worth a thousand words." In order to reach a consensus, the first task that is required for the team of robots is to globally identify these "words". In this chapter we address the problem of finding global correspondences between the observations of all the robots in a distributed manner. At the beginning, each robot finds correspondences only with the robots that can directly communicate with it. This is done using existing matching techniques for pairs of images. After that, we propose a distributed algorithm that propagates the local correspondences through the network. We formally demonstrate the main properties of the algorithm and prove that after executing our method, the team of robots finishes with a globally consistent data association. The performance of the algorithm is tested with extensive simulations and real images at the end of the chapter.

3.1 Introduction

An important issue involving a team of robots in perception tasks consists of establishing correspondences between the features perceived by all the robots. The standard consensus linear iteration, eq. (2.2), does not consider this problem, because it assumes the existence of a single feature (of arbitrary dimension) that all the robots observe and share. When the perception system is composed by cameras, the number of features that each robot sees can be considerably big. An example of this is given in Figure 3.1, where there are many faces in the same picture. Before a consensus on the people's position is possible [145], the robots require to know which faces detected by the different robots correspond to the same person. Therefore, the first step in the consensus process is the identification of common features observed by different robots so that they can combine them.

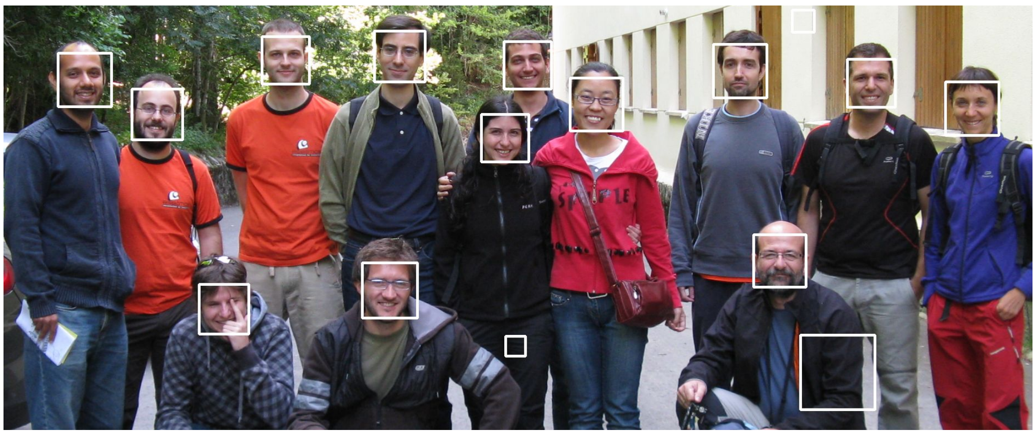


Figure 3.1: Multiple features observed in the same image. Before the consensus can be achieved, the robots require to identify which of the observed features have also been seen by other members of the team.

We assume that the pairs of robots with a communication link can establish local correspondences of the features in their images using existing matching methods, e.g., [29, 57]. However, distant robots may have observed common features as well. Therefore, additional mechanisms are required to obtain higher level information than pair to pair matches. This information can be obtained by the proper propagation of the local matches through the network. Additionally, due to the presence of spurious local matches, there may appear inconsistent global correspondences, which are detected when chains of local matches create a path between two features from the same robot (see Fig. 3.2). These situations must also be correctly identified and solved, otherwise, the consensus achieved in the following steps will be wrong.

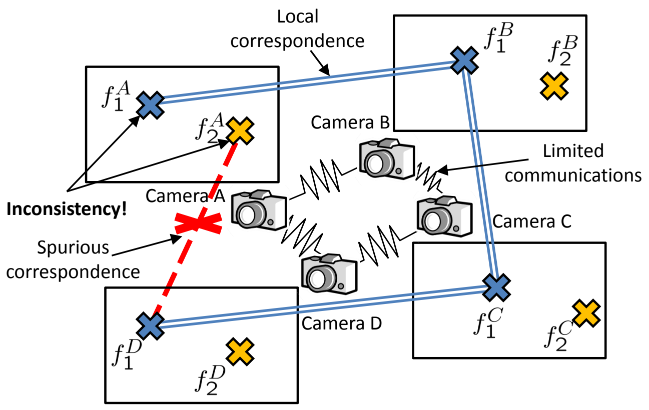


Figure 3.2: Example of one inconsistency. The network is composed by four robots A, B, C and D. The chain of local matches between direct neighbors leads to a situation in which two features observed by the robot A are associated, which is not possible. In this chapter we propose a method to distributively detect and solve these situations.

The problem of finding correspondences between two images has been deeply studied from different perspectives, depending on the features used. Point or line features and geometric constraints are used [57, 103]. Three dimensional points with uncertainties are matched using the Nearest Neighbor, and the Maximum Likelihood, in terms of the Euclidean or the Mahalanobis distance [55, 65]. Another popular method is the Joint Compatibility Branch and Bound [102], which considers the compatibility of many associations simultaneously. And there are many combinations of all the previous techniques with RANSAC [43] or any of its variations [26] for higher robustness. There also exists a vast literature dealing with the problem of matching image templates [112] representing, e.g., faces [38], people [52] or objects [29].

Centralized solutions are common in order to manage the correspondence problem of multiple views. Triplets of matches are considered in [157] and they are characterized as consistent when the three matches are reliable. Correspondences between features observed in two different sequences are found in [22] where spatial and temporal parameters define the matching between the sequences. Multiple views and rigid constraints are considered in [50, 108, 160]. The work in [36], from the target tracking literature, simultaneously considers the association of all local sets of features. A multi-view matching is presented in [41] where every pair of views are compared among them. Then, their results are arranged in a graph where associations are propagated and conflicts are solved. The k-dimensional matching problem tries to find the maximal matching in a k-partite k-uniform balanced hyper-graph [58]. Since the optimal solution for this problem for three or more dimensions (views) is an NP-hard problem [114], a greedy algorithm is used in [133] to find the correspondences along a sequence. The algorithm is also used in [134] to associate trajectories observed by several moving cameras.

There are also different works that propose distributed solutions to this problem. An early approach of distributed matching using range views can be found in [14]. With vision cameras, the problem of surveillance has given different distributed solutions [122, 152]. In this case the cameras are fixed and usually they have been previously calibrated, so there is some knowledge about the relations between the cameras. In [8] a considerably large set of images is considered. Initially a subset of the 2-by-2 associations is performed guarantying connectedness of the correspondences using the theory of random graphs. The rest of the matches are found by propagation of the initial correspondences. Finally, local conflicts are solved by performing additional 2-by-2 associations. However, the resolution method provided requires any image to be able to be associated with any other, which is not always possible in networks with limited communications.

In this chapter, we address the problem of discovering the global correspondences between the set of images in a distributed way, solving also the possible inconsistencies that may appear. We contribute to existing state of the art algorithms by proposing a fully distributed solution to these problems. Given the local matches established between neighbors, our propagation algorithm allows each robot to find correspondences with all the other robots in the network, even if they cannot directly communicate. We show how the inconsistencies are detected and propose two different resolution algorithms to break them. The first one considers the quality of the local matches, using a variant of a max-consensus algorithm to find the link with the largest error that breaks the inconsistency. The second one builds different independent spanning trees that ensure that the alternative data association is free of inconsistencies.

More in detail, the contributions of this chapter are:

- A distributed algorithm to propagate the local matches, providing all the robots with the global correspondences with the other robots, even if they cannot directly communicate;
- A mechanism to detect inconsistent associations and two distributed algorithms to resolve these inconsistencies through the deletion of local matches, in function of their quality;
- A rigorous study of the properties of the whole procedure which proves that is fully distributed, requires low communication and finishes in finite time. In addition, the method makes mild assumptions on the local matching functions, and thus can be combined with a wide variety of features and local matchers.

This chapter has been partially published in [7, 86].

3.2 Problem Description

3.2.1 Matching between two robots

As previously stated, our method consists of correctly propagating the local matches of neighboring robots through the network in a distributed fashion. For a better understanding of the algorithms presented in the chapter, we introduce the notation and describe the properties the local matcher must satisfy. The r^{th} feature observed by the i^{th} robot is denoted as f^i_r. Given a matrix $\mathbf{A}$, the notation $\mathbf{A}_{ij}$ denotes the block (i, j) of the matrix whereas $[\mathbf{A}]_{r,s}$ corresponds to the component (r, s) of the matrix. Other parts of the chapter require the use of rows, in this cases $\mathbf{a}^i_r$ represents a row with the information corresponding to feature f^i_r, and $[\mathbf{a}^i_r]_s$ the s^{th} component of the row.

Consider two robots i and j, that observe two sets $\mathcal{S}_i$ and $\mathcal{S}_j$ of m_i and m_j features respectively,

$$\mathcal{S}_i = \{f^i_1, \ldots, f^i_{m_i}\}, \quad \mathcal{S}_j = \{f^j_1, \ldots, f^j_{m_j}\}. \tag{3.1}$$

We let F be the local matching function, such that for any two sets of features, $\mathcal{S}_i$ and $\mathcal{S}_j$, $F(\mathcal{S}_i, \mathcal{S}_j)$ returns an association matrix $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$ where

$$[\mathbf{A}_{ij}]_{r,s} = \begin{cases} 1 & \text{if } f^i_r \text{ and } f^j_s \text{ are associated,} \\ 0 & \text{otherwise,} \end{cases}$$

for $r = 1, \ldots, m_i$ and $s = 1, \ldots, m_j$. The function F must satisfy the following conditions.

Assumption 3.2.1 (Self Association). *When F is applied to the same set $\mathcal{S}_i$, it returns the identity, $F(\mathcal{S}_i, \mathcal{S}_i) = \boldsymbol{I}$.*

Assumption 3.2.2 (Unique Association). *The association $\boldsymbol{A}_{ij}$ has the property that the features are matched in a one-to-one way,*

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

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

Assumption 3.2.3 (Symmetric Association). *For any 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 about the quality of each association. The management of this information about quality is discussed later in the chapter, in section 3.4.

We do not make any assumptions about the sets of features. However, we point out that the better the initial matching is, the better the results of 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, [57], [103],
- 3D points computed using mapping techniques matched with the Joint Compatibility Branch and Bound (JCBB) [102],
- Image templates of people, faces, objects, etc. matched with sums of absolute differences of the pixels or correlation methods [112].
- ...

3.2.2 Centralized matching between N robots

Let us consider now the situation in which there are N robots 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 matching 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 if and only if $[\mathbf{A}_{ij}]_{r,s} = 1$.

For a perfect matching function, the matching graph, $\mathcal{G}_{cen}$, exclusively contains disjoint cliques, identifying features observed by multiple robots (Fig. 3.3 (a)). However, in real situations, the matching function will miss some matches and will consider as good correspondences some spurious matches (Fig. 3.3 (b)). As a consequence, inconsistent associations relating different features from the same set $\mathcal{S}_i$ may appear.

Definition 3.2.4 (Association Sets and Inconsistencies). *Given a matching graph, $\mathcal{G}_{cen}$, 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 robot. A feature is* inconsistent *or* conflictive *if it belongs to an inconsistent association.*

Centralized solutions to overcome this problem are found in [41], [8]. The latter one is also well suited for a distributed implementation but yet requires that any pair of images can be matched. In robotic networks this implies global communications, which are not always possible.

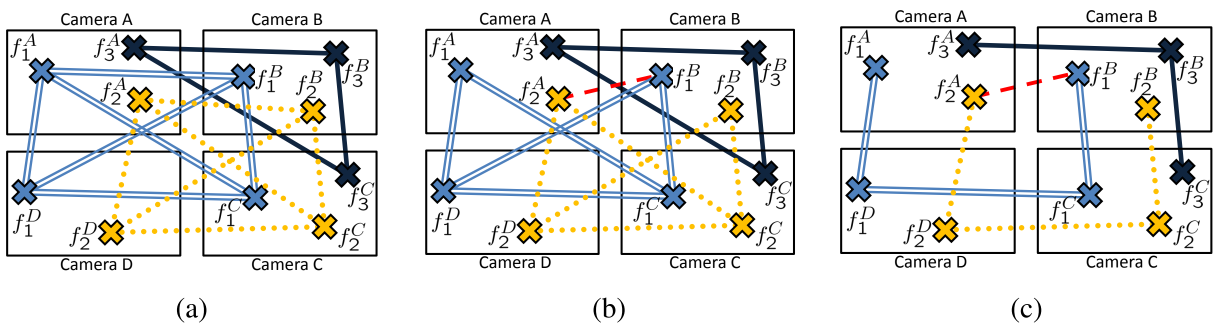


Figure 3.3: Different association graphs. (a) Centralized matching with perfect association function. The graph is formed by disjoint cliques. (b) Centralized matching with imperfect association. Some edges are missed, (f_1^A, f_1^B) and (f_2^A, f_2^B), and spurious matches appear, (f_2^A, f_1^B). As a consequence, a subset of the features form a *conflictive set*. (c) Matching with limited communications. Now, the matches between A and C, and B and D, cannot be computed because they are not neighbors in $\mathcal{G}$. Moreover, the information available to each robot is just the one provided by its neighbors.

3.2.3 Matching between N robots with limited communications

Let us consider now that there is no central unit with all the information and there are N robots, with limited communication capabilities following the model described in the previous chapter, section 2.2.2. In this case, local matches can only be found within direct neighbors in the communication graph. 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. 3.3 (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,

$$\mathcal{E}_{dis} = \{(f_r^i, f_s^j) \mid (f_r^i, f_s^j) \in \mathcal{E}_{cen} \wedge (i,j) \in \mathcal{E}\}. \tag{3.2}$$

In the thesis, 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 $\mathcal{G}_{dis}$, This diameter satisfies $d_f \leq m_{sum}$. We name $\mathbf{A} \in \mathbb{N}^{m_{sum} \times m_{sum}}$ the adjacency matrix of $\mathcal{G}_{dis}$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{N1} & \dots & \mathbf{A}_{NN} \end{bmatrix}, \tag{3.3}$$

where

$$\mathbf{A}_{ij} = \begin{cases} F(\mathcal{S}_i, \mathcal{S}_j) & \text{if } j \in \{\mathcal{N}_i \cup i\}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3.4}$$

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 first problem studied in the thesis is formulated as follows:

Definition 3.2.5 (**The data association problem**). *Given a network with limited communications 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.*

3.3 Propagation of Matches and Detection of Inconsistencies

Considering Definition 3.2.4 we observe that in order to match features of robots that are not neighbors and to detect any inconsistency between features, the paths that exist among the elements in $\mathcal{G}_{dis}$ should be computed.

As the following lemma states [18], given a graph $\mathcal{G}$, the powers of its adjacency matrix contains the information about the number of paths existing between the nodes of $\mathcal{G}$:

Lemma 3.3.1 (Lemma 1.32 [18])**.** *Let $\mathcal{G}$ be a weighted graph of order $|\mathcal{V}|$ with un-weighted adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, and possibly with self loops. For all $r, s \in \{1,\ldots,|\mathcal{V}|\}$ and $t \in \mathbb{N}$ the (r, s) entry of the t^{th} power of $\boldsymbol{A}$, $[\boldsymbol{A}^t]_{r,s}$, equals the number of paths of length t (including paths with self loops) from node r to node s.*

The powers of the adjacency matrix in (3.3) can be computed in a distributed way. Let each robot $i \in \mathcal{V}$ have a set of variables $\mathbf{X}_{ij}(t) \in \mathbb{N}^{m_i \times m_j}$, $j = 1,\ldots,N$, $t \geq 0$, initialized as

$$\mathbf{X}_{ij}(0) = \begin{cases} \mathbf{I}, & j = i, \\ \mathbf{0}, & j \neq i, \end{cases} \tag{3.5}$$

and updated, at each time step, with the following rule

$$\mathbf{X}_{ij}(t) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik}\mathbf{X}_{kj}(t-1), \tag{3.6}$$

with $\mathbf{A}_{ik}$ as defined in (3.4). It is observed that the algorithm is fully distributed because the robots only use information about its direct neighbors in the communication graph. The equivalency with the powers of $\mathbf{A}$ is true because $\mathbf{A}_{ik} = \mathbf{0}$ for $k \notin \{\mathcal{N}_i \cup i\}$. Therefore eq. (3.6) is equivalent to $\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t)$ executed by blocks and, since $\mathbf{X}(0) = \mathbf{I}$, then $\mathbf{X}(t) = \mathbf{A}^t$. The advantages are that the robots only manage the rows corresponding to their features, the computation is done distributively and computation finishes in at most d_f iterations.

The robots are able to detect all the features associated with its own set from their own blocks $\mathbf{X}_{ij}(t)$. If there is a conflictive feature, they can also detect it and know the rest of features that belong to the conflictive set, independently of who observed such features. Inconsistency detection is done using two rules. A feature f_r^i is conflictive if and only if one of the following conditions is satisfied:

- There exists another feature $f_{r'}^i$, with $r \neq r'$, such that

$$[\mathbf{X}_{ii}(t)]_{r,r'} > 0; \tag{3.7}$$

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

$$[\mathbf{X}_{ij}(t)]_{r,s} > 0 \text{ and } [\mathbf{X}_{ij}(t)]_{r,s'} > 0. \tag{3.8}$$

However, this way to detect the inconsistencies has several drawbacks. The powers of $\mathbf{A}$ may contain large values, but 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. For that reason, we propose a propagation algorithm that overcomes these limitations reducing the complexity of the operations, the number of execution steps and the amount of transmitted information.

Algorithm 1 shows the proposed method. 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 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 the 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.

Algorithm 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 $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t), \forall r \in \mathcal{S}_i$
```

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 can simply avoid the execution of line 6 because it already has all the information from $\mathbf{y}_r^i(t-1)$, which is exactly equal to $\mathbf{y}_r^i(t)$.

The modified algorithm reduces the complexity of the operations with respect to the powers of the adjacency matrix by replacing the products of matrices by logical operations between rows. This reduction allows to avoid the large numbers that may appear when computing powers of the adjacency matrix and also allows us to reduce the amount of transmitted data. In the following we describe the main properties of the algorithm.

Proposition 3.3.2 (Limited Communications). *The amount of information exchanged by the network during the whole execution of Algorithm 1 can be upper bounded by* $2m_{sum}^2$.

Proposition 3.3.3 (Correctness). *After execution of Algorithm 1 all the paths between features have been found and they are available to all the robots with features involved in them.*

Theorem 3.3.4 (Limited Iterations). *All the robots end the execution of the Algorithm 1 in at most* $\min(d_f, 2N)$ *iterations.*

Before illustrating the behavior of the algorithm with an example, we discuss some aspects of the decentralized inconsistency detection algorithm.

Remark 3.3.5 (Conservative Bounds). *The bounds provided in Proposition 3.3.2 and Theorem 3.3.4 are conservative. In practice we should expect a better performance of the algorithm. In order to send* $2m_{sum}^2$ *data, it is required for the association graph to be strongly connected, i.e., for any pair of features there is a path of arbitrary length connecting them. This situation is unlikely to happen, since it would mean that all the features are associated with each other. Regarding the number of iterations, the bound does not take into account the simultaneous exchange of data by the robots. In practice we have observed that in less than* N *iterations the algorithm always finishes.*

Remark 3.3.6 (Higher Level Matches). *With the presence of additional images, it is possible to find better matches using higher level constraints, e.g., the trifocal tensor with triplets of images. Our algorithms are compatible with other multi-camera matches, as long as the communications allow the computation of these matches and the Assumptions 3.2.1 to 3.2.3 are satisfied between pairs of cameras. However, in order to keep the communication graph free of constraints, we have only considered matching functions using pairs of images.*

Remark 3.3.7 (Multiple Matches). *In the literature, when dealing with two images, it is common to consider multiple hypotheses when associating the features [102]. Although this possibility would be of high interest* *here, this is not affordable by now. By considering multiple associations, the problem grows exponentially with the number of cameras and features, whereas with a one-to-one association grows linearly. Moreover, the match of one feature with two or more features observed by other camera is directly an inconsistency, which requires further processing to solve it in a second step.*

Remark 3.3.8 (**Detectable Spurious**)**.** *The existence of one inconsistency implies the existence of at least one spurious match in the local matching process. However, the opposite is not necessarily true. There might be local matches which are spurious but that do not lead to one inconsistency, e.g., one match between only two robots. These spurious matches cannot be detected with our algorithm, and therefore cannot be corrected during the data association. A more detailed treatment of spurious and outliers is given in the next chapter.*

Example of execution

Figure 3.4 shows an example of how the algorithm is applied. The example shows the execution of the algorithm for the associations shown in Fig. 3.3 (c). Each robot has only the information about the rows corresponding to the features it has observed. In Fig. 3.4 (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 1 the rows have the form of Fig. 3.4 (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, f_2^A, 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. 3.4 (c). Now the process is repeated, obtaining the matrix in Fig. 3.4 (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.

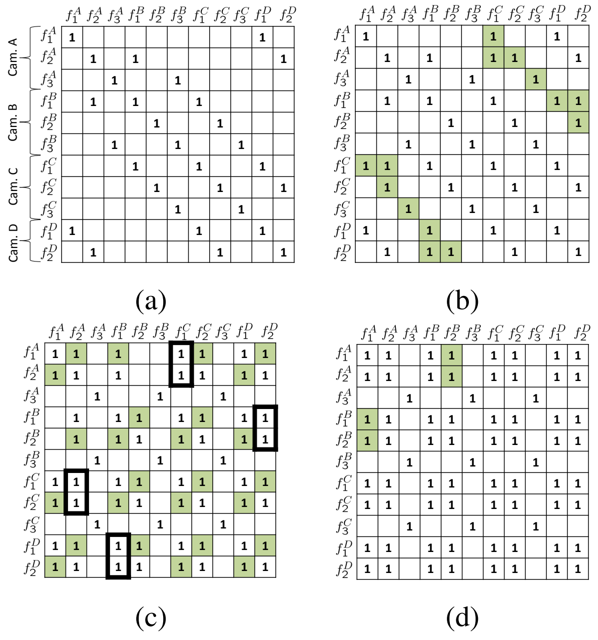


Figure 3.4: Example of execution of the propagation algorithm and the detection of inconsistencies. Figures (a)-(d) show the four steps that the algorithm requires to propagate all the correspondences.

3.4 Decentralized Resolution of Inconsistent Associations

The existence of one inconsistency implies the existence of at least one spurious match in the local matching process. Therefore, the resolution of inconsistent associations is carried out by deleting edges from $\mathcal{G}_{dis}$, ideally the spurious matches, so that the resulting graph is conflict-free.

Definition 3.4.1 (Conflict Free Graph). *Let C denote the number of conflictive sets in $\mathcal{G}_{dis}$. The robots that detect a conflictive set $\mathcal{C}$ is $\mathcal{V}_{inc} \subseteq \mathcal{V}$. The number of features from each robot $i \in \mathcal{V}_{inc}$ involved in $\mathcal{C}$ is $\tilde{m}_i$ and the number of total features involved in $\mathcal{C}$ is denoted as c. 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

$$\bigcup_q \mathcal{C}_q = \mathcal{C}, \text{ and } \mathcal{C}_q \cap \mathcal{C}_{q'} = \emptyset, q \neq q'.$$

Note that, even with the full knowledge of the association graph that generates $\mathcal{C}$, finding the optimal partition that solves the inconsistency is an NP-Hard problem. If there were only two inconsistent features f_r^i, $f_{r'}^i$, it could be approached as a max-flow min-cut problem [115]. However, in general there will be more inconsistent features, $\tilde{m}_i \geq 2$, within C associated to each robot. The application of [115] 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. For that reason we focus on proposing heuristic methods such that the communication constraints are respected. The number of conflict-free components is a priori unknown but can be lower bounded with the following proposition:

Proposition 3.4.2 (Number of Partitions). *Let $\mathcal{V}_{inc}$ be the set of robots that detect $\mathcal{C}$ and $i_\star$ be the robot with the most features in $\mathcal{C}$,*

$$i_\star = \arg\max_{i \in \mathcal{V}_{inc}} \tilde{m}_i. \tag{3.9}$$

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

In the rest of the section we provide two different distributed algorithms to solve the inconsistencies in $\mathcal{G}_{dis}$. The first one, the *Maximum Error Cut*, considers the weights in the association graph in order to find the edge with the largest error that breaks a given inconsistency. The second method is based on a greedy deletion of edges to construct different *Spanning Trees* free of inconsistencies.

3.4.1 Resolution using the *Maximum Error Cut*

Most of the matching functions in the literature are based on errors between the matched features, e.g., the Sampson distance for the epipolar constraint, or the sum of the absolute differences for template matching. These errors can be used to find a partition of $\mathcal{C}$. Let $\mathbf{E}$ be the weighted symmetric association matrix

$$[\mathbf{E}]_{r,s} = \begin{cases} e_{rs} & \text{if } [\mathbf{A}]_{r,s} = 1, \\ -1 & \text{otherwise}, \end{cases} \tag{3.10}$$

with e_{rs} the error of the match between r and s.

Assumption 3.4.3 (Properties of the Errors). *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 3.4.4 (Bridges and Cuts). *Given a conflictive set, a* bridge *is an edge whose deletion divides the set in two connected components, i.e., it does not belong to a cycle. Given two conflictive features, we define a* cut *as a bridge that, if it is deleted, then the conflict between the features is solved.*

Note that not all the bridges in one inconsistency are cuts. There are bridges that, if deleted, will not break the inconsistency because they do not belong to the path between the features to separate. Our goal is, for each pair of conflictive features, find and delete the cut with the maximum error.

Algorithm 2 shows the solution we propose to find the cuts using local interactions. We explain in detail how it works. As we did in the detection algorithm, let each robot initialize its own $\tilde{m}_i$ rows of elements as

Algorithm 2 *Maximum Error Cut* - Robot i

```
Require: Set of $C$ different conflictive sets
Ensure: $\mathcal{G}_{dis}$ is conflict free
 1: for all $\mathcal{C}$ do
 2:     – Error transmission
 3:     $\mathbf{z}_r(0) = \{[\mathbf{E}_{\mathcal{C}}]_{r,1}, \ldots, [\mathbf{E}_{\mathcal{C}}]_{r,c}\}$, $r = 1, \ldots, \tilde{m}_i$
 4:     repeat
 5:         $\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C},\ [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0}(\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs})$
 6:     until $\mathbf{z}_r(t+1) = \mathbf{z}_r(t), \forall r \in \tilde{m}_i$
 7:     – Edge Deletion
 8:     while robot $i$ has conflictive features $r$ and $r'$ do
 9:         Find the cuts $(s, s')$ :
10:             (a) $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}, s \neq s'$,
11:             (b) For all $s'' \neq s$, $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_{s''}$,
12:             (c) For all $s'' \neq s'$, $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_{s''}$
13:         Select the cut with largest error
14:         Send message to break it
15:     end while
16: end for
```

$\mathbf{z}_r(0) = \{[\mathbf{E}_{\mathcal{C}}]_{r,1}, \ldots, [\mathbf{E}_{\mathcal{C}}]_{r,c}\}$. The update rule executed by every robot and every feature is

$$\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C},\ [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0} (\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs}), \tag{3.11}$$

where the maximum is done element-wise 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. (3.10). Equivalently, for a given feature r, we can put eq. (3.11) as a function of its elements. The u^{th} component, $[\mathbf{z}_r(t+1)]_u$, is updated as follows:

$$[\mathbf{z}_r(t+1)]_u = \begin{cases} \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_s) \text{ if } [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0 \wedge u = r \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_r) \text{ if } [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0 \wedge u = s \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_u) \text{ if } [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0 \wedge r \neq u \neq s. \end{cases} \tag{3.12}$$

As the following results state, the presented method converges in finite time. We also show the convergence values of the different elements. For clarity, we separate the analysis in two parts: the first result gives the values reached by the components that belong to bridges in the graph; the second result consider the features that form part of a cycle in the association graph.

Proposition 3.4.5 (Convergence). *The dynamic system defined in* (3.11) *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 $\boldsymbol{z}_s\boldsymbol{P}_{rs}$. In addition, for any $r \in \mathcal{C}$, $[\boldsymbol{z}_r(t)]_r = 0, \forall t \geq 0$.*

Theorem 3.4.6 (Values for Bridges). *Let us consider one bridge, (s, u). Let $d(r, s)$ be the minimal distance in edges to reach node s starting from node r, then for all r such that $d(r, s) < d(r, u)$*

$$[\boldsymbol{z}_r(t)]_u \to [\boldsymbol{E}_\mathcal{C}]_{s,u} = e_{su}. \tag{3.13}$$

Equivalently, for all r such that $d(r, s) > d(r, u)$

$$[\boldsymbol{z}_r(t)]_s \to [\boldsymbol{E}_\mathcal{C}]_{u,s} = e_{us} = e_{su}. \tag{3.14}$$

Theorem 3.4.7 (Values for Cycles). *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. For a given feature r*

$$[\boldsymbol{z}_r(t)]_s \to \max_{u,u' \in \mathcal{C}_\ell} e_{uu'}, \quad \forall s \in \mathcal{C}_\ell \setminus \arg\min_{s' \in \mathcal{C}_\ell} d(r, s'). \tag{3.15}$$

Corollary 3.4.8. *If there is a cycle $\mathcal{C}_\ell$ in the association graph, after the execution of Algorithm 2, for every feature r there exist at least two features s, s' in $\mathcal{C}_\ell$ as in* (3.15) *for which the elements $[\boldsymbol{z}_r]_s, [\boldsymbol{z}_r]_{s'}$ reach the same value.*

At this point we are ready to define the cuts in terms of the variables $\mathbf{z}_r$ and to propose a criterion to select the best cut to delete. The cuts, (s, s'), for any pair of conflictive features r and r' satisfy

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

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

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

The first condition comes from Theorem 3.4.6 and the other two come from Theorem 3.4.7. For any cut, the error of the cut is the same as the value of $[\mathbf{z}_r]_s$, $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'} = e_{ss'}$. Therefore, each robot can look in a local way at its own rows and choose the best cut that breaks the conflict, the one with the largest error. Let us note that, by the definition of a cut, the algorithm considers single-edge deletions, attempting to minimize the number of broken links. Also with this approach, cycles in the association graph are not broken, which is another good property. These cycles are sets of features strongly associated, and therefore, it is better not to delete edges there.

In case one robot has more than two features in the same conflict, the algorithm chooses two of the $\tilde{m}_i$ inconsistent features and selects the best cut for them. The cut 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. This is a simple way to proceed that satisfies the resolution of complex inconsistencies, with the payoff of the probable deletion of more edges than necessary. It is possible that two robots decide to delete two different links, when breaking only one of them would be enough to break both inconsistencies, as happens in the example in Fig. 3.5 (b). Nevertheless, recall that finding the optimal partition is NP-Hard and requires the full knowledge of the graph, which implies expensive communications and computations, whereas this approach is fully distributed and easy to compute.

The main limitation of the algorithm appears when two conflictive features belong to one cycle in the association graph. In this case there are no cuts and the proposed algorithm cannot solve the inconsistency. However, since the algorithm is able to detect this situation, a different approach can be used to solve it.

3.4.2 Resolution based on *Spanning Trees*

We propose an alternative algorithm to deal with the situations that the *Maximum Error Cut* does not solve. The method is based on the computation of different spanning trees in each conflictive set and, although the cuts done in the association graph are arbitrary, it has the property that it is able to solve all the inconsistencies. The algorithm constructs $\tilde{m}_{i_\star}$ spanning trees free of inconsistencies. Initially, each robot i detects the conflictive sets for which it is the root using its local information $\mathbf{y}_r^i, r = 1, \ldots, m_i$. The root robot for a conflictive set is the one with the most inconsistent features involved, eq. (3.9). In case two robots have the same number of inconsistent features, the one with the lowest identifier is selected.

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 $[\mathbf{A}_{ji}]_{s,r} = 1$, 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. The process is summarized in Algorithm 3.

Theorem 3.4.9 (Properties of the *Spanning Trees* resolution). *Let us consider that each robot executes Algorithm 3 on one conflict $\mathcal{C}$,*

(i) after 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;

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

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

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

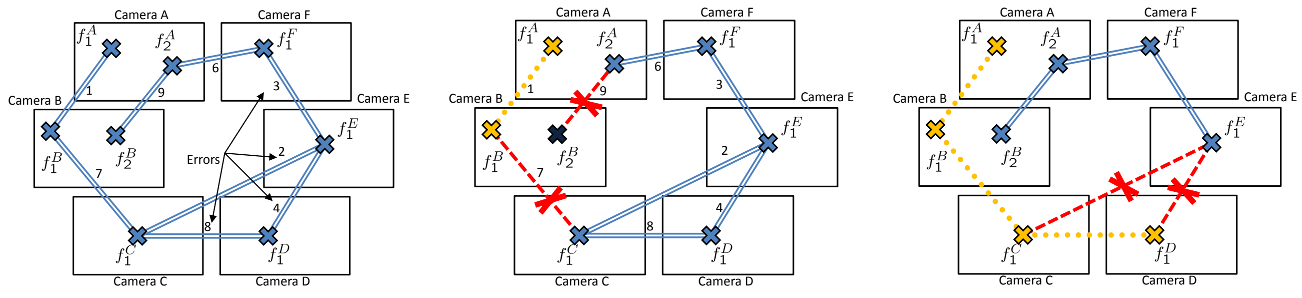
The algorithm ends its execution after no more than N communication rounds. When the algorithm finishes, each original conflictive set $\mathcal{C}$ has been partitioned into $\tilde{m}_{i_\star}$ disjoint, conflict-free components. It may happen that a subset of features remains unassigned. These features may still be conflictive. The detection and resolution algorithms can be executed on the subgraph defined by this smaller subset of features obtaining in the end an association graph free of all the inconsistencies. Empirical comparisons between the two methods are provided in section 3.5.

Algorithm 3 *Spanning Trees* - Robot i

```
Require: Set of $C$ different conflictive sets
Ensure: $\mathcal{G}_{dis}$ is conflict free
 1: – Initialization
 2: for all $\mathcal{C}$ such that $i$ 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: – Algorithm
 8: for each component request from $f_s^j$ to $f_r^i$ do
 9:     if (b) or (c) then
10:         $[\mathbf{A}_{ij}]_{r,s} = 0$
11:         send reject message to $j$
12:     else if (d) then
13:         assign $f_r^i$ to the component
14:         send component request to all its neighboring features
15:     end if
16: end for
17: for each component reject from $f_s^j$ to $f_r^i$ do
18:     $[\mathbf{A}_{ij}]_{r,s} = 0$
19: end for
```

Example of execution

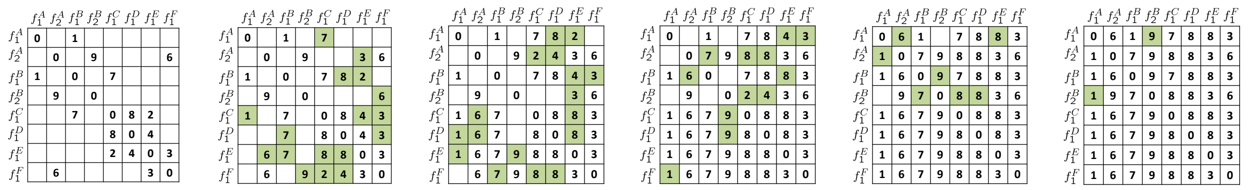
Let us consider one inconsistency as the one depicted in Fig. 3.5 (a) where the communication graph is a ring with an additional edge between robots C and E.



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

Figure 3.5: Example of execution of the resolution of one inconsistency using the two approaches. (a) Inconsistency. (b) Solution obtained using the *Maximum Error Cut* approach. (c) Solution obtained using the *Spanning Trees* algorithm. A detailed explanation can be found in section 4.3.

Figure 3.5 (b) shows the solution obtained using the *Maximum Error Cut* algorithm. The evolution of the $\mathbf{z}_r$ vectors is shown in Figure 3.6. Each one of the figures 3.6 (a)-(f) represents a new iteration of the algorithm in (3.11). The -1 values are omitted for clarity. As an example of how it works, the third row in figure 3.6 (b), corresponding to f_1^B is obtained as follows. f_1^B executes (3.11) and updates its row in Fig. 3.6 (a) with the 1st and 5th rows in Fig. 3.6 (a), 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 in Fig. 3.6 (b) (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 (3.11) has finished, robots A and B look for the cuts to break their inconsistencies (Fig. 3.7). For robot B the best cut is the one matching features f_2^A and f_2^B. For the robot A the largest error is in the column associated to f_2^B. However, this is not a cut 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 because it is a cut and the match between f_1^B and f_1^C is deleted.



	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0		1					
f_2^A		0		9				6
f_1^B	1		0		7			
f_2^B		9		0				
f_1^C			7		0	8	2	
f_1^D					8	0	4	
f_1^E					2	4	0	3
f_1^F		6					3	0

(a)

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0		1		7			
f_2^A		0		9			3	6
f_1^B	1		0		7	8	2	
f_2^B		9		0				6
f_1^C	1		7		0	8	4	3
f_1^D			7		8	0	4	3
f_1^E		6	7		8	8	0	3
f_1^F		6		9	2	4	3	0

(b)

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0		1		7	8	2	
f_2^A		0		9	2	4	3	6
f_1^B	1		0		7	8	4	3
f_2^B		9		0			3	6
f_1^C	1	6	7		0	8	8	3
f_1^D	1	6	7		8	0	8	3
f_1^E	1	6	7	9	8	8	0	3
f_1^F		6	7	9	8	8	3	0

(c)

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0		1		7	8	4	3
f_2^A		0	7	9	8	8	3	6
f_1^B	1	6	0		7	8	8	3
f_2^B		9		0	2	4	3	6
f_1^C	1	6	7	9	0	8	8	3
f_1^D	1	6	7	9	8	0	8	3
f_1^E	1	6	7	9	8	8	0	3
f_1^F	1	6	7	9	8	8	3	0

(d)

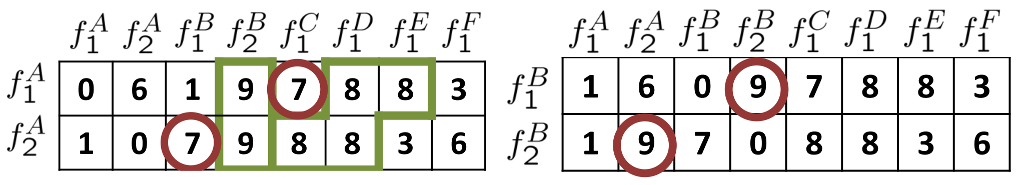
	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0	6	1		7	8	8	3
f_2^A	1	0	7	9	8	8	3	6
f_1^B	1	6	0	9	7	8	8	3
f_2^B		9	7	0	8	8	3	6
f_1^C	1	6	7	9	0	8	8	3
f_1^D	1	6	7	9	8	0	8	3
f_1^E	1	6	7	9	8	8	0	3
f_1^F	1	6	7	9	8	8	3	0

(e)

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0	6	1	9	7	8	8	3
f_2^A	1	0	7	9	8	8	3	6
f_1^B	1	6	0	9	7	8	8	3
f_2^B	1	9	7	0	8	8	3	6
f_1^C	1	6	7	9	0	8	8	3
f_1^D	1	6	7	9	8	0	8	3
f_1^E	1	6	7	9	8	8	0	3
f_1^F	1	6	7	9	8	8	3	0

(f)

Figure 3.6: Example of execution of (3.11) for the inconsistency in Fig. 3.5 (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 edges should be deleted to solve them. For more details see Section 4.3.



	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0	6	1	9	7	8	8	3
f_2^A	1	0	7	9	8	8	3	6

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^B	1	6	0	9	7	8	8	3
f_2^B	1	9	7	0	8	8	3	6

Figure 3.7: Decision about which edges should be deleted to solve the inconsistency. Robot B chooses the edge (f_2^A, f_2^B). Robot A discards the elements with values 8 and 9 because they belong to a cycle and to an edge 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.

The *Spanning Trees* solution is shown in Fig. 3.5 (c). 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 edges are broken. After this point the algorithm has ended its execution and the new association graph is conflict free.

3.5 Experiments

3.5.1 Simulations

We have designed a simulation environment using MatLab to evaluate the performance of the proposed algorithms. The environment considers a set of N robots observing the same m features. To find the local matches, we start from the perfect association graph. After that, we randomly remove a percentage of the perfect associations (p_m Missing Edges) and add a percentage of spurious matches (p_s Spurious Edges). The errors for the good matches are randomly assigned between 0 and 10. For the spurious matches we use a parameter, ϵ, so that the error is randomly generated between 0 and $(1+\epsilon)10$.

By varying the three parameters, p_m, p_s and ϵ, we can model different types of matching without committing to a specific feature or function. For example, matching templates using the intensity of the pixels returns many spurious associations, but the errors are quite discriminative. Therefore, this matching function is characterized by a small value of p_m and large values of p_s and ϵ. Other example, the epipolar constraint returns a very robust match, at the price of missing many good matches. The threshold to filter the outliers makes all the errors very similar. In this case we have a big value of p_m and small values of p_s and ϵ.

At each trial, we assume the local matching to be deterministic, i.e., for the same matching function and pair of images, the local matching is always the same. In this way we can repeat the experiment considering different network topologies. The networks are generated as random graphs, where each communication link has independent probability of exist, δ. We call this parameter the network density, because values close to 1 create networks with many links whereas small values of δ imply very sparse networks.

Since all the robots are observing all the features, we can define a quantitative metric to measure the quality of the global matching. We define a full match as an association set in which the N robots of the network match the same feature. The optimal solution is found when the network finds the m full matches. In Fig. 3.8, 3.9 and 3.10 we show the percentage of full matches using two different matching functions, $F1 = [p_m, p_s, \epsilon] = [0.1, 0.1, 0.2]$, and $F2 = [p_m, p_s, \epsilon] = [0.5, 0.05, 0.05]$, simulating the two examples above mentioned and varying the number of features, of robots and the density of the network.

Influence of the number of features: Fig. 3.8 shows the percentage of full matches after the propagation (P), and after executing the Maximum Error Cut (MEC) and the Spanning Trees (ST) resolution methods for different values of m. The number of robots is fixed and equal to $N = 8$ and the density of the network is $\delta = 0.5$. For each number of features we have repeated the experiment 100 times with different initial configurations. Independently of the matcher used, the number of features is a parameter without much influence on the obtained results. This makes sense because each association set is treated independently, and in general, with more features there are more sets, but not more complex inconsistencies. Therefore, the increase on the number of features only implies the communication of larger messages and more computational demands. We can also see that there is a difference in the values depending on the local matcher, but we leave this analysis for later in the section.

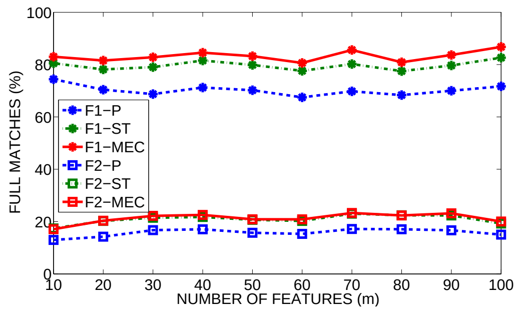


Figure 3.8: Number of matches against number of features.

Influence of the number of robots: In Fig. 3.9 we show the results for the same experiment fixing m to 15 features and varying the number of robots N. As the number of robots is increased, the percentage of full matches after the propagation step is decreased because there are more outliers (and more inconsistencies). On the other hand, using any of the resolution algorithms, the percentage is kept at good values.

Influence of the density of the network: In Fig. 3.10 we show the results considering different densities of the network and fixed number of robots, $N = 8$, and features, $m = 50$. With more communication links between robots our algorithms have a better performance. With few communication links it is more probable for a spurious match to pass undetected, whereas with more links, it will be easier to detect inconsistencies. This detection allows us to improve the results by deleting more spurious edges.

Influence of the local matching: The quality of the function used for the local matching is a parameter that plays a fundamental role in our algorithm. In Figs. 3.8, 3.9 and 3.10 we can see that two different matchers can return very different results. In Fig. 3.11 we have considered 9 different matchers and evaluate their performance with $N = 8$, $m = 50$, $\delta = 0.5$ and $\epsilon = 0.2$. The matchers have different values

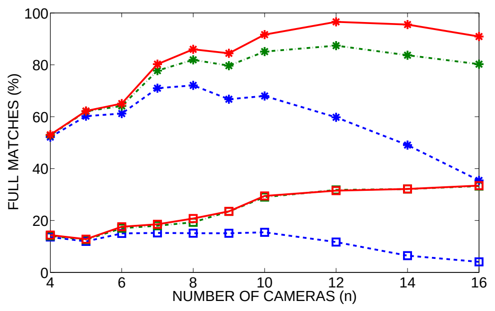


Figure 3.9: Number of matches against number of robots.

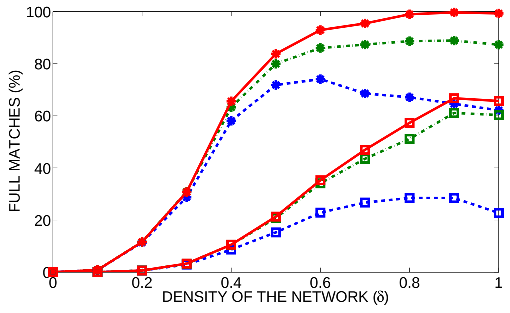


Figure 3.10: Number of matches varying the network density.

of p_s, $[S1, S2, S3] = [0.05, 0.15, 0.25]$, and different values of p_m, $[M1, M2, M3] = [0.1, 0.25, 0.5]$. As expected, the performance is decreased when the number of missing links or the number of spurious links is increased. This makes sense, because if the local matching is poor, so will be the global association. It is interesting to note how the method is more sensitive to missing links than spurious links. This happens because with fewer links, and taking into account that we are only able to delete links, it is harder to obtain a full match, whereas with more links, even if there are several spurious, cutting the appropriate links the full matches are recovered. Nevertheless, in all the cases, using the resolution methods we obtain better results than just considering the matching given by the propagation of the local matches.

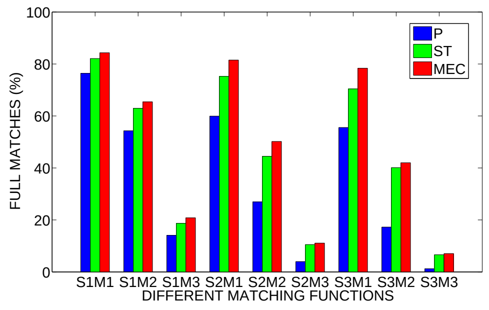


Figure 3.11: Performance of different matching functions.

In Fig. 3.12 we have repeated the same experiment but considering different values of ϵ, to see how the MEC is affected by this parameter. In this case we only show the performance obtained using the MEC method, because the propagation and the ST are not affected by this parameter. As expected, increasing ϵ the percentage of full matches is increased, because it is easier to recognize the spurious matches. However, this increase is very small, which introduces the question of why the MEC has a better performance than the ST, when the quality of the links does not seem to be an important factor in the final result. The answer to this question is that by looking to individual links that break the inconsistencies, the MEC algorithm is implicitly considering the topology of the associations, enforcing features strongly associated to remain that way.

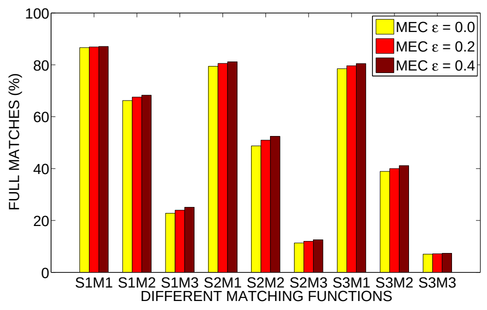


Figure 3.12: Quality of the *Maximum Error Cut* with different errors.

Finally, the graphic in Fig. 3.13 shows the percentage of spurious links (blue bars) and good links (red and yellow bars) deleted by the two resolution algorithms and the percentage of inconsistencies that the MEC was not able to solve (green bar), considering the same conditions as in Fig. 3.11. In this plot we can see that the MEC is able to delete a bigger percentage of spurious links than the ST. On the other hand, both methods delete more or less the same number of good links, which almost in all the cases represents a small percentage of the total of good links. Finally, we observe in the green bar that the MEC is almost always able to manage the inconsistencies.

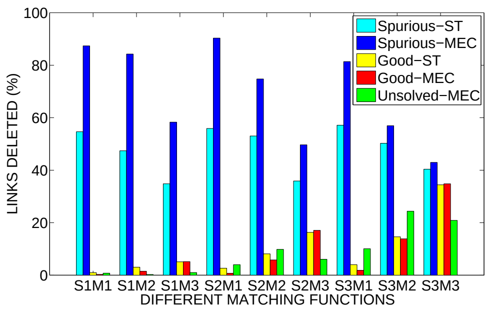


Figure 3.13: Percentage of links deleted.

In conclusion, our algorithms are able to improve the initial data association, in the sense that they are able to identify and delete a great proportion of the spurious links introduced by errors in the local matching. As a consequence, the global data association we obtain is better than the one obtained by just propagating the local associations. In addition, the whole process is done in a distributed way.

3.5.2 Experiments with real images

We have also tested our proposal with real images considering different scenarios such as teams of mobile robots with cameras, intelligent cell phones or surveillance camera networks. In each example we have used different features and functions to find the local correspondences.

Data association using geometric constraints

Two examples are reported for this kind of constraints. In the first experiment there are 6 robots moving in formation (around 5 m away from each other). Each robot acquires one image with its camera and extracts SURF features [13] (Fig. 3.14). The epipolar constraint plus RANSAC [57] is used for the local matching. The detection and resolution of inconsistencies is analyzed for four different typical communication graphs (Fig. 3.15). The error function used for the *Maximum Error Cut* algorithm is the Sampson distance.

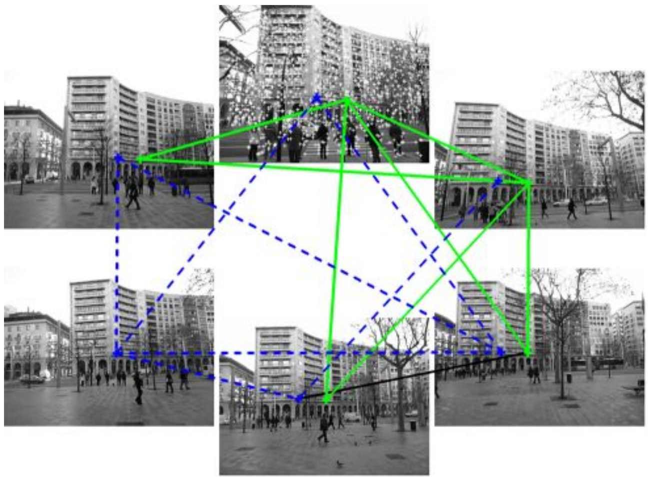
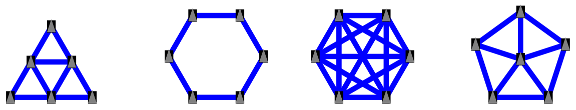


Figure 3.14: Images acquired by 6 robots moving in formation. We show an example of one inconsistency solved with the *Spanning Trees* algorithm by deleting the black line. The blue lines show a full match and the green lines a partial match. For clarity, we do not show the rest of the matches or inconsistencies.



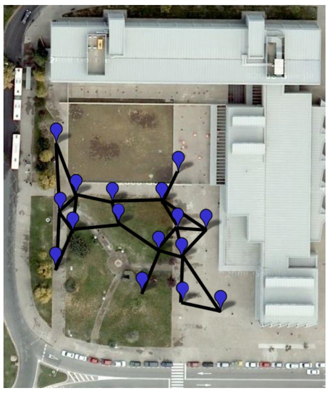
(a) Pyramidal (b) Ring (c) Complete (d) Star-Ring

Figure 3.15: Formations used in the experiment.

The second example considers a set of images using one Iphone outside one building of the University of Zaragoza. These pictures usually contain a GPS tag of where they were taken. We have used these tags to define a proximity graph (Fig 3.16 (a)), with distances between the images of 10 to 20 meters. In this case we have used the homography constraint to execute the local matches. The error of the matches has been computed with $\|\mathbf{f}_1 - \mathbf{H}_{12}\mathbf{f}_2\|$, normalizing with respect to the homogeneous coordinate.

We have chosen man made scenarios to be able to manually classify the matches, see Fig. 3.14 and Fig 3.16 (b). Although ground truth is not available in this examples, by looking at the correspondences we have counted the amount of full matches. Since this number is very small or even zero due to missing matches and occlusions caused by the trees in the images, For that reason we also define a partial match when a subset larger than 50% of the robots (4 robots in the first example and 9 in the second) associates the feature without spurious links. Finally, a wrong match is defined when there is at least one spurious link in the set but there is no inconsistency. This last type of associations cannot be improved with the methods presented in this Chapter.

The results of the two experiments can be seen in Table 3.1. The number of direct matches and the number of features involved increases with the number of edges in $\mathcal{G}$. Having more local matches, the propagation is able to find more associations and inconsistencies between robots that cannot communicate. Finding more inconsistencies is something good because it means the detection of a spurious match, which can be corrected in the global matching. In all the cases there are more full and partial matches after solving the conflicts, independently on the method used. Since the resolution algorithms delete the links using only partial information, it is also natural to have more wrong matches after the resolution. Nevertheless, the increase of partial and full matches is in all the cases almost the same as the number of inconsistencies, which means that the resolution


(a) Communication Network

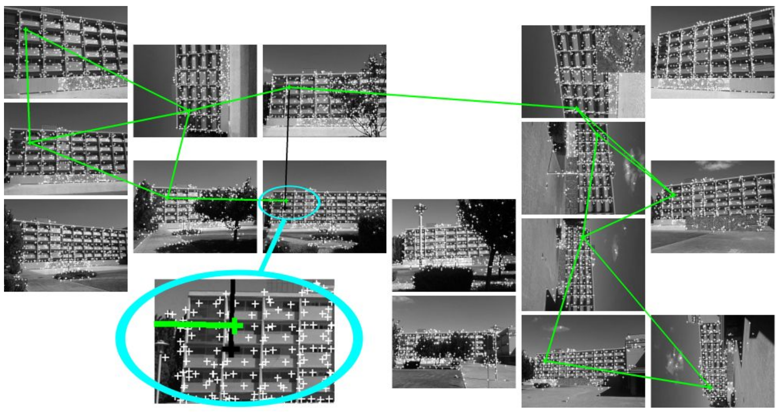

(b) Example of one inconsistency

Figure 3.16: 16 images of one building captured with one Iphone. (a) Communication network used in the experiment and GPS positions of the images provided by the Iphone. (b) One inconsistency solved using the *Maximum Error Cut*. The algorithm deletes the spurious association (black line): the macth between a feature in the fourth floor with another feature in the third floor (zoomed region). Although the inconsistency only appears in one image it affects almost all the captured images. The rest of the SURF points are in the images but for clarity we have removed the rest of the matches.

algorithms are in general able to obtain a good partition of the inconsistencies.

The number of inconsistencies also depends on the number of cycles in $\mathcal{G}$ because each cycle can generate inconsistencies independently of the rest of the communication network. The size of the cycles also affect the conflicts. Small cycles will cause more inconsistencies because the number of local matches required to find a conflict is also small. On the other hand, the inconsistencies that appear because of small cycles in the communication graph are usually easier to handle than inconsistencies caused by large cycles because they contain a fewer number of features.

Data association using image templates

Another motivating example to test our algorithms is the association of people across multiple views in surveillance tasks. We present an example to show the possibilities of our algorithm in this field of research.

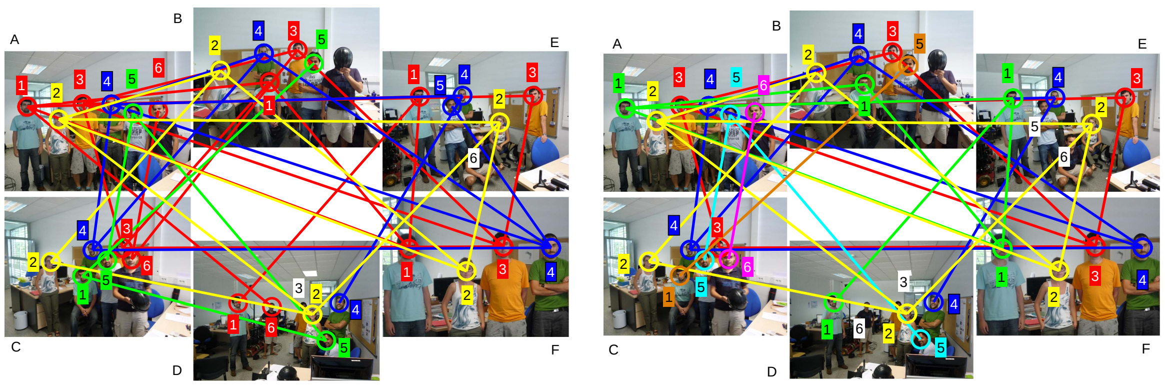
In order to show in a clear way the behavior of the methods, in the experiment we have considered 6 pictures with 6 people. The faces have been extracted using a Haar classifier and the implementation available in Open CV. Each patch containing a face has been resized to a fixed dimension of 100x100 pixels. The premises for the local matching are that the pictures are acquired in relatively close instants of time, therefore we can expect similar conditions of lightning and appearance. However, we do not make any assumption about the geometry of the environment or of the people visible in the images, i.e., there is no database to recognize the people and geometric constraints to match cannot be used. With all these considerations, the local matching is carried out computing the absolute differences between pairs of patches, weighted using an Epanichov kernel to give more importance to the center of the patch than to the edges. We have used this matching function for simplicity but more recent and robust functions can be used, see e.g., [52, 112].

In Fig. 3.17 (a) we show the matches found between neighbor robots after the propagation algorithm. For a better interpretation we have manually classified the faces, assigning them identifiers. Each color in the lines represent one association set. Robots without direct links between them, e.g., B and D, are robots that cannot communicate. There are 4 different association sets and two unassigned features (D3 and E6). On of the sets is a full match (feature 2), and the other 3 are inconsistencies, containing a total of 6 spurious links (A1-D6, B1-C3, B3-C6, B5-C1, C1-D5 and E5-F4). In Fig. 3.17 (b) we show the final association after using the Maximum Error Cut. In this case there are 7 association sets and 3 unassigned features. From the 7 sets there are 2 full matches (features 2 and 4), 2 partial matches (features 1 and 3, being matched in 5 robots),

Table 3.1: Associations for the different communication graphs

Comm. graph	Fig. 3.15 (a)	(b)	(c)	(d)	Fig. 3.16 a
AFTER PROPAGATION					
Total Features	1985	1704	2518	2144	4854
Total Links	1398	1017	2305	1582	3825
Inconsistencies	17	2	55	30	26
Incons. feats.	115	23	448	188	251
Incons. Links	106	20	518	175	266
Full Matches	11	5	11	14	0
Partial Matches	31	16	48	34	50
Wrong Matches	6	2	11	7	2
SPANNING TREES					
Deleted Links	20	2	112	46	40
Full Matches	11	6	16	14	0
Partial Matches	40	17	88	50	58
Wrong Matches	12	2	36	15	4
MAXIMUM ERROR CUT					
Incons. Not Solved	0	0	4	1	0
Deleted Links	24	2	103	38	36
Full Matches	13	6	18	15	0
Partial Matches	41	17	87	52	60
Wrong Matches	8	2	26	13	3

other 2 associations(feature 5 with 3 robots and feature 6 with 2 robots), and one wrong association (B5-C1). The algorithm has been able to solve all the inconsistencies and has removed 5 of the 6 spurious links without removing any inlier.



(a) Data association after propagation

(b) Data association after resolution

Figure 3.17: Matching faces across images.

In Fig. 3.18 we show one wrong association after the resolution. Note that in this inconsistency 2 links should be removed in order to get the good association, whereas by removing only one we can break the inconsistency. This shows the difficulty of the problem of solving inconsistencies, because even having the knowledge of the whole graph (which we do not), every possible partition can be the right one, independently on the number of removed edges.

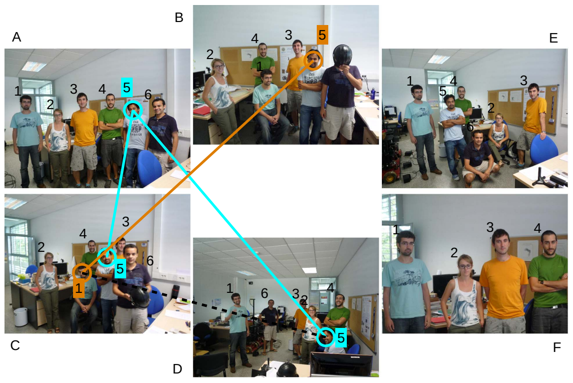


Figure 3.18: Inconsistency containing 2 spurious links. The inconsistency is solved by deleting only one of them.

Data association of stochastic maps

In this experiment each robot has explored a section of the environment and it has built a stochastic map using a SLAM algorithm. When the exploration finishes, the local maps are merged into a global map of the environment [5]. If the robots start the merging process using only its local associations, and there is any inconsistency, at some point a robot will be forced to fuse two or more of its features into a single one. To avoid this situation, they execute the presented algorithm solving any inconsistent association before merging the maps.

We use a data set [49] with bearing information obtained with vision (Sony EVI-371DG). The landmarks are vertical lines extracted from the images (Fig. 3.19). The measurements are labeled so that we can compare our results with the ground-truth data association. We select 8 sections of the whole path for the operation of 8 different robots. A separate SLAM is executed on each section, producing the 8 local maps (Fig. 3.20). As in many real scenarios, here the landmarks are close to each other, and the only information available for matching them are their cartesian coordinates. The local data associations are computed using the Joint Compatibility Branch and Bound (JCBB) [102] since it is very convenient for clutter situations like the considered scenario. 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 3.21).

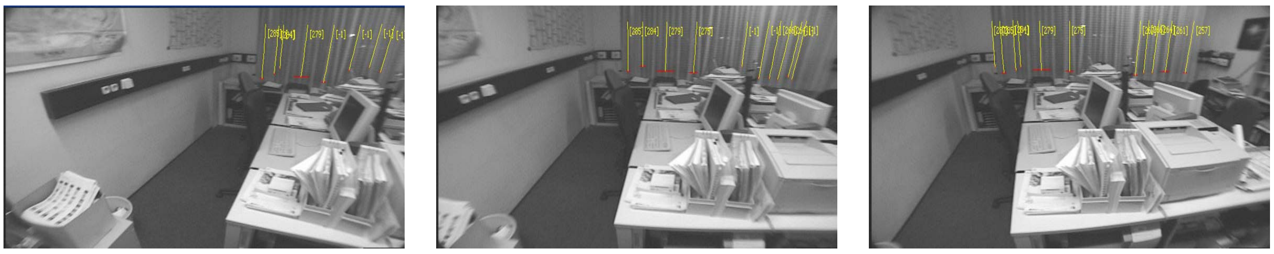


Figure 3.19: An example of the images used by the 8 robots during the navigation to test the proposed method [49]. We test the algorithm using the lines extracted from natural landmarks (in yellow)

Table 3.2 shows the results for the different network topologies in Fig. 3.21. In Fig. 3.20 we can see the local matches obtained under the communication graph in Fig. 3.21 (a). We assign to each edge an error that depends on the number of matches between the local maps. Thus, we assume that an edge that belongs to a set with many jointly compatible matches has many chances of being a good edge. Between the edges belonging to the same set of jointly compatible associations, we use the individual Mahalanobis distance to differentiate their errors. Then, we apply the two resolutions algorithms to solve the inconsistencies. The *Spanning Trees* approach, which does not take into account the errors associated to the edges, produces good results. For the

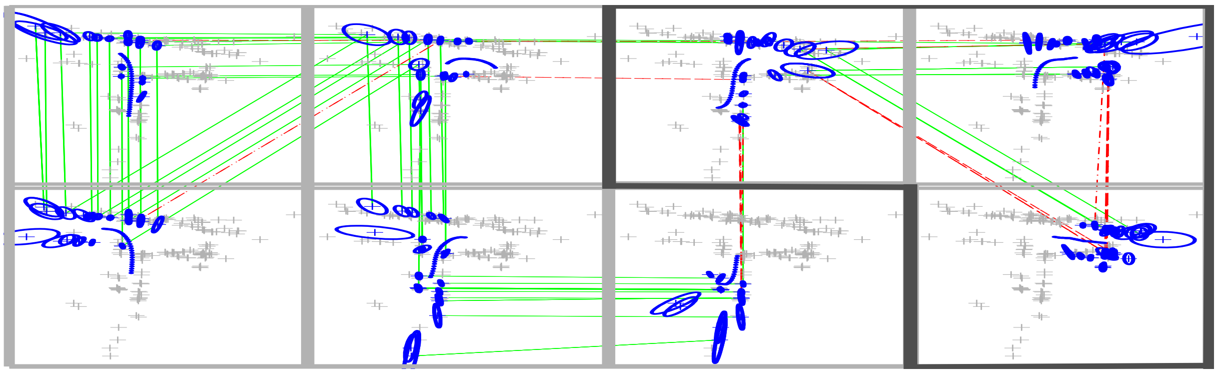


Figure 3.20: Local maps acquired by 8 robots (blue) during their exploration. We also display the features observed by all the robots (gray crosses) to give an idea of the region explored by each robot. Each robot solves a local data association with its neighbors in the communication graph in Fig. 3.21 (a). Although many of the local edges are good (green solid lines), there are also some spurious matches (red dashed lines) that give rise to an inconsistency between 3 of the local maps (inside the dark gray area).

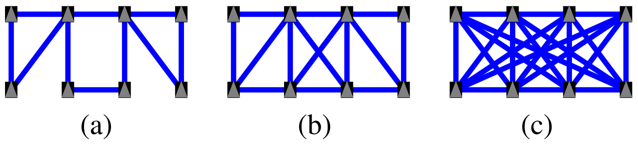


Figure 3.21: Communication graphs between the 8 robots used for evaluating the data association of stochastic maps.

three communication schemes, it improves the amount of full and partial matches. However, the *Maximum Error Cut* algorithm produces better results. The total number of edges deleted by this approach is lower, whereas the number of full matches is higher than for the spanning trees method.

3.6 Discussion

Summing up, this chapter has made possible for a team of robots with multiple observations to distinguish common features among the robots in a distributed fashion.

Using as input the local correspondences found between robots that can communicate, we have proposed a fully decentralized method to compute all the paths between local associations. We have proven that the algorithm is fully distributed, requires low communication and finishes in finite time. We have also dealt with the problem of solving the inconsistencies that occur because of spurious local matches. In order to break the inconsistencies, we have presented two different algorithms. One of them considers the quality of each local match, when this information is available, finding and deleting the local match with the maximum error that breaks the inconsistency. The other algorithm computes different spanning trees free of conflicts using a breadth first search technique. Additionally, an extensive evaluation of the proposed algorithms has shown that they can be applied with a wide variety of features and local matchers.

With the execution of these algorithms, the team of robots has the knowledge of the real number of features observed and which observations should be mixed in the consensus process. On the other hand, robustness issues have not been completely handled. It is true that our algorithms are able to detect and eliminate a large percent of the spurious matches introduced by the local matching function. However, as stated in Remark 3.3.8, those spurious that do not belong to an inconsistency will pass undetected in the global data association. Moreover, there still might be a small percentage of outliers that our resolution algorithms did not delete. Therefore, additional mechanisms are required during the consensus computation, such that they make the system robust to these kind of errors.

Table 3.2: Results for the data association of stochastic maps

Comm. graph	(a)	(b)	(c)
AFTER PROPAGATION			
Total Features	194	194	194
Total Links	82	93	111
Inconsistencies	2	6	8
Incons. feats.	8	35	49
Incons. Links	6	33	53
Full Matches	55	49	46
Partial Matches	4	3	2
Wrong Matches	10	7	6
SPANNING TREES			
Deleted Links	2	10	16
Full Matches	55	53	50
Partial Matches	6	5	6
Wrong Matches	12	13	15
MAXIMUM ERROR CUT			
Incons. Not Solved	0	0	1
Deleted Links	2	7	14
Full Matches	55	55	52
Partial Matches	4	3	3
Wrong Matches	12	11	11

Proofs

Proof of Proposition 3.3.2 (Limited Communications)

By the definition of the operations in lines 6 and 9 of Algorithm 1 we can see that the components of $\mathbf{y}_r^i$ change their value at most once during the whole execution, for all $i \in \mathcal{V}, r \in \mathcal{S}_i$. 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. Therefore, in the worst case, the amount of transmitted information through the network during the whole execution of the algorithm now is $2m_{sum}^2$. ■

Proof of Proposition 3.3.3 (Correctness)

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 includes the features r. 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 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, conditions (3.7)-(3.8) can be applied to $[\mathbf{y}_r^i]_u$ to detect conflictive features. ■

Proof of Theorem 3.3.4 (Limited Iterations)

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, $\mathcal{G}$, 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 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 1. The other robot is the one with the common feature, 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 $2N$. ■

Proof of Proposition 3.4.2 (Number of Partitions)

Each conflict-free component can contain, at most, one feature observed by each robot $i \in \mathcal{V}_{inc}$. Then there must be, at least, $\max_{i \in \mathcal{V}_{inc}} \tilde{m}_i = \tilde{m}_{i_\star}$ components. ■

Proof of Proposition 3.4.5 (Convergence)

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 [18]. For any $r, s \in \mathcal{C}$ such that $[\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0$, by eq. (3.11) 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

$$\mathbf{z}_r \geq \mathbf{z}_s \mathbf{P}_{rs} \text{ and } \mathbf{z}_s \geq \mathbf{z}_r \mathbf{P}_{sr}. \tag{3.16}$$

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. (3.16) yields to $\mathbf{z}_r = \mathbf{z}_s\mathbf{P}_{rs}$.

For any feature, r, taking into account eq. (3.12), the update of the r^{th} element of $\mathbf{z}_r$, $[\mathbf{z}_r(t+1)]_r$, is

$$[\mathbf{z}_r(t+1)]_r = \max_{s \in \mathcal{C},\ [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0} ([\mathbf{z}_r(t)]_r, [\mathbf{z}_s(t)]_s). \tag{3.17}$$

Recalling the first point in assumption 3.4.3, the initial value of $[\mathbf{z}_r(0)]_r = e_{rr} = 0$, for all r, then $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$. ■

Proof of Theorem 3.4.6 (Values for Bridges)

First note that (s, u) is a bridge, therefore it creates a partition of $\mathcal{C}$ in two strongly connected, disjoint subsets

$$\mathcal{C}_s = \{r \in \mathcal{C} \mid d(r, s) < d(r, u)\},$$

$$\mathcal{C}_u = \{r \in \mathcal{C} \mid d(r,u) < d(r,s)\}. \tag{3.18}$$

In the above equations it is clear that $s \in \mathcal{C}_s$ and $u \in \mathcal{C}_u$.

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

$$[\mathbf{z}_r(t)]_u, r \in \mathcal{C}_s, \text{ and } [\mathbf{z}_r(t)]_s, r \in \mathcal{C}_u.$$

In the first case, for any $r \in \mathcal{C}_s \setminus s$, update rule (3.12) is equal to

$$[\mathbf{z}_r(t+1)]_u = \max_{r' \in \mathcal{C}_s,\ [\mathbf{E}_\mathcal{C}]_{r,r'} \geq 0} ([\mathbf{z}_r(t)]_u, [\mathbf{z}_{r'}(t)]_u), \tag{3.19}$$

because $r \neq u \neq r'$. The nodes in $\mathcal{C}_u$ are not taken into account because that would mean that (u,s) belongs to a cycle and it is not a bridge. The special case of feature s has an update rule equal to

$$[\mathbf{z}_s(t+1)]_u = \max_{r' \in \mathcal{C}_s, [\mathbf{E}_\mathcal{C}]_{s,r'} \geq 0} ([\mathbf{z}_s(t)]_u, [\mathbf{z}_{r'}(t)]_u, [\mathbf{z}_u(t)]_s). \tag{3.20}$$

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

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

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

$$[\mathbf{z}_r(t)]_u, [\mathbf{z}_{r'}(t)]_s \to \max_{r \in \mathcal{C}_s,\ r' \in \mathcal{C}_u} ([\mathbf{z}_r(0)]_u, [\mathbf{z}_{r'}(0)]_s). \tag{3.21}$$

Since all the features $r \in \mathcal{C}_s \setminus s$ are not associated with u, $[\mathbf{z}_r(0)]_u = -1$. Analogously, for all the features $r \in \mathcal{C}_u \setminus u$, $[\mathbf{z}_r(0)]_s = -1$. Finally, for the features u and s, by the second and third point of assumption 3.4.3, $[\mathbf{z}_u(0)]_s = e_{us} = e_{su} = [\mathbf{z}_s(0)]_u \geq 0 > -1$. Therefore this subset of elements of the state vectors converge to the error of the edge (u,s), e_{us}. ■

Proof of Theorem 3.4.7 (Values for Cycles)

We will use the following lemma to proof the result

Lemma 3.6.1. *Let us consider a feature u, such that it is an articulation vertex, defined as a node in $\mathcal{C}$ whose deletion increases the number of connected components of $\mathcal{C}$. Denote $\mathcal{C}_u$ and $\mathcal{C}_{u'}$ the two partitions generated by its deletion. For any $r \in \mathcal{C}_u$, $s \in \mathcal{C}_{u'}$, $[z_r(t)]_s$ will converge to the same value as $[z_u(t)]_s$ and $[z_s(t)]_r$ will converge to the same value as $[z_u(t)]_r$. Moreover $[z_r(t)]_u$ will converge to a different value than $[z_s(t)]_u$.*

Proof. Considering the fact that u is the only feature that connects $\mathcal{C}_u$ and $\mathcal{C}_{u'}$, the permutations in (3.12) for elements in $\mathcal{C}_u$ matched with u will not change the value of the components related to elements in $\mathcal{C}_{u'}$ and viceversa. On the other hand the permutations will affect the value of the $[\mathbf{z}_r(t)]_u$ and $[\mathbf{z}_s(t)]_u$ for features matched to u, shifting it to different positions in the two partitions. Then using Proposition 3.4.5 the result holds.

Proof of Theorem 3.4.7

Let $\bar{\mathcal{C}}_\ell = \mathcal{C} \setminus \mathcal{C}_\ell$ be the rest of the features in the inconsistency. Given a feature $r \in \bar{\mathcal{C}}_\ell$ there exists a unique $s \in \mathcal{C}_\ell$ such that there is at least one path of features in $\bar{\mathcal{C}}_\ell$ that ends in s. The uniqueness of s comes from the fact that if there were another feature $s' \in \mathcal{C}_\ell$, reachable from r without passing through s, that would mean that r is also part of the cycle. Note that this does not discard the possibility that r and s belong to another cycle different than $\mathcal{C}_\ell$. Also note that $s = \arg\min_{s' \in \mathcal{C}_\ell} d(r, s')$.

Since s is the only connection with $\mathcal{C}_\ell$, then it is an articulation vertex and, by Lemma 3.6.1, for any $s' \in \mathcal{C}_\ell \setminus s$, $[\mathbf{z}_r]_{s'}$ will have final value equal to $[\mathbf{z}_s]_{s'}$. Therefore, if we show that (3.15) is true for the features belonging to the cycle then the theorem is proved.

Let us see what happens to features inside the cycle. First note that $r = \arg\min_{s \in \mathcal{C}_\ell} d(r, s)$, $\forall r \in \mathcal{C}_\ell$, and therefore, by Proposition 3.4.5, this element is always zero. Now, for any $r \in \mathcal{C}_\ell$, if we consider another element $s \in \mathcal{C}_\ell$, such that r is not directly matched to it, the update rule (3.12) is

$$[\mathbf{z}_r(t+1)]_s = \max_{u \in \mathcal{C}_\ell, [\mathbf{E}_\mathcal{C}]_{r,u} \geq 0} ([\mathbf{z}_r(t)]_s, [\mathbf{z}_u(t)]_s). \tag{3.22}$$

We have omitted other possible features that are directly matched to r and that do not belong to $\mathcal{C}_\ell$ because they cannot be matched to s, otherwise they would belong to $\mathcal{C}_\ell$, and then, because of Lemma 3.6.1, they do not affect to the final value of $[\mathbf{z}_r]_s$.

The special case of features in the cycle, s', directly matched to s has update rule equal to

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

Due to the permutation, $[\mathbf{z}_{s'}]_s$ depends on the value of $[\mathbf{z}_s]_{s'}$. Then, by Proposition 3.4.5 and the connectedness of the cycle, in the end $[\mathbf{z}_r]_s$ will have the same value for all $r \in \mathcal{C}_\ell \setminus s$, and equal to the final value of $[\mathbf{z}_s]_{s'}$, for any s' in the cycle directly associated to s. By applying the same argument for any other element corresponding to a feature in $\mathcal{C}_\ell$ we conclude that after the execution of enough iterations of (3.12), for any $r \in \mathcal{C}_\ell$, $[\mathbf{z}_r]_s = [\mathbf{z}_r]_{s'}, \forall s, s' \in \mathcal{C}_\ell \setminus r$. 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 edges, and (3.15) is true. ■

Proof of Theorem 3.4.9 (Properties of the *Spanning Trees* resolution)

(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 between any two features belonging to a conflictive set $\mathcal{C}$. Therefore, there is also a path in $\mathcal{G}_a$ between any two features assigned to the same component $\mathcal{C}_q$. Since the algorithm does not delete edges from $\mathcal{G}_a$ within a component (case (a)), then $\mathcal{C}_q$ it is also connected in the new partition.

(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 Assumption 3.2.1, 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. ■

Chapter 4

De-RANSAC: Distributed Robust Consensus

"Robustness is the ability of a system to cope with errors during the execution." This property is essential in any robotic system. A reliable robotic network must be able to fuse its perception of the world in a robust way. Data association mistakes and measurement errors are some of the factors that can contribute to an incorrect consensus value. In this chapter, we present a distributed scheme for robust consensus in autonomous robotic networks. The method is inspired by the RANSAC (RANdom SAmple Consensus) algorithm. We propose a distributed version of this algorithm that enables the robots to detect and discard the outlier observations during the computation of the consensus. The basic idea is to generate different hypotheses and vote for them using a dynamic consensus algorithm. Assuming that at least one hypothesis is initialized with only inliers, we show theoretically and with simulations that our method converges to the consensus of the inlier observations.

4.1 Introduction

In the previous chapter we provided the network with an algorithm able to find correspondences between all the observations of the robots. In a perfect setup, this would be enough to compute the consensus. Unfortunately, in a real scenario there is still the possibility that the given data association contains spurious correspondences. This can be due to inconsistencies wrongly solved in our previous algorithm or by mistakes caused by the local matcher and undetected by our method, as mentioned in Remark 3.3.8. One of the most important properties that any robotic system requires is robustness against different types of failures. A major drawback of the standard distributed averaging algorithm is the lack of robustness to erroneous values. That is, the consensus value can be severely affected by wrong sensor measurements or outliers. Without proper mechanisms that ensure the rejection of the spurious information while fusing the data, the final consensus estimate can be highly unreliable. An example of this problem is illustrated in Fig. 4.1. Five robots are looking for an exit door

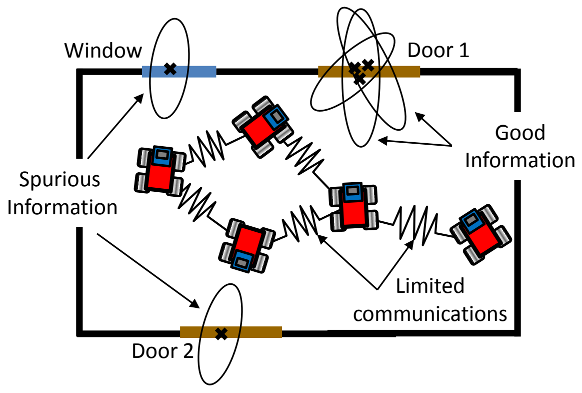


Figure 4.1: Five robots are looking for the exit door of a room. Two of the robots have wrong estimates that must be discarded in the fusion algorithm. The proposed algorithm solves this problem in a decentralized way.

of a room. Due to perception or data association mistakes, two of the robots have measurements of another door and a window. If the information is fused together without additional control mechanisms, the identified exit door location will be biased and unreliable. This chapter focuses precisely on this aspect and aims to find a distributed algorithm that can solve this problem.

Different approaches have been proposed to cope with noisy information in distributed consensus algorithms. The works in [59, 72, 118] are focused on the optimal mix of the measures minimizing some error function, but do not consider situations in which some of the measurements are spurious and must be discarded from the fusion. Similarly, channel noise is addressed in [73, 154]. These works assume that each message introduces additive noise in the estimations of the nodes and propose solutions to guarantee convergence to the desired result. Faulty nodes or malicious agents are treated in [45, 46, 116, 139, 140]. The main idea of these approaches is that the faulty nodes' actions depart from the expected behavior and the good nodes are able to detect them.

Compared to all these approaches, we assume that the communication protocol used by the robots guarantees the reception of the sent information without noise. The robots in the network are cooperative and behave as expected; however, some of them have outlier information. We deal with the problem of identifying the robots with outlier information and discarding their contributions to the final consensus value.

In computer vision, RANSAC (RANdom SAmple Consensus) [43] is a widely used algorithm for robust matching between pairs of images. In the last few years, new variations to the basic algorithm have been presented [26, 28, 104]. However, none of these algorithms are implementable over distributed networks. The closest approach to a distributed scenario can be found in [156]. There, the information to be fitted is separated in several non-overlapping subsets and the hypotheses are generated choosing data from them. However, the whole process is still centralized.

The main contribution of this chapter is a new robust distributed consensus method, *De-RANSAC*, in which the outlier observations are identified and the consensus is achieved discarding their information. Our approach extends RANSAC to a distributed setup. Briefly, the RANSAC steps can be summarized as follows:

- Create random hypotheses with subsets of observations.
- Choose the best hypothesis using a voting process.
- Compute a better model considering only the observations that voted the best hypothesis as good.

Our algorithm executes the same three steps in a fully decentralized fashion, computing a robust consensus. Moreover, in our approach robots are allowed to change their opinion, making the voting process dynamic and executing the three steps simultaneously. The algorithm expands the utility of RANSAC by adding a new set of applications where it can be used in the field of sensor networks, such as robust localization of events of interest [131], face recognition [145], distributed labeling [69] or collaborative sensor-bias calibration [19]. As such, our algorithm can be seen as a contribution to this body of literature.

Additionally, the chapter presents two other contributions:

- Two extensions of the averaging rule to allow robots without an observation to participate in the consensus process and to guarantee convergence in finite time when the consensus value has finite precision.
- A distributed averaging primitive to compute the number of active robots in a network.

Since *De-RANSAC* makes use of these extensions, we start the chapter by explaining them in detail and then focus on the outlier detection problem. The work presented here has been discussed in [87–89]

4.2 Distributed Averaging using homogeneous coordinates

As we have seen in the previous chapter, using our data association algorithm there might be elements for which not all the robots have an observation. The general averaging algorithm requires all the robots to have some initial value to introduce in the algorithm. We propose a modification of the algorithm to obtain an average of the measurements of a subset of the robots without excluding from the linear iteration any of the participating robots. The modification makes use of another classic concept in computer vision, homogeneous coordinates.

The homogeneous coordinates of a d_1-dimensional vector are defined as a (d_1+1)-dimensional vector where the additional coordinate is used as a scaling factor [57]. In this way all the vectors with the form $(\mathbf{x}, 1) \equiv (\rho\mathbf{x}, \rho)$, are equivalent in the projective space for any $\rho \neq 0$. Let $x_i^h(t)$ be the scale coordinate of the robot i at time instant t. We will define the extended measurements of the robots, $\mathbf{x}_i^e(t) = [\mathbf{x}_i(t), x_i^h(t)]$, as the measurements expressed in homogeneous coordinates. Let $\mathcal{V}_h \in \mathcal{V}$ be the set of robots that have some initial information to fuse in the consensus process. Each robot initializes its extended measurement with the following rule:

$$\mathbf{x}_i^e(0) = \begin{cases} [\mathbf{x}_i(0)^T,\ 1]^T \text{ if } i \in \mathcal{V}_h \\ \mathbf{0}^T \text{ otherwise} \end{cases} \tag{4.1}$$

Since the robots know if they belong to $\mathcal{V}_h$, there is no synchronization required for this.

Theorem 4.2.1 (Consensus with homogeneous coordinates)**.** *If $\mathcal{V}_h \neq \varnothing$, then the iteration rule* (2.2)*, applied to $\boldsymbol{x}_i^e(t)$ with the initial conditions defined in* (4.1) *converges, in normalized coordinates, to*

$$\boldsymbol{x}_i^e(t) \to [\frac{1}{|\mathcal{V}_h|}\sum_{j\in\mathcal{V}_h} \boldsymbol{x}_j(0)^T,\ 1]^T \quad as\ t \to \infty \quad \forall i \in \mathcal{V}.$$

Let us note that this idea can also be used in situations in which one robot has lost its sensor due to any kind of failure but its communication capabilities remain intact or, as we will see, when there is one or more robots with information that we do not want to mix in the consensus.

4.2.1 Averaging in Finite-Time

For general graph-switching sequences, the asymptotic convergence of (2.2) does not achieve convergence in a finite number of steps. However, when consensus involves integers or real numbers with a finite precision this is possible.

Definition 4.2.2. *We will say that $\Phi \subseteq \mathbb{R}^{d_2}$ is a φ-set, $\varphi \in \mathbb{R}$, $d_2 \in \mathbb{N}$ if $\forall\, \boldsymbol{x}, \boldsymbol{x}' \in \Phi$*

$$\frac{|x_k - x_k'|}{\varphi} \in \mathbb{N}, \quad \forall k = 1, \dots, d_2, \tag{4.2}$$

with x_k, x_k' being the k^{th} component of $\boldsymbol{x}$, $\boldsymbol{x}'$ respectively.

Considering the examples above mentioned, the set of integers, $\mathbb{Z}$, is defined as a 1-set and the set of reals with s decimal digits as a 10^{-s}-set.

We introduce next a modification of the standard consensus algorithm when dealing with elements in a φ-set. The idea of the new rule is to keep the distributed update in (2.2) and combine it with a rounding rule that returns the closest element in Φ. If the average, or the value of some function evaluated in the average, belongs to Φ, then the algorithm will reach the exact consensus value for all the robots in finite time.

Theorem 4.2.3 (Distributed Averaging in Finite-Time)**.** *Let $g(\boldsymbol{x})$ be a continuous function, $g : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ and $\boldsymbol{x}_i(0) \in \mathbb{R}^{d_1}$ be initial conditions with $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i\in\mathcal{V}} \boldsymbol{x}_i(0)$ their average. Given Φ as a known φ-set, if $g(\bar{\boldsymbol{x}}) \in \Phi$, then the iteration*

$$\begin{cases} \boldsymbol{x}_i(t+1) = w_{ii}(t)\boldsymbol{x}_i(t) + \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\boldsymbol{x}_j(t), \\ \hat{\bar{\boldsymbol{x}}}(t+1) = \arg\min_{\boldsymbol{x}\in\Phi} \|\boldsymbol{x} - g(\boldsymbol{x}_i(t))\|_2, \end{cases} \tag{4.3}$$

leads to the finite-time convergence of the variables $\hat{\hat{\boldsymbol{x}}}$ to the consensus value $g(\bar{\boldsymbol{x}})$. That is,

$$\exists t^* > 0 \mid \forall t > t^*,\ \hat{\hat{\boldsymbol{x}}}(t) = \hat{\hat{\boldsymbol{x}}}(t^*) = g(\bar{\boldsymbol{x}}),\ \forall i \in \mathcal{V}.$$

The proposed rule converges in finite time to the same value for all the robots. The additional required information is the value of φ, which is easy to know before running the application. Examples of different functions that can be used within this algorithm are provided ahead in the chapter.

4.2.2 A Distributed Primitive for Robot Counting

The number of robots participating in the consensus is another information that in many situations will be useful for the team, but that may not always be available.

We propose a distributed algorithm that allows a network to compute the number of robots participating in the consensus process. The algorithm is based on distributed averaging combined with a max-consensus rule. The idea of the algorithm is to make the network evolve in such a way that the sum of the initial conditions is equal to one. Since the average value is divided by the total number of robots involved in the computation, the system will tend to $1/N$ and every robot will be able to know how many robots are participating.

Let $N_i(t), i \in \mathcal{V}$, be the estimated value of N that robot i has at time instant t. In order to make $N_i(t) \to N$ the robot exchange a 2-dimensional vector, $\boldsymbol{\beta}$, initialized as

$$\boldsymbol{\beta}_i(0) = (\mathrm{ID}_i, 1)^T, \tag{4.4}$$

where ID_i is the identifier of robot i in the network, which by Assumption 2.2.1 is different to all the others identifiers. At each time step the robots update their values

$$\beta_{i1}(t+1) = \max(\beta_{i1}(t), \beta_{j1}(t)), \quad j \in \mathcal{N}_i(t), \tag{4.5}$$

$$\beta_{i2}(t+1) = w_{ii}(t)\beta_{i2}(t) + \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\beta_{j2}(t) + u_i(t+1), \tag{4.6}$$

denoting $\beta_{i1}(t)$ and $\beta_{i2}(t)$ as the first and second component of $\boldsymbol{\beta}(t)$ that robot i has at time instant t and $w_{ij}(t)$ as defined in Assumption 2.3.3. In eq. (4.6), the initial input is set to zero, $u_i(0) = 0$, and for the rest of time instants is defined as

$$u_i(t+1) = \begin{cases} -1, & \text{if } \beta_{i1}(t) \neq \mathrm{ID}_i \text{ and } \sum_{s=0}^{t} u_i(s) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

Theorem 4.2.4 (Distributed Robot Counting). *If all the robots have initial values, $\boldsymbol{\beta}$, defined in* (4.4) *and update their states using* (4.5) *and* (4.6)*, then, for all $i \in \mathcal{V}$,*

1. $\beta_{i1}(t) \to \max_{i\in\mathcal{V}} \mathrm{ID}_i$*, in finite time.*
2. $\beta_{i2}(t) \to \frac{1}{N}$ *as $t \to \infty$.*
3. *The iteration rule*

$$N_i(t) = \arg\min_{n\in\mathbb{N}} |n - \frac{1}{\beta_{i2}(t)}|, \tag{4.8}$$

converges to N in finite time.

4.3 Outlier Detection with RANSAC

Before proposing our robust distributed algorithm, let us first consider the problem of robust estimation and outlier detection in a centralized setup. In this case a central computer has access to the samples of all the robots, $X = \{x_i(0)\}$, $i \in \mathcal{V}$. In centralized scenarios, a widely used algorithm for robust estimation and outlier detection is the RANSAC algorithm [43]. This algorithm was proposed to estimate the parameters of a model that can be used to explain a set of sampled data in the presence of spurious information.

The basic idea of RANSAC is to generate a limited number, $K > 0$, of hypothetical models, also called hypotheses, using random subsets of samples, and then select one of the models using a voting system. The algorithm makes the following assumption, which we keep in our distributed version of the method.

Assumption 4.3.1. *RANSAC assumes that:*

- *Each sample has independent probability,* $0 < p_{\text{in}} \leq 1$, *of being inlier information.*
- *There exists a procedure to estimate a hypothetical model to fit the data using at least* c *samples.*
- *If a model is generated using* c *inlier samples, then the model fits all the inliers.*

In order to decide how many hypotheses generate, RANSAC uses a stochastic approach. Each hypotheses is generated by randomly choosing c samples from X and computing the model that fits this subset. Considering Assumption 4.3.1, the probability of one hypothesis to be composed only by inliers is $(p_{\text{in}})^c$. Therefore, the probability that one hypothesis contains at least one outlier is $1-(p_{\text{in}})^c$. Defining p_{suc} as the desired probability to generate one hypothesis using only inlier samples, then K must be

$$K = \frac{\log(1 - p_{suc})}{\log(1 - (p_{\text{in}})^c)}. \tag{4.9}$$

The larger p_{suc} is chosen, the better the algorithm will work but at the cost of having a larger K, which will imply more computations.

After that, the algorithm uses a voting system to rank all the hypotheses. Given a threshold $\tau > 0$, and an error function, we say that $x \in X$ is an inlier with respect to one hypothesis, h, if $e(x, h) \leq \tau$, and then votes for it

$$\text{vote}(x, h) = \begin{cases} 1, & e(x, h) \leq \tau, \\ 0, & e(x, h) > \tau. \end{cases} \tag{4.10}$$

The selection of τ depends on X and the model to fit. Small values of τ will return more accurate results but they might discard some possible inliers.

With all the hypotheses voted, RANSAC chooses the one with the most votes,

$$h^* = \arg \max_{k=1,\ldots,K} (\sum_{\mathbf{x} \in X} \text{vote}(\mathbf{x}, h_\ell)) \tag{4.11}$$

and determines a better model using the subset of inliers for h^*; i.e., $X_{h^*} = \{\mathbf{x} \in X \,|\, e(\mathbf{x}, h^*) \leq \tau\}$.

Considering again the distributed scenario, note that, even if all the robots had access to all the samples, RANSAC could not be used independently by each robot to detect the outliers because it is a non deterministic algorithm. Two different executions of the algorithm can return different results and therefore, additional mechanisms are required to ensure that all the robots achieve the same result.

4.4 *De-RANSAC*: Robust Consensus Algorithm

In this section we provide a distributed solution to detect the outliers and compute the robust consensus. The process is inspired by the RANSAC algorithm. In order to make it distributed, several issues need to be addressed:

- The first problem is the generation of the hypotheses by the network.
- The second issue to consider is how these hypotheses are voted for by all the robots.
- The last problem, once the best hypothesis has already been selected, is the computation of the final solution using only inlier information.

Along the section we explain how these issues are handled in our algorithm.

4.4.1 Distributed Generation of Random Hypotheses

The number of hypotheses required to have a probability of success equal to p_{suc} is the same as in the centralized case, eq. (4.9). However, each robot initially has only one sample, its own information, $\mathbf{x}_i(0)$. We will denote by $\mathcal{V}_{\text{in}} \subset \mathcal{V}$ the subset of robots with inlier information.

Following the RANSAC principles, we assume that each observation has equal probability of being a good observation p_{in} independent from the probability of the rest of observations. Therefore, the number of hypotheses, K, is exactly the same as in the centralized case, eq. (4.9). The value of K depends on the number of samples required to generate one hypothesis. Since we are considering the average as our consensus objective function, we define one hypothesis as the average of a subset of the observations. Then, the number of samples, c, can be chosen arbitrarily. We discuss the selection of this parameter in the simulations.

For one hypothesis, h, let $\varnothing \neq \mathcal{V}_h \subseteq \mathcal{V}$ be the subset of c robots whose observations generate the hypothesis. Given a fixed c, to build one hypothesis h we require c robots to belong to $\mathcal{V}_h$, $|\mathcal{V}_h| = c$. We use a max consensus algorithm with random initial conditions to decide which c robots form the subset.

Initially, each robot generates a random number $h_i > 0$ and the hypothesis set $\mathbf{h}_i(0) = \{h_i\}$, which is updated using

$$\mathbf{h}_i(t+1) = \max{}^c(\mathbf{h}_i(t) \bigcup_{j \in \mathcal{N}_i(t)} \mathbf{h}_j(t)), \tag{4.12}$$

where $\max^c$ selects the c maximum elements of the set. After the execution of (4.12), $\mathbf{h}_i$ will converge to the c maximum values of the network for all i. Assuming that each robot generates a different random number the subset $\mathcal{V}_h$ is established as

$$\mathcal{V}_h = \{i \in \mathcal{V} \mid h_i \in \mathbf{h}_i(d_v)\}. \tag{4.13}$$

Let us note that each robot knows if it belongs to $\mathcal{V}_h$ in a local way. The process can also be executed for all the hypotheses in parallel. Once the robots know if they should contribute with their observations to generate a specific hypothesis, they can initialize their value for that hypothesis using eq. (4.1).

The hypotheses are computed by the whole network using distributed averaging. We denote by $\mathbf{x}^h$ the average of the observations of the robots in $\mathcal{V}_h$. By Theorem 4.2.1, we can assure that all the robots will eventually have the value of each hypothesis.

4.4.2 Distributed Voting of the hypotheses

With all the hypotheses created, the robots must vote for them in order to select the best one. We propose a general procedure to vote the hypotheses based on distributed averaging.

In order to vote for the set of hypotheses, each robot initializes a voting vector, $\boldsymbol{v}_i \in \mathbb{R}^K, i = 1, \dots, N$, with as many elements as hypotheses to be voted for. Initially, for every $h = 1, \dots, K$, the h^{th} component of $\boldsymbol{v}_i$, is specified as

$$v_i^h(0) = \text{vote}(\mathbf{x}_i, h) = \begin{cases} 1, & \|\mathbf{x}_i - \mathbf{x}^h\| \leq \tau, \\ 0, & \|\mathbf{x}_i - \mathbf{x}^h\| > \tau, \end{cases} \tag{4.14}$$

where $\tau > 0$ is the threshold in RANSAC to determine the votes. We are considering the Euclidean distance as the error function to assign the votes but other functions depending on the context could be used, e.g., the Mahalanobis distance.

After this, the vote vectors are updated using distributed averaging of the initial values of the $\boldsymbol{v}_i$.

Proposition 4.4.1 (Distributed Voting in Finite Time). *Given initial conditions as in* (4.14)*, the rule* (4.3) *with $g(\boldsymbol{v}) = N\boldsymbol{v}$ converges to the sum of the initial conditions*

$$\arg \min_{\boldsymbol{v} \in \mathbb{N}} \|\boldsymbol{v} - N\boldsymbol{v}_i(t)\|_2 \rightarrow \sum_{i \in \mathcal{V}} \boldsymbol{v}_i(0),$$

which is the total number of votes for each hypothesis, in finite time.

Once the iteration has converged, the largest entry of $\boldsymbol{v}_i$ will correspond to the hypothesis with the most number of votes. Since the hypotheses are sorted in the vector, if there exists a tie between two or more hypotheses every robot will keep the one with the smallest index,

$$h^* = \min(\arg \max_{h=1,\dots,K} v_i^h), \tag{4.15}$$

4.4.3 Consensus of the Inliers

The last step in the RANSAC algorithm is a refinement of the solution considering the set of measurements that voted the best hypothesis. This way the algorithm obtains a final consensus erasing the influence of the spurious information.

Let $\mathcal{V}_{\text{in}} = \{i \in \mathcal{V} \mid \|\mathbf{x}_i - \mathbf{x}^{h^*}\| \leq \tau\} \subseteq \mathcal{V}$ be the set of inlier robots, which is the set that must contribute to the final consensus value. The desired final value is

$$\mathbf{x}_i(t) \rightarrow \frac{1}{|\mathcal{V}_{\text{in}}|} \sum_{j \in \mathcal{V}} \mathbf{x}_j(0),\ \forall i \in \mathcal{V}. \tag{4.16}$$

We can use again the averaging rule using homogeneous coordinates to compute the consensus of the inlier robots. If each robot initializes its extended measurement with the following rule,

$$\mathbf{x}_i^e(0) = \begin{cases} [\mathbf{x}_i(0)^T,\ 1]^T \text{ if } i \in \mathcal{V}_{\text{in}} \\ \mathbf{0}^T \text{ otherwise} \end{cases}, \tag{4.17}$$

then, applying the standard averaging, eq. (2.2), and normalizing by the homogeneous coordinate, by Theorem 4.2.1 we assure that the average of the inliers is obtained by the whole network.

4.4.4 Analysis of the Algorithm

De-RANSAC is summarized in Algorithm 4. We compare now our approach with the centralized version of RANSAC and provide an analysis of the complexity of the algorithm in terms of computations and communications.

Algorithm 4 *De-RANSAC* scheme - Robot i

```
Ensure: Computation of the average of the data fitting the most voted hypothesis
 1: Compute $K = \frac{\log(1-p_{suc})}{\log(1-p_{in}^c)}$
 2: – Do this process in parallel for the K hypotheses
 3: Generate a random number $h_i$
 4: Compute $\mathcal{V}_h$ by $\mathbf{h}_i(t+1) = \max^c(\mathbf{h}_i(t) \bigcup_{j \in \mathcal{N}_i(t)} \mathbf{h}_j(t))$
 5: Initialize the hypothesis using eq. (4.1)
 6: Compute the hypothesis using distributed averaging
 7: Initialize $v_i^h(0)$,
        $$v_i^h(0) = \text{vote}(\mathbf{x}_i, h) = \begin{cases} 1, & \|\mathbf{x}_i - \mathbf{x}^h\| \le \tau, \\ 0, & \|\mathbf{x}_i - \mathbf{x}^h\| > \tau, \end{cases}$$
 8: Decentralized voting
        $$v_i^h(t+1) = w_{ii}(t)v_i^h(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)v_i^h(t)$$
 9: Select $h^* = \min(\arg\max_{h=1,\ldots,K} v_i^h)$
10: – At this point only one hypothesis remains
11: Initialize the final consensus value
        $$\mathbf{x}_i^e(0) = \begin{cases} [\mathbf{x}_i(0)^T,\ 1]^T \text{ if } \|\mathbf{x}_i - \mathbf{x}^{h^*}\| \le \tau \\ \mathbf{0}^T \text{ otherwise} \end{cases}$$
12: Compute the consensus value
        $$\mathbf{x}_i^e(t+1) = w_{ii}(t)\mathbf{x}_i^e(t) + \sum_{j \in \mathcal{N}_i} w_{ij}(t)\mathbf{x}_j^e(t),$$
13: Normalize the consensus value dividing by $x_i^h$
```

If we compare *De-RANSAC* with the centralized RANSAC, the properties of both algorithms are the same. For the same input parameters and number of hypotheses both algorithms have the same probability of computing a good solution and detect all the outliers because they rely on the same assumptions and execute the same steps.

Regarding the computational complexity of our approach, note that the distributed algorithm does not require heavy computational requirements. The three steps of the algorithm require standard averaging executions. Therefore, the algorithm is suitable for low computation capabilities.

In terms of communications the algorithm is more demanding than the standard averaging consensus because it requires 3 different computations of average values (hypotheses, votes and inliers). The size of the messages is also bigger than for a general averaging procedure. The hypotheses generation computes K different averages in parallel, then in this step the size of the messages is K times the size of the standard messages. The voting procedure computes the average of a K dimensional vector and the consensus on the inliers requires one additional component in the vectors, the homogeneous coordinate. Nevertheless, the amount of extra communications of the algorithm is justified by the robustness provided by RANSAC, essential in robotic applications.

4.5 *De-RANSAC* using Dynamic Voting

The previous approach requires three different consensus steps, one for the generation of the hypotheses, another one for the distributed voting process and a last one to compute the ML of the inlier observations. Using dynamic consensus techniques [163] we can execute the three steps of *De-RANSAC* at the same time, reducing the times distributed averaging needs to be executed from three to just one. We explain the process for the problem of computing the maximum likelihood (ML) of a set of uncertain measurements. We will show that, if one hypothesis is generated only using robots in $\mathcal{V}_{\text{in}}$, then the ML of the inlier robots will be obtained by the algorithm in only one consensus step.

We consider that each robot has a noisy initial measurement, $\mathbf{x}_i \in \mathbb{R}^{d_1}$, with uncertainty contained in the symmetric, semi-definite positive covariance matrix $\mathbf{\Lambda}_i \in \mathbb{R}^{d_1 \times d_1}$. The maximum likelihood (ML), $\boldsymbol{\theta}_{\text{ML}}$, is estimated using a weighted least-squares approximation from the robot measurements as

$$\boldsymbol{\theta}_{\text{ML}} = \left(\sum_{i=1}^{N} \mathbf{\Lambda}_i^{-1} \right)^{-1} \sum_{i=1}^{N} \mathbf{\Lambda}_i^{-1} \mathbf{x}_i. \tag{4.18}$$

Since there are some observations that are outliers, we need to discard them from the computation of the ML. We follow again the steps of RANSAC to detect and discard the outliers, i.e., generate a set of hypotheses, vote for them and pick the one with the most votes. However, we are going to execute the three steps at the same time. The difference now is the use of a dynamic voting approach where the robots vote for (or not) a hypothesis when their observations pass (or not) a distance test at each iteration t. In order to make the presentation clearer, we describe the algorithm just for one hypothesis, omitting the superscript h.

The robots that initially configure the hypothesis are chosen as in the static version of *De-RANSAC*. In this case, the hypothesis is defined as the ML of a subset of the observations,

$$\boldsymbol{\theta}_{\text{ML}} = \left(\sum_{i \in \mathcal{V}_h} \mathbf{\Lambda}_i^{-1} \right)^{-1} \sum_{i \in \mathcal{V}_h} \mathbf{\Lambda}_i^{-1} \mathbf{x}_i. \tag{4.19}$$

Proposition 4.5.1 (**Distributed Computation of a Partial ML**). *Let the variables* $[\boldsymbol{P}_i(0), \boldsymbol{q}_i(0)]$ *be initialized by*

$$[\boldsymbol{P}_i(0), \boldsymbol{q}_i(0)] = \begin{cases} [\mathbf{\Lambda}_i^{-1}, \mathbf{\Lambda}_i^{-1} \boldsymbol{x}_i] \textit{ if } i \in \mathcal{V}_h \\ [\boldsymbol{0}, \boldsymbol{0}] \textit{ otherwise} \end{cases} . \tag{4.20}$$

Then, the variable $\boldsymbol{\theta}_i(t) = (\boldsymbol{P}_i(t))^{-1} \boldsymbol{q}_i(t)$, *with updates of* $\boldsymbol{P}$ *and* $\boldsymbol{q}$ *using* (2.2)*, asymptotically converges to* (4.19) *for all* $i \in \mathcal{V}$.

This means that the network is able to compute partial maximum likelihoods of different subsets of the robots in a decentralized way. The covariance associated to the partial ML will be $\mathbf{\Lambda}_{\text{ML}} = \frac{1}{N} \lim_{t \to \infty} (\mathbf{P}_i(t))^{-1}$.

In contrast with the static voting approach, now the voting process starts at the same time as the hypotheses generation, eq. (4.20). The initial votes are

$$v_i(0) = \begin{cases} 1, & \text{if } i \in \mathcal{V}_h, \\ 0, & \text{otherwise.} \end{cases} \tag{4.21}$$

Instead of waiting until the computation of the partial ML, the new local updates for each robot are

$$\begin{aligned}\mathbf{P}_i(t+1) &= w_{ii}(t)\mathbf{P}_i(t) + \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\mathbf{P}_j(t) + \mathbf{u}_i^{\mathbf{P}}(t),\\ \mathbf{q}_i(t+1) &= w_{ii}(t)\mathbf{q}_i(t) + \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\mathbf{q}_j(t) + \mathbf{u}_i^{\mathbf{q}}(t),\\ \upsilon_i(t+1) &= w_{ii}(t)\upsilon_i(t) + \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\upsilon_j(t) + \mathbf{u}_i^{\upsilon}(t),\end{aligned} \tag{4.22}$$

where $\mathbf{u}_i^{\mathbf{P}}(t), \mathbf{u}_i^{\mathbf{q}}(t)$ and $\mathbf{u}_i^{\upsilon}(t)$ are the dynamic inputs in the consensus rule.

In order to decide the inputs, each robot executes the Chi-square test with the current value of $\boldsymbol{\theta}_i(t) = \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t)$. With a slightly abuse of notation, we denote this test by

$$\chi(\mathbf{x}_i, \boldsymbol{\theta}_i(t), \boldsymbol{\Lambda}_i) = \chi_i(t) = \begin{cases} 1, & \text{if } \sqrt{(\mathbf{x}_i - \boldsymbol{\theta}_i(t))^T(\boldsymbol{\Lambda}_i)^{-1}(\mathbf{x}_i - \boldsymbol{\theta}_i(t))} \leq \chi^2_{d_1,p},\\ 0, & \text{otherwise },\end{cases} \tag{4.23}$$

where $\chi^2_{d,p}$ is the value of the Chi square distribution for d_1 degrees of freedom and confidence probability p.

There are two issues to comment about this test. Firstly, for the time instants t for which $\mathbf{P}_i^{-1}(t)$ does not exist, we cannot compute $\boldsymbol{\theta}_i(t)$. At these instants we assign $\chi_i(t) = 0$. Secondly, although this test is very similar to compute the Mahalanobis distance, which defines how correlated two stochastic variables are, it is not exactly the same. We are omitting in our distance function the covariance associated to $\boldsymbol{\theta}_i(t)$, $\boldsymbol{\Lambda}_i(t) = (\mathbf{P}_i(t))^{-1}$, which implies that our distance is going to be larger than the Mahalanobis distance. We have chosen this conservative solution because in the first iterations of the algorithm the estimation of $\mathbf{P}_i(t)$ is highly unreliable. Usually, at these times $\mathbf{P}_i(t)$ is multiplied by weights very close to zero, resulting in large covariances due to the inverse $\mathbf{P}_i^{-1}(t)$. When this happens, the Mahalanobis distances are close to zero and spurious votes can appear in the algorithm, whereas by omitting this term the problem is avoided.

Denote the set of time instants in which robot i changes its opinion as follows:

$$\begin{aligned}\mathcal{T}_i^+ &= \{t \in \mathbb{N} \mid \chi_i(t) = 1 \ \wedge \ \chi_i(t-1) = 0\},\\ \mathcal{T}_i^- &= \{t \in \mathbb{N} \mid \chi_i(t) = 0 \ \wedge \ \chi_i(t-1) = 1\}.\end{aligned} \tag{4.24}$$

The control inputs of robot i are given by:

$$[\mathbf{u}_i^{\mathbf{P}}(t), \mathbf{u}_i^{\mathbf{q}}(t), \mathbf{u}_i^{\upsilon}(t)] = \begin{cases} [\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, 1] & \text{if } t \in \mathcal{T}_i^+,\\ -[\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, 1] & \text{if } i \in \mathcal{T}_i^-,\\ [\mathbf{0}, \mathbf{0}, 0] & \text{otherwise.}\end{cases} \tag{4.25}$$

Proposition 4.5.2 (Convergence of the Dynamic Voting Consensus). *If $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ are finite for all $i \in \mathcal{V}$ then the rule* (4.22) *with control inputs* (4.25) *converges to*

$$\lim_{t\to\infty} \boldsymbol{\theta}_i(t) = (\sum_{j\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_j^{-1})^{-1} \sum_{j\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_j^{-1}\boldsymbol{x}_j, \tag{4.26}$$

$$\lim_{t\to\infty} \upsilon_i(t) = \frac{|\mathcal{V}_{\text{con}}|}{N}. \tag{4.27}$$

where $\mathcal{V}_{\text{con}} = \{i \in \mathcal{V} \mid \chi_i(t) = 1,\ t > t_{\max}\}$ and $t_{\max}$ is a time instant such that $t_{\max} > t, \forall t \in \mathcal{T}_i^+, \mathcal{T}_i^-, \forall i \in \mathcal{V}$.

If the robots are not indefinitely changing their vote, then the algorithm will achieve convergence to the ML of the subset of robots that have voted for it. At the end, the hypothesis with the larger value of υ_i^h will be the one selected by all the robots as the good one. Note that with this approach, once the hypothesis is selected there is no need to compute additional ML estimates. What remains to be done now is to determine conditions that guarantee the convergence to the ML of the inliers.

Conditions to reach the Maximum Likelihood of the inliers

We derive a set of reasonable conditions such that, if $\mathcal{V}_h \subseteq \mathcal{V}_{\text{in}}$ for one hypothesis, then the assumptions in Proposition 4.5.2 are met and $\mathcal{V}_{\text{con}} = \mathcal{V}_{\text{in}}$ for that hypothesis.

First, we impose a condition on the inlier observations. Since they are different measurements of the same vector, they have to be close to each other.

Condition 1. *It holds that $\chi(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\Lambda}_i) = 1$ for any pair of robots, $i, j \in \mathcal{V}_{\text{in}}$.*

Lemma 4.5.3 (Bounded Distances in the Convex Hull). *Let $\text{CH}(\mathcal{V}_{\text{in}})$ be the convex hull of the inlier observations. If Condition 1 is satisfied, then, for any robot $i \in \mathcal{V}_{\text{in}}$ and any $\boldsymbol{x} \in \text{CH}(\mathcal{V}_{\text{in}})$, we have $\chi(\boldsymbol{x}_i, \boldsymbol{x}, \boldsymbol{\Lambda}_i) = 1$.*

This means that we have a set of points voted for by all the inliers, which leads to a second condition

Condition 2. *For any $\mathcal{V}_h \subseteq \mathcal{V}_{\text{in}}$, $\boldsymbol{\theta}^h_{\text{ML}} \in \text{CH}(\mathcal{V}_{\text{in}})$.*

The lemma also suggests a restriction to impose to the outlier observations.

Condition 3. *For all $\boldsymbol{x} \in \text{CH}(\mathcal{V}_{\text{in}})$ and $k \notin \mathcal{V}_{\text{in}}$ it holds that $\chi(\boldsymbol{x}_k, \boldsymbol{x}, \boldsymbol{\Lambda}_k) = 0$.*

However, let us note that because our algorithm is not convex, the above conditions are not enough to ensure that one hypothesis instantiated with inliers will end up with all the inliers voting for it and all the outliers rejecting it. It could be possible that some outliers vote for it at some intermediate estimation or that one or more inliers constantly change their vote and convergence does not occur.

An additional condition to ensure convergence to the desired result is imposed. First let us notice that $\boldsymbol{\theta}_i(t)$ and the error distance, $\chi_i(t)$, can be written as functions of a vector $\mathbf{w} = (w_1, \ldots, w_N) \in [0,1]^N$, which represents the weights of the linear combination (not necessarily convex) of the different observations.

$$\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{P}_i^{-1}(\mathbf{w})\mathbf{q}_i(\mathbf{w}) = (\sum_{i\in\mathcal{V}} w_i \boldsymbol{\Lambda}_i^{-1})^{-1}(\sum_{i\in\mathcal{V}} w_i \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i),$$

$$\chi_i(\mathbf{w}) = \chi(\mathbf{x}_i, \boldsymbol{\theta}_i(\mathbf{w}), \boldsymbol{\Lambda}_i) = \begin{cases} 1, & \text{if } \sqrt{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \boldsymbol{\Lambda}_i^{-1}(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))} \le \chi^2_{d_1,p}, \\ 0, & \text{otherwise} , \end{cases}$$

Both functions are well defined for any $\mathbf{w} \neq \mathbf{0}$. However, let us recall that if $\mathbf{w} = \mathbf{0}$, we have defined $\chi_i(\mathbf{w}) = 0$. The values of the different w_i are hard to compute as a function of t because they depend on the weights $w_{ij}(t)$ in eq. (4.22) and the network topology at each time instant. However, we can analyze the behavior of $\chi_i(\mathbf{w})$ over a compact subset of $[0,1]^N$. If the behavior in the set is the desired one, we will be able to ensure that for any t the algorithm will return the desired results.

Without loss of generality, let us assume that the robots are ordered so that we can separate the different elements of $\mathbf{w}$ in $\mathbf{w}_{\text{in}} \in [0,1]^{|\mathcal{V}_{\text{in}}|}$, the components corresponding to the inliers and $\mathbf{w}_{\text{out}}$ the components of the outliers, $\mathbf{w} = [\mathbf{w}_{\text{in}}, \mathbf{w}_{\text{out}}]$.

Condition 4 (Local maxima of the derivative). *For any $i \in \mathcal{V}$, the partial derivatives of the square root in $\chi_i(\boldsymbol{w})$,*

$$\frac{\partial \sqrt{(\boldsymbol{x}_i - \boldsymbol{\theta}_i(\boldsymbol{w}))^T \boldsymbol{\Lambda}_i^{-1}(\boldsymbol{x}_i - \boldsymbol{\theta}_i(\boldsymbol{w}))}}{\partial w_j} = \frac{(\boldsymbol{x}_i - \boldsymbol{\theta}_i(\boldsymbol{w}))^T \boldsymbol{\Lambda}_i^{-1} \boldsymbol{P}_i^{-1}(\boldsymbol{w}) \boldsymbol{\Lambda}_j^{-1}(\boldsymbol{x}_j - \boldsymbol{\theta}_i(\boldsymbol{w}))}{\chi_i(\boldsymbol{w})}, \tag{4.28}$$

satisfy, $\forall j \in \mathcal{V}$ and $\boldsymbol{w}_{\text{out}} = \boldsymbol{0}$, that

$$\frac{\partial \sqrt{(\boldsymbol{x}_i - \boldsymbol{\theta}_i(\boldsymbol{w}))^T \boldsymbol{\Lambda}_i^{-1}(\boldsymbol{x}_i - \boldsymbol{\theta}_i(\boldsymbol{w}))}}{\partial w_j} = 0 \Leftrightarrow \begin{cases} \boldsymbol{\theta}_i(\boldsymbol{w}) = \boldsymbol{x}_i, \boldsymbol{x}_j, & \textit{if } i,j \in \mathcal{V}_{\text{in}}, \\ \boldsymbol{\theta}_i(\boldsymbol{w}) = \boldsymbol{x}_i, & \textit{if } i \in \mathcal{V}_{\text{in}}, j \notin \mathcal{V}_{\text{in}}, \\ \boldsymbol{\theta}_i(\boldsymbol{w}) = \boldsymbol{x}_j, & \textit{if } i \notin \mathcal{V}_{\text{in}}, j \in \mathcal{V}_{\text{in}}. \end{cases} \tag{4.29}$$

The derivation of (4.28) is included with the proofs of the theoretical results at the end of the chapter.

Theorem 4.5.4 (Convergence to the ML of the Inliers). *Under Conditions 1-4, for any $\mathcal{V}_h \subseteq \mathcal{V}_{\text{in}}$, the following holds:*

- *All the inliers eventually vote for the hypothesis*

$$\exists\, t^* \mid \forall t > t^*, \forall i \in \mathcal{V}_{\text{in}},\ \chi_i(t) = 1. \tag{4.30}$$

- *The outliers do not vote for the hypothesis at any time*

$$\chi_k(t) = 0,\ \forall t > 0, \forall k \notin \mathcal{V}_{\text{in}}. \tag{4.31}$$

This means, by Proposition 4.5.1 that (4.22) *will converge and that* $\mathcal{V}_{\text{con}} = \mathcal{V}_{\text{in}}$.

Let us note that the conditions to know with certainty that the algorithm will converge to the desired result are relatively easy to occur in real scenarios. The first condition requires that the inlier observations are close to each other, which is easy to happen because they are good measurements of the same vector.

The second condition is that the maximum likelihood of partial sets of inliers falls in the convex hull of the inliers. For a very small number of inliers (2 or 3) this will be hard to be true, but for a larger number of inliers this condition is almost sure to happen because the ML is similar to a weighted average.

The third condition requires the outliers to be far away from the convex hull of the inliers. This makes sense, otherwise one may argue that they are not real outliers as they would be posing a good observation of the feature.

The last condition is that the derivatives of the Mahalanobis distance only vanish at the points that correspond to the inlier observations. For the observation of any robot i, a global minimum is obtained when $\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i$, with distance equal to zero. For the inliers this is not a problem because they must vote the hypotheses. For the outliers, it would be a problem that this happened, but it is almost impossible that a combination of inlier observations satisfying Conditions 1-3 returns the outlier. A global extreme can also appear if there exists $\mathbf{w}$ such that $(\mathbf{x}_k - \boldsymbol{\theta}(\mathbf{w}))$ is orthogonal to all $(\mathbf{x}_i - \boldsymbol{\theta}(\mathbf{w}))$ with respect to $\boldsymbol{\Lambda}_k^{-1}\mathbf{P}_k^{-1}(\mathbf{w})\boldsymbol{\Lambda}_i^{-1}$. Nevertheless, we have not encountered this situation in any of the simulations we have carried out and, provided that this extreme satisfies that $\chi_k(\mathbf{w}) = 0$, the algorithm would still converge.

Finally, let us remark that the algorithm may still converge to the desired result if some of these conditions are not met. In the next section we analyze this situation.

4.6 Applications of *DE-RANSAC*

De-RANSAC can be used in different consensus scenarios. In this section we present some possible applications of the algorithm using simulated data and real images.

4.6.1 Distributed Robust Event Localization

As an example of how *De-RANSAC* works, let us consider a sensor network composed by 40 nodes and deployed in a square environment like the one in Fig. 4.2. The goal of the network is to detect and localize events of interest that take place in the area, e.g., fires or intruders. In this example we consider that there are two simultaneous events localized at coordinates $(-40, -40)$ and $(70, 30)$ respectively.

Each sensor is only able to measure one of the events with some noise in the localization of the event. In addition, there might be some sensors that place the event in a totally random position due to failures in their measurement. Moreover, there is no way for the nodes to know which of the events they are measuring. In Figure 4.2 we show the measurement taken by each sensor. There are 20 sensors that measure the first

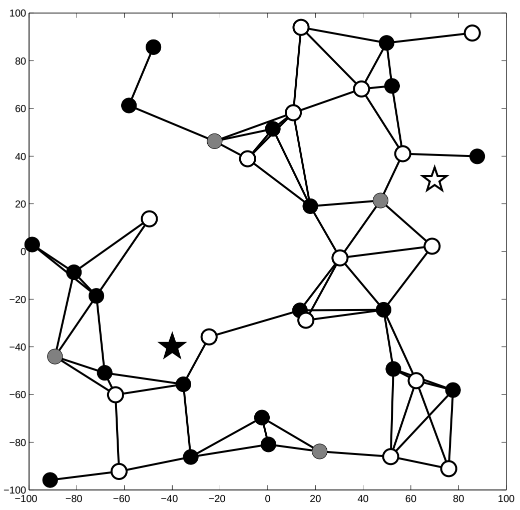


Figure 4.2: Sensor network deployed in the environment to detect events of interest. There are 20 sensors that measure the first event (black circles and black star), 16 sensors that measure the second event (white circles and white star) and 4 sensors that measure a random value (grey circles).

event (black circles and black star), 16 sensors that measure the second event (white circles and white star) and 4 sensors that measure a random value (grey circles). If all the measurements are mixed using distributed averaging, then the mean position of the event is placed at $(3.72, -13.85)$, which is a wrong estimation. Next we show how our algorithm can overcome this problem.

Initially the nodes do not know how many other sensors are active due to failures, initialization, etc. Figure 4.3 shows the evolution on the consensus about the number of active nodes using the finite-time rule. The final value is 40, the exact number of active sensors, for all the nodes in the network.

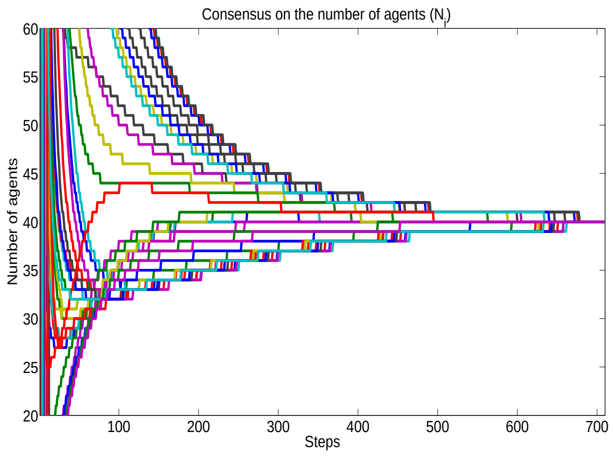


Figure 4.3: Finite-time consensus about the number of active sensors.

The hypotheses are defined as possible locations of the event of interest. We generate the hypotheses computing the average location of sets of two samples, i.e., $c = 2$. We have run two simulations considering two different values of the probability of a sensor having inlier information, p_{in}. In the first one we have set p_{in} to 0.8 and in the second to 0.3. The probability chosen to decide the total number of hypotheses, eq. (4.9), in order to be successful in creating one good hypothesis has been taken to be $p_{suc} = 0.99$. The threshold for voting one hypothesis considers the Euclidean distance between the location of the hypothetical event and the sensor observation of the event the hypothesis has been set to 2 meters.

The first column of Figure 4.4 shows the evolution of the number of votes of the best hypothesis using the finite-time consensus rule and the achievement of the final consensus value (X and Y coordinates). The best hypothesis gets a total of 20 positive votes, which is exactly the number of sensors perceiving the event located in $(-40, 40)$. Regarding the final consensus value, considering only the 20 inliers, we observe that using the homogeneous coordinates all the 40 sensors reach the same value, the correct location of the event.

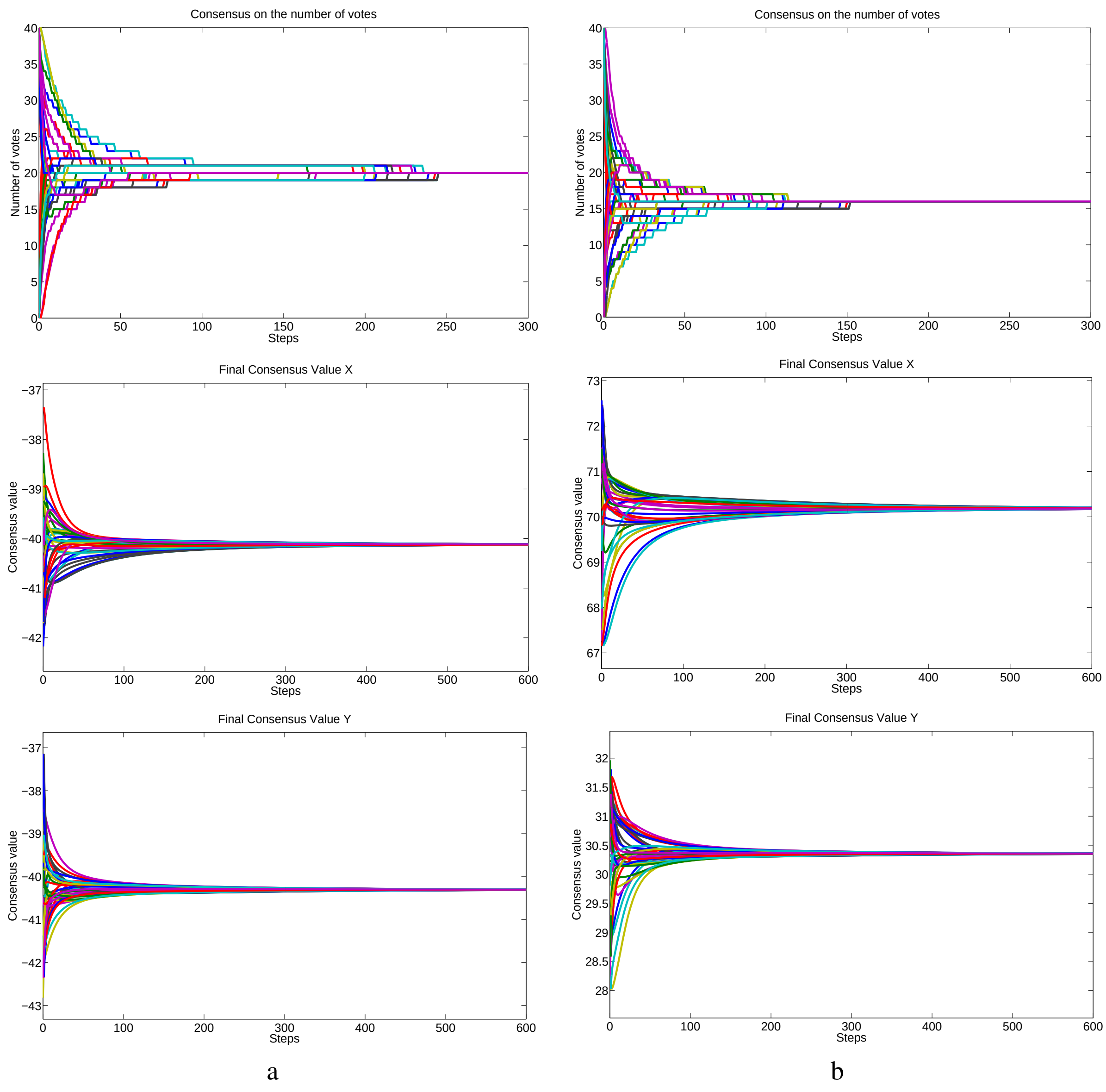


Figure 4.4: Consensus in the different steps of *De-RANSAC* for the robust event localization. The first column (a) shows, from top to bottom, the finite-time consensus on the number of votes of the best hypothesis (20 votes) and the final consensus value (X and Y coordinates) mixing only the inlier measurements, using the normalized coordinates. The second column (b) shows the same values for the next best hypothesis. In this case the number of votes is 16 and the final consensus value is the location of the second event of interest.

More detailed results of the two simulations are in Table 4.1. The Avrg column shows the location of the event mixing the information of all the sensors, which is equal and wrong in both cases because of the influence of the outliers. Rob_Av represents the value of the most voted hypotheses. Since *De-RANSAC* is non deterministic, this value differs for different executions. Nevertheless, let us notice that in both cases the distance with respect to the real value $(-40, -40)$ is smaller than the tolerance fixed. Fin_Av shows the consensus value considering only the inlier nodes. In this case both experiments return the same result (the average of the inliers) because all the inliers have been identified and they are computing the average of the same set of measurements. K is the number of hypotheses created. By choosing a bigger value of p_{in} we can reduce the total number of hypotheses, and therefore, of required communications.

Table 4.1: Results of the Event Localization using De-RANSAC

Case	p_{in}	Avrg	Rob_Av	Fin_Av	Votes	K
1	0.8	(3.72, -13.85)	(-40.36, -38.66)	(-40.12, -40.3)	21	7
2	0.3	(3.72, -13.85)	(-39.27, -39.91)	(-40.12, -40.3)	21	74

Another advantage of the evaluation of multiple hypotheses with *De-RANSAC* is that without additional computations we can identify the position of the second event of interest, located in $(70, -30)$. Once the best hypothesis has been identified, the sensors select the next most voted hypothesis such that:

- They did not vote for it if they were inliers.
- They voted for it if they were outliers.

The number of votes and the average of the positive votes for this hypothesis are shown in the second column of Figure 4.4. We can see that in this case the number of votes is 16 and the average of the positive votes for this hypothesis reaches the correct value.

4.6.2 People Identification in Camera Networks

We reintroduce the problem of face recognition in multiple images first discussed in the previous chapter. We show how *De-RANSAC* can be applied as a second step in the procedure to improve the initial data association.

Figure 4.5 shows the initial correspondences of the network found using our data association distributed algorithm. In this example there are no inconsistencies and all the cameras are able to assign one face to each association set. However, there are some outlier matches, e.g., faces 2 and 4 have been wrongly matched in cameras A and B. We can use *De-RANSAC* to discard mismatches and improve the data association.

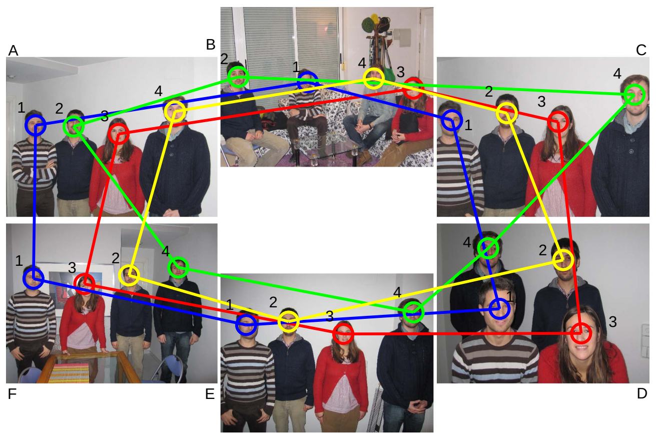


Figure 4.5: Initial People Association in Camera Networks.

Since *De-RANSAC* considers only one datum per node, the algorithm is executed in parallel for each of the association sets given by the data association, represented with different colors in Fig. 4.5. We define one hypothesis as a weighted average of two patches. Assuming the initial data association is relatively good, we set p_{in} to 0.9, which implies that each camera only generates one hypothesis. The error function to detect outliers is the same function used to match the faces (absolute differences between pairs of patches) and the threshold has been empirically set considering the errors obtained comparing patches in the data association step.

The result of applying *De-RANSAC* is shown in Fig. 4.6. The algorithm has been able to detect that faces 2 and 4 were outliers in cameras A and B and has removed these matches. Also, in faces 1 and 3 the algorithm has not removed any match. On the other hand, the algorithm has also considered as an outlier face 4 in camera C, which was not. This happens because of choosing a restrictive threshold for the voting. A possible way to improve this could be to use a smaller threshold in the error function to vote for the hypotheses. However, in a real application the lack of spurious matches prevails over missing some good correspondences.

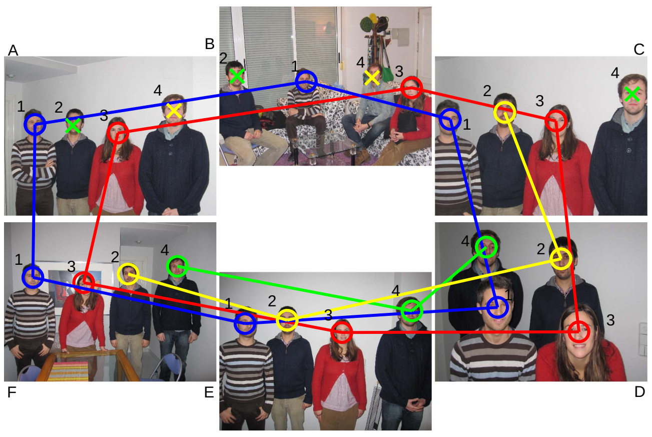


Figure 4.6: Robust People Association in Camera Networks.

4.6.3 Maximum Likelihood Estimation using Dynamic Voting

We have also tested the dynamic version of the algorithm for the computation of the maximum likelihood in a simulated environment. In this case we have considered a robotic network composed by ten robots with communications as described in Fig. 4.7.

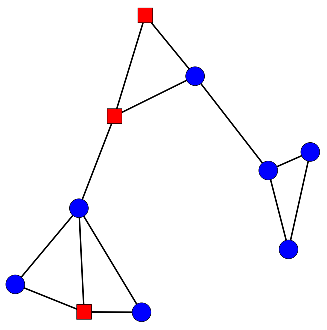
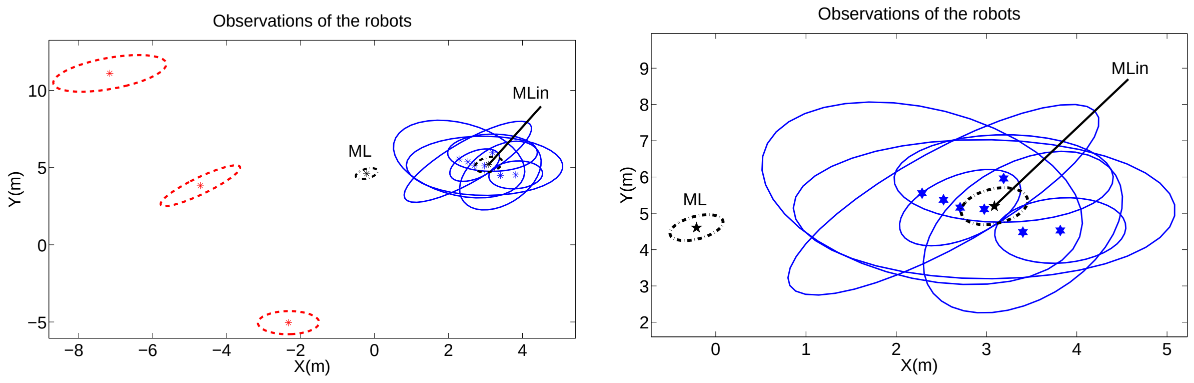


Figure 4.7: Communication network of the ten robots. Blue circles represent the robots with inlier information and red squares the outliers.

In Fig. 4.8, we show the observations of a two-dimensional feature by the ten robots. We have chosen a two-dimensional feature in order to have a good visualization of the results, however, the algorithm is not restricted to this case and can be used with descriptors of any dimension. Seven robots have good observations of the feature (blue crosses and solid ellipses) and 3 robots have outlier information (red crosses and dashed ellipses). If all the measurements are considered in the computation of the Maximum Likelihood, the obtained result is the black cross and dash-dotted ellipse with the ML mark at value $(-0.21, 4.60)$ while the ML of the inlier robots is $(3.08, 5.19)$ (MLin).

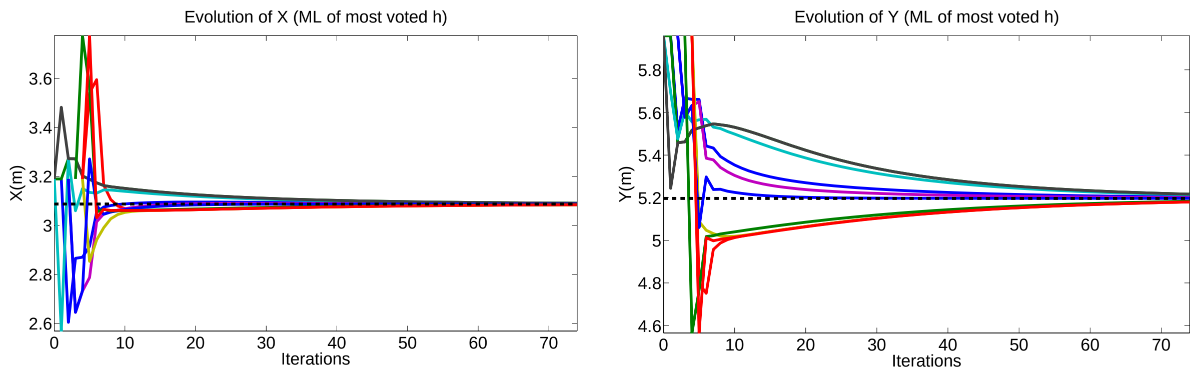


(a) Observations of the robots (b) Observations of the inliers (zoom)

Figure 4.8: Robust sensor fusion of a bi-dimensional feature observed by ten robots. (a) Observations of the ten robots. Seven robots have a good observation of the feature (blue crosses and solid ellipses) whereas three robots have observed another features (red crosses and dashed ellipses). If all the measurements are considered in the computation of the Maximum Likelihood, the obtained result is the black cross and dash-dot ellipse with the ML mark. The good maximum likelihood is the one in the middle of the inlier observations (MLin). (b) Zoom of the inlier observations and both ML.

In the first step, the robots generate the different subsets $\mathcal{V}_h$ that will initialize the hypotheses. In the experiment we have set the probability of being an inlier to 0.6 and the probability of success in RANSAC to 0.99. As we mentioned in Section 4.4.1, the value of c can be arbitrarily chosen. Larger values of c make many robots to plug their observations at the beginning, which is good if the number of inliers is large. However, the larger the c, the more hypotheses will be required to succeed, and for each additional hypothesis the amount of information to be communicated among robots grows linearly. For this reason we have set $c = 1$ resulting in a total of 6 hypotheses generated by the algorithm. In this example, the conditions stated in section 4.5 are also satisfied, ensuring convergence to the ML of the inliers if one hypothesis is instantiated by robots in $\mathcal{V}_{\text{in}}$.

In Fig. 4.9, we show the evolution of the two coordinates of $\boldsymbol{\theta}_i(t)$ for the most voted hypothesis. The values of the different robots converge to the value of the ML of all the inliers (depicted in black dashed line in the graphics). In Fig. 4.10, the evolution of $\boldsymbol{v}_i(t)$ for the same hypothesis is depicted. Eventually all the robots



(a) (b)

Figure 4.9: Evolution of the Maximum Likelihood, $\boldsymbol{\theta}_i(t)$, for the hypothesis that has obtained the most number of votes in the end. The dashed black line is the value of the ML of the robots with inlier information. It is observed that in both coordinates the values of $\boldsymbol{\theta}_i(t)$ asymptotically converge to it.

reach the value 0.7, which is exactly the fraction of robots with inlier information. It is also remarkable that the number of iterations in which the robots change their opinion is considerably small. In less than 10 iterations, the graphics do not have discontinuities due to the inputs (4.22). After that point, the algorithm behaves as a static consensus algorithm.

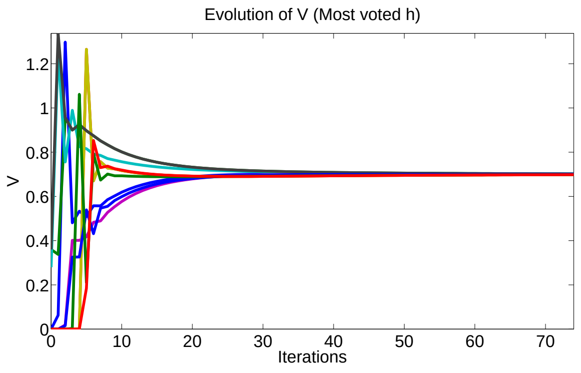


Figure 4.10: Evolution of $v_i^{h^*}$. It converges to 0.7 for all the robots, which is exactly the fraction of robots with inlier information.

We have also run a Monte Carlo simulations considering more general situations where the conditions of Section 4.5 do not always hold. We have compared the results of the new algorithm with the robust consensus based on static voting and with the distributed consensus algorithm proposed in [155] to compute the ML of all the observations. We have run 1000 trials in which 20 robots have been considered. The probability for each robot to have inlier information has been set to 0.8, and only 3 hypotheses were generated in each trial. The inlier robots have measurements of the feature with gaussian error of zero mean and standard deviation of 2 meters. For the outlier robots we have assigned a deviation of 10 meters. The covariance matrices have also been randomly generated with eigenvalues of mean 0.5 and standard deviation 0.5. Regarding the communication topology we have changed it at each iteration of each trial.

The results obtained in the simulation can be seen in Table 4.2. The results using the non robust algorithm [155] are in the first column. In this case all the outliers participate in the different trials (a total of 4015). As a consequence, the average error in the estimation of the ML is large (2.06 meters). If the static voting method is used, the results are clearly improved. Only 375 false positive votes appear and the average error is reduced to 0.444 meters, with only 472 inliers thinking they have outlier information. Finally, if the proposed algorithm is used, 355 false positive votes are counted and the average error is of 0.438 meters. Moreover, since the robots are voting a dynamic observation which tends to the good ML, a fewer number of false negatives is registered (192). Although the results are similar to those of the static voting, the dynamic algorithm is much faster because it requires one third of communication rounds. The dynamic voting algorithm condenses the three steps of RANSAC in only one, requiring 100 iterations per trial. The only drawback of the robust algorithms with respect to [155] is that the size of the messages grows linearly with the number of hypotheses. However, this limitation is also found in the centralized version of RANSAC, where for each hypothesis a different solution must be computed and voted for.

4.7 Discussion

Throughout this chapter we have analyzed in detail the problem of robustness in the computation of the consensus. We have proposed a new algorithm, *De-RANSAC*, able to detect and discard outlier measurements during the computation of the average consensus. The algorithm follows the philosophy of the RANSAC algorithm, a very common algorithm for robust matching between pairs of images. However, our algorithm is designed to be fully distributed, and then suitable for teams of robots with limited communications.

Table 4.2: Comparison of the different algorithms

Method	ML [155]	Static Voting	Dynamic Voting
Trials	1000	1000	1000
False Positive Votes	4015	375	355
False Negative Votes	0	472	192
Avrg. norm of error	2.063	0.444	0.438
Std. deviation of error	2.502	1.275	1.256
Iterations per trial	100	300	100

Another good property of the proposed algorithm is its modularity. The use of distributed averaging techniques to detect the outliers allows to introduce variations of the standard algorithm able to handle, e.g., asynchronous communications [84] or robustness against malicious agents [45]. In this way, our method can handle other communication problems. This is actually what we have done to using the finite-time averaging to count the number of votes or the number of active robots. In addition, we have seen that homogeneous coordinates can be very helpful to compute the average when some of the robots do not have information, or when we do not want to introduce it in the fusion.

We have also used dynamic consensus techniques to allow the robots to change their opinion during the voting. As a consequence, we have seen that the three steps of RANSAC can be executed in a single step with our algorithm. This reduces the number of communication rounds the robots need to exchange information.

On the other hand, the robustness in the consensus implies the counterpart of communicating more information. Each hypothesis that the network generates increases the size of the messages exchanged by the robots. This can be a bit problematic when using visual information, since the amount of data required to exchange in such cases is generally big. Then, the next issue to analyze in our system is the reduction of the number of messages exchanged required to compute the average and the choice of a good representation of the information, so that it takes up a small amount of space.

Proofs

Proof of Theorem 4.2.1 (Consensus with homogeneous coordinates)

The scale coordinate converges to

$$x_i^h(t) \to \frac{1}{N} \sum_{j \in \mathcal{V}_h} 1 = \frac{|\mathcal{V}_h|}{N}, \tag{4.32}$$

and the regular coordinates converge to

$$\mathbf{x}_i(t) \to \frac{1}{N} \sum_{j \in \mathcal{V}_h} \mathbf{x}_j(0). \tag{4.33}$$

The normalized coordinates of an homogeneous vector are its coordinates divided by the scale coordinate, $\mathbf{x}_i^e(t)/x_i^h(t)$, hence

$$\frac{\mathbf{x}_i^e(t)}{x_i^h(t)} \to [\frac{1}{|\mathcal{V}_h|} \sum_{j \in \mathcal{V}_h} \mathbf{x}_j(0)^T,\ 1]^T \; \forall i \in \mathcal{V}. \tag{4.34}$$

■

Proof of Theorem 4.2.3 (Distributed Averaging in Finite-Time)

Taking into account Assumptions 2.2.6 and 2.3.3, we know that $\mathbf{x}_i(t)$ in (4.3) will asymptotically converge to $\bar{\mathbf{x}}$ for all the robots. Given $\varepsilon > 0$, the asymptotic convergence allows us to find t^* such that

$$\forall t > t^* \quad ||\bar{\mathbf{x}} - \mathbf{x}_i(t)||_2 < \varepsilon, \quad \forall i \in \mathcal{V}.$$

In addition, the continuity of g in $\mathbf{x}$ implies that

$$\forall \psi > 0, \; \exists \, \varepsilon > 0 \mid \|\bar{\mathbf{x}} - \mathbf{x}_i(t)\|_2 < \varepsilon \Rightarrow\Rightarrow \|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2 < \psi.$$

Considering again the definition of the φ-set and that $g(\bar{\mathbf{x}}) \in \Phi$, by choosing $\psi < \frac{\varphi}{2}$, there exists t^* depending on ε, and thus on ψ, such that for all $t > t^*$

$$\|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2 < \frac{\varphi}{2},$$

and therefore

$$\arg\min_{\mathbf{x}\in\Phi} \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 < \frac{\varphi}{2}.$$

Now, let us suppose that there exists some $\mathbf{x} \in \Phi, \mathbf{x} \neq g(\bar{\mathbf{x}})$, such that for some $t > t^*$

$$\|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 < \|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2.$$

This would imply

$$\|\mathbf{x} - g(\bar{\mathbf{x}})\|_2 \leq \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 + \|g(\mathbf{x}_i(t)) - g(\bar{\mathbf{x}})\|_2 < \varphi,$$

which by definition of Φ is not possible. Then

$$g(\bar{\mathbf{x}}) = \arg\min_{\mathbf{x}\in\Delta} \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2.$$

■

Proof of Theorem 4.2.4 (Distributed Robot Counting)

1) The rule in (4.5) is a max consensus update rule. Under Assumption 2.2.6, the max consensus algorithm is proved to converge in a finite number of iterations for all the robots in the network [78].

2) The sum of the initial conditions is $\sum_i \beta_{i2}(0) = N$. In order to compute the sum of the inputs, taking into account 1) and Assumption 2.2.1, we have that after some time instant $t^* < \infty$, $\beta_{i1} \neq \mathrm{ID}_i, \forall i \setminus \max_{i\in\mathcal{V}} \mathrm{ID}_i$. Since the initial input is equal to zero for all the robots, then, by (4.7), every robot but the leader (the one with ID equal to the max consensus) will have an input $u_i(t) = -1$ for some t within the interval $[2, t^*]$. After that, the sum of the previous inputs will be different from zero and, therefore, the future inputs will also be equal to zero. Thus

$$\sum_{t=0}^{t^*} \sum_{i\in\mathcal{V}} u_i(t) = -N + 1 \tag{4.35}$$

Then, $\sum_i \beta_{i2}(t^*) = 1$ and $u_i(t) = 0, \; \forall \, i \in \mathcal{V}, t > t^*$, which means that (4.6) behaves like (2.2) and $\beta_{i2}(t) \to \frac{1}{N}$ as $t \to \infty$.

3) Choosing $g(x) = \frac{1}{x}$, which is continuous for any $x \neq 0$ and satisfies that $g(\bar{x}) = N \in \mathbb{N}$, by direct application of Theorem 4.2.3 the result is proved. ■

Proof of Proposition 4.4.1 (Distributed Voting in Finite Time)

First let us notice that $v_i^h(0)$ is in the 1-set, $v_i^h(0) \in \mathbb{N}, \forall i \in \mathcal{V}, \forall h$. In this way, $\sum_{i\in\mathcal{V}} v_i^h(0) \in \mathbb{N}$. We also know that $\sum_{i\in\mathcal{V}} \boldsymbol{v}_i(0) = N\bar{\boldsymbol{v}} = g(\bar{\boldsymbol{v}})$. The function satisfies all the conditions of Theorem 4.2.3 and then convergence in finite time follows. ■

Proof of Proposition 4.5.1 (Distributed Computation of a Partial ML)

As stated in [155], $\mathbf{P}_i(t)$ and $\mathbf{q}_i(t)$ converge to the average of the initial values of all the $\mathbf{P}_j(t)$ and $\mathbf{q}_j(t), j \in \mathcal{V}$. However, the initial values for any robot $j \notin \mathcal{V}_h$ are zero by eq. (4.20), therefore, for all $i \in \mathcal{V}$, it holds that

$$\begin{aligned}\lim_{t\to\infty} \boldsymbol{\theta}_i(t) &= \lim_{t\to\infty} (\mathbf{P}_i(t))^{-1}\mathbf{q}_i(t) = \\ &= (\frac{1}{N}\sum_{j\in\mathcal{V}} \boldsymbol{\Lambda}_j^{-1})^{-1}\frac{1}{N}(\sum_{j\in\mathcal{V}} \boldsymbol{\Lambda}_j^{-1}\mathbf{x}_i) = \\ &= (\frac{1}{N}\sum_{j\in\mathcal{V}_h} \boldsymbol{\Lambda}_j^{-1})^{-1}\frac{1}{N}(\sum_{j\in\mathcal{V}_h} \boldsymbol{\Lambda}_j^{-1}\mathbf{x}_i) = \boldsymbol{\theta}_{\mathrm{ML}}^h\end{aligned}$$

■

Proof of Proposition 4.5.2 (Convergence of the Dynamic Voting Consensus)

The sets $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ are finite. This means that there is some time instant, $t_{\max}$, that upper bounds $\mathcal{T}_i^+$ and $\mathcal{T}_i^-, \forall i \in \mathcal{V}$. Moreover, $\forall t > t_{\max}$ and $i \in \mathcal{V}$, the value of $\chi_i(t)$ remains constant, which means that the robots do not change their opinion after $t_{\max}$.

Let us analyze the evolution of $\mathbf{P}_i(t)$. After $t_{\max}$, the iteration rule (4.22) behaves like (2.2) because $\mathbf{u}_i^{\mathbf{P}}(t) = \mathbf{0}$ for all i and t, and therefore, $\mathbf{P}_i(t)$ will converge to $\frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{P}_i(t_{\max})$. The sum of the values of $\mathbf{P}_i(t_{\max})$ can be written as

$$\sum_{i\in\mathcal{V}} \mathbf{P}_i(t_{\max}) = \sum_{i\in\mathcal{V}_h} \boldsymbol{\Lambda}_i^{-1} + \sum_{i\in\mathcal{V}}\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t), \tag{4.36}$$

where the sum of the inputs for each robot is

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t) = \sum_{t\in\mathcal{T}_i^+} \boldsymbol{\Lambda}_i^{-1} - \sum_{t\in\mathcal{T}_i^-} \boldsymbol{\Lambda}_i^{-1} = (|\mathcal{T}_i^+| - |\mathcal{T}_i^-|)\boldsymbol{\Lambda}_i^{-1}. \tag{4.37}$$

By the eq. (4.24), for any robot i, $-1 \leq |\mathcal{T}_i^+| - |\mathcal{T}_i^-| \leq 1$. At the beginning, for the robots in $\mathcal{V}_h$, it holds that $\chi_i(0) = 1$, because $\boldsymbol{\theta}_i(0) = \mathbf{x}_i$. Therefore, for any $i \in \mathcal{V}_h$,

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t) = \begin{cases} -\boldsymbol{\Lambda}_i^{-1} & \text{if } \chi_i(t_{\max}) = 0, \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{4.38}$$

The robots that do not belong to $\mathcal{V}_h$ cannot compute the distance because the inverse of $\mathbf{P}_i(0)$ is not defined; but this situation in our algorithm is equivalent to $\chi_i(0) = 0$. Then we have that for any $i \notin \mathcal{V}_h$,

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t) = \begin{cases} \boldsymbol{\Lambda}_i^{-1} & \text{if } \chi_i(t_{\max}) = 1, \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{4.39}$$

Putting together (4.36), (4.38) and (4.39), in the limit we have

$$\lim_{t\to\infty} \mathbf{P}_i(t) = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{P}_i(t_{\max}) = \frac{1}{N}\sum_{i\in\mathcal{V}_{\text{con}}} \mathbf{\Lambda}_i^{-1}, \tag{4.40}$$

Applying the same argument to $\mathbf{q}_i(t)$, we obtain

$$\lim_{t\to\infty} \mathbf{q}_i(t) = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{q}_i(t_{\max}) = \frac{1}{N}\sum_{i\in\mathcal{V}_{\text{con}}} \mathbf{\Lambda}_i^{-1}\mathbf{x}_i, \tag{4.41}$$

and then eq. (4.26) holds. Finally, following the same reasoning with $\boldsymbol{v}_i(t)$ eq. (4.27) is obtained and the proof is complete. $\blacksquare$

Proof of Lemma 4.5.3 (Bounded Distances in the Convex Hull)

Let us note that $\mathbf{x}_i \in CH(\mathcal{V}_{\text{in}})$ for all i. For any point in the convex hull, the maximum distance to points inside the hull is achieved at one of the corner points. Since these points are observations that belong to $\mathcal{V}_{\text{in}}$

$$\sqrt{(\mathbf{x}_i - \mathbf{x})^T \mathbf{\Lambda}_i^{-1}(\mathbf{x}_i - \mathbf{x})} \le \max_{\mathbf{x}_j\in\mathcal{V}_{\text{in}}} \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{\Lambda}_i^{-1}(\mathbf{x}_i - \mathbf{x}_j)} \le \chi^2_{d_1,p},$$

and then $\chi(\mathbf{x}_i, \mathbf{x}, \mathbf{\Lambda}_i) = 1$. $\blacksquare$

Computation of the derivative in eq. (4.28) (Local maxima of the derivative)

Let

$$d_i(\mathbf{w}) = \sqrt{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \mathbf{\Lambda}_i^{-1}(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))} \tag{4.42}$$

The distance function can be rewritten as

$$\begin{aligned} d_i(\mathbf{w}) &= \sqrt{d_2(\mathbf{w})}, \\ d_2(\mathbf{w}) &= (\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \mathbf{\Lambda}_i^{-1}(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w})), \\ \boldsymbol{\theta}_i(\mathbf{w}) &= \mathbf{P}_i^{-1}(\mathbf{w})\mathbf{q}_i(\mathbf{w}). \end{aligned} \tag{4.43}$$

We compute the partial derivative applying the chain rule.

The partial derivative of $\boldsymbol{\theta}_i(\mathbf{w})$ with respect to w_j is a function of $\mathbf{P}_i(\mathbf{w})$ and $\mathbf{q}_i(\mathbf{w})$, whose partial derivatives are

$$\frac{\partial \mathbf{P}_i(\mathbf{w})}{\partial w_j} = \mathbf{\Lambda}_j^{-1}, \; \frac{\partial \mathbf{q}_i(\mathbf{w})}{\partial w_j} = \mathbf{\Lambda}_j^{-1}\mathbf{x}_j. \tag{4.44}$$

By passing the inverse matrix to the left member we have

$$\frac{\partial \mathbf{P}_i(\mathbf{w})}{\partial w_j}\boldsymbol{\theta}_i(\mathbf{w}) + \mathbf{P}_i(\mathbf{w})\frac{\partial \boldsymbol{\theta}_i(\mathbf{w})}{\partial w_j} = \frac{\partial \mathbf{q}_i(\mathbf{w})}{\partial w_j}. \tag{4.45}$$

Clearing $\partial\boldsymbol{\theta}_i(\mathbf{w})/\partial w_j$ in (4.45) and plugging (4.44) yields

$$\frac{\partial \boldsymbol{\theta}_i(\mathbf{w})}{\partial w_j} = \mathbf{P}_i^{-1}(\mathbf{w})\mathbf{\Lambda}_j^{-1}(\mathbf{x}_j - \boldsymbol{\theta}_i(\mathbf{w})). \tag{4.46}$$

The partial derivative of $d_2(\mathbf{w})$ with respect to $\boldsymbol{\theta}_i(\mathbf{w})$ is obtained as

$$\begin{aligned} &\frac{\partial(\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1}(\mathbf{x}_i - \boldsymbol{\theta}_i(w))}{\partial \boldsymbol{\theta}_i(w)} = \\ &(\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-T} + (\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1} = \\ &2(\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1}. \end{aligned} \tag{4.47}$$

Finally, computing the partial of $d_i(\mathbf{w})$ with respect to $d_2(\mathbf{w})$ and applying the chain rule we get

$$\frac{\partial d_i(\mathbf{w})}{\partial w_j} = \frac{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \boldsymbol{\Lambda}_i^{-1} \mathbf{P}_i^{-1}(\mathbf{w}) \boldsymbol{\Lambda}_j^{-1} (\mathbf{x}_j - \boldsymbol{\theta}_i(\mathbf{w}))}{d_i(\mathbf{w})}. \tag{4.48}$$

The derivative is well defined at any point but the set of points such that $\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i$, because $d_i(\mathbf{w}) = 0$. However, at this points we already now that $d_i(\mathbf{w})$ has a global minimum because the distance is always positive (or zero). Therefore, they do not affect our analysis in Theorem 4.5.4.

Proof of Theorem 4.5.4 (Convergence to the ML of the Inliers)

Along this proof $\mathbf{e}_i$ will denote the i^{th} vector of the canonical basis of $\mathbb{R}^N$ and we will make use of eq. (4.42) and the property $d_i(\mathbf{w}) = d_i(\lambda \mathbf{w}), \lambda > 0$. Observe also that $\boldsymbol{\theta}_i(\mathbf{e}_i) = \mathbf{x}_i$.

By the theory of analysis of multivariate functions [81], given a closed compact set $C \subset \mathbb{R}^N$ and a continuous function $f : \mathbb{R}^N \to \mathbb{R}$, f has local extremes in C only where the derivatives are zero or are not defined. If f has no local extremes in C, then the maximum (minimum) values of f are in the frontier of C. In our case the functions to analyze are the different distances $d_i(\mathbf{w})$. The compact set, C_ε, where we will analyze the function is, $\mathbf{w}_{\text{out}} = \mathbf{0}$ and $\mathbf{w}_{\text{in}} \in [0,1]^{|\mathcal{V}_{in}|} \setminus [0, \varepsilon)^{|\mathcal{V}_{in}|}$, for $\varepsilon \simeq 0$ sufficiently small.

Due to the definition of C_ε, its frontier is a set of surfaces in $\mathbb{R}^{N-1}$. By applying the same result recursively to the different frontier surfaces to differentiable functions, if the partial derivatives of f do not vanish inside C_ε, we obtain that the maximum (minimum) values of f are at the corners of C_ε. For example, for two inliers, C_ε has 6 different corners at positions $\{(0,1),(1,0),(1,1),(0,\varepsilon),(\varepsilon,0),(\varepsilon,\varepsilon)\}$. For a general number of dimensions, the set of corners of C_ε is characterized as

$$\mathbf{w}^* = \{\sum_{i \in \mathcal{S}} \mathbf{e}_i, \mathcal{S} \subseteq \mathcal{V}_{\text{in}}\} \bigcup \{\varepsilon \sum_{i \in \mathcal{S}} \mathbf{e}_i, \mathcal{S} \subseteq \mathcal{V}_{\text{in}}\}. \tag{4.49}$$

Note that, by Condition 4, if $\partial d_i(\mathbf{w})/\partial w_j = 0$, then there exists a corner $\mathbf{w}^* \in \{\mathbf{e}_i, \mathbf{e}_j\}$ where $d_i(\mathbf{w}^*) = d_i(\mathbf{w})$.

Let us start showing that the outliers do not vote for the hypothesis at any time. Let us consider any robot $k \notin \mathcal{V}_{\text{in}}$. At the beginning $\mathbf{P}_k(0) = \mathbf{0}$ because $\mathcal{V}_h \subseteq \mathcal{V}_{\text{in}}$, and therefore k does not vote the hypothesis. Let us denote now k the first outlier for which $\mathbf{P}_k(t) \neq \mathbf{0}$. At this moment $\mathbf{w}_{\text{out}} = \mathbf{0}$ and $\mathbf{w}_{\text{in}} \in C_\varepsilon$ for a small enough ε. Let us see that for any $\mathbf{w}_{\text{in}} \in C_\varepsilon, d_k(\mathbf{w}_{\text{in}}) > \chi^2_{d_1,p}$, then $\chi(\mathbf{w}_{\text{in}}) = 1$. By Condition 4, the partial derivatives do not vanish, or, if they do, is at points $\mathbf{x}_i \in \mathcal{V}_{\text{in}}$, which correspond to corners of C_ε. We have then to analyze the value of $d_k(\mathbf{w}^*)$. Condition 2 together with Condition 3 implies that

$$d_k(\sum_{i \in \mathcal{S}} \mathbf{e}_i) = d_k(\varepsilon \sum_{i \in \mathcal{S}} \mathbf{e}_i) > \chi^2_{d,p}, \quad \forall \mathcal{S} \subseteq \mathcal{V}_{\text{in}}.$$

All the corner points of C_ε are covered and for all of them the Mahalanobis distance is larger than $\chi^2_{d,p}$. Therefore, the outlier will not vote for the hypotheses. As successive outliers have $\mathbf{P}_k(t) \neq \mathbf{0}$, still $\mathbf{w}_{\text{out}} = \mathbf{0}$ and the same argument applies. This means that none of the outliers will vote the hypotheses and eq. (4.31) is true.

A similar argument can be applied for the inliers. We already know that $\mathbf{w}_{\text{out}} = \mathbf{0}$ for any time instant. Looking at the corners of C_ε, by Conditions 1 and 2, for any $i \in \mathcal{V}_{\text{in}}$

$$d_i(\sum_{j\in\mathcal{S}} \mathbf{e}_j) = d_i(\varepsilon \sum_{j\in\mathcal{S}} \mathbf{e}_j) \leq \chi^2_{d,p}, \quad \forall \mathcal{S} \subseteq \mathcal{V}_{\text{in}},$$

then for any $\mathbf{w}_{\text{in}} \in C_\varepsilon$, $d_i(\mathbf{w}_{\text{in}}) \leq \chi^2_{d,p}$. This means that once $\mathbf{P}_i(t) \neq \mathbf{0}$, robot i will vote for the hypotheses and will keep doing so. By taking t^+ the first time instant for which for all $i \in \mathcal{V}_{\text{in}}$ $\mathbf{P}_i(t) \neq \mathbf{0}$, eq. (4.30) is satisfied and the result holds. $\blacksquare$

Chapter 5

Fast Consensus with Chebyshev Polynomials

"No matter how fast your computer system runs, you will eventually think of it as slow." When the number of robots in the network is large, distributed averaging methods usually have a slow convergence rate. In this chapter we analyze the use of Chebyshev polynomials in the distributed consensus problem to reduce the number of iterations required to achieve a good consensus. We propose a distributed linear iteration using these polynomials that compared to other approaches, is able to achieve the average of the initial conditions in a small number of iterations. In the chapter we characterize the main properties of the algorithm for both, fixed and switching communication topologies. Additionally, we provide a second algorithm for the adaptive selection of the parameters to maximize the convergence rate. We validate our results with extensive simulations.

5.1 Introduction

When the number of robots in the network is large, distributed averaging methods usually have a slow convergence rate. This means that the number of iterations before obtaining a good approximation of the consensus is considerably large. Moreover, because of the use of visual sensors, the size of the messages the robots exchange is in general big and, as we have seen in the previous chapter, if we want the consensus to be robust we require the sending of even more information. If we put everything together, we find that the standard linear iteration, although in theory is very useful, in practice it is not so much. The goal of this chapter is to devise a distributed linear iteration that requires a small number of iterations to reach the consensus.

The convergence rate of linear iterations has been studied under a variety of scenarios [4, 21, 42, 76, 109, 110, 117, 162, 164]. All these papers point out the slow convergence rate of linear iterations for large or sparsely connected networks. For that reason a lot of research has been devoted to provide solutions that reduce the convergence time to achieve the consensus. Some works achieve consensus in finite time using continuous-time non linear methods [30, 62, 151] or link scheduling [67]. The use of numerical integrators affects the number of iterations in the non-linear approaches and there might also be situations in which not all the links are feasible. Other approaches speed up convergence by sending additional information in the messages [63, 158]. Unfortunately, depending on the network topology it might be required to send big messages. The design of the adjacency matrix has been the focus of several works [9, 64, 66, 153], improving the convergence speed. Nevertheless, they can still be combined with additional techniques in order to accelerate even more the consensus.

The distributed evaluation of polynomials speeds up the consensus, keeping the good properties found in linear methods. The minimal polynomial of the adjacency matrix is used in [138] and [159]. Once this polynomial is known, the network can achieve the consensus in a finite number of communication rounds. Unfortunately, the use of the minimal polynomial presents several disadvantages: it only works for a fixed communication topology, it requires a full and precise knowledge of the network and, for a large number of robots or sparsely connected networks, it is unstable [91].

An algorithm using a polynomial of low degree, computed using optimization techniques, is proposed in [68]. However, the whole network is also required to solve the optimization problem and the algorithm has to be executed in blocks of iterations of the size of the degree. If the topology changes during the execution of the blocks there are no guarantees of convergence, which forces the degree of the polynomial to be small.

Polynomials described by recurrent relations overcome these limitations and considerably speed up the consensus [111, 129]. In this chapter we propose a consensus algorithm using these kind of approaches that increases the convergence rate with respect to existing approaches. To design the fast consensus algorithm we consider Chebyshev polynomials of the first kind [83]. These polynomials are a powerful mathematical tool that has proven to be very helpful in many different fields of science. For example, they are used to model chemical reaction [101], in aeronautics [127], numerical methods [148] and computer vision [113].

The contributions of this chapter are:

- A distributed algorithm based on the second order difference equation that describes the Chebyshev polynomials of first kind. The algorithm has no limitations on the degree of the polynomial and only requires an approximated knowledge of the maximum and minimum eigenvalues of the weight matrix to reach the consensus.
- A complete study of the properties of the algorithm. For the case of fixed topology, we find the parameters to achieve the optimal convergence rate and we give bounds on the selection of these parameters to achieve a faster convergence than using the standard consensus algorithm. For the case of switching topologies, we theoretically show that there are always parameters that make the algorithm converge to the consensus.
- An adaptive method to distributively compute the optimal parameters of our fast algorithm. The algorithm is proved to converge to the algebraic connectivity of the network, which corresponds with the parameter that speeds up the most the consensus algorithm.
- An empirical validation of the theoretical results with extensive simulations, where we compare our algorithm with existing methods.

This chapter has been partially published in [90–92].

5.2 Background on Chebyshev Polynomials and Polynomial Filtering

5.2.1 Chebyshev Polynomials

We denote the Chebyshev polynomial of degree n by $T_n(x)$. These polynomials satisfy

$$T_n(x) = \cos(n \arccos x), \text{ for all } x \in [-1, 1], \tag{5.1}$$

and $|T_n(x)| > 1$ when $|x| > 1$, for all $n \in \mathbb{N}$. Moreover $|T_n(x_1)| > |T_n(x_2)|$ for all $|x_1| > |x_2| > 1$. A more general way to define these polynomials in the real domain is using a second order recurrence,

$$\begin{array}{l} T_0(x) = 1,\ T_1(x) = x \\ T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n \geq 2. \end{array} \tag{5.2}$$

By the theory of difference equations [2], the direct expression of (5.2) is determined by the roots τ_1 and τ_2 of the characteristic equation,

$$T_n(x) = \frac{1}{2}(\tau_1(x)^n + \tau_2(x)^n), \tag{5.3}$$

where $\tau_1(x) = x - \sqrt{x^2-1}$ and $\tau_2(x) = x + \sqrt{x^2-1} = 1/\tau_1(x)$. In the Thesis we take

$$\tau(x) = \left\{ \begin{array}{ll} x - \sqrt{x^2-1}, & \text{if } x \geq 0 \\ x + \sqrt{x^2-1}, & \text{if } x < 0 \end{array} \right. , \tag{5.4}$$

so that $|\tau(x)| < 1$ and $|\tau(x)|^{-1} > 1$ for all $|x| > 1$, and therefore,

$$T_n(x) = \frac{1}{2}(\tau(x)^n + \tau(x)^{-n}) = \frac{1}{2}\tau(x)^{-n}(1 + \tau(x)^{2n}). \tag{5.5}$$

It is clear that if $|x| > 1$, then $T_n(x)$ goes to infinity as n grows. If $|x| < 1$, then $\tau(x)$ is a complex number with $|\tau(x)| = |x \pm (\sqrt{1-x^2})i| = 1$ and $|T_n(x)| \leq 1,\ \forall n$, as stated in eq. (5.1).

5.2.2 Polynomial Filtering

The distributed evaluation of polynomials provides an easy way to speed up the consensus, keeping the good properties found in standard methods. The main idea consists in designing a distributed linear iteration such that the execution of a fixed number of n steps is equivalent to the evaluation of some polynomial, $P_n(x)$, in the fixed matrix $\mathbf{W}$ and multiplied by the initial conditions, $\mathbf{x}(0)$, [68, 138]. This evaluation is defined as

$$\begin{aligned} P_n(\mathbf{W})\mathbf{x}(0) &= \alpha_0\mathbf{x}(0) + \alpha_1\mathbf{W}\mathbf{x}(0) + \ldots + \alpha_n\mathbf{W}^n\mathbf{x}(0) \\ &= \alpha_0\mathbf{x}(0) + \alpha_1\mathbf{x}(1) + \ldots + \alpha_n\mathbf{x}(n) \\ &= P_n(1)\gamma_1\mathbf{v}_1 + P_n(\lambda_2)\gamma_2\mathbf{v}_2 + \ldots + P_n(\lambda_N)\gamma_N\mathbf{v}_N, \end{aligned} \tag{5.6}$$

with $\mathbf{v}_i$ and γ_i the eigenvectors and their coefficients as in eq. (2.7) and α_i the coefficients of the polynomial.

In order to achieve the consensus, the polynomial must satisfy that $P_n(1) = 1$ and $|P_n(x)| < 1$ if $|x| < 1$. The convergence speed is given by $\max_{\lambda_i} |P_n(\lambda_i)|$, with λ_i the eigenvalues of $\mathbf{W}$. Let us note that the iteration (2.3) is in fact a polynomial filter $\mathbf{x}(n) = P_n(\mathbf{W})\mathbf{x}(0)$ [1] with $P_n(x) = x^n$. With full knowledge of the weight matrix (and assuming it does not change), the minimal polynomial of $\mathbf{W}$ is clearly the best choice because $P_N(\lambda_i) = 0, \forall i \neq 1$ [138]. Unfortunately, this solution fails when the topology changes or when N is large [91]. This is shown in Figure 5.1, where the minimal polynomial fails to compute the average for a network composed by 35 nodes.

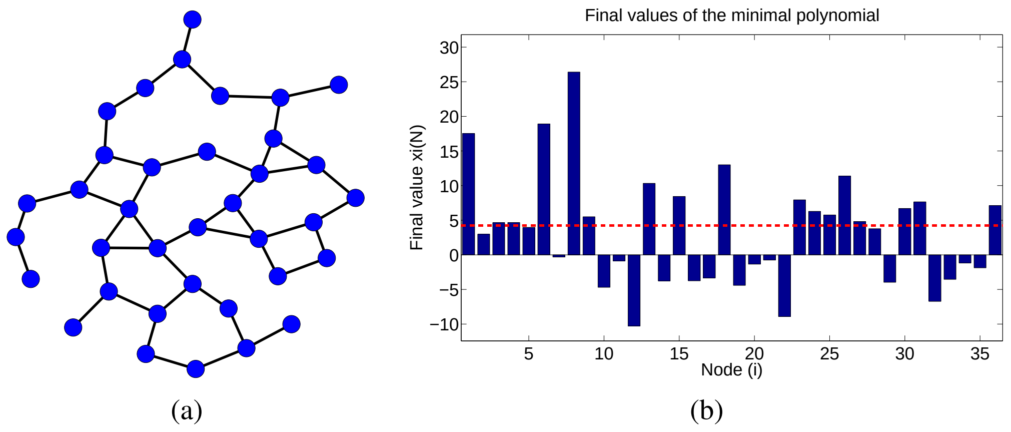


Figure 5.1: Numerical errors in the evaluation of the minimal polynomial lead to errors in the final estimation. (a) Communication network of 35 nodes. (b) Estimations of the 35 nodes of the network evaluating the minimal polynomial in a distributed way. The red dashed line shows the average of the initial conditions and the bars are the values of the final estimations. The rounding errors make the algorithm fail.

This can be overcome by designing a polynomial of a lower degree, $k \ll N$, using optimization techniques to minimize $Q_k(\lambda_i)$ [68]. However, the matrix $\mathbf{W}$ is still required and the algorithm has to be executed in blocks of k iterations to get $\mathbf{x}(k) = Q_k(\mathbf{W})\mathbf{x}(0)$, $\mathbf{x}(2k) = Q_k(\mathbf{W})\mathbf{x}(k), \ldots$ Moreover, if the topology changes during the execution of one of the k iterations of one block, then the convergence is not guaranteed.

[1] In order to keep a consistent notation with Chebyshev polynomials of degree n, $T_n(x)$, in this Chapter we replace the time parameter, t, in the linear iterations for the degree of the polynomial, n.

5.3 Consensus algorithm using Chebyshev Polynomials

Let us assume the eigenvalues of $\mathbf{W}$ are unknown. In this situation it seems natural to look for a polynomial that minimizes the uniform norm in the interval that contains all the eigenvalues, i.e., find a polynomial, $P_n(x)$, that minimizes $\sup_{x\in(-1,1)} |P_n(x)|$. However, this way of proceeding does not guarantee a good convergence rate because for all $P_n(x)$ such that $P_n(1) = 1$,

$$\sup_{x\in(-1,1)} |P_n(x)| = \max_{x\in[-1,1]} |P_n(x)| \geq 1.$$

Instead of that, we can assume that the eigenvalues are located all along a closed interval $[\lambda_m, \lambda_M] \subset (-1, 1)$. With this assumption we can look for the polynomial that minimizes the uniform norm in such an interval, i.e., find a polynomial, $P_n(x)$, that minimizes $\max_{x\in[\lambda_m,\lambda_M]} |P_n(x)|$.

This polynomial is precisely the shifted-scaled Chebyshev polynomial of degree n. Given two real coefficients λ_m, λ_M, with $-1 < \lambda_m < \lambda_M < 1$, the shifted-scaled Chebyshev polynomial of degree n is defined by

$$P_n(x) = \frac{T_n(cx-d)}{T_n(c-d)}, \text{ with } c = \frac{2}{\lambda_M - \lambda_m}, \quad d = \frac{\lambda_M + \lambda_m}{\lambda_M - \lambda_m}. \tag{5.7}$$

This polynomial satisfies that $P_n(1) = 1$, $P_n(\lambda_M+\lambda_m-1) = (-1)^n$, $|P_n(x)| < 1$ for all $x \in (\lambda_M+\lambda_m-1, 1)$ and $|P_n(x)| \geq 1$ otherwise, for all n.

Using (5.2) it is possible to evaluate (5.7) in a stable way by means of the recurrence

$$\begin{aligned} P_n(x) &= \frac{T_n(cx-d)}{T_n(c-d)} = \frac{2(cx-d)T_{n-1}(cx-d) - T_{n-2}(cx-d)}{T_n(c-d)} = \\ &= 2(cx-d)\frac{T_{n-1}(c-d)}{T_n(c-d)}P_{n-1}(x) - \frac{T_{n-2}(c-d)}{T_n(c-d)}P_{n-2}(x). \end{aligned} \tag{5.8}$$

Therefore, the value $\mathbf{x}(n) = P_n(\mathbf{W})\mathbf{x}(0)$, can be computed as

$$\begin{aligned} \mathbf{x}(1) &= P_1(\mathbf{W})\mathbf{x}(0) = \frac{T_1(c\mathbf{W} - d\mathbf{I})}{T_1(c-d)}\mathbf{x}(0) = \frac{(c\mathbf{W} - d\mathbf{I})}{c-d}\mathbf{x}(0), \\ \mathbf{x}(n) &= P_n(\mathbf{W})\mathbf{x}(0) = \left(2\frac{T_{n-1}(c-d)}{T_n(c-d)}(c\mathbf{W} - d\mathbf{I})P_{n-1}(\mathbf{W}) - \frac{T_{n-2}(c-d)}{T_n(c-d)}P_{n-2}(\mathbf{W})\right)\mathbf{x}(0) \\ &= 2\frac{T_{n-1}(c-d)}{T_n(c-d)}(c\mathbf{W} - d\mathbf{I})\mathbf{x}(n-1) - \frac{T_{n-2}(c-d)}{T_n(c-d)}\mathbf{x}(n-2), \; n \geq 2, \end{aligned} \tag{5.9}$$

with $\mathbf{I}$ the identity matrix of dimension N. The individual update that each robot executes is

$$\begin{aligned} x_i(1) &= \frac{1}{T_1(c-d)}\left(w'_{ii}x_i(n-1) + \sum_{j\in\mathcal{N}_i(n)} w'_{ij}x_j(n-1)\right), \\ x_i(n) &= 2\frac{T_{n-1}(c-d)}{T_n(c-d)}\left(w'_{ii}x_i(n-1) + \sum_{j\in\mathcal{N}_i(n)} w'_{ij}x_j(n-1)\right) - \frac{T_{n-2}(c-d)}{T_n(c-d)}x_i(n-2), \end{aligned} \tag{5.10}$$

with $w'_{ij} = c\,w_{ij}$, $w'_{ii} = (c\,w_{ii} - d)$ and w_{ij} are the elements of the weight matrix $\mathbf{W}$.

Compared to other polynomial filters the algorithm presents some advantages. It is well known that the evaluation of Chebyshev polynomials using (5.2) is stable [83]. Then, at each iteration we get the value of $P_n(\mathbf{W})\mathbf{x}(0)$, i.e., there are no limitations on the degree of the polynomial and the algorithm is not required to be executed in blocks of k iterations, for some fixed k. In addition, when the topology of the network changes, the recurrent evaluation of Chebyshev polynomials (5.10) can still be used. The time-varying version of the algorithm is equivalent to (5.9) replacing the constant weight matrix $\mathbf{W}$ by the weight matrix at each step $\mathbf{W}(n)$. Although this is no longer equivalent to the distributed evaluation of a Chebyshev polynomial, a theoretical analysis about its convergence is still possible. In addition, compared to (2.2), we can see that the iteration (5.10) transmits exactly the same information, $x_i(n-1)$. In terms of space requirements the robots only need to store one extra datum, $x_i(n-2)$, and the computation demands are also very similar because $T_n(c-d)$ can be computed locally by each robot using (5.2).

To design the algorithm we have assumed that the eigenvalues of $\mathbf{W}$ were in the interval $[\lambda_m, \lambda_M]$. However, in practice we do not know if these assumption is going to be true. In the next two sections we analyze the main properties of the algorithm as a function of the parameters and the eigenvalues of $\mathbf{W}$, for fixed and switching communication topologies.

Algorithm 5 Consensus algorithm using Chebyshev polynomials - Robot i

```
Require: $x_i(0)$, MaxIt $\in \mathbb{N}$, $\lambda_m$, $\lambda_M$,
1: – Initialization
2: $c = 2/(\lambda_M - \lambda_m)$;    $d = (\lambda_M + \lambda_m)/(\lambda_M - \lambda_m)$;
3: $T(0) = 1$;    $T(1) = c - d$;
4: – First Communication Round
       $x_i(1) = \frac{1}{T(1)}(c \sum_{j \in \mathcal{N}_i(n)} w_{ij} x_j(0) + (c\, w_{ii} - d) x_i(0))$;
5: for $n = 2, \ldots,$MaxIt do
6:     $T(n) = 2(c-d)T(n-1) - T(n-2)$;
7:     – Communication Between Neighbors
       $x_i(n) = 2\frac{T(n-1)}{T(n)}(c \sum_{j \in \mathcal{N}_i(n)} w_{ij} x_j(n-1) + (c\, w_{ii} - d) x_i(n-1)) - \frac{T(n-2)}{T(n)} x_i(n-2)$;
8: end for
```

5.4 Analysis of the algorithm with Fixed Topologies

In this section we analyze the main properties of the proposed algorithm when the network topology is fixed. In particular we first study the convergence conditions of the algorithm. Next, we find the parameters that maximize the convergence speed and we give bounds on the selection of these parameters to achieve the consensus faster than (2.2). Finally, we analyze the convergence for directed graphs and the parameters to assign in regular graphs and spanning trees.

Theorem 5.4.1 (Convergence of the algorithm). *For any $\mathbf{W}$ fulfilling Assumption 2.3.2 and parameters λ_m and λ_M such that $1 > \lambda_M > \lambda_m > -1$ and $\lambda_N > \lambda_m + \lambda_M - 1$, the recurrence in eq.* (5.9) *converges asymptotically to the average of the initial conditions, with convergence rate*

$$\|\boldsymbol{x}(n) - \gamma_1 \boldsymbol{v}_1\|_2 \leq \max_{\lambda_i \neq 1} \frac{|T_n(c\lambda_i - d)|}{T_n(c-d)} \|\boldsymbol{x}(0) - \gamma_1 \boldsymbol{v}_1\|_2. \tag{5.11}$$

Note that the conditions in Theorem 5.4.1 are easy to fulfill without the necessity of knowing the eigenvalues of the matrix $\mathbf{W}$. A symmetric selection of the parameters, i.e., $-\lambda_m = \lambda_M,\ 0 < \lambda_M < 1$, always satisfies the condition in Theorem 5.4.1. However, it is interesting to know the optimal selection of λ_m and λ_M to maximize the convergence speed. From Theorem 5.4.1 we know that the convergence rate is given by the factor

$$\max_{\lambda_i \neq 1} \frac{|T_n(c\lambda_i - d)|}{T_n(c-d)} \leq \max\left\{ \frac{|T_n(c\lambda_N - d)|}{T_n(c-d)}, \frac{|T_n(c\lambda_2 - d)|}{T_n(c-d)}, \frac{1}{T_n(c-d)} \right\}. \tag{5.12}$$

The first two terms consider the case when either λ_2 or λ_N lay outside of the interval $[\lambda_m, \lambda_M]$, because in such case $|T_n(c\lambda - d)|$ is monotonically increasing with $|\lambda|$. When all the eigenvalues are contained in the interval $[\lambda_m, \lambda_M]$, then $|T_n(c\lambda_i - d)| \leq 1$ for all $i \neq 1$ and the convergence rate is given by the third term of (5.12). A simple calculation using eq. (5.5) leads to

$$\frac{|T_n(c\lambda - d)|}{T_n(c-d)} = \left(\frac{\tau(c-d)}{|\tau(c\lambda - d)|} \right)^n \frac{1 + \tau(c\lambda - d)^{2n}}{1 + \tau(c-d)^{2n}}, \text{ for all } \lambda \in \mathbb{R}. \tag{5.13}$$

It is clear that when $n \to \infty$, the second fraction in the right side of (5.13) goes to 1 for $|\lambda| \notin [\lambda_m, \lambda_M]$. Therefore, the convergence rate is determined by

$$\nu(c,d) = \begin{cases} \tau(c-d), & \text{if } [\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M] \\ \max\left\{ \frac{\tau(c-d)}{|\tau(c\lambda_N - d)|}, \frac{\tau(c-d)}{|\tau(c\lambda_2 - d)|} \right\}, & \text{otherwise.} \end{cases} \tag{5.14}$$

The optimum values of λ_m and λ_M will be those that lead to the minimum value of $\nu(c,d)$.

Theorem 5.4.2 (Optimal parameters). *The convergence rate $\nu(c,d)$ attains its minimum value for the parameters c, d such that $\lambda_M = \lambda_2$ and $\lambda_m = \lambda_N$*

This implies that in order to achieve the maximum convergence speed, instead of requiring to know the whole matrix, only the maximum and minimum eigenvalues, λ_2 and λ_N, of the weight matrix are necessary. Note that using the optimal parameters, the conditions of Theorem 5.4.1 are almost always satisfied. The only situation where this does not hold is when $\lambda_2 = \ldots = \lambda_N$, because that would imply $\lambda_m = \lambda_M$. This happens for example for a complete graph. However, in this case we can choose $\lambda_m = \lambda_N + \epsilon$ and $\lambda_M = \lambda_2 - \epsilon$ with $\epsilon \simeq 0$ and achieve very good convergence rates.

Finally, we provide bounds for the symmetric assignation of parameters, $\lambda_M = -\lambda_m$, to achieve faster convergence than (2.3).

Theorem 5.4.3 (Faster convergence than $\mathbf{W}^n$). *For any matrix **W** satisfying Assumption 2.3.2, let $\lambda = \max(|\lambda_2|, |\lambda_N|)$ be the convergence rate in* (2.3). *For any*

$$0 < \lambda_M < \frac{2\lambda}{\lambda^2 + 1}, \textit{ and } \lambda_m = -\lambda_M, \tag{5.15}$$

$P_n(\lambda)$ goes to zero faster than λ^n when n goes to infinity. Therefore the algorithm in eq. (5.9) *converges to the consensus faster than the one in eq.* (2.3).

Remark 5.4.4 (Fast convergence with the optimal parameters). *The above result shows that there always exist parameters that make the proposed algorithm faster than* (2.3). *Therefore, if the algorithm is executed using the optimal parameters, it will also converge to the average faster than* (2.3).

Finally, a graphical comparison of x^n, $T_n(x)$ and $P_n(x)$ using $\lambda_m = -0.95$ and $\lambda_M = 0.95$ is depicted in Fig. 5.2 for $n = 4$, in the interval $[-1, 1]$. Note that $T_n(x)$ cannot be used in the consensus process because at some points it would not reduce the error. On the other hand, as we have shown along the section, $P_n(x)$ satisfies the conditions required to achieve consensus. Also notice that $P_n(x)$ has closer values to zero than x^n in points close to -1 and 1, which supports the theory that the error associated to eigenvalues in that regions will be reduced faster.

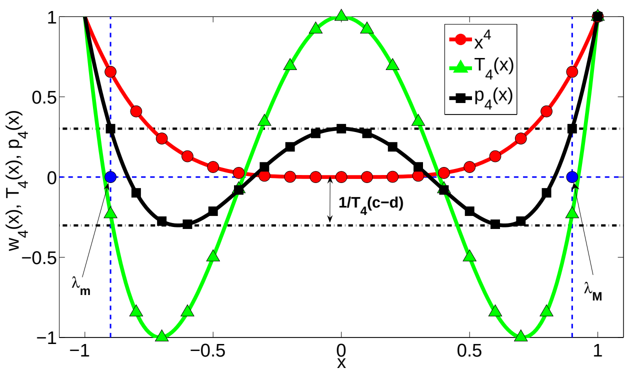


Figure 5.2: Plot of the polynomials x^n, $T_n(x)$ and $P_n(x)$. In the figure $n = 4$, $\lambda_m = -0.95$ and $\lambda_M = 0.95$.

5.4.1 Special Topologies

We consider an extension of the algorithm to directed communication graphs. We also analyze the selection of the parameters λ_M and λ_m for some common topologies.

Directed graphs

If the communication graph is directed, then the weight matrix is not symmetric anymore. This implies that the matrix might no longer be diagonalizable and its eigenvalues could be in the complex plane. We analyze the implications of these facts in the convergence of the algorithm.

Chebyshev polynomials, $T_n(z), z \in \mathbb{C}$, on the complex plane can also be expressed by (5.5), where $\tau(z)$ is defined now by

$$\tau(z) = \begin{cases} z - \sqrt{z^2 - 1}, & \text{if } |z - \sqrt{z^2-1}| < 1 \\ z + \sqrt{z^2 - 1}, & \text{otherwise} \end{cases} . \tag{5.16}$$

In this case $|\tau(z)| \leq 1$ and $|\tau(z)|^{-1} \geq 1$ for all z, which implies that $T_n(z)$ go always to infinity with n for $z \notin \mathbb{R}$.

Proposition 5.4.5 (Convergence for directed graphs). *Let **W** be diagonalizable and row stochastic, and parameters λ_m and λ_M such that $1 > \lambda_M > \lambda_m > -1$. If the minimum real eigenvalue of **W** satisfies $\lambda_N > \lambda_m + \lambda_M - 1$ and the complex eigenvalues, λ_z, of **W** satisfy $|\tau(c\lambda_z - d)| > \tau(c - d)$, then the recurrence in eq.* (5.9) *converges to the consensus state.*

The condition on the complex eigenvalues has some geometrical meaning [83]. Imposing that $|\tau(c\lambda_z - d)| > \tau(c - d)$ is equivalent to require that λ_z is inside an ellipse in the complex plane centered at $(d/c, 0)$, or equivalently $((\lambda_M + \lambda_m)/2, 0)$, and with semi-axis $e_1 = (c - d)/c$ and $e_2 = (\sqrt{(c-d)^2 - 1})/c$ (see Fig 5.3). The condition of the weight matrix being diagonalizable is required to factorize **W** in the proof of the Proposition. Although there is no direct way of knowing if **W** is diagonalizable without the knowledge of the whole matrix, in our simulations we have not found any convergence trouble.

Spanning trees

In order to give bounds on the parameters λ_m and λ_M for topologies with a spanning tree configuration, we will consider the best and the worst cases and study the eigenvalues of their weight matrices using the local degree weights [153].

The topology with the best connectivity is the star topology, in which there is one robot connected to all the others. For a star graph of N robots it can be proved that **W** has eigenvalues $1 = \lambda_1 > (N-1)/N = \lambda_2 = \ldots = \lambda_{N-1} > \lambda_N = 0$.

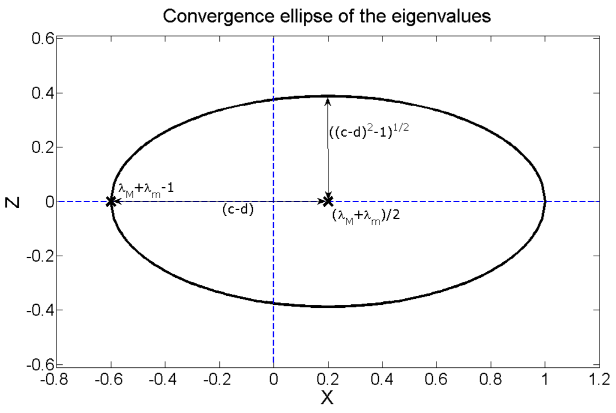

Convergence ellipse of the eigenvalues
((c-d)^2-1)$^{1/2}$
λ_M+λ_m-1
(c-d)
(λ_M+λ_m)/2
Z
X

Figure 5.3: Ellipse where all the eigenvalues must be contained in order to achieve the consensus. In this particular example we have chosen $\lambda_M = 0.9$ and $\lambda_m = -0.5$. Note that when the imaginary part of the eigenvalues is zero convergence is achieved if $\lambda_M + \lambda_m - 1 > \lambda > 1$ as stated in Theorem 5.4.1.

The worst possible spanning tree configuration is a chain. For a chain graph of N robots, using the local degree weights, the eigenvalues of **W** satisfy [18]

$$\lambda_i = \frac{1 - 2\cos(\frac{i\pi}{N})}{3}, i = 1, \dots, N.$$

Combining the constraints on the two graphs we conclude that, for $N \geq 3$, the eigenvalues λ_2 and λ_N must satisfy

$$\frac{N-1}{N} \leq \lambda_2 \leq \frac{1 - 2\cos(\frac{(N-1)\pi}{N})}{3} < 1 \text{ and } -\frac{1}{3} < \frac{1 - 2\cos(\frac{\pi}{N})}{3} \leq \lambda_N \leq 0.$$

And then by choosing $\lambda_m = -1/3$ and $\lambda_M = \frac{1-2\cos(\frac{(N-1)\pi}{N})}{3}$ convergence is ensured.

Regular graphs

A regular graph of degree r is a graph in which all the robots have exactly r neighbors. For this kind of graphs, the worst connectivity appears when $r = 2$, which is only possible if the network forms a ring. For the ring topology, the eigenvalues are [18]

$$\lambda_i = \frac{1 + 2\cos(\frac{2i\pi}{N})}{3}, i = 1, \dots, N.$$

The best topology occurs when $r = N - 1$, which implies a fully connected graph. In this case $\lambda_2 = \dots = \lambda_N = 0$. However, this is a trivial scenario in which the consensus is achieved in one iteration and only serves us to find the upper bound of λ_m. Instead of considering the weight matrix, let us consider the adjacency matrix. In [17] it is shown that, for $r \geq 3$, the second largest eigenvalue of the adjacency matrix is contained in the interval centered in $r^{3/4}$ with radius $r^{1/2}$. Then, a lower bound for this eigenvalue is $r^{3/4} - r^{1/2}$. Using the local degree weights we find the relation $\mathbf{W} = (\mathbf{I} + \mathbf{A})/(r + 1)$. All this together yields to the bounds on λ_2 and λ_N for regular graphs,

$$\frac{1 + r^{3/4} - r^{1/2}}{d+1} \leq \lambda_2 \leq \max_i \frac{1 + 2\cos(\frac{2i\pi}{N})}{3} < 1, \text{ and } -\frac{1}{3} < \min_i \frac{1 + 2\cos(\frac{2i\pi}{N})}{3} \leq \lambda_N \leq 0,$$

which provides values for the parameters λ_m and λ_M.

5.5 Analysis of the algorithm with Switching Topologies

We study now the recursive evaluation of (5.9) when the topology of the network, and therefore the matrix **W**, changes at different iterations. Given initial conditions $\mathbf{x}(0)$, the distributed recurrence now is:

$$\begin{aligned}\mathbf{x}(1) &= \frac{1}{T_1(c-d)}(c\mathbf{W}(1) - d\mathbf{I})\mathbf{x}(0),\\ \mathbf{x}(n) &= 2\frac{T_{n-1}(c-d)}{T_n(c-d)}(c\mathbf{W}(n) - d\mathbf{I})\mathbf{x}(n-1) - \frac{T_{n-2}(c-d)}{T_n(c-d)}\mathbf{x}(n-2),\ n \geq 2.\end{aligned} \tag{5.17}$$

Note that this recurrence is suitable for switching weight matrices. However, the evaluation of the recurrence is no longer equivalent to $P_n(\mathbf{W})\mathbf{x}(0)$, for some matrix **W**. This means that we are not exactly evaluating the transformed Chebyshev polynomials at the eigenvalues of some matrix anymore. Nevertheless, a theoretical analysis is still possible.

Recall that the evaluation of $P_n(\mathbf{W})\mathbf{x}(0)$ can be separated into the evaluation of its eigenvalues and eigenvectors, eq. (5.6), $P_n(\lambda_i)\mathbf{v}_i = T_n(c\lambda_i - d)/T_n(c-d)\mathbf{v}_i$. In the switching case we must take into account that both λ_i and $\mathbf{v}_i$ change at each iteration. Moreover, since the eigenvectors of different matrices are related we must also consider these relations. For the moment, as a first simplification of the problem, let us forget about the changes in $\mathbf{v}_i$ and the parameters c and d and let us study the scalar evaluation of the Chebyshev recurrence (5.2) with different λ_i at each iteration. That is,

$$T_0(L) = 1,\ T_1(L) = \lambda(1),\ T_n(L) = 2\lambda(n)T_{n-1}(L) - T_{n-2}(L), \tag{5.18}$$

where $L = \{\lambda(n)\},\ n \in \mathbb{N}$ is a sequence of arbitrary real numbers, with $\lambda(n) \in \mathbb{R}$ the n^{th} term in the sequence. Specifically, we are interested in the behavior of $|T_n(L)|$.

Proposition 5.5.1 (Upper bound for the time-varying Chebyshev recurrence). *Suppose there exists values $\delta_{\min}$ and $\delta_{\max}$ such that $\lambda(n) \in [\delta_{\min}, \delta_{\max}]$, $\forall n \in \mathbb{N}$, $\delta_{\min} < 0 < \delta_{\max}$ and $|\delta_{\min}| \leq \delta_{\max}$. Then*

$$|T_n(L)| \leq |T_n(L^*)| \tag{5.19}$$

where $L^* = \{\lambda^*(n)\}$ is a succession defined by

$$\lambda^*(n) = \begin{cases} \delta_{\max} & \text{if } n \text{ odd,} \\ \delta_{\min} & \text{if } n \text{ even,} \end{cases} \tag{5.20}$$

Corollary 5.5.2. *If $|\delta_{\min}| > \delta_{\max}$ then the bound in eq.* (5.19) *is true taking $L^* = \{\lambda^*(n)\}$ with*

$$\lambda^*(n) = \begin{cases} \delta_{\max} & \text{if } n \text{ even,} \\ \delta_{\min} & \text{if } n \text{ odd,} \end{cases} \tag{5.21}$$

The previous proposition reveals that the Chebyshev recurrence evaluated in a succession of different real numbers does not keep the behavior shown when it is evaluated with a constant value. The next Lemma provides a bound for the direct expression of this behavior.

Lemma 5.5.3 (Direct expression for the bounded time-varying Chebyshev recurrence). *Let us suppose that the conditions of Proposition 5.5.1 are true. Then*

$$|T_n(L^*)| \leq \kappa_1(\delta_{\max})^n,\ \textit{where}\ \kappa_1(\delta_{\max}) = \delta_{\max} + \sqrt{\delta_{\max}^2 + 1} \tag{5.22}$$

This direct expression (5.22) will be helpful in the development of the convergence analysis dealing with changing matrices and the parameters c and d. We provide now the main result, showing the convergence of the algorithm for the switching case.

Theorem 5.5.4 (Convergence with time-varying topologies). *Allow the communication graph, $\mathcal{G}(n)$, to arbitrarily change in such a way that it is connected for all n, with the weight matrices, $\mathbf{W}(n)$, designed according to Assumption 2.3.3. Let us denote $\lambda_i(n),\ i = 1, \dots, N$, the eigenvalues of $\mathbf{W}(n)$ and*

$$\lambda_{\max} = \max_{n} \max_{i=2,\dots,N} \lambda_i(n), \text{ and } \lambda_{\min} = \min_{n} \min_{i=2,\dots,N} \lambda_i(n). \tag{5.23}$$

Given fixed parameters c and d, a sufficient condition to guarantee convergence to consensus of iteration (5.17) *is*

$$\kappa_1(\max\{|c\lambda_{\max} - d|, |c\lambda_{\min} - d|\})\tau(c - d) < 1. \tag{5.24}$$

The next corollaries give more specific values of λ_M and λ_m, and therefore on c and d, that satisfy the condition in the theorem to achieve convergence.

Corollary 5.5.5 (Convergence with symmetric parameters). *Assume $|c\lambda_{\max} - d| > |c\lambda_{\min} - d|$ and a symmetric assignation, $-\lambda_m = \lambda_M = \lambda$, of the parameters. Then if*

$$\lambda^2 < (1 - \lambda_{\max}^2), \tag{5.25}$$

the algorithm converges.

If we prefer to assign non-symmetric values to the parameters, the following corollary provides a possible assignation that satisfies Theorem 5.5.4.

Corollary 5.5.6 (Convergence with non symmetric parameters). *Assume now that the values of $\lambda_{\max}$ and $\lambda_{\min}$, or some bounds, are known. If λ_M and λ_m satisfy that*

$$\lambda_M + \lambda_m = \lambda_{\max} + \lambda_{\min}, \tag{5.26}$$

and

$$\lambda_M - \lambda_m < \sqrt{4(1 - \lambda_{\max})(1 - \lambda_{\min})}, \tag{5.27}$$

then the algorithm achieves the consensus.

We discuss now in detail the meaning of the theorem and its implications.

Remark 5.5.7 (Sufficient condition). *Note that the theorem provides just a sufficient condition to ensure convergence. This means that although the given bounds seem very restrictive, in practice, even if we choose large values of λ_M and λ_m, there can be convergence. Moreover, an important consequence of corollaries 5.5.5 and 5.5.6 is that, independently on the changes of the network topology, there are always parameters such that the method converges to the consensus.*

Remark 5.5.8 (Comparison with the fixed case). *It is also interesting to note the different behavior of the algorithm when the topology changes with respect to the fixed case. In the latter case, in general it is better to select the parameters λ_M and λ_m with large modulus to ensure that all the eigenvalues of the weight matrix are included in the interval $[\lambda_m, \lambda_M]$. However, in the switching case, it is necessary to choose them small so that $c - d$ is large enough to guarantee convergence. This happens because the more variation on the eigenvalues of the weight matrices, the larger $\kappa_1(\max\{|c\lambda_{\max} - d|, |c\lambda_{\min} - d|\})$ is. Therefore, the larger N, the smaller (in modulus) λ_M and λ_m should be chosen.*

Remark 5.5.9 (Application to general recurrences). *The analysis followed to prove convergence of our algorithm is also interesting because it can be applied to more general consensus algorithms based on recurrences of order greater than one. Given a recurrence similar to* (5.17)*, if a scalar difference equation is found such that its solution bounds the original one in the worst case, a convergence result using the behavior of this recurrence can be obtained.*

Finally, we provide a discussion about the assumptions we have made to proof convergence.

- *Extension to directed graphs:* If the weight matrices are not symmetric, then we cannot ensure that the norm of the matrices used to change the base of eigenvectors is equal to 1. In such a case the convergence condition in Theorem 5.5.4 would be $\zeta\kappa_1(\max\{|c\lambda_{\max}-d|, |c\lambda_{\min}-d|\})\tau(c-d) < 1$, with $\zeta \geq 1$ some positive constant. It is also important to remark that, in this situation, the left eigenvector associated to $\lambda_1(n)$ is not constant anymore for different matrices. This makes the theoretical analysis of the behavior more tedious because at each iteration it is affected by these eigenvectors.

- *Connectivity of the graphs:* The assumption about the connectivity of each graph is more restrictive than in other approaches, e.g., [61], where only joint connectivity is imposed. In our analysis, if one graph is disconnected, then $\lambda_{\max} = 1$ and the sufficient condition (5.24) is never satisfied. This, of course, is caused because we are considering the worst case scenario, so that we can model the behavior of the Chebyshev recurrence as the n^{th} power of some quantity. However, even though we have not been able to prove convergence when some graphs are disconnected, we show in section 5.7 through simulations that in practice the algorithm turns out to be convergent.

5.6 Adaptive Parameters and Algebraic Connectivity Estimation

In this section we present the adaptive consensus algorithm using the Chebyshev Polynomials that leads to the estimation of the algebraic connectivity. The proposed algorithm is based on the bisection method. We describe the process considering a symmetric choice of the parameters, $-\lambda_m = \lambda_M = \lambda$, and therefore, $c = 1/\lambda$ and $d = 0$. The algorithm uses control data to adapt the parameter. For simplicity we explain the algorithm assuming that $\mathbf{x}(0)$ are exactly these control data. The execution of the algorithm with regular initial conditions can be executed in parallel. Each robot randomly initializes $x_i(0) \in \{0, 1\}$. We require the following assumption:

Assumption 5.6.1 (Non-null Fiedler eigenvector). *The initial conditions $\boldsymbol{x}(0)$ expressed as a sum of eigenvectors of $\boldsymbol{W}$, $\boldsymbol{x}(0) = \sum_{i=1,\dots,N} \gamma_i \boldsymbol{v}_i$, satisfy $\boldsymbol{v}_2 \neq \boldsymbol{0}$.*

Let us suppose we run the algorithm (5.9) for a fixed number, n, of iterations choosing $c = 1/\lambda$, with $\lambda \in (0, 1)$. Let us define

$$\kappa_n(c) = T_n(c)\frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty} \tag{5.28}$$

as an indicator of the position of the eigenvalues with respect to the parameter c. The vector $\mathbf{v}_1$ is the eigenvector associated to λ_1, i.e., the average of the initial values and $\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty$ and $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty$ are the initial and current error in the averaging process. The distributed computation of these data is explained later in the text.

Proposition 5.6.2 (Eigenvalues position indicator). *Given λ and $c = 1/\lambda$ as the parameter for the iteration* (5.9)*, if $\lambda_i \in [-\lambda, \lambda]$, $\forall i = 2, \dots, N$, then $\kappa_n(c)$ is bounded by*

$$\kappa_n(c) \leq \frac{\sum_{i=2}^N \|\boldsymbol{v}_i\|_\infty}{\|\sum_{i=2}^N \boldsymbol{v}_i\|_\infty}, \; \forall n. \tag{5.29}$$

Otherwise, at least $\lambda_2 \notin [-\lambda, \lambda]$ and $\lim_{n\to\infty} \kappa_n(c) = \infty$.

Taking this into account, for a sufficiently large n we are able to discern when all the eigenvalues of **W** are contained in the interval $[-\lambda, \lambda]$ and when they are not.

We can now propose the adaptive algorithm to iteratively choose λ and c using $\kappa_n(c)$ and a bisection method. The algorithm starts with an interval defined by the two extremes, $\lambda_{\min}$ and $\lambda_{\max}$, such that

$$\begin{aligned} &\lambda_i \in [-\lambda_{\max}, \lambda_{\max}],\ \forall i = 2, \ldots, N, \\ &\text{at least } \lambda_2 \notin [-\lambda_{\min}, \lambda_{\min}]. \end{aligned} \tag{5.30}$$

If there is no knowledge about the network, the initial values of these parameters can be $\lambda_{\min} = 0$ and $\lambda_{\max} = 1$. Following the bisection approach, the first parameter to run the consensus algorithm (5.9) is $\lambda = (\lambda_{\min} + \lambda_{\max})/2 = 0.5,\ c = 1/\lambda$.

At each consensus round, eq. (5.9) is run for n iterations using c. This number, n, must be chosen in such a way that $\kappa_n(c)$ has time to diverge when there is some eigenvalue outside the range $[-\lambda, \lambda]$. The bound in eq. (5.29) is unknown so an estimation, κ is used in the algorithm. If $\kappa_n(c) \leq \kappa$, we know that all the eigenvalues are contained in the interval $[-\lambda, \lambda]$. On the other hand, if $\kappa_n(c) > \kappa$, it means that there is some eigenvalue outside the range. Once we have detected which of the two situations is happening, the bisection parameters are updated according to it,

$$\begin{aligned} \lambda_{\max} &= \lambda, \text{ if } \kappa_n(c) \leq \kappa \\ \lambda_{\min} &= \lambda, \text{ if } \kappa_n(c) > \kappa. \end{aligned} \tag{5.31}$$

After that the consensus process is repeated, computing at each round a new estimation of the parameter $\lambda = (\lambda_{\min} + \lambda_{\max})/2$.

Proposition 5.6.3 (Convergence of the bisection method). *Assuming n is large enough, the algorithm* (5.31) *is convergent to λ_2, that is* $(\lambda_{\max} + \lambda_{\min})/2 \to \lambda_2$.

Therefore, the adaptive consensus algorithm updates the parameter c of eq. (5.9) to optimize the convergence speed of the method. Moreover, as new values of λ are computed, a better estimation of the algebraic connectivity is available.

Remark 5.6.4 (Computation of λ_N). *Once a good approximation of λ_2 is available the same process can be applied to estimate λ_N. We need to consider again the standard parametrization of* (5.9) *with two parameters. The last estimation, λ, of λ_2 such that $\lambda > \lambda_2$ is assigned to λ_M and the new parameter to adaptively tune is λ_m. The iterative use of* (5.31) *to update λ_m will eventually assign the value of λ_N to this parameter.*

We have not discussed the problem of computing the initial, $\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty$, and final, $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty$, errors or the exact value of $\mathbf{v}_1$ yet. It is also convenient to discuss the selection of κ and n to have convergence guarantee.

The use of control data, composed by integers, allows us to use the ideas presented in Chapter 4 to reach the consensus in finite-time. Let us recall that each robot has initial value of $x_i(0) \in \{0, 1\}$. Therefore the average will be j/N, for some $j = 0, 1, \ldots, N$, and $\mathbf{v}_1 = j/N\mathbf{1}$. Let us assume the number of robots has been computed as explained in Chapter 4.2.2. By using Theorem 4.2.3, the approximation

$$y_i(n) = \frac{1}{N}\left[N x_i(n)\right], \tag{5.32}$$

with $[\cdot]$ a rounding operator to the closest integer, will provide us the exact value of $\mathbf{v}_1$, for all n such that $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty \leq 1/N$.

Since during the first iterations the exact value of $\mathbf{v}_1$ is not available, the errors cannot be computed either. Nevertheless, we can get a good approximation of the infinity norm by

$$\begin{aligned} \|\mathbf{x}(n) - \mathbf{v}_1\|_\infty &\simeq \max_i\{|x_i(n) - y_i(n)| + |y_i(n) - y_i(n-1)|\}, \\ \|\mathbf{x}(0) - \mathbf{v}_1\|_\infty &\simeq \max_i\{|x_i(0) - y_i(n)| + |y_i(n) - y_i(n-1)|\}, \end{aligned}$$

which can be easily computed using a max consensus algorithm in a fixed number of iterations equal to the diameter of $\mathcal{G}$. Since $|x_i(n) - y_i(n)|$ is upper bounded by $1/N$ for all n and i, during the first iterations it will not provide a real estimation of the errors. For that reason the term $|y_i(n) - y_i(n-1)|$ is introduced in the estimation. Also note that, after a finite number of iterations, $\mathbf{y}(n) = \mathbf{y}(n-1) = \mathbf{v}_1$, and the above estimation of the error will be exact whereas using $|x_i(n) - x_i(n-1)|$ it would never be.

The only parameter that cannot be exactly computed is κ, but we can give some more accurate bounds of the value of $\kappa_n(c)$ that can be used by the algorithm as values of κ.

Lemma 5.6.5 (Bound of the estimator). *When $c\lambda_i < 1$, $\kappa_n(c)$ is bounded by $\kappa_n(c) \leq \sqrt{N}$*

Therefore, assigning $\kappa = \sqrt{N}$ will assure that the algorithm will not overestimate the value of λ_2. On the other hand, by choosing a conservative value of κ will require to choose bigger values of n to make $\kappa_n(c)$ diverge. In this way the algorithm also has time to have a good estimation of $y_i(n)$, and therefore of the errors.

The whole process is synthesized in Algorithm 6.

Algorithm 6 Consensus Algorithm with Adaptive Parameters

```
Require: $\kappa$, $n$ and $\mathbf{x}(0)$, s.t. $x_i(0) \in \{0,1\}$
1: Initialize $\lambda_{\min} = 0$, $\lambda_{\max} = 1$
2: while $\lambda_{\max} - \lambda_{\min} >$ tolerance do
3:     $\lambda = (\lambda_{\min} + \lambda_{\max})/2,\ c = 1/\lambda$
4:     for it= $1, \ldots, n$ do
5:         Compute x(it) using (5.9)
6:     end for
7:     Use max consensus to estimate the errors
8:     Compute $\kappa_n(c)$
9:     if $\kappa_n(c) \leq \kappa$ then $\lambda_{\min} = \lambda$
10:    else $\lambda_{\max} = \lambda$
11:    end if
12: end while
```

5.6.1 Variants of the Estimator

We also provide three different variants of the basic algorithm that can be used to improve its behavior. The first two variants can be used to improve the convergence rate to the algebraic connectivity and the third one can be used to detect changes in the communication topology, making the algorithm to converge at each step to the current best parameter.

Direct estimation of λ_2

The first variation we present makes use of $\kappa_n(c)$ to provide a direct estimation of λ_2 when $c\lambda_2 > 1$. Let us recall that in this situation, $\kappa_n(c) \to \infty$ with speed determined by $T_n(c\lambda_2)$. The direct expression of $T_n(c\lambda_2)$ is characterized [83] by

$$T_n(c\lambda_2) = \frac{1+\tau^{2n}}{2\tau^n},\ \tau = c\lambda_2 - \sqrt{(c\lambda_2)^2 - 1}. \tag{5.33}$$

Therefore, the value of λ_2 can be estimated as follows:

1. Compute $\kappa_n(c)$ and assume $\kappa_n(c) \simeq T_n(c\lambda_2)$

2. Using (5.33) compute the value of τ from the second degree equation $2\kappa_n(c)\tau^n = 1 + \tau^{2n}$,

$$\tau = \left(\kappa_n(c) - \sqrt{\kappa_n^2(c) - 1}\right)^{1/n} \tag{5.34}$$

3. Finally, clear λ_2 from (5.33) using (5.34)

$$\lambda_2 \simeq \lambda \frac{\tau^2 + 1}{2\tau}. \tag{5.35}$$

Note that the value obtained is still an approximation of the real value of λ_2. Therefore, the bisection iteration still needs to be executed. Whenever a direct estimation of λ_2 is available, the interval is updated by

$$\begin{aligned} \lambda_{\max} &= \lambda_2 + \min(\lambda_{\max} - \lambda_2, \lambda_2 - \lambda_{\min}), \\ \lambda_{\min} &= \lambda_2 - \min(\lambda_{\max} - \lambda_2, \lambda_2 - \lambda_{\min}), \\ \lambda &= \lambda_2, \end{aligned} \tag{5.36}$$

where λ_2 here is the estimation obtained in (5.35). In this way the interval to look for the algebraic connectivity is significantly reduced and so is the number of iterations and consensus rounds.

In order to have a good direct estimation of λ_2, it is desirable to have $T_n(c\lambda_i)\mathbf{v}_i/T_n(c) \simeq 0,\ i = 3, \ldots, N$, or at least $|T_n(c\lambda_2)\mathbf{v}_2| \gg |T_n(c\lambda_i)\mathbf{v}_i|$. To make this happen, at each new consensus round we update the initial conditions by $\mathbf{x}(0) = \mathbf{x}(n)$. With this update the average, $\mathbf{v}_1$, is preserved, but the initial conditions are closer to it, which is the same as to say that $\mathbf{v}_i$ is closer to zero. In addition, the estimations of the errors are also improved by the update because $y_i(n)$ will be closer to the average. Therefore, we are also obtaining a more exact value of $\kappa_n(c)$, improving even more the estimation.

However, we only do the update when $c\lambda_2 > 1$. The reason is that the convergence to zero is faster for the eigenvalues contained within $[-\lambda, \lambda]$ than for those outside the interval. Since λ_2 is not contained in the interval, $\mathbf{v}_2$ is not reduced as much as the other eigenvectors and $\mathbf{v}_2 \gg \mathbf{v}_i$. If we update the initial conditions when $c\lambda_2 < 1$, it is possible that the component associated to $\mathbf{v}_2$ is reduced by a larger factor than for other eigenvectors.

Speed up using k-section method

The bisection method has the property of reducing the estimation error, $|\lambda - \lambda_2|$, by a constant factor of 0.5. In robotic networks, the cost of sending several small messages is usually bigger than the cost of sending a unique message with more information. Taking this into account our method can execute several copies of the consensus algorithm in parallel with different parameters. In this way, the bounds of λ_2 are delimited with more accuracy and the optimal convergence rate is reached sooner. Specifically, given $\lambda_{\min}$ and $\lambda_{\max}$, the k-section method executes $k - 1$ consensus in parallel with parameters

$$\lambda_j = j(\lambda_{\min} + \lambda_{\max})/k,\ c_j = 1/\lambda_j,\ j = 1, \ldots, k - 1. \tag{5.37}$$

Once all the different estimations of $\kappa_n(c_j)$ have been computed, the interval to consider in the next consensus iteration is defined by

$$\begin{aligned} \lambda_{\min} &= \max_j \lambda_j \text{ s.t. } \kappa_n(c_j) \leq \kappa, \\ \lambda_{\max} &= \min_j \lambda_j \text{ s.t. } \kappa_n(c_j) > \kappa. \end{aligned} \tag{5.38}$$

With this algorithm, the size of the messages exchanged by the nodes will increase in $k - 1$ additional elements instead of the one sent by the bisection method. However, the error in the estimation of λ_2 will be reduced by a factor of $1/k$ after each update in the estimation.

Detecting changes in the communication topology

Since the evaluation of the Chebyshev polynomial requires the topology to be fixed, we impose that during the n iterations used to estimate $\kappa_n(c)$ the topology remains fixed. However, let us assume that within different consensus rounds the communication topology can change. If we consider a network of mobile sensors, this situation could appear, for example, making the sensors remain static during the computation of $\mathbf{x}(n)$ and letting them move during the computation of $\kappa_n(c)$.

Whenever the eigenvalue λ_2 is contained in the interval $[\lambda_{\min}, \lambda_{\max}]$, the standard method will converge to the right value, even if it changes between estimations of $\kappa_n(c)$. However, as we approach to the algebraic connectivity, the interval $[\lambda_{\min}, \lambda_{\max}]$ will be small, and it will be very likely that a change of the topology displaces λ_2 outside of it. In such case, the standard method will not converge to λ_2.

Using a similar approach to the k-section method we can detect when the eigenvalue we are looking for has left the considered interval. Let us define $c_{\min} = 1/\lambda_{\min}$ and $c_{\max} = 1/\lambda_{\max}$. If we run the consensus iteration in parallel for the three parameters, $c_{\min}, c$ and $c_{\max}$, and $\lambda_2 \in [\lambda_{\min}, \lambda_{\max}]$, it must hold that $\kappa_n(c_{\min}) > \kappa$ and $\kappa_n(c_{\max}) \leq \kappa$. If one of these conditions does not hold we will know that the topology has changed and the algebraic connectivity has left the interval.

In case $\kappa_n(c_{\max}) < \kappa$ that means that λ_2 is above the interval and we can use (5.35) to obtain a direct estimation of it. If $\kappa_n(c_{\min}) \geq \kappa$ then the value of the algebraic connectivity is below the interval we are analyzing. In this case we do not have any means to estimate an approximate value. The policy we follow is to assign $\lambda_{\max} = \lambda_{\min}$ and $\lambda_{\min} = 0$. Although it is a conservative policy, it ensures that the eigenvalue we are looking for is again within the interval.

5.7 Simulations

We have analyzed our algorithm in a simulated environment. Monte Carlo experiments have been designed to study the convergence of the algorithm using Chebyshev polynomials and the influence of the parameters λ_m and λ_M in the convergence properties.

5.7.1 Evaluation with a fixed communication topology

In a first step we study the algorithm when the topology of the network is fixed. We analyze the convergence speed for different weight matrices, comparing Chebyshev polynomials with other approaches.

In the experiments we have considered 100 random networks of 100 nodes. For each network the nodes have been randomly positioned in a square of 200×200 meters. Two nodes communicate if they are at a distance less than 20 meters. The networks are also forced to be connected so that the algorithms converge. After that, 100 different random initial values have been generated in the interval $(0, 1)^N$, giving a total of 10000 trials to test each of the algorithms.

Convergence speed

For each communication network we have computed 4 different weighted adjacency matrices. The first one, $\mathbf{W}_{ld}$, uses the "local degree weights", the second one, $\mathbf{W}_{bc}$, uses the "best constant factor" and the third one, $\mathbf{W}_{os}$, computes the "optimal symmetric weights". This computation is done running an optimizer on the second largest eigenvalue with the "best constant factor" as the initial approximation to the solution of the problem. For more information about these matrices we refer to [153]. These three matrices are symmetric, for that reason we have included in the experiment a fourth non-symmetric matrix, $\mathbf{W}_{ns}$, computed by $w_{ij} = 1/(\mathcal{N}_i + 1)$ if $j \in \mathcal{N}_i \cup i$ and $w_{ij} = 0$ otherwise.

We have compared our method using Chebyshev polynomials (*Chebyshev*) with the standard method that computes the powers of the matrices using (2.3) (*Standard*), the polynomial of degree 4 proposed in [68] using semi-definite programming optimization (*SDP*), and the second order recurrence with fixed weights proposed in [129], $F_n(x) = \beta x F_{n-1}(x) + (1-\beta)F_{n-2}(x)$ (*SOFixed*). In the last method we have used the value $\beta = 2/(1+\sqrt{1-\lambda_2^2})$, which gives the best convergence rate. For the Chebyshev polynomials we have also assigned the optimal parameters $\lambda_M = \lambda_2$ and $\lambda_m = \lambda_N$. We have measured the average number of iterations required to obtain an error smaller than a given tolerance.

Table 5.1 shows the results of the experiment. For any matrix our algorithm reaches the consensus faster than all the other algorithms. Moreover, considering that the initial error is upper bounded by 1, note that our algorithm is able to reduce the error by five orders of magnitude (10^{-5}) in around $N = 100$ iterations ($103.0, 109.9, 103.4$ and 94.1 iterations in the table), which is the size of the network.

Table 5.1: Number of iterations for different algorithms and tolerances

Method\Tolerance	10^{-2}	10^{-3}	10^{-4}	10^{-5}	Method\Tolerance	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Standard($\mathbf{W}_{ld}$)	396.1	899.0	1422.9	1902.9	SDP($\mathbf{W}_{ld}$)	149.9	248.8	349.1	450.1
Standard($\mathbf{W}_{bc}$)	470.5	892.4	1307.4	1691.5	SDP($\mathbf{W}_{bc}$)	170.7	283.1	396.4	510.5
Standard($\mathbf{W}_{os}$)	390.8	735.1	1092.0	1446.0	SDP($\mathbf{W}_{os}$)	166.2	265.5	372.4	489.9
Standard($\mathbf{W}_{ns}$)	308.9	698.4	1116.7	1521.2	SDP($\mathbf{W}_{ns}$)	305.4	414.8	484.9	555.7
SOFixed($\mathbf{W}_{ld}$)	45.7	71.9	98.0	124.2	Chebyshev($\mathbf{W}_{ld}$)	41.8	62.2	82.6	103.0
SOFixed($\mathbf{W}_{bc}$)	45.2	67.4	91.2	114.6	Chebyshev($\mathbf{W}_{bc}$)	44.6	66.4	88.1	109.9
SOFixed($\mathbf{W}_{os}$)	42.2	62.9	83.3	103.6	Chebyshev($\mathbf{W}_{os}$)	42.1	62.6	83.0	103.4
SOFixed($\mathbf{W}_{ns}$)	40.8	63.9	86.8	109.8	Chebyshev($\mathbf{W}_{ns}$)	38.6	57.1	75.6	94.1

An interesting detail is that our algorithm converges faster using the "local degree weights", $\mathbf{W}_{ld}(103.0)$, and the "non-symmetric weights", $\mathbf{W}_{ns}(94.1)$, than using the other two matrices (109.9 and 103.4), even though the second largest eigenvalue of $\mathbf{W}_{bc}$ and $\mathbf{W}_{os}$ is smaller. This behavior happens because the eigenvalues of $\mathbf{W}_{bc}$ and $\mathbf{W}_{os}$ are symmetrically placed with respect to zero whereas for $\mathbf{W}_{ld}$ and $\mathbf{W}_{ns}$ $|\lambda_N| < \lambda_2$ (an example can be found in [153]). As a consequence, $c-d$ is larger and the algorithm converges faster. This is indeed very convenient because the "local degree weights" and the "non-symmetric weights" can be easily computed in a distributed way without global information, whereas the other two require the knowledge of the whole topology.

Regarding the non-symmetric weights, we have observed that λ_2 is, in general, small compared to the second eigenvalue of the symmetric matrices. Since the eigenvalues of $\mathbf{W}_{ns}$ also satisfy that $|\lambda_N| < \lambda_2$, the convergence for this matrix is the fastest. Also note that these matrices are the easiest to compute. On the other hand, when using symmetric weight matrices the convergence value is known to be the average of the initial conditions whereas when using non-symmetric weights the convergence value depends on the matrix.

Dependence on the parameters λ_M and λ_m

So far we have evaluated the convergence speed of our algorithm only considering the optimal parameters, which implies the knowledge of the eigenvalues of the weight matrix. However, in most situations the nodes will have no knowledge about these eigenvalues. We analyze now the convergence rates of Chebyshev polynomials algorithm when it is run using sub-optimal parameters. In this case, for simplicity we have only considered $\mathbf{W}_{ld}$ in the experiment.

The results are in Table 5.2. The table shows the average number of iterations required to have an error lower than 10^{-3}. The number of iterations is in all the cases larger than in Table 5.1 (62.2 iterations) but the results are in most cases still good because the number of iterations is smaller than using the powers of $\mathbf{W}_{ld}$ (899.0 iterations in Table 5.1). The results compared to *SOFixed* evaluated with the optimal parameter (71.9 it. in Table 5.1) seem to be poor. However, the optimal β requires the knowledge of λ_2 which, right now,

Table 5.2: Number of iterations using sub-optimal parameters and tolerance 10^{-3}

$\lambda_m \backslash \lambda_M$	0.5	0.8	0.9	0.95	0.999
-0.5	630.4	397.2	279.0	194.5	75.9
-0.8	690.6	435.2	305.6	213.1	83.1
-0.9	709.5	447.0	314.0	219.0	85.4
-0.95	718.8	453.0	318.1	221.8	86.5
-0.999	726.0	457.6	321.3	224.0	87.4
β	0.5	0.8	0.9	0.95	0.999
SOFixed	672.4	463.9	320.9	227.1	93.0

we are assuming it is unknown. For that reason, in the last row of Table 5.2 we have included the results using F_n evaluated with $\beta = 2/(1 + \sqrt{1 - \lambda_M^2})$, i.e., with the same estimation of λ_2 used for the Chebyshev polynomials. In this case we observe again that both methods present a similar performance when using the same parameters. The degree of freedom given by λ_m is what differs in the algorithms. By adjusting this parameter we can reduce the number of iterations in our algorithm.

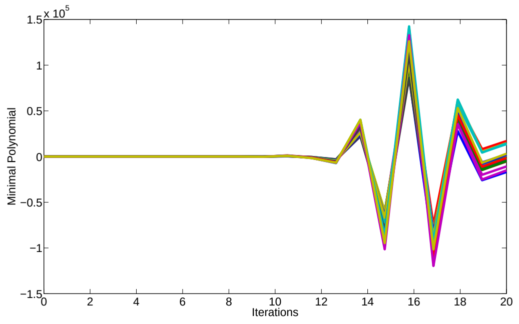
Another advantage of using Chebyshev polynomials with the weight matrix $\mathbf{W}_{ld}$, besides the computation using local information, is that usually its smallest eigenvalue, λ_N, is a negative value close to zero (in our simulations it has never valued less than -0.5). The second largest eigenvalue depends on how many nodes has the network and the number of links, but in general this eigenvalue is close to one. Therefore by choosing $\lambda_m = -0.5$ and $\lambda_M \simeq 1$ there is a great chance to obtain a good convergence rate and almost no risk of divergence, see for example the cell in the second row and sixth column of Table 5.2 (75.9). A safer choice of parameters is $\lambda_m = -\lambda_M$, which we know that always converges. In this case we can see that the larger the value of λ_M, the fastest the convergence rate.

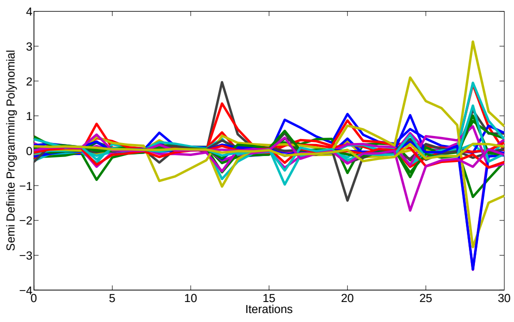
5.7.2 Evaluation with a switching communication topology

We see now how the Chebyshev polynomials work when the topology of the network changes at different iterations. We start by showing the convergence in an illustrative example where the conditions of Theorem 5.5.4 are satisfied. After that, we run again Monte Carlo experiments to analyze the algorithm in more realistic situations.

Illustrative Example

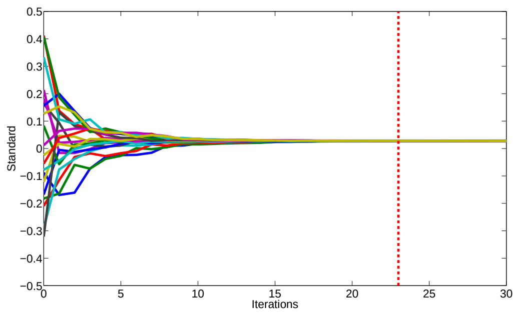
We have considered a connected communication network composed by 20 nodes. In order to satisfy the conditions of Theorem 5.5.4 at each iteration we have randomly added some links to the network. In this way all the topologies remain connected and the parameters $\lambda_{\max}$ and $\lambda_{\min}$ correspond to the second maximum and the smallest eigenvalues of the initial weight matrix. Using the local degree weights, which return a symmetric matrix, these parameters are $\lambda_{\max} = 0.9477$ and $\lambda_{\min} = -0.1922$. In Figure 5.4 we show the evolution of the state using different algorithms and Chebyshev polynomials with different parameters. In the first row we can see that the finite-time consensus algorithm using the minimal polynomial [138] and the semi-definite programming polynomial [68] do not reach the consensus when the topology changes. On the other hand, the standard method and the second order fixed recurrence achieve the consensus. The vertical line represents when the algorithms reach the consensus with a tolerance smaller than 10^{-3}. In the second row of Fig. 5.4 we show the evolution of the state using Chebyshev polynomials with parameters defined, from left to right, in Corollary 5.5.5, $\lambda_M = -\lambda_m = 0.3190$, Corollary 5.5.6, $\lambda_M = 0.6274$, $\lambda_m = 0.1282$, non-symmetric with good convergence rate for fixed topology, $\lambda_M = 0.9, \lambda_m = -0.5$, and symmetric with large modulus,

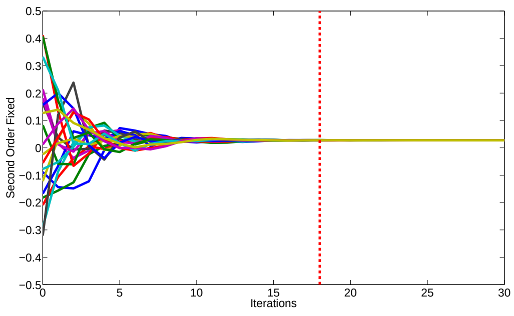




Minimal polynomial

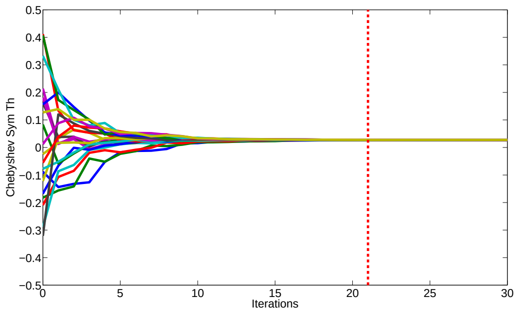
SDP polynomial

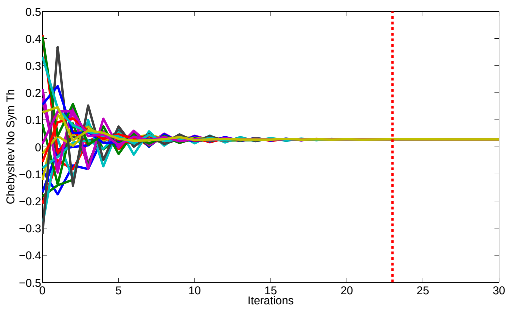




Standard

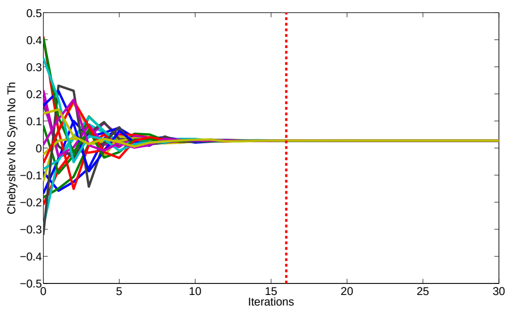
SO Fixed

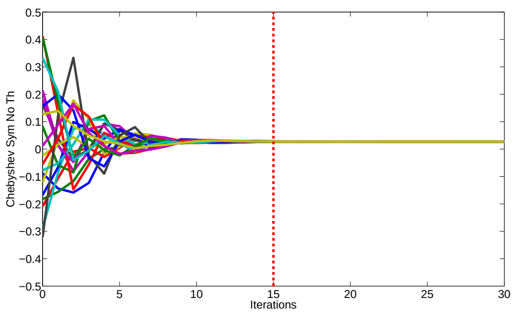




$\lambda_M = -\lambda_m = 0.3190$

$\lambda_M = 0.6274,\ \lambda_m = 0.1282$





$\lambda_M = 0.8,\ \lambda_m = -0.8$

$\lambda_M = 0.9,\ \lambda_m = -0.5$

Figure 5.4: Convergence speed of different algorithms with a switching communication topology. The minimal polynomial and the SDP polynomial do not reach the consensus when the topology changes. On the other hand, the standard method and the second order fixed recurrence achieve the consensus. The vertical line represents when the algorithms reach the consensus with a tolerance smaller than 10^{-3}. The second row shows the use of Chebyshev polynomials with the parameters of Corollaries 5.5.5 ($\lambda_M = -\lambda_m = 0.3190$) and 5.5.6 ($\lambda_M = 0.6274,\ \lambda_m = 0.1282$), and parameters with good behavior in fixed topology ($\lambda_M = 0.8,\ \lambda_m = -0.8$ and $\lambda_M = 0.9,\ \lambda_m = -0.5$). Note that in these cases the convergence to the consensus is also faster than using other methods.

$\lambda_M = 0.8, \lambda_m = -0.8$. The last two examples show that the condition of Theorem 5.5.4 is a sufficient condition and that the algorithm also converges to the consensus choosing parameters with large modulus. Moreover, note that in these last cases the convergence is also faster than using the other tested methods.

Convergence depending on the evolution of the network and the parameters

We have generated again 100 random networks of 100 nodes like in the fixed topology case. To model the changes in the communication topology we have considered three different scenarios in the experiment. The first one assumes a fixed initial communication topology and, at each iteration the links can fail with constant probability equal to 0.05 (Link Failures). This is a usual way to model networks with unreliable or noisy communications. In the second scenario we consider a set of mobile agents that randomly move in the environment. In this way, at each iteration the communication topology evolves with the proximity graph defined by the new positions of the agents (Evolution with Motion). The last scenario assumes a new random network at each iteration (Random Network). Although in real situations may be uncommon, it is interesting to analyze it in order to study the properties of our algorithm. In the three scenarios we have used the local degree weights to define the weight matrix at each iteration. We have not worried about the network connectivity, letting the experiment to possibly have several iterations with disconnected networks. We have set a maximum of 3000 iterations per trial.

Table 5.3: Iterations for Link Failures (LF), Evolution with Motion (EWM) and Random Networks (RN)

$\lambda_m \backslash \lambda_M$	0.5			0.75			0.9			0.95		
	LF	EWM	RN	LF	EWM	RN	LF	EWM	RN	LF	EWM	RN
-0.5	≥ 3000	1765.2	8.9	1328.6	665.6	11.6	418.9	461.9	22.3	293.5	306.5	42.1
-0.75	≥ 3000	1793.5	9.6	1356.6	703.6	11.8	452.3	506.3	21.7	316.8	309.8	37.8
-0.9	≥ 3000	1813.0	10.0	1321.0	708.5	12.0	470.9	564.9	21.7	330.0	311.0	36.8
-0.95	≥ 3000	1818.0	10.1	1326.4	710.4	12.0	476.9	564.9	21.7	334.5	311.5	36.5
Standard	LF = 1087.2				EWM = 1032.4					RN = 9.4		

The number of iterations required to achieve the consensus with accuracy 10^{-3} using Chebyshev polynomials with different parameters is shown in Table 5.3. In the last row of the Table we have included the number of iterations the standard method requires to converge with the same precision. Similarly to the case with fixed topology, an appropriate choice of the parameters leads to convergence rates faster than the standard method, e.g., considering Link Failures (LF) and choosing $\lambda_M = 0.95$ and $\lambda_m = -0.5$ only 293.5 iterations are required, compared to the 1087.2 iterations required with the standard method (more than 4 times faster). On the other hand, we can see that a wrong choice of the parameters may lead to a slow convergence rate. This is the case, for example, of the cells with "$\geq$ 3000" iterations (around 4 times slower). Nevertheless, even in these cases the algorithm still is able to achieve the consensus, even when the conditions required in Theorem 5.5.4 are violated, whereas we already know that other polynomial approaches fail to converge.

It is also surprising which parameters achieve the fastest convergence in the different scenarios. For the Link Failures and the Evolution with Motion, the best parameters are exactly the worst parameters for the Random Networks scenario, i.e., $\lambda_M = 0.95$ and $\lambda_m = -0.5$. On the other hand, the best parameters for the Random Networks are those who give the slowest convergence rate for the other two scenarios. The explanation for this phenomenon appears in the variability of the eigenvectors of the weight matrices. When the topology changes arbitrarily at each iteration, there is a great variability in the eigenvectors of the weight matrices, which turns out in a great variability of $\mathbf{x}(n)$. This situation is closer to the worst case we have shown in section IV to proof the convergence of the algorithm. Therefore, a good convergence rate requires a large value of $c - d$, achieved when λ_M and λ_m have small modulus. When the topology changes smoothly, as in the Link Failures and the Motion Evolution, the eigenvectors do not change much and the algorithm behaves similarly to the fixed case. For that reason, the parameters that achieve the best convergence rate are the same as in the fixed case. However, we must be careful because for large values of λ_M the algorithm may diverge.

A final detail is that, in all the cases, the convergence seems to be more affected by λ_M than λ_m. This is explained by the use of the local degree weights. As we have mentioned earlier, these matrices do not have symmetric eigenvalues with respect to zero. In these matrices $\lambda_{\max}$ dominates the convergence rate, so the convergence is more sensible to the parameter λ_M.

In conclusion, when the topology of the network changes, the parameters should be chosen taking into account the nature of these changes. For small changes similar parameters to the fixed case should be assigned whereas if the network is expected to change a lot we should pick small parameters for the algorithm to guarantee convergence.

5.7.3 Evaluation of the adaptive parameter estimation

In the simulations to estimate the algebraic connectivity we have generated networks of different sizes. For each different number of robots, we have tried a hundred random networks. In each network the robots have been randomly positioned in a square of 200×200 meters. The communication radius has been set again to 20 meters. The networks have also been forced to be connected. Ten different random initial values have been tested for each network, giving a total of 1000 trials for each number of robots. In all the experiments the matrix **W** has been computed using the "local degree weights".

The value of κ has been set to $\sqrt{N}$ and we have used the value of c in order to decide the number of iterations n executed at each round. We have chosen the minimum n such that $T_n(c) > 100$. In this way, the algorithm always executes enough iterations to converge to the right value.

The results obtained using the bisection method are in Table 5.4. The second column (λ_2) shows the mean algebraic connectivity of all the networks analyzed and the third column (Diam) shows the mean diameter. The column "Rounds" represents the number of estimations of λ before satisfying that $\lambda_{\max} - \lambda_{\min} \leq 10^{-2}$ Therefore, the method is expected to have a tolerance of 10^{-2} in the estimation of λ_2. Since the error is reduced by the same factor at each round, using this method the number of rounds is constant for any size of the network. The total number of iterations (including the max consensus to estimate the errors) is written in the column "Iter" whereas next column (n) shows only the iterations required for the consensus part. Finally, the last column shows the mean of the estimations, λ.

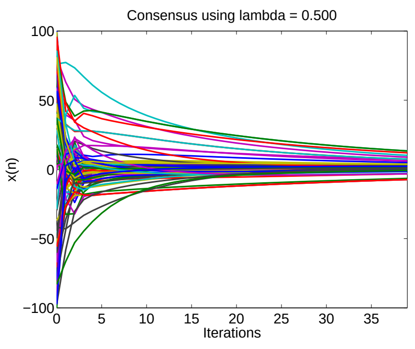
Table 5.4: Results for the standard bisection method

N	λ_2	Diam	Rounds	Iter	n	λ
10	0.910	4.68	6	118.08	90.00	0.917
50	0.982	10.52	6	170.56	107.45	0.978
100	0.985	13.62	6	198.67	116.95	0.982
250	0.992	18.64	6	240.84	129.00	0.993
500	0.992	43.00	6	376.40	150.20	0.994

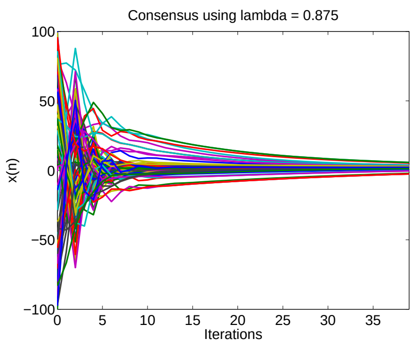
Looking at the results, we can extract some interesting conclusions. First of all, the large values of the mean λ_2 indicate that consensus algorithms evaluated in these networks will require a large number of iterations to achieve consensus. Although our method uses several consensus rounds to estimate the algebraic connectivity, each one of these rounds requires a considerably smaller number of iterations than a standard consensus method. Since the algorithm is intended to be executed at the same time as the standard consensus with real data, we conclude that within one consensus execution with real data we will have the algebraic connectivity.

It is also remarkable how well the method escalates with the size of the network. For large networks in less than N iterations the algorithm reaches a good estimation of λ_2. Another thing to remark is that the number of iterations used in the max consensus represents a large fraction of the total, specially as N grows. This happens because the choice of n does not depend on the size of the network but on the connectivity. Therefore, for large networks, the bottleneck of the algorithm, in terms of communications, is the diameter of $\mathcal{G}$.

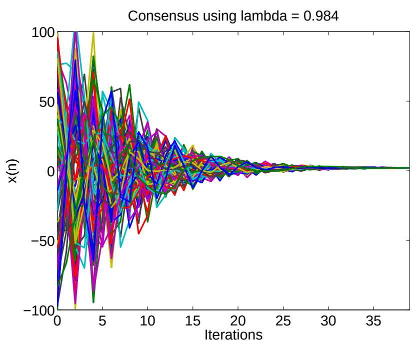
Figure 5.5 depicts three executions of the adaptive consensus in a network of 100 nodes using real initial conditions. The three pictures consider the same initial conditions, the same number of iterations and different input parameters, estimated using bisection. It can be seen that as new estimations of λ are computed, the consensus is reached faster.



(a) Round 1



(b) Round 3



(c) Round 6

Figure 5.5: Adaptive consensus with parameter estimation. Evolution of $\mathbf{x}(n)$ using the same initial conditions and number of iterations but different parameters estimated using pure bisection. The algebraic connectivity of the network is 0.987. As new estimations of λ are computed, the consensus is reached faster.

Table 5.5: Results with direct estimation of λ_2

N	λ_2	Diam	Rounds	Iter	n	λ
10	0.910	4.68	4.40	86.62	66.20	0.915
50	0.982	10.52	4.02	118.68	76.60	0.979
100	0.985	13.62	4.45	155.02	94.36	0.982
250	0.992	18.64	5.23	215.66	118.31	0.994
500	0.991	43.00	5.10	322.00	129.72	0.994

In Table 5.5 we show the results for the same experiment but using the direct estimation of λ_2 presented in section 5.6.1. In this case the number of rounds depends on the computed values of λ_2. Note that this number is smaller than the number of rounds required by the standard bisection to obtain the same tolerance error. As a consequence the number of iterations is also reduced. Finally, it is worth noticing that the estimation of λ_2 is also very precise.

Estimation with a switching topology

To end, we show an experiment where we allow the communication graph to change, showing that our method tracks the algebraic connectivity using the k-section method.

We have considered a random network composed by 50 nodes. Since during the computation of $\mathbf{x}(n)$ the network must be static, we have only modified the network during the estimation of the errors. The evolution of the estimations is depicted in Fig. 5.6. We can see that the algorithm detects the changes in the topology and adjust the intervals in consequence. If the topology remains fixed for enough iterations, the method estimates the value of the algebraic connectivity. When the algebraic connectivity is reduced, instead of assigning $\lambda_{\min} = 0$ we have used $\lambda_{\min} = \lambda_{\max} - 0.1$. Although sometimes, e.g., just before iteration 200, more than one consensus round is required to adapt the interval, with this assignment, the convergence is in general faster to the real value of λ_2.

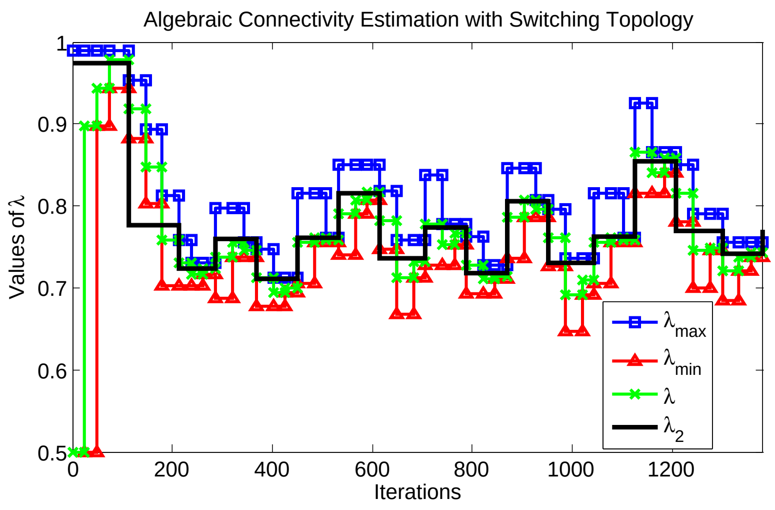


Figure 5.6: Estimation of the algebraic connectivity under a switching communication network. The algorithm detects the changes and adapts the interval to estimate at each round the algebraic connectivity.

5.8 Discussion

Summing up, we have proposed a linear iteration using Chebyshev polynomials of first kind that requires a small number of iterations to reach the consensus. In this way, we can compensate the extra communication load that visual information usually requires, or the voting for multiple hypotheses to reach a robust consensus using *De-RANSAC*.

The proposed algorithm has the same characteristics of a standard linear iteration. The robots need to send the same amount of data and only need to compute sums and products of the variables. The main difference is the use of a second order recurrence iteration, which has small additional space requirements in the memory of the robots but significantly accelerates the convergence rate compared to existing approaches. In the chapter we have characterized the speed up with theoretical results and extensive simulations considering fixed and switching communication topologies.

Our algorithm requires two additional input parameters that define the convergence speed of the method. We have analyzed the influence of these parameters in the convergence speed, showing that the fastest version of our algorithm is achieved when they correspond with the smallest and the second largest eigenvalues of the weight matrix. For some specific topologies like trees or regular graphs we have given direct expressions to assign these parameters. For general topologies, the exact value of the parameters requires the knowledge of the whole network topology. For that reason we have also proposed a distributed solution to estimate the parameters using a bisection technique. The result is an adaptive distributed linear iteration able to reach the maximum convergence speed.

With this chapter we conclude the first part of the Thesis, where we have focused on modifying the standard consensus algorithm so that it can be used with visual information of any type. It is clear that the choice of the appropriate representation of the information is another important factor to achieve a good consensus. For example, in perception tasks, the use of line descriptors extracted from the images will have different implications in the computation of the consensus than the use of SURF descriptors. In the rest of the Thesis we will study how the consensus computation can be simplified in perception and control tasks by using well known computer vision methods.

Proofs

Proof of Theorem 5.4.1 (Convergence of the Algorithm)

We introduce an auxiliary result to prove the convergence.

Lemma 5.8.1. *Given $x_1 > 1,$ for any x_2 such that $|x_2| < x_1$ it holds that*

$$\lim_{n\to\infty} \frac{T_n(x_2)}{T_n(x_1)} = 0. \tag{5.39}$$

Proof. For $|x_2| \leq 1$, $|T_n(x_2)| \leq 1$, $\forall n$, and since $T_n(x_1) \to \infty$ with n, eq. (5.39) is true. Now, if $1 < |x_2| < x_1$, then using (5.5) we have

$$\frac{T_n(x_2)}{T_n(x_1)} = \frac{\tau(x_1)^n}{\tau(x_2)^n}\frac{1+\tau(x_2)^{2n}}{1+\tau(x_1)^{2n}}. \tag{5.40}$$

But in this case, $1 > |\tau(x_2)| > \tau(x_1) > 0$ and the result holds immediately.

Proof of Theorem 5.4.1. Let $\mathbf{Q} = \mathbf{W} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, whose eigenvalues are $\lambda_1 = 0$, with $\mathbf{v}_1 = \mathbf{1}$ its corresponding eigenvector, and $\lambda_2, \ldots, \lambda_N$ with the same eigenvectors as $\mathbf{W}$. Considering that $\gamma_1 = 1/N\mathbf{1}^T\mathbf{x}(0)$, then $\frac{1}{N}\mathbf{1}\mathbf{1}^T(\mathbf{x}(0) - \gamma_1\mathbf{v}_1) = 0$. Taking this into account it is easy to see that

$$\mathbf{W}^k(\mathbf{x}(0) - \gamma_1\mathbf{v}_1) = \mathbf{Q}^k(\mathbf{x}(0) - \gamma_1\mathbf{v}_1), \quad \forall k \in \mathbb{N}, \tag{5.41}$$

and therefore $P_n(\mathbf{W})(\mathbf{x}(0)-\gamma_1\mathbf{v}_1) = \sum_{k=0}^n \alpha_k\, \mathbf{W}^k(\mathbf{x}(0)-\gamma_1\mathbf{v}_1) = \sum_{k=0}^n \alpha_k\, \mathbf{Q}^k(\mathbf{x}(0)-\gamma_1\mathbf{v}_1) = P_n(\mathbf{Q})(\mathbf{x}(0)-\gamma_1\mathbf{v}1)$, with α_k the coefficients of P_n.

Also $\mathbf{W}\mathbf{v}_1 = \mathbf{v}_1$ and $P_n(1) = 1$, then $P_n(\mathbf{W})\mathbf{v}_1 = \mathbf{v}_1$ and

$$\|\mathbf{x}(n) - \gamma_1\mathbf{v}_1\|_2 = \|P_n(\mathbf{W})(\mathbf{x}(0) - \gamma_1\mathbf{v}_1)\|_2 = \|P_n(\mathbf{Q})(\mathbf{x}(0) - \gamma_1\mathbf{v}_1)\|_2 \leq \|P_n(\mathbf{Q})\|_2\|\mathbf{x}(0) - \gamma_1\mathbf{v}_1\|_2. \tag{5.42}$$

In addition, $\mathbf{Q}$ is symmetric, and so is $P_n(\mathbf{Q})$, which implies that its spectral norm coincides with the spectral radius,

$$\|P_n(\mathbf{Q})\|_2 = \rho(P_n(\mathbf{Q})) = \max_{i\neq 1}|P_n(\lambda_i)| = \max_{i\neq 1}\frac{|T_n(c\lambda_i - d)|}{T_n(c-d)}. \tag{5.43}$$

For any $x \in (\lambda_M + \lambda_m - 1, 1)$ we have that $|cx - d| < c - d$, then for all the eigenvalues of $\mathbf{W}$ but λ_1, $|c\lambda_i - d| < c - d$. Finally, noting that $c - d$ is strictly larger than 1, by Lemma 5.8.1 $P_n(\lambda_i) \to 0$ for all $i \neq 1$, which proves the convergence of the algorithm. ∎

Proof of Theorem 5.4.2 (Optimal Parameters)

In order to prove Theorem 5.4.2 we will use the following auxiliary results.

Lemma 5.8.2. *Let λ_m, λ_M such that $[\lambda_N, \lambda_2] \not\subseteq [\lambda_m, \lambda_M]$ and $|c\lambda_N - d| < c\lambda_2 - d$. Then, for fixed c, $\nu(c,d)$ is a decreasing function of d.*

Proof. Let us see that $\partial\nu(c,d)/\partial d < 0$.

$$\nu(c,d) = \frac{\tau(c-d)}{|\tau(c\lambda_2 - d)|} = \frac{\tau(c-d)}{\tau(c\lambda_2 - d)} > 0$$

Then

$$\frac{\partial\nu}{\partial d} = \frac{-\tau'(c-d)\tau(c\lambda_2 - d) + \tau(c-d)\tau'(c\lambda_2 - d)}{\tau(c\lambda_2 - d)^2}.$$

But since for $x > 0$, $\tau'(x) = -\tau(x)/\sqrt{x^2-1}$, then

$$\frac{\partial \nu}{\partial d} = \frac{\tau(c-d)}{\tau(c\lambda_2 - d)}\left[\frac{1}{\sqrt{(c-d)^2-1}} - \frac{1}{\sqrt{(c\lambda_2-d)^2-1}}\right]$$

which is negative because $1 < (c\lambda_2 - d)^2 < (c-d)^2$.

Lemma 5.8.3. *Let $\lambda_m,\ \lambda_M$ such that $[\lambda_N, \lambda_2] \not\subseteq [\lambda_m, \lambda_M]$ and $|c\lambda_N - d| > |c\lambda_2 - d|$ with $c\lambda_N - d < 0$. Then, for fixed c, $\nu(c, d)$ is an increasing function of d.*

Proof. The proof is very similar to that of Lemma 5.8.2 and is omitted.

Proposition 5.8.4. *Let $\lambda_m,\ \lambda_M$ such that $\lambda_M - \lambda_m = 2/c$ is fixed and $[\lambda_N, \lambda_2] \not\subseteq [\lambda_m, \lambda_M]$. Then*

i) If $\lambda_2 - \lambda_N > \lambda_M - \lambda_m$, $\nu(c, d) \geq \nu(c, d^*)$, d^* being the value such that $\lambda_M + \lambda_m = \lambda_2 + \lambda_N$, that is, for a fixed c, $\nu(c, d)$ is minimum when λ_m, λ_M are symmetrically placed with respect to λ_N, λ_2.

ii) If $\lambda_2 - \lambda_N \leq \lambda_M - \lambda_m$ and $\lambda_M < \lambda_2$ then $\nu(c, d) \geq \nu(c, d^*)$, d^* being such that $\lambda_M = \lambda_2$, and in this case $[\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M]$

iii) If $\lambda_2 - \lambda_N \leq \lambda_M - \lambda_m$ and $\lambda_m > \lambda_N$ then $\nu(c, d) \geq \nu(c, d^*)$, d^* being such that $\lambda_m = \lambda_N$, and in this case $[\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M]$

Proof. i) The result follows from Lemmas 5.8.2 and 5.8.3. If $\lambda_2 > \lambda_M$, then $c\lambda_2 - d > |c\lambda_N - d|$ and $\nu(c, d)$ is a decreasing function of $d = (\lambda_M + \lambda_m)c/2$ which means that it decreases as λ_M increases. The maximum value of λ_M for which these conditions hold is $\lambda_M = 1/c + (\lambda_2 + \lambda_N)/2$ for which $c\lambda_2 - d = |c\lambda_N - d|$.

If $\lambda_N < \lambda_m$, then $c\lambda_2 - d < |c\lambda_N - d|$ and $\nu(c, d)$ is an increasing function of $d = (\lambda_M + \lambda_m)c/2$ which means that it increases when λ_M increaseses. The minimum value of λ_M for which these conditions hold is $\lambda_M = 1/c + (\lambda_2 + \lambda_N)/2$ for which $c\lambda_2 - d = |c\lambda_N - d|$.

ii) In this case $c\lambda_2 - d > |c\lambda_N - d|$, and $\nu(c, d)$ is a decreasing function of $d = (\lambda_M + \lambda_m)c/2$ which means that it decreases when λ_M increases. The maximum value of λ_M for which these conditions hold is $\lambda_M = \lambda_2$.

iii) In this case $c\lambda_2 - d < |c\lambda_N - d|$, and $\nu(c, d)$ is an increasing function of $d = (\lambda_M + \lambda_m)c/2$ which means that it increases when λ_m increases. The minimum value of λ_m for which these conditions hold is $\lambda_m = \lambda_N$.

Proof of Theorem 5.4.2. If $[\lambda_2, \lambda_N] \subseteq [\lambda_m, \lambda_M]$ the result was proved in [91]. Let us suppose then that $[\lambda_2, \lambda_N] \not\subseteq [\lambda_m, \lambda_M]$. If $\lambda_2 - \lambda_N \leq \lambda_M - \lambda_m$, it has been shown in Proposition 1.1 that $\nu(c, d)$ has smaller values for c, d such that $[\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M]$, and in this case $\lambda_2 = \lambda_M$ and $\lambda_N = \lambda_m$ yields to the minimum $\nu(c, d)$.

If $\lambda_2 - \lambda_N > \lambda_M - \lambda_m$, we have seen in Proposition 1.1 that $\nu(c, d)$ is smaller for c, d such that λ_m, λ_M are symmetrically placed with respect to λ_N, λ_2, that is, $\lambda_M = \lambda_2 - \alpha$ and $\lambda_m = \lambda_N + \alpha$, $\alpha \geq 0$. Let us see that $\nu(c, d)$ is minimum for $\alpha = 0$. First, note that

$$c = \frac{2}{\lambda_M - \lambda_m} = \frac{2}{\lambda_2 - \lambda_N - 2\alpha}, \text{ and } d = \frac{\lambda_M + \lambda_m}{\lambda_M - \lambda_m} = \frac{\lambda_2 + \lambda_N}{\lambda_2 - \lambda_N - 2\alpha}.$$

Thus

$$\nu(c, d) = \frac{\tau(c-d)}{\tau(c\lambda_2 - d)} = \frac{\tau(c-d)}{-\tau(c\lambda_N - d)}$$

and taking into account that

$$\frac{\partial}{\partial \alpha}(c\lambda - d) = 2\frac{2\lambda - \lambda_2 - \lambda_N}{(\lambda_2 - \lambda_N - 2\alpha)^2} = 2\frac{c\lambda - d}{(\lambda_2 - \lambda_N - 2\alpha)},$$

$$\frac{\partial \nu(c,d)}{\partial \alpha} = \frac{-2\tau(c-d)}{\tau(c\lambda_2 - d)(\lambda_2 - \lambda_N - 2\alpha)} \left[\frac{c-d}{\sqrt{(c-d)^2-1}} - \frac{c\lambda_2 - d}{\sqrt{(c\lambda_2-d)^2-1}} \right] > 0.$$

Then $\nu(c,d)$ is increasing with α and the minimum value is obtained for $\alpha = 0$. ■

Proof of Theorem 5.4.3 (Faster convergence than $\mathbf{W}^n$)

By eq. (5.7) we have that $c-d = 1/\lambda_M,\ c\lambda - d = \lambda/\lambda_M$. To prove the faster convergence of our algorithm we show that the quotient between λ^n and $P_n(\lambda)$ goes to infinity with n. That is

$$\lim_{n\to\infty} \left| \frac{\lambda^n}{P_n(\lambda)} \right| = \lim_{n\to\infty} \left| \frac{\lambda^n T_n(c-d)}{T_n(c\lambda - d)} \right| = \infty. \tag{5.44}$$

If $\lambda \in [\lambda_m, \lambda_M]$, then $c\lambda - d \in (-1,1)$ and $|T_n(c\lambda - d)| \leq 1$. Using (5.5)

$$\left| \frac{\lambda^n T_n(c-d)}{T_n(c\lambda - d)} \right| \geq |\lambda^n T_n(c-d)| = \left| \lambda^n \frac{1+\tau(c-d)^{2n}}{2\tau(c-d)^n} \right|, \tag{5.45}$$

where τ is the function in eq. (5.5), which at $c-d$ is smaller than one. Then, in order to fulfill (5.44) it must hold that

$$\lim_{n\to\infty} \left(\frac{\lambda}{\tau(c-d)} \right)^n = \infty \Leftrightarrow \lambda > \tau(c-d) \Leftrightarrow \lambda_M \lambda > 1 - \sqrt{1-\lambda_M^2}, \tag{5.46}$$

which is satisfied if (5.15) holds.

On the other hand, if $\lambda \notin [\lambda_m, \lambda_M]$, using again (5.5) we substitute the value of $T_n(c-d)$ and $T_n(c\lambda - d)$

$$\frac{\lambda^n}{P_n(\lambda)} = \lambda^n \frac{\tau(c\lambda - d)^n}{\tau(c-d)^n} \frac{1+\tau(c-d)^{2n}}{1+\tau(c\lambda-d)^{2n}}. \tag{5.47}$$

The second term of (5.47) goes to one as n goes to infinity. This means that, in order to fulfill (5.44) it must hold that

$$\lim_{n\to\infty} \left(\frac{\lambda\tau(c\lambda - d)}{\tau(c-d)} \right)^n = \infty \Leftrightarrow \left| \frac{\lambda\tau(c\lambda-d)}{\tau(c-d)} \right| > 1. \tag{5.48}$$

Replacing τ for its value and doing some calculations using the radical conjugates we obtain

$$\frac{\lambda\tau(c\lambda-d)}{\tau(c-d)} = \lambda \frac{1+\sqrt{1-\lambda_M^2}}{\lambda + \sqrt{\lambda^2 - \lambda_M^2}}. \tag{5.49}$$

Using (5.49) we obtain that (5.48) is equivalent to $\lambda_M(-1-\lambda^2) + 2\lambda > 0$, which by (5.15) is always true, and therefore, the proof is complete. ■

Proof of Proposition 5.4.5 (Convergence for directed graphs)

Lemma 5.8.5. *Given $x > 1$, for any complex number z, such that $|\tau(z)| = \min\{|z+\sqrt{z^2-1}|, |z-\sqrt{z^2-1}|\} > \tau(x)$, then $\lim_{n\to\infty} T_n(z)/T_n(x) = 0$.*

Proof. It is a straightforward consequence of (5.40).

Proof of Proposition 5.4.5. Let $\mathbf{Q} = \mathbf{W} - \mathbf{1}\mathbf{w}_1^T/\mathbf{w}_1^T\mathbf{1}$, with $\mathbf{w}_1$ the left eigenvector associated to λ_1. Proceeding as in the Proof of Theorem 5.4.1, using $\gamma_1 = \mathbf{w}_1^T\mathbf{x}(0)/\mathbf{w}_1^T\mathbf{1}$, we arrive at

$$\|\mathbf{x}(n) - \gamma_1\mathbf{v}_1\|_2 \leq \|P_n(\mathbf{Q})\|_2\|\mathbf{x}(0) - \gamma_1\mathbf{v}_1\|_2. \tag{5.50}$$

Since $\mathbf{W}$ is diagonalizable, so is $\mathbf{Q}$, which implies that $\mathbf{Q}$ can be decomposed, $\mathbf{Q} = \mathbf{R}\mathbf{D}\mathbf{R}^{-1}$, with $\mathbf{D}$ =diag$(0, \lambda_2, \ldots, \lambda_N$ Using that $\mathbf{Q}^n = \mathbf{R}\mathbf{D}^n\mathbf{R}^{-1}$, we get that $P_n(\mathbf{Q}) = \mathbf{R}P_n(\mathbf{D})\mathbf{R}^{-1}$, and then

$$\|P_n(\mathbf{Q})\|_2 \leq \|\mathbf{R}\|_2\, \rho(P_n(\mathbf{Q}))\, \|\mathbf{R}^{-1}\|_2 = K \max_{i\neq 1}|P_n(\lambda_i)| = K \max_{i\neq 1}\frac{|T_n(c\lambda_i - d)|}{T_n(c-d)}, \tag{5.51}$$

with K the condition number of $\mathbf{P}$.

For any $x \in (\lambda_M + \lambda_m - 1, 1)$ we have that $|cx - d| < c - d$, then for all the real eigenvalues of $\mathbf{W}$ but λ_1, $|c\lambda_i - d| < c - d$. Noting that $c - d$ is strictly larger than 1 and $\tau(c-d) < \tau(c\lambda_z - d)$, for any complex eigenvalue λ_z, by Lemmas 5.8.1 and 5.8.5, $P_n(\lambda_i) \to 0$ for all $i \neq 1$, which proves the convergence of the algorithm. ∎

Proof of Proposition 5.5.1 (Upper bound for the time-varying Chebyshev recurrence)

For abbreviation, in the proof we will denote the sign of $T_n(\Lambda)$ by $s(T_n)$.

Let us note that, if $s(T_{n-1}) = s(T_{n-2})$, by choosing $\lambda(n) < 0$, then

$$|T_n(\Lambda)| = |2\lambda(n)T_{n-1}(\Lambda) - T_{n-2}(\Lambda)| = |2\lambda(n)T_{n-1}(\Lambda)| + |T_{n-2}(\Lambda)|, \tag{5.52}$$

independently of n. The choice of $\lambda(n) > 0$ when $s(T_{n-1}) = s(T_{n-2})$ implies that

$$|T_n(\Lambda)| = |2\lambda(n)T_{n-1}(\Lambda) - T_{n-2}(\Lambda)| < |2\lambda(n)T_{n-1}(\Lambda)| + |T_{n-2}(\Lambda)|. \tag{5.53}$$

Taking these two facts into account we can see that

$$s(T_{n-1}) = s(T_{n-2}) \Rightarrow \arg\max_{\lambda(n)}|T_n(\Lambda)| = \delta_{\min}. \tag{5.54}$$

Besides, in this situation, choosing $\lambda(n) < 0$ yields $s(T_n) \neq s(T_{n-1})$.

Now, if $s(T_{n-1}) \neq s(T_{n-2})$ and $\lambda(n) > 0$, then eq. (5.52) is again true. On the other hand, choosing $\lambda(n) < 0$ in this situation implies (5.53). Thus,

$$s(T_{n-1}) \neq s(T_{n-2}) \Rightarrow \arg\max_{\lambda(n)}|T_n(\Lambda)| = \delta_{\max}. \tag{5.55}$$

Also, if $s(T_{n-1}) \neq s(T_{n-2})$ and $\lambda(n) > 0$, then $s(T_n) = s(T_{n-1})$.

Finally, noting that inequality (5.19) holds for $n = 0$ and 1, and $s(T_0(\Lambda^*)) = s(T_1(\Lambda^*))$, then using (5.54) and (5.55) the succession (5.20) is obtained and the result is proved. ∎

Proof of Lemma 5.5.3 (Direct expression for the bounded time-varying Chebyshev recurrence)

Let us define the recurrence

$$T_0^*(\lambda) = 1,\ T_1^*(\lambda) = \lambda,\ T_n^*(\lambda) = 2\lambda T_{n-1}^*(\lambda) + T_{n-2}^*(\lambda), \tag{5.56}$$

which satisfies that

$$|T_n(\Lambda^*)| \leq T_n^*(\delta_{\max}). \tag{5.57}$$

According to recurrence (5.56), the succession $\{T_n^*(\delta_{\max}),\ n = 0, 1, \ldots\}$ satisfies the homogeneous difference equation $T_n^*(\delta_{\max}) - 2\delta_{\max} T_{n-1}^*(\delta_{\max}) - T_{n-2}^*(\delta_{\max}) = 0$. By the theory of difference equations [2], the solution to this equation is determined by the roots κ_1 and κ_2 of the characteristic polynomial. In this case

$$\kappa_1(\delta_{\max}) = \delta_{\max} + \sqrt{\delta_{\max}^2 + 1} > 1, \text{ and } \kappa_2(\delta_{\max}) = \delta_{\max} - \sqrt{\delta_{\max}^2 + 1} = -1/\kappa_1(\delta_{\max}). \tag{5.58}$$

Since $\kappa_1(\delta_{\max}) \neq \kappa_2(\delta_{\max})$, the direct expression of $T_n^*(\delta_{\max})$ is

$$T_n^*(\delta_{\max}) = A\kappa_1(\delta_{\max})^n + B\kappa_2(\delta_{\max})^n \tag{5.59}$$

where A and B depend on the initial conditions $T_0^*(\delta_{\max})$ and $T_1^*(\delta_{\max})$. In our case $A = B = 1/2$ and

$$|T_n(\Lambda^*)| \leq T_n^*(\delta_{\max}) = \frac{1}{2}(\kappa_1(\delta_{\max})^n + (-1/\kappa_1(\delta_{\max}))^n) \leq \kappa_1(\delta_{\max})^n. \tag{5.60}$$

■

Proof of Theorem 5.5.4 (Convergence with time-varying topologies)

First of all, let us state the notation we will follow along the proof. For any weight matrix $\mathbf{W}(n)$ we denote its eigenvectors by $\mathbf{v}_i(n),\ i = 1, \ldots, N$. Let us denote $\mathbf{V}(n) = [\mathbf{v}_1(n), \ldots, \mathbf{v}_N(n)]$ the matrix with all the eigenvectors of $\mathbf{W}(n)$. Since $\mathbf{W}(n)$ is symmetric, it is diagonalizable and we can choose the base of eigenvectors in such a way that $\mathbf{V}(n)$ is orthogonal. Therefore, $\mathbf{v}_1(n)^T\mathbf{v}_i(n) = 0, \forall i = 2, \ldots, N$, and $\mathbf{v}_1(n) = \mathbf{1}/\sqrt{N} = \mathbf{v}_1$, for all n. We define $\bar{\mathbf{x}}$ as the vector containing the average of the initial conditions in all its components, $\bar{\mathbf{x}} = \mathbf{1}^T\mathbf{x}(0)\mathbf{1}/N$.

Let $\mathbf{Q}(n) = \mathbf{W}(n) - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, whose eigenvalues are 0, with $\mathbf{v}_1(n) = \mathbf{1}/\sqrt{N}$ its corresponding eigenvector, and $\lambda_2(n), \ldots, \lambda_N(n)$, with the same eigenvectors as $\mathbf{W}(n)$. Thus, $\mathbf{Q}(n)\mathbf{V}(n) = \mathbf{V}(n)\mathbf{D}(n)$, with $\mathbf{D}(n) = \operatorname{diag}(0, \lambda_2(n), \ldots, \lambda_N(n))$. Taking all of this into account it is easy to see that $\mathbf{1}\mathbf{1}^T(\mathbf{x}(0) - \bar{\mathbf{x}}) = 0$, and $\mathbf{W}(n)(\mathbf{x}(n) - \bar{\mathbf{x}}) = \mathbf{Q}(n)(\mathbf{x}(n) - \bar{\mathbf{x}})$.

Given two consecutive matrices, $\mathbf{Q}(n)$ and $\mathbf{Q}(n-1)$, let $\mathbf{P}(n)$ be the matrix such that $\mathbf{V}(n-1) = \mathbf{V}(n)\mathbf{P}(n)$, that is, the matrix that changes from the base of eigenvectors of $\mathbf{Q}(n-1)$ to the base of eigenvectors of $\mathbf{Q}(n)$. In a similar way, $\mathbf{R}(n)$ will be such that $\mathbf{V}(n-2) = \mathbf{V}(n)\mathbf{R}(n)$. The orthogonality of $\mathbf{V}(n)$, implies that the matrices $\mathbf{P}(n) = \mathbf{V}(n)^{-1}\mathbf{V}(n-1)$ and $\mathbf{R}(n) = \mathbf{V}(n)^{-1}\mathbf{V}(n-2)$ are also orthogonal, and $\|\mathbf{P}(n)\|_2 = \|\mathbf{R}(n)\|_2 = 1$.

We define the scaled error at the nth iteration by $\mathbf{e}(n) = T_n(c-d)(\mathbf{x}(n) - \bar{\mathbf{x}})$. Using the Chebyshev recurrence (5.17), and the equivalence

$$\bar{\mathbf{x}} = 2\frac{T_n(c-d)}{T_{n+1}(c-d)}(c\mathbf{W}(n) - d\mathbf{I})\bar{\mathbf{x}} - \frac{T_{n-1}(c-d)}{T_{n+1}(c-d)}\bar{\mathbf{x}}, \tag{5.61}$$

$\mathbf{e}(n)$ satisfies the recurrence

$$\mathbf{e}(0) = (\mathbf{x}(0) - \bar{\mathbf{x}}),\ \mathbf{e}(1) = (c\mathbf{Q}(1) - d\mathbf{I})\mathbf{e}(0),\ \mathbf{e}(n) = 2(c\mathbf{Q}(n) - d\mathbf{I})\mathbf{e}(n-1) - \mathbf{e}(n-2). \tag{5.62}$$

Each vector $\mathbf{e}(n)$ can be expressed as a linear combination of the eigenvectors of $\mathbf{Q}(n)$,

$$\mathbf{e}(n) = \sum_{i=1}^{N} \alpha_i(n)\mathbf{v}_i(n) = \mathbf{V}(n)\boldsymbol{\alpha}(n). \tag{5.63}$$

We will next prove that $\|\boldsymbol{\alpha}(n)\|_2/T_n(c-d)$ goes to zero with n. Replacing $\mathbf{e}(n)$ by (5.63) in (5.62),

$$\begin{aligned}\mathbf{e}(n) &= 2(c\mathbf{Q}(n) - d\mathbf{I})\mathbf{V}(n-1)\boldsymbol{\alpha}(n-1) - \mathbf{V}(n-2)\boldsymbol{\alpha}(n-1)\\ &= 2(c\mathbf{Q}(n) - d\mathbf{I})\mathbf{V}(n)\mathbf{P}(n)\boldsymbol{\alpha}(n-1) - \mathbf{V}(n)\mathbf{R}(n)\boldsymbol{\alpha}(n-2)\\ &= 2\mathbf{V}(n)(c\mathbf{D}(n) - d\mathbf{I})\mathbf{P}(n)\boldsymbol{\alpha}(n-1) - \mathbf{V}(n)\mathbf{R}(n)\boldsymbol{\alpha}(n-2)\\ &= \mathbf{V}(n)[2(c\mathbf{D}(n) - d\mathbf{I})\mathbf{P}(n)\boldsymbol{\alpha}(n-1) - \mathbf{R}(n)\boldsymbol{\alpha}(n-2)] = \mathbf{V}(n)\boldsymbol{\alpha}(n).\end{aligned} \tag{5.64}$$

Therefore, the vectors $\boldsymbol{\alpha}(n)$ satisfy the recurrence

$$\boldsymbol{\alpha}(0) = \boldsymbol{\alpha}(1),\ \boldsymbol{\alpha}(n) = 2(c\mathbf{D}(n) - d\mathbf{I})\mathbf{P}(n)\boldsymbol{\alpha}(n-1) - \mathbf{R}(n)\boldsymbol{\alpha}(n-2). \tag{5.65}$$

Taking spectral norms,

$$\begin{aligned}\|\boldsymbol{\alpha}(n)\|_2 &= \|2(c\mathbf{D}(n) - d\mathbf{I})\mathbf{P}(n)\boldsymbol{\alpha}(n-1) - \mathbf{R}(n)\boldsymbol{\alpha}(n-2)\|_2 \leq \\ &\leq 2\|(c\mathbf{D}(n) - d\mathbf{I})\|_2\|\mathbf{P}(n)\|_2\|\boldsymbol{\alpha}(n-1)\|_2 + \|\mathbf{R}(n)\|_2\|\boldsymbol{\alpha}(n-2)\|_2 \leq \\ &\leq (2\max_i |c\lambda_i(n) - d|\|\boldsymbol{\alpha}(n-1)\|_2 + \|\boldsymbol{\alpha}(n-2)\|_2).\end{aligned} \tag{5.66}$$

By Lemma 5.5.3 we can bound the norm of $\boldsymbol{\alpha}(n)$ by $\kappa_1(x_{\max})^n\|\boldsymbol{\alpha}(0)\|_2$, where the parameter $x_{\max}$ in this case is

$$\begin{aligned}x_{\max} &= \max_n \max_{i=2,\ldots,N} |c\lambda_i(n) - d| = \max_n\{|c\lambda_2(n) - d|, |c\lambda_N(n) - d|\} = \\ &= \max\{|c\lambda_{\max} - d|, |c\lambda_{\min} - d|\}.\end{aligned} \tag{5.67}$$

Therefore, in order to make the error go to zero we require that

$$\lim_{n\to\infty} \frac{\kappa_1(x_{\max})^n}{T_n(c-d)} = 0. \tag{5.68}$$

Using (5.5)

$$\frac{\kappa_1(x_{\max})^n}{T_n(c-d)} = \frac{\kappa_1(x_{\max})^n\tau(c-d)^n}{1+\tau(c-d)^{2n}}, \tag{5.69}$$

which goes to zero if $\kappa_1(x_{\max})\tau(c-d) < 1$. When this happens $\lim_{n\to\infty}\mathbf{x}(n) = \bar{\mathbf{x}}$, and the consensus is achieved. ■

Proof of Corollary 5.5.5 (Convergence with symmetric parameters)

Recall that with this assignation $c = 1/\lambda$ and $d = 0$. Substituting κ_1 and τ by their values in eq. (5.24) and doing some simplifications eq. (5.25) is obtained. ■

Proof of Corollary 5.5.6 (Convergence with non symmetric parameters)

If we know the values of $\lambda_{\max}$ and $\lambda_{\min}$, the choice of λ_m and λ_M can be done in such a way that

$$|c\lambda_{\min} - d| = |c\lambda_{\max} - d|. \tag{5.70}$$

With this assignation we are minimizing the value of $\max\{|c\lambda_{\max} - d|, |c\lambda_{\min} - d|\}$ and therefore, the convergence condition is easier to fulfill. Clearing (5.70) yields (5.26). With this first condition, doing some calculations in eq. (5.24) the second condition (5.27) is obtained. ■

Proof of Proposition 5.6.2 (Eigenvalues position indicator)

The initial error is

$$\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty = \|\sum_{i=2}^N \mathbf{v}_i\|_\infty. \tag{5.71}$$

The error after n iterations is equal to

$$\|\mathbf{x}(n)-\mathbf{v}_1\|_\infty = \|\sum_{i=2}^{N}\frac{T_n(c\lambda_i)}{T_n(c)}\mathbf{v}_i\|_\infty \tag{5.72}$$

Using these two expressions we get

$$\kappa_n(c) = T_n(c)\frac{\|\mathbf{x}(n)-\mathbf{v}_1\|_\infty}{\|\mathbf{x}(0)-\mathbf{v}_1\|_\infty} = \frac{\|\sum_{i=2}^{N}T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N}\mathbf{v}_i\|_\infty} \tag{5.73}$$

If $\lambda_i \in [-\lambda, \lambda]$, it means that $c\lambda_i \leq 1$. In this case, the evaluation of the Chebyshev polynomial is upper-bounded (in norm) by 1, therefore,

$$\kappa_n(c) = \frac{\|\sum_{i=2}^{N}T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N}\mathbf{v}_i\|_\infty} \leq \frac{\sum_{i=2}^{N}\|\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N}\mathbf{v}_i\|_\infty}, \; \forall n. \tag{5.74}$$

On the other hand, when $c\lambda_i > 1$, for some λ_i, $T_n(c\lambda_i)$ goes to infinity as n grows and

$$\lim_{n\to\infty}\kappa_n(c) = \lim_{n\to\infty}\frac{\|\sum_{i=2}^{N}T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N}\mathbf{v}_i\|_\infty} = \infty \tag{5.75}$$

■

Proof of Proposition 5.6.3 (Convergence of the bisection method)

By Proposition 5.6.2, we know that for a sufficiently large n, $\kappa_n(c)$ correctly discriminates if $\lambda_i \in [\lambda, \lambda]$, $\forall i = 2, \ldots, N$. The algorithm in (5.31) is based on bisection and is convergent because of the use of $\kappa_n(c)$. The value of convergence is the border between the two limit situations, and it is $\lambda = (\lambda_{\max} + \lambda_{\min})/2 \to \max_{i\neq 1}\lambda_i = \lambda_2$. ■

Proof of Proposition 5.6.5 (Bound of the estimator)

The bound is obtained from the following inequality:

$$\begin{aligned}\|\mathbf{x}(n)-\mathbf{v}_1\|_\infty &\leq \|\mathbf{x}(n)-\mathbf{v}_1\|_2 = \\ &= \|T_n(c\mathbf{W})/T_n(c)(\mathbf{x}(0)-\mathbf{v}_1)\|_2 \leq \\ &\leq \max_{i\neq 1}T_n(c\lambda_i)/T_n(c)\|\mathbf{x}(0)-\mathbf{v}_1\|_2 \leq \\ &\leq \max_{i\neq 1}T_n(c\lambda_i)/T_n(c)\|\mathbf{x}(0)-\mathbf{v}_1\|_\infty\sqrt{N}.\end{aligned}$$

Regrouping terms and considering that $T_n(c\lambda_i) \leq 1$ yields $\kappa_n(c) \leq \sqrt{N}$. ■

Chapter 6

Distributed Consensus in Perception using Homographies

"A good map is both a useful tool and a magic carpet to far away places." In the Thesis we have studied how to modify the consensus iteration to handle different perception issues. In this chapter we present an application of such algorithms in the problem of cooperative mapping with cameras. The approach builds topological maps from the sequences of images acquired by each robot, grouping the features in planar regions and fusing them using consensus. The use of planar regions to represent the map has many advantages both in the mapping task and in the achievement of the consensus. First of all, using inter-image homographies, the individual maps are easy to create and the data association between different maps is simple. The computation of a global reference frame to represent the features, which is in general quite complicated, but necessary to reach a consensus, is reduced to a simple max-consensus method multiplying different homographies. Finally, homographies between images can be computed without knowing the internal parameters of the cameras, which makes the approach robust to calibration issues. The result is a simple but very effective distributed algorithm that creates a global map using the information of all the robots. Experiments with real images in complex scenarios show the good performance of our distributed solution.

6.1 Introduction

In the previous chapters of the Thesis, we have seen that consensus methods can be used to efficiently fuse the information of the different robots. We have also provided mechanisms to handle some important problems to reach the agreement using vision sensors. In this chapter we focus on presenting an application in which consensus algorithms can be of high interest. We present a solution to the problem of distributed map building with vision sensors using consensus algorithms. The proper representation of the environment where the robot is going to work is a fundamental issue in any robotic application. That is, the robots need a common map of the environment to navigate and localize themselves.

The problem of mapping the environment considering a single robot has been deeply studied by the research community. A common approach is to simultaneously localize the robot and map the environment (SLAM) [27, 39]. In these approaches the map is usually represented by a set of 3D features, whose position is updated every time they are observed in a new image. To make this process more robust, view-based maps [70] introduce geometric constraints between pairs of images in the SLAM algorithm. Dealing with multiple robots, we also find centralized [23, 55, 137] and distributed methods [6]. Computing the structure of the scene, usually defined with the positions of the features and the cameras, requires the exact knowledge of the intrinsic parameters of the cameras [1] and makes the errors and drift grow with the size of the map. Additionally, in a multi-robot scenario, with these approaches the robots need to compute a global reference frame to represent their positions and the positions of the observed features. On the other hand, topological approaches [71] overcome these limitations because no explicit metric information for the global map is computed.

Topological visual maps can be built from conventional [3, 47] or omnidirectional [100, 165] images. The whole image can be stored but usually only the extracted features are saved. Most recent and successful works use visual words [120] to represent the scene in a more compact way. Although topological maps based on images give good results, the space required to manage these maps is considerably large. If we store all the features, many of them will be seen in several images, so that the map will take up a lot of repeated data for the same features. Additionally, from a distributed point of view, achieving a consensus about different sets of images is unclear because the average of two or more images does not make sense.

A better approach in this context consists in grouping the information in planes. A plane is defined as a set of features that belong to the same planar region in the scene. Plane detection in images is a common problem in computer vision [48, 136, 160]. There are several advantages of using planes for individual robots and for the whole team:

- Advantages for the consensus process:
  - The data association between different maps is simple and robust using inter-image homography constraints. The global data association can then be obtained using the algorithms presented in Chapter 3.
  - Instead of using complex procedures, the computation of a global reference frame to represent the features, necessary to achieve a valid consensus, is done just by multiplying different homographies.
  - Finally, homographies between images can be computed without knowing the internal parameters of the cameras, which makes the approach robust to the use of different cameras by different robots.
- Advantages for individual robots:
  - The errors in different planar regions are uncorrelated because the extraction of each one is independent of the others.
  - Features are stored only once independently of the number of images in which they are observed.
  - The complexity of the graph that defines the map is also reduced. Graphs made from raw images are usually dense because of the number of connections among close images whereas with the proposed maps the number of connections between planes is considerably smaller.
  - Planes also provide a good semantic information meaningful for humans. If the robots need to cooperate with a human or simply receive orders from him, they will be able to understand some basic human concepts such us walls or building facades.

In addition to the previous advantages, it is well known that the structure estimation is improved in terms of accuracy and stability when considering the scene represented by planes [141]. Moreover, there are several works in the literature that assume the presence of planes in the environment to solve different tasks such as visual servoing [24], [40], [74], visual navigation with maps [123], [34], structure reconstruction [130], pose estimation [20] or camera calibration [85]. In all these approaches the 3D structure of the scene is not required and only sets of coplanar features in the image domain are used. Since the proposed solution based on planes follows these guidelines it can also be useful in the above situations.

The main contributions of this chapter are:

- A proposal in the context of consensus for perception of scenes using a geometrical constraint as the homography.
- A method for feature matching and homography computation using a triple set plane-image-image. the planar regions are tracked along the sequence and new features are added to them once they are detected. Homology constraints are used to detect new planes and also give a geometric criterion to relate them.

- A new organization of visual information in a graph of planar regions where there exists a geometric relation between the different planes of the scene.
- A solution for the multi-robot scenario, giving a technique to fuse several maps based on distributed consensus without knowledge about the cameras, the positions of the robots or the data association between the individual maps.

Part of this work has been previously published in [94–96].

6.2 Single robot topological map

Let $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots\}$ be the set of images acquired by a 6DOF camera mounted on a mobile robot. Let us assume that several rigid planar regions appear in the scene. There is no knowledge about either the internal parameters of the camera, represented by the calibration matrix **K**, nor about the motion between consecutive frames, **R** and **t**. For an easy understanding of the section, subscript indices will correspond to images in $\mathcal{I}$ whereas superscript indices will correspond to the planar regions, for example $\boldsymbol{\pi}_k^m$ will represent the features of the m^{th} planar region seen in the k^{th} image.

The topological map is managed using a graph, $\mathcal{G}^\pi = (\mathcal{P}, \mathbf{\Pi}, \mathbf{C})$, represented by a finite non empty set of planes $\mathcal{P}$ with cardinality $|\mathcal{P}| = P$, a vector $\mathbf{\Pi}$ containing the features observed in the planar regions and a matrix of relations between the planes $\mathbf{C} \in \{0,1\}^{P \times P}$. If $\mathbf{C}(m,n) = 1$ then there exists a relation between the planes m and n, whereas for planes with no relation, $\mathbf{C}(m,n) = 0$.

If one plane, m, is visible in two consecutive images of the sequence, $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$, it is possible to compute a projective mapping (inter-image homography), $\mathbf{H}^m_{k,k+1}$, that relates the features belonging to the plane, $\boldsymbol{\pi}_k^m = \mathbf{H}^m_{k,k+1}\boldsymbol{\pi}^m_{k+1}$. This homography is defined up to a scale factor and has the form

$$\mathbf{H}^m_{k,k+1} = \mathbf{K}(\mathbf{R}_{k,k+1} - \frac{\mathbf{t}_{k,k+1}(\mathbf{n}^m_{k+1})^T}{d^m_{k+1}})\mathbf{K}^{-1}, \tag{6.1}$$

with d^m_{k+1} and $\mathbf{n}^m_{k+1}$ the distance and normal of the m^{th} plane in the $(k+1)^{th}$ frame, respectively. The homography can be estimated from four correspondences without prior knowledge about the scene or the calibration [57].

The planes are extracted from the sequence with the following scheme:

1. The two initial frames of the sequence, $\mathcal{I}_1$ and $\mathcal{I}_2$, are picked up and all the planes seen in both images (Fig. 6.1-a) are extracted using DLT+RANSAC [57]. In order to perform the matching between features we have chosen SURF descriptors [13]. Nevertheless, the algorithm will work with any descriptor as long as it is able to extract enough features to compute homographies.
2. For any plane, m, visible in the first two images:
   (a) All the features from the plane, $\boldsymbol{\pi}^m$, are stored expressed in the coordinates of the first image where they were detected. This first image is marked as the reference image of the plane, $\mathcal{I}_{r^m}$. The identifiers of the features in every image are also stored to make automatic the future search of these features.
   (b) The next image in the list is picked up, $\mathcal{I}_3$, and the correspondences with $\mathcal{I}_2$ are found. From the whole set, only those matches that already belong to $\boldsymbol{\pi}^m$ are chosen, searching a new homography among this subset. By looking for the homography only among this subset, fewer hypotheses are required in RANSAC, because the probability of a sample to be an inlier is larger than considering the whole set of features.

(c) The homography with respect to the points in the reference image, $\mathbf{H}^{m}_{r^m,3}$, is also computed so that the voting procedure is more robust, enforcing every feature to support both homographies instead of just one (Planes m and o in Fig. 6.1-b). With the two homographies new features are added to the existing plane (Plane o in Fig 6.1-c).

3. Once all the matches belonging to existing planes have been processed, new planes between the remaining matches are extracted (Plane p in Fig 6.1-c).

4. The next image is selected and the method is recursively repeated until all the images are processed.

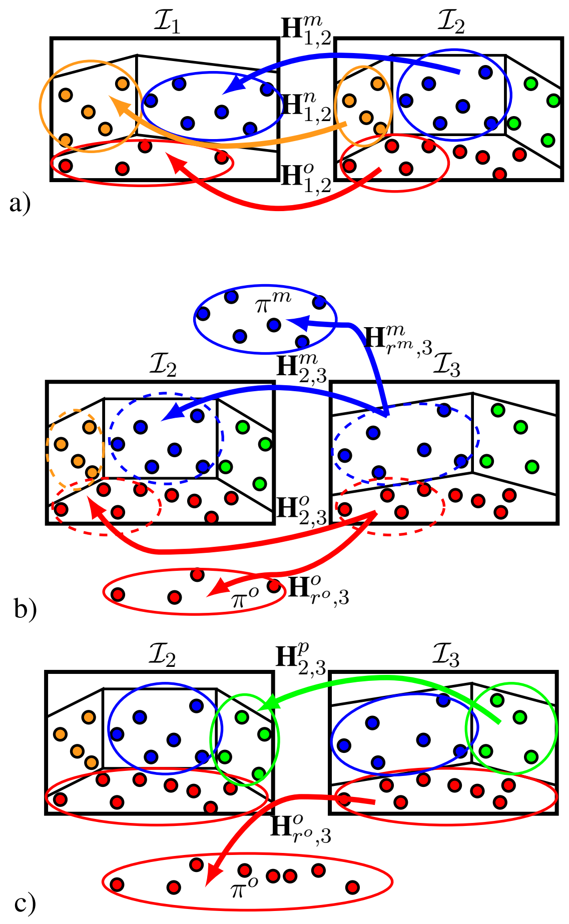


Figure 6.1: Scheme of the plane segmentation. a) Extraction of the initial planes, m, n, o. b) Triple match Plane-Image-Image for homography computation with the previous and the reference image. c) Addition of new points to the existing planes and detection of new planar regions within the remaining matches.

Every time a new plane is detected it is added to the topological map. One plane, m, is formally added to the graph by

$$\begin{cases} \mathbf{C} = [\mathbf{I}_P \mid \mathbf{0}_P]^T \ \mathbf{C} \ [\mathbf{I}_P \mid \mathbf{0}_P], \\ \mathbf{\Pi} = (\mathbf{\Pi}^T, \boldsymbol{\pi}^m)^T, \\ \mathcal{P} = \mathcal{P} \cup \{m\}, \end{cases} \tag{6.2}$$

with $\mathbf{I}_P$ the identity matrix of $P \times P$ dimensions and $\mathbf{0}_P$ a null vector of dimension P.

The final information used to represent one plane is the set of features that belong to the plane with their SURF descriptors. The coordinates of each feature are expressed in the reference image of the plane.

6.2.1 Links between planes

Two planes m and n are defined as co-visible if there are at least two consecutive images in which both planes are detected. This idea of co-visibility has a great interest for navigation and localization tasks in future uses of the map. If a robot localizes one plane of the set it will know which other planes it might see when it moves. Therefore, the space searched during the execution of the task will be reduced.

When two planar regions are visible together in two consecutive images it is possible to extract multi-plane constraints between them. A homology matrix, also called "relative homography" captures the relative motion between the images through two planes visible in the two images. Let us suppose that m and n are both visible in $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$. The homology is obtained by multiplying one of the homographies by the inverse of the other one, $\mathcal{H}^{mn}_{k,k+1} = (\mathbf{H}^m_{k,k+1})^{-1}\mathbf{H}^n_{k,k+1}$. The homology is the chosen criterion to create a link between planes in the topological map. When two planes have been detected together in two consecutive images the algorithm sets the connecting edges to 1:

$$\mathbf{C}(m,n) = \mathbf{C}(n,m) = 1 \Leftrightarrow \exists\, \mathcal{I}_k, \mathcal{I}_{k+1} \in \mathcal{I} \mid \exists\, \mathcal{H}^{mn}_{k,k+1}. \tag{6.3}$$

Let us note that although the link is created considering a geometric criterion between the planes, the map does not include any metric information.

The homology has also some properties that can be useful for robust detection of new planes in the sequence. Using the Sherman-Morrison formula [143], as in [160], the homology matrix can be decomposed in $\mathcal{H}^{mn}_{k,k+1} = \mathbf{I} + \mathbf{v}\mathbf{p}^T$, where

$$\mathbf{v} = (v_1, v_2, v_3)^T = \mathbf{K}\frac{\mathbf{R}^{-1}_{k,k+1}\mathbf{t}_{k,k+1}}{1 + \frac{(\mathbf{n}^m_{k+1})^T}{d^m_{k+1}}\mathbf{R}^{-1}_{k,k+1}\mathbf{t}_{k,k+1}} \tag{6.4}$$

is a view dependent vector and

$$\mathbf{p} = (p_1, p_2, p_3)^T = (\frac{(\mathbf{n}^m_{k+1})^T}{d^m_{k+1}} - \frac{(\mathbf{n}^n_{k+1})^T}{d^n_{k+1}})\mathbf{K}^{-1} \tag{6.5}$$

is a plane dependent vector. The homology is used to separate real planes from false and repeated ones. This is done using its eigenvalues, $\{\lambda_1, \lambda_2, \lambda_3\}$, which for a correct homology must have the form $(1, 1, 1 + v_1p_1 + v_2p_2 + v_3p_3)$. Before adding a new plane to the map the eigenvalues of the homology must hold

$$|\lambda_1 - 1| \le \epsilon,\ |\lambda_2 - 1| \le \epsilon,\ |\lambda_3 - 1| \ge \epsilon, \tag{6.6}$$

for a sufficient small ϵ. If the three eigenvalues are close to the unity it means that the two planes are actually the same one (the homology is an identity matrix), so instead of creating a new plane, the new features are added to the existing one. On the other hand, if two of the three eigenvalues are not close enough to the unity there is an homography that is not describing a real plane. In this second case the new plane is ignored. Let us notice that the test is pure image-based and the method still does not need any information about neither the camera calibration nor the motion between the images.

6.2.2 Loop closing

The last problem considered to build the individual maps is to detect when a plane appears in the sequence because the robot is revisiting the same place (loop closing). To consider this situation every time a new plane is detected, the algorithm matches the features of the new plane with the rest of existing planes and tries to compute a robust homography between them. If for some plane the corresponding homography exists and it is supported by most of the matches it means that both planes are the same and must be merged. The merging process is performed by adding to the existing plane the new features and by updating **C** through eq. (6.3).

In the end, all the method can be summarized in the Algorithm 7

Algorithm 7 Single robot topological map

```
1: Extract planes from $\mathcal{I}_1$ and $\mathcal{I}_2$
2: Create $\mathcal{G}^\pi$ with the initial planes
3: $\mathbf{C}(m,n) = \mathbf{C}(n,m) = 1,\ \forall m \neq n$
4: for all $\mathcal{I}_k \in \mathcal{I}$ do
5:     Match features in $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$
6:     for all visible $m \in \mathcal{P}$ do
7:         Select the matches that belong to $\boldsymbol{\pi}^m$
8:         Compute $\mathbf{H}^m_{k,k+1}$ and $\mathbf{H}^m_{r^m,k+1}$ with DLT+Ransac
9:         Add new features to $\boldsymbol{\pi}^m$ using $\mathbf{H}^m_{r^m,k+1}$
10:    end for
11:    Search for new planes in the remaining matches
12:    if new plane was already in the map then
13:        Update $\mathcal{G}^\pi$
14:    else
15:        Add the new plane to $\mathcal{G}^\pi$ (eq. (6.2))
16:    end if
17:    Modify C with the new homologies (eq. (6.3)-(6.6))
18: end for
```

6.3 Multi-Camera Distributed Topological map

Let us consider the full team of N robots with communications defined by a fixed graph $\mathcal{G}$ as defined in Chapter 2.2.2. Each robot manages an individual topological map $\mathcal{G}_i^\pi = \{\mathcal{P}_i, \mathbf{\Pi}_i, \mathbf{C}_i\}$, $i = 1, \ldots, N$. We want to make all the robots compute an identical global map, $\mathcal{G}_*^\pi = (\mathcal{P}^*, \mathbf{\Pi}^*, \mathbf{C}^*)$, containing the information acquired by the whole set of robots. The computation of the global map can be divided in two parts. On one hand the information about the graph of planes and their relations ($\mathcal{P}^*$ and $\mathbf{C}^*$), and on the other hand the reference image of each planar region, how many features they contain and their coordinates ($\mathbf{\Pi}^*$).

A distributed consensus approach is followed to compute the global map. A leader election (max-consensus) approach is followed to obtain the consensus on the global graph and the SURF descriptors, whereas for the feature coordinates a distributed averaging rule is used. In order to use these techniques several properties must be ensured:

1. Information of the robots: it is required that all the robots have an initial value of the information.
2. Data association: in order to perform the fusion it is necessary to know which planes in the local maps are associated. The data association between features belonging to the same plane in different maps is also required.
3. Common reference frame: given two planes from two different maps which correspond to the same planar region in the world, the corresponding sets of features must be expressed in the same reference frame.

However, local maps composed by planes and image features do not satisfy the properties above mentioned. Solutions to overcome these problems are proposed, obtaining a common global map equal for all the robots.

6.3.1 Information of the robots

It is supposed that none of the robots has the information of all the planes seen by the whole team. Therefore, in the first step the local maps, $\mathcal{G}_i^\pi$, are augmented so that the size and the order of each of them is the same,

$P_i^* = P_j^*$ and $|\boldsymbol{\Pi}_i^*| = |\boldsymbol{\Pi}_j^*|$, $\forall i, j \in \mathcal{V}$.

The ordering of the planes is done by univocal identification of both, the robots and the planes. By Assumption 2.2.1 each robot has a unique identifier, whereas the planes are ordered as they were detected in the local sequences. These two elements define a global order of the whole set of maps. The function $O : \mathcal{P} \to \mathbb{N}$ is defined in such a way that it returns the order of a given plane in the map. For example, the first plane observed by the third robot will have a smaller position in the map than the fourth plane observed by the same robot but a larger value in the global order than any plane observed by the second or the first robot.

The different size of the initial maps is solved creating fictitious planes in the local maps so that all the robots have a final map of the same size. A fictitious plane, $\tilde{m}$, is a plane with no relations in the graph and for which all the coordinates of all its features, $\boldsymbol{\pi}^{\tilde{m}}$, are initialized to zero. Every robot creates as many fictitious planes as the total number of planes observed by the other robots. This is done exchanging local messages so that, every robot i eventually has a feature vector

$$\boldsymbol{\Pi}_i^* = (\mathbf{0}_1^T \dots \boldsymbol{\Pi}_i^T \dots \mathbf{0}_N^T)^T \tag{6.7}$$

and an adjacency matrix $\mathbf{C}_*^i$ with the form

$$\mathbf{C}_*^i = \begin{bmatrix} \mathbf{0}_{11} & \dots & \mathbf{0}_{1i} & \dots & \mathbf{0}_{1N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0}_{i1} & \dots & \mathbf{C}_i & \dots & \mathbf{0}_{iN} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0}_{N1} & \dots & \mathbf{0}_{Ni} & \dots & \mathbf{0}_{NN} \end{bmatrix}, \tag{6.8}$$

where $\mathbf{0}_j$ is a vector of zeros with dimension $|\boldsymbol{\Pi}_j|$ and $\mathbf{0}_{ij}$ is a matrix of zeros with dimension $P_i \times P_j, j = 1, \dots, N, j \neq i$. Let us note that in order to create a fictitious plane it is only necessary to know which robot has seen it, the order in the local map and the number of features it contains. This reduces considerably the size of the exchanged messages.

When new messages containing information from other robots are received, the fictitious planes are added to the local maps,

$$\mathcal{P}_i^* = \mathcal{P}_i^* \cup \tilde{m}. \tag{6.9}$$

Regarding the adjacency matrices, $\mathbf{C}_i^*$, for any new fictitious plane a new row and column with zeros is created

$$\mathbf{C}_i^* = \mathbf{P}_{O(\tilde{m})P_i} \; [\mathbf{I}_{P_i} \mid \mathbf{0}]^T \; \mathbf{C}_i^* \; [\mathbf{I}_{P_i} \mid \mathbf{0}] \; \mathbf{P}_{O(\tilde{m})P_i}, \tag{6.10}$$

where the middle matrices augment the adjacency matrix as in eq. (6.2) and $\mathbf{P}_{O(\tilde{m})P_i}$ is a permutation matrix that moves the last row, P_i, to the row $O(\tilde{m})$ and displaces all the rows in between one position down. Finally, $\boldsymbol{\Pi}_i^*$ is also updated by adding fictitious features, with coordinates equal to zero, in the corresponding position. Since the communication graph is fixed, after d_v rounds of exchanging information, all the robots will know the size of the global map, and all the vectors $\boldsymbol{\Pi}_i^*$ and matrices $\mathbf{C}_i^*$ will have the same dimension and the form of eqs. (6.7) and (6.8) respectively. This process is summarized in Algorithm 8.

6.3.2 Data association

Here two issues must be addressed. The first one is the data association between planes in different maps and the data association of features between matched planes. Once the data association is known, the second issue is to reduce the adjacency matrices (respectively feature vectors) so that they have the correct number of elements and in the right order.

The data association is performed using the algorithm presented in Chapter 3. The function to locally match the planes is the homography between the invariant features of the planes. That is, two planar regions are

Algorithm 8 Augment local maps - Robot i

```
1: Send information about the local map to all $j \in \mathcal{N}_i$
2: for $it = 1 \dots d_v$ do
3:     Receive information from all $j \in \mathcal{N}_i$
4:     Create fictitious planes (eq. (6.9))
5:     Augment $\mathbf{C}_i^*$ (eq. (6.10))
6:     Send the new information to all $j \in \mathcal{N}_i$
7: end for
```

associated if a robust homography exists between them. The procedure to compute a homography is exactly the same one applied to pairs of images (DLT+RANSAC) and, as a consequence, is robust to the use of different cameras. Additionally, note that the same function serves to associate the features observed in the planes.

The next step is to reduce the size of the maps considering the correspondences found. Given a plane, m, let $B(m)$ be the set of planes associated to m and $\bar{n}(m) = \arg\min_{n \in B(m)} O(n)$ be the plane with the lowest value in the global order. For all the associations of the robot i, the adjacency matrix $\mathbf{C}_i^*$ is updated to put together all the associated planes

$$\mathbf{C}_i^* = \mathbf{I}_n(\mathbf{C}_i^* \vee \mathbf{P}_{\bar{n}(m),n}\mathbf{C}_i^* \vee \mathbf{C}_i^*\mathbf{P}_{\bar{n}(m),n})\mathbf{I}_n^T, \tag{6.11}$$

$\forall\ m \in \mathcal{P}_i, n \in B(m)$, where $\mathbf{P}_{\bar{n}(m),n}$ is a permutation matrix of the rows $\bar{n}(m)$ and n, and $\mathbf{I}_n$ is an identity matrix where the n^{th} row has been deleted. The symbol $\vee$ represents the *or* operation between the matrices, which can be done taking into account that all the elements of the matrices are in the set $\{0, 1\}$. Let us note that row $\bar{n}(m)$ will be the same for all the robots with planes in the set $B(m)$. This means that all the robots will move the information of each association to the same row and will delete the rest of the rows, maintaining the size and the order of their maps.

The last problem is to combine the associations of planes in which there is no plane belonging to $\mathcal{P}_i$. To solve this problem a similar exchange of messages like the one to augment the local maps is carried out. In this case the exchanged information are the sets of associated planes $B(m)$. After d_v iterations all the associations are received by all the robots and using eq. (6.11) all the updates are done. Algorithm 9 schematizes the data association step for planar regions.

Algorithm 9 Data association and map reduction for planes - Robot i

```
1: Exchange local maps with neighbors
2: –Distributed Data Association
3: Execute Algorithm 1
4: –Map Reduction
5: Update $\mathbf{C}_i^*$ (eq. (6.11))
6: for $it = 1 \dots d_v$ do
7:     Send block associations to $j \in \mathcal{N}_i$
8:     Receive block associations from $j \in \mathcal{N}_i$
9:     Update $\mathbf{C}_i^*$ (eq. (6.11))
10: end for
```

Regarding the association of the features a similar process is performed for all the features belonging to the same planar region. After such process all the vectors $\mathbf{\Pi}^*$ are updated, having the same size and order.

6.3.3 Common reference

At this point, the local maps of planes have been associated and all of them have the same size and are equally sorted. Although the adjacency matrices are ready to execute the consensus algorithm, the features still require a common reference image. Otherwise posterior consensus will give erroneous results. In order to solve this problem a max consensus is used to decide, for each plane, which one of the local reference images is defined as the common reference image. In the same process the homography that transforms the features from the local references to the global one is also computed.

The number of observed features in the plane, $|\boldsymbol{\pi}|$, is the criterion chosen to decide the reference. For every plane, algorithm 10 is executed by every robot. In the algorithm, $\mathbf{H}_{j,i}$ represents the homography that transforms the plane from the coordinates in the local reference to the coordinates of the neighbor's reference. This homography is the one computed in the local data association step. The variables *feats*$_j$ represent the number of real features that each plane, j, has. For the global reference no subscript is used. After d_v iterations all the robots know the common reference (the one which contains more real features) and the homography to transform their coordinates. Applying this transformation and normalizing the coordinates, all the robots have their features in the same reference frame. The fictitious planes are not affected by changes of the reference. Taking this into account the robots with a fictitious plane do not participate in the max consensus algorithm of such plane.

Algorithm 10 Choice of a common reference - Robot i

```
1: RefPlane = $i$; feats = $|\boldsymbol{\pi}_i|$; $\mathbf{H}_{r,i} = \mathbf{I}$
2: for $it = 1 \dots d_v$ do
3:     Send [RefPlane,feats,$\mathbf{H}_{r,i}$] to all $j \in \mathcal{N}_i$
4:     Receive [RefPlane$_j$,feats$_j$,$\mathbf{H}_{r,j}$] from all $j \in \mathcal{N}_i$
5:     if feats$_j >$ feats then
6:         RefPlane = RefPlane$_j$; feats = feats$_j$; $\mathbf{H}_{r,i} = \mathbf{H}_{r,j}\mathbf{H}_{j,i}$
7:     end if
8: end for
9: Transform the features' coordinates of $\boldsymbol{\pi}$ using $\mathbf{H}_{r,i}$
```

6.3.4 Consensus on the global map

From here on all the robots have the information needed to perform the consensus. The graphs $\mathcal{G}_i^*$ satisfy now all the requirements to apply distributed consensus algorithms.

The adjacency matrices of the local maps are updated with the following rule

$$\mathbf{C}_i^*(m,n) = \mathbf{C}_i^*(m,n) \vee \mathbf{C}_j^*(m,n), \forall\, j \in \mathcal{N}_i \tag{6.12}$$

Theorem 6.3.1 (**Convergence of $\mathbf{C}_i^*$**). *The set of adjacency matrices $\mathbf{C}_i^*$, under iteration rule* (6.12)*, converges in d_v iterations to a common matrix $\mathbf{C}^*$ that includes all the links between planes.*

Proof. Let us consider separately each element of the matrices $\mathbf{C}_i^*$. Considering that the initial value of the elements is $\{0,1\}$, then it holds that

$$\mathbf{C}_i^*(m,n) \vee \mathbf{C}_j^*(m,n) = \max(\mathbf{C}_i^*(m,n), \mathbf{C}_j^*(m,n)). \tag{6.13}$$

Then eq. (6.12) can also be seen as an update of a max consensus algorithm, which is proved to converge in d_v iterations [78].

Since $\mathbf{C}_i^*(m,n) = 1$ implies that there is a link between planes m and n and $\max(0,1) = 1$ then all the links of the adjacency matrices are preserved.

With respect to the features, since no observations are assumed to be better than others, a distributed averaging of the matched observations is computed. Let us note that although the fictitious planes do not provide any information to the final consensus, they affect it in the sense that the final value is divided by the total number of robots. The third coordinate of the features, which is the homogeneous coordinate, plays here a fundamental role. Let us recall that for the fictitious features the third coordinate, usually related with the scale of the point, was also set to zero. Recalling the consensus algorithm using homogeneous coordinates presented in Chapter 4.2, the real average of the feature coordinates will be achieved.

Let us note that the consensus is only carried out for the coordinates of the features and not for the whole descriptor. For the SURF descriptors a leader election algorithm is used so that every robot has the same set of SURF descriptors after d_v steps.

6.4 Experimental Results

Several experiments have been carried out in order to evaluate the properties and the behavior of the whole method. We have tested it using different real image data sets that correspond to different locations of man-made environments with plenty of planar regions. The first data set has been recorded indoors (House data set). It consists of 3600 frames from different rooms. The second data set is composed by nine different sequences recorded outdoors in a downtown Zaragoza area (Downtown data set). Figure 6.2 shows a view of the map where these sequences have been acquired and the topology used in the experiments.

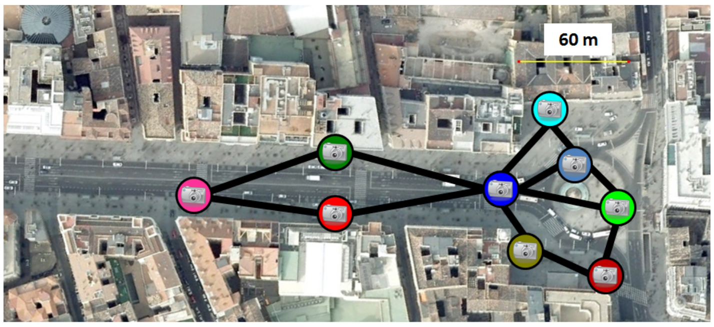


Figure 6.2: Map of Zaragoza (Downtown data set), where the nine sequences have been acquired and the communication graph among the agents. Each camera represents one robot with its local map and the black edges are the communication links in the network.

The camera used in all the cases has been a *Panasonic Lumix FX-500*. In all the cases the camera has moved with 6DOF. For all the images we have used SURF descriptors [13] for matching. The computation of the homographies has been done using DLT+RANSAC algorithm. It is well known that under pure rotations or small motions all the features can be fitted to the same homography. In video sequences with high frame rates this is a common situation. To avoid this problem we have followed the idea of [128] to select key frames among the sequence:

- There are as many images as possible between the key frames $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$.
- There are at least M matches between the key frames $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$.
- There are at least N matches between the key frames $\mathcal{I}_k$ and $\mathcal{I}_{k+2}$.

The results are divided in three sections. In the first experiment (sec. 6.4.1) we analyze how the triple matching step works and the properties of the extracted planes. In the second experiment (sec. 6.4.2) we analyze the properties of the individual maps created using the sequences of images from each camera separately. We have compared the resulting graphs obtained for both data sets with graphs made by images [165]. For the latter approach we have stored the SURF descriptors of each image and we have imposed the homography constraint between frames to observe the pros and cons of using images or planes. In the last experiment (sec. 6.4.3) we have tested the multi-robot distributed approach for the Downtown data set.

6.4.1 Extraction of the planar regions

The House data set has helped us to test the detection of planes since it has a lot of different planar regions. In Fig. 6.3 some of the segmented regions are depicted. We observe that although the method detects several planes which are the same in the ceiling, there are no wrong planes segmented.

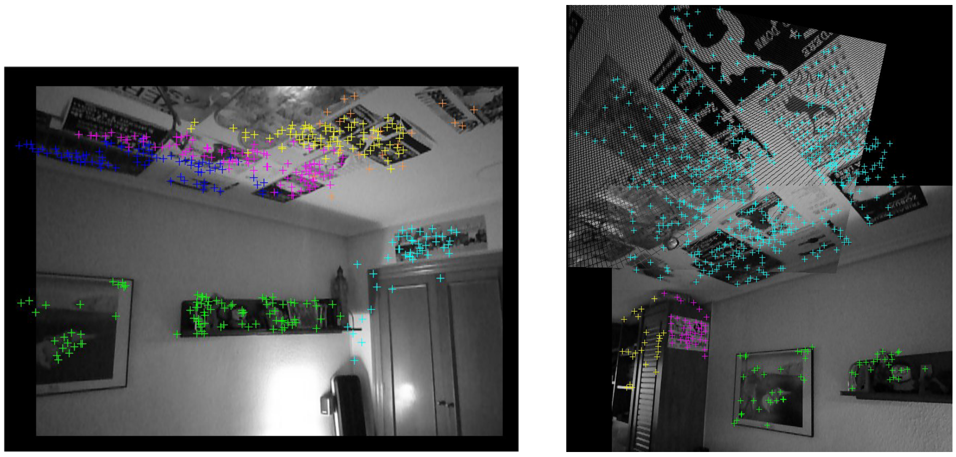


Figure 6.3: Planar regions extracted from the House data set. Each region is represented with a different color. In the bottom figure we have added points corresponding to the ceiling transformed with the computed homography.

The nine sequences in the Downtown data set are more challenging because in this case the scene contains dynamic elements such as people and cars. There are also a lot of trees that occlude the building facades and, in some cases, there are also illumination changes when the camera points towards the sun. Figure 6.4 and 6.5 depict two examples of two different planes extracted using our method. Even when the extracted planes contain some outlier features the results are still quite good and, what is more important, we observe that the method maintains the planar regions correctly adding new features as they appear.

We have observed that sometimes in a real scenario many dynamic objects may generate planar regions. These planar regions are undesirable in a practical situation. A lower bound on the number of features of a plane after the map generation clears almost all the undesired planes. Other times the algorithm considers as a planar region a set of features belonging to different planes but coplanar between them. Usually in the next steps the algorithm grows this plane considering only the biggest number of real coplanar features. The last problem observed comes from the homology test. The homology depends both on the motion between the images and the parameters of the planes. If the motion is small, it is possible that 2 different planes are fused because the three eigenvalues of the homology will be close to one. The algorithm used to select the key frames [128] mitigates this problem. Since the algorithm tries to skip as many images as possible between two consecutive key-frames, consecutive frames will be, in general, far away from each other in terms of distance. Therefore, the homology will only be close to the identity when the planes must actually be fused.

6.4.2 Single camera topological map of planar regions

Using the planar regions extracted with our algorithm from the sequences of images, we have computed the associated graphs of planes. We have compared the resulting graphs with the image graphs created following the approach in [165]. Table 6.1 shows the comparison of the graphs generated using the House data set and Table 6.2 shows the same comparison for the nine sequences of the Downtown data set.

In both cases the graph made of planes has less nodes and edges than the graph composed by images. The amount of space for storing the information is drastically reduced using our approach (Tables 6.1 and 6.2).

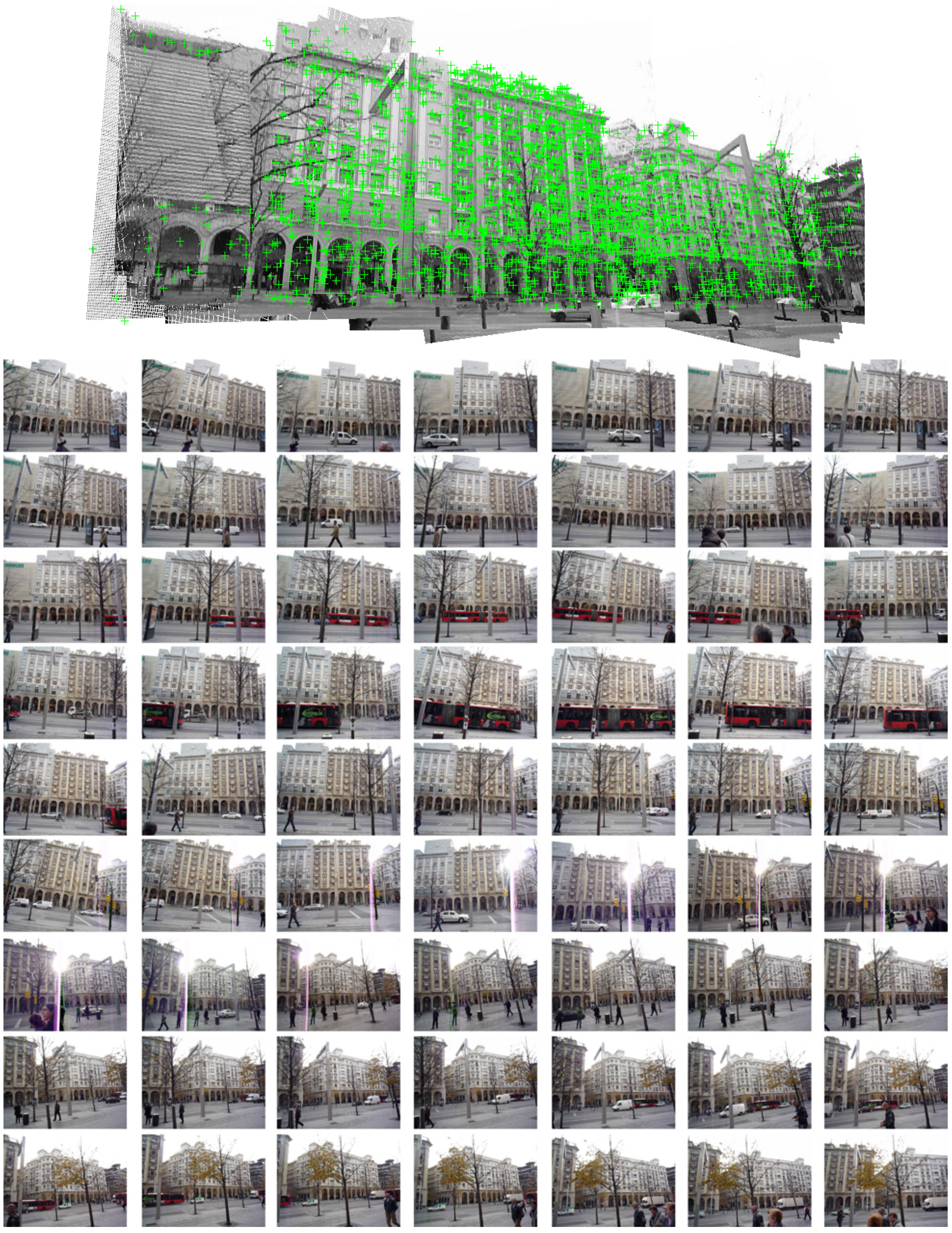


Figure 6.4: Planar region extracted from one sequence of the Downtown data set. The algorithm is able to track and grow the plane over 63 different frames with dynamic elements and some illumination changes. The top figure represents the plane with the 482 detected features. The plane has been observed in all the images depicted below.

Notice that in our approach the size of each plane is not bounded and there can be big differences between nodes. In a visual memory made by images all the nodes will have similar size (the features per node can be assumed to be bounded) whereas the graph made of planes may contain very small planar regions with just a few features, and other nodes can represent large planar regions with hundreds of features and many

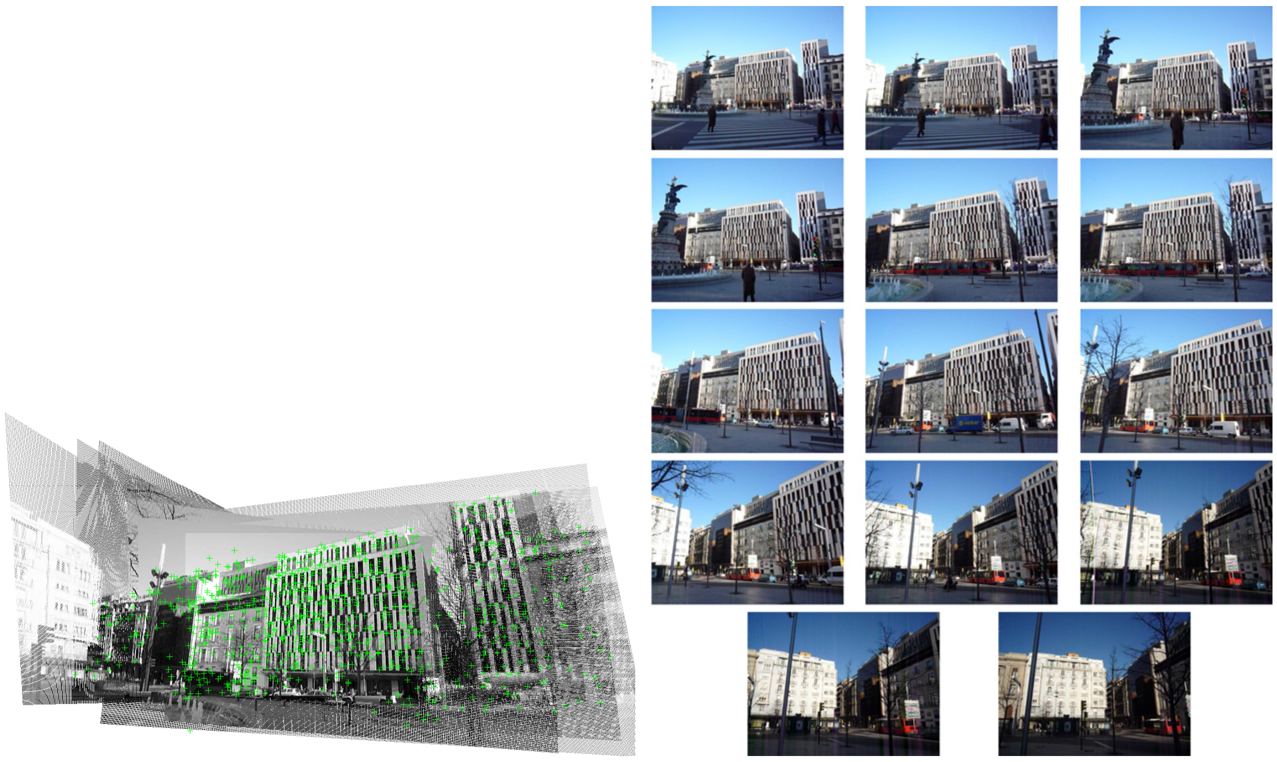


Figure 6.5: Another planar region extracted from one sequence of the Downtown data set.

Table 6.1: Results for the house sequence

Map	Nodes	Edges	Feats	Feats/node	Size (MB)
Images	140	1728	78124	558	29.0
Planes	30	78	10832	361	4.0

homographies.

If the camera moves too fast or if there is a sequence of images in which there are no planar regions the topological map will be unconnected. To prevent these situations, we have also imposed consecutive planes to be connected in the local maps.

6.4.3 Multi-Camera Distributed Topological map

The distributed building of a topological map has been done using the maps generated from the Downtown data set. The limited communications between the robots are shown in Fig 6.2. The diameter of the graph is 4, which means that most part of the algorithms will finish only in four steps.

As previously commented, the triple matching algorithm finds a lot of small planes which are a mix of different outliers (trees, buses, people and noisy features). These planes do not apport real information and it is better to discard them. In order to do so we have set a threshold of 50 features, so that planes containing less features than the threshold are not considered for the distributed global map. As shown in Table 6.3, after erasing the small planes, only 132 of the 287 planes take part in the distributed process. These planes amount a total of 46981 features.

Initially the robots exchange the information about their maps to create fictitious planes. After four steps, every robot has a map with 132 planes and 46981 features. Then the robots exchange their maps with their neighbors and perform the local data association step. Figure 6.6 shows one plane seen by three different

Table 6.2: Results for the downtown Zaragoza sequences

Agent	Map	Nodes	Edges	Feats	Feats/node	Size (MB)
1	Images	300	3966	315830	1052	117.4
	Planes	104	340	15694	150	5.89
2	Images	78	796	58219	746	21.6
	Planes	17	62	3958	232	2.28
3	Images	33	230	29777	902	11.1
	Planes	8	28	1084	135	0.4
4	Images	465	7192	431607	928	160.6
	Planes	92	288	18497	201	6.95
5	Images	86	820	74123	861	27.5
	Planes	34	110	4104	120	1.54
6	Images	101	1434	79700	781	29.6
	Planes	11	40	409	361	1.68
7	Images	32	304	29258	914	10.9
	Planes	6	14	1611	268	0.61
8	Images	35	230	30434	869	11.3
	Planes	9	18	2021	224	0.77
9	Images	32	318	29237	913	10.9
	Planes	6	12	1162	193	0.44
Total	Images	1162	15290	1078185	927	400.9
	Planes	287	912	48540	169	20.20

Table 6.3: Evolution of the global map's size

Step	Nodes	Feats
Initial global map	287	48540
After erasing small planes	132	46981
Data Association	121	46351

neighbor robots with the found matches. The local associations delete a total of 11 planes and 340 features, remaining 121 planes and 46351 features (third row in Table 6.3). We have observed that the small number of associations is mainly due to the different points of view of the trajectories and not because of mismatching. Even so, in the multi-robot mapping it is advisable to use larger RANSAC thresholds.

After data association the robots execute the max consensus algorithm in order to fix the common references for each plane. In the example of the Figure 6.6 the reference plane is the top one because is the one with the most features.

Finally, the robots execute the consensus rule to reach an average on the features and a consensus on the adjacency matrices of the topological maps. Figure 6.7 shows the consensus evolution of the three coordinates by the nine robots; we can see that they do not reach the desired average in f_x and f_y because only three of the nine robots have information about the feature but the average considers the value of the nine. However, the normalized coordinates (Fig. 6.8) converge to the desired value.

A comparison with the maps made by images has been done in order to analyze the amount of information transmitted through the network. This analysis has been carried out analytically considering the network topology and the information exchanged. An upper bound on the number of messages required to transmit some information to the considered network (flooding) is 14. That means that for the map made of images, the 400.9 megabytes (Table 6.2) are transmitted 14 times, giving a total of 5612 transmitted megabytes. The breakdown of the transmitted information using topological maps is in Table 6.4. The results show that the total information transmitted through the network using processed planes is considerably smaller (20%). Also the final global map is better using planar regions, since every match mixes the information reducing the total

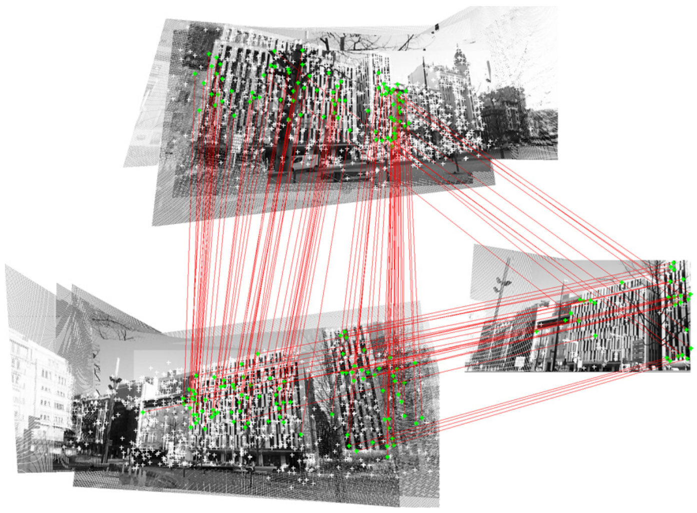


Figure 6.6: Multi-Robot Distributed Topological Map. Example of a planar region viewed in three different sequences. Red lines are the matches among the planes. In this example the reference plane is the top one.

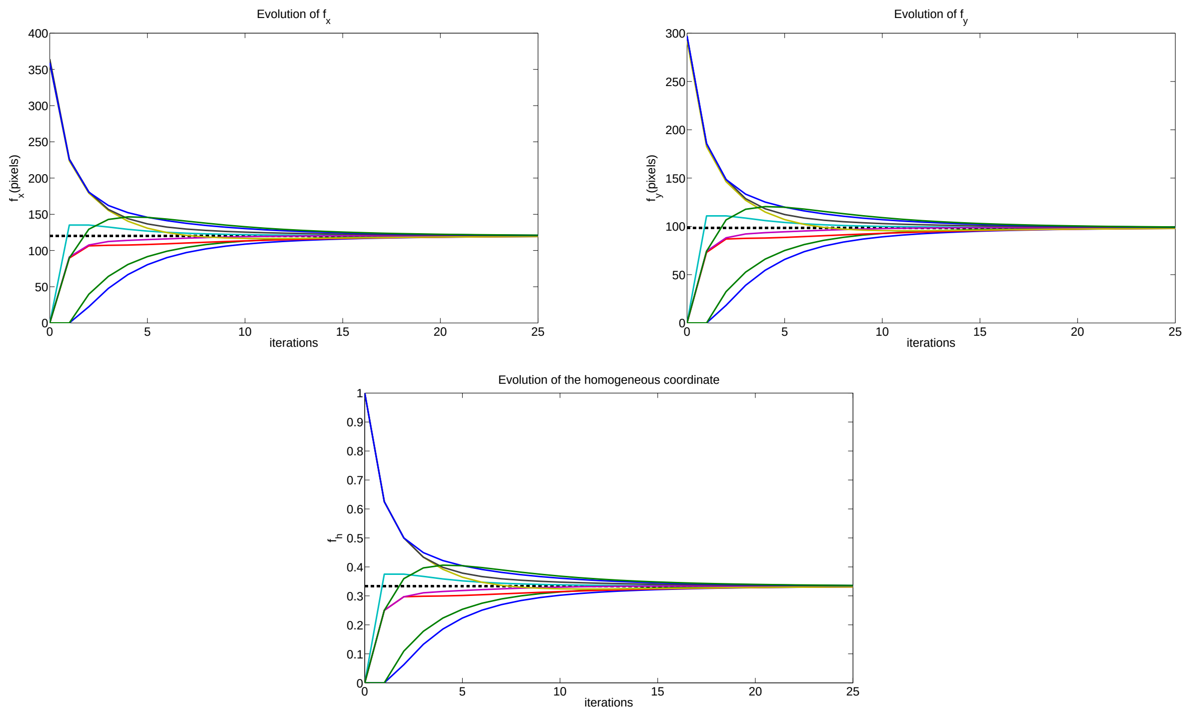


Figure 6.7: Consensus process for feature coordinates. Initially each agent has a different value of the feature coordinates. The nine agents exchange the information they have with their neighbors. It is observed that after 25 iterations consensus has been achieved and all the agents have the same value of the coordinates.

amount of data, whereas with images the total size is the sum of the local maps.

6.5 Discussion

In this chapter we have studied the problem of distributed map building by a team of robots using planes as the features to represent the map. The idea of storing planes as information in the top graph rather than

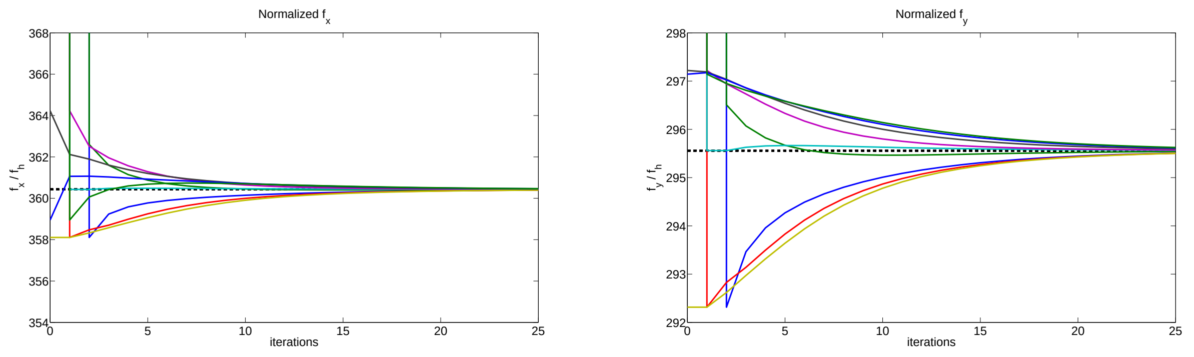


Figure 6.8: Consensus process for normalized feature coordinates. Evolution in normalized coordinates of the value of the same feature than in figure 6.7. Only three agents have a real observation of the feature, which are $(358.1072, 292.3148)^T$, $(364.2285, 297.2203)^T$ and $(358.9520, 297.1431)^T$. The rest of the agents have fictitious features with initial value equal to zero. At the beginning the agents with a fictitious coordinate have ∞ values. We can see that eventually the nine agents reach the same value, which corresponds to the average of the measurements of the three agents that observed the feature.

Table 6.4: Amount of information transmitted (MB)

Total using images	$\simeq 5612$
Augment maps	< 1
Share maps	45.2
Data Association	13.2
Block Association	< 1
Leader Election	< 1
Consensus	1091.5
Total using planes	$\simeq 1150$

whole images presents several advantages for the individual robots and for the whole team. We have presented a fully distributed solution using distributed consensus methods and inter-image homographies to relate the information in different images and different maps.

At the beginning, each robot creates a local map from the sequence of images it acquires. The planar regions are extracted considering the information from previous images using a triple set plane-image-image for feature matching, tracking and growing the planar regions as new areas of the plane become visible. The planes are organized in a graph, built simultaneously to the extraction, where homologies are used to detect new planes when they appear and define a relation of co-visibility between the planes.

The second part of the chapter has dealt with the problem of fusing the individual maps of the robots using distributed consensus. We have seen that the use of homographies also simplifies some of the most important problems in this context. Homographies can be used to solve the data association problem, providing with a robust mechanism to detect planes in different maps. The common reference frame, required to compute the consensus, only requires a max-consensus and the multiplication of different homographies. Finally, since homographies already use homogeneous coordinates, the final consensus can be achieved by all the robots, even if most of them have not seen a specific plane. In the end, we show that the whole team of robots achieves the consensus about the global map in a simple and efficient way.

To conclude, we can see that well known computer vision techniques can play a fundamental role in achieving a consensus about the visual information perceived by all the robots. An interesting question is if we can use similar computer vision techniques, which have been very useful in perception tasks, to simultaneously control to team of robots to achieve a specific configuration, e.g., flocking behavior (consensus in the attitude) of the robots. This will be the focus of the next chapter.

Chapter 7

Distributed Consensus in Control using Epipoles

"Fine art is knowledge made visible." In this chapter we give a distributed solution to the problem of making a team of non-holonomic robots achieve the same heading (attitude consensus problem) using monocular cameras. The use of cameras with constrained field of view limits the information the robots perceive compared to other omnidirectional sensors. This makes the coordination problem more complicated, because the robots will not always be able to observe their neighbors, specially if they intend to observe the environment. By using structure from motion computed from images, the robots can estimate their relative orientations from common observations of the environment without the necessity of directly observe each other. In this way, the robots are able to achieve the consensus in their heading while performing exploration tasks. In addition, the control is robust to changes in the topology of the network and does not require to know the calibration of the cameras in order to achieve the desired configuration. Finally, we have tested our controller in simulations using a virtual environment and with real robots moving in indoors and outdoors scenarios.

7.1 Introduction

The algorithms we have proposed are mainly oriented to achieve a consensus on the visual information in perception tasks. While this is the main interest of the Thesis, vision sensors can also be used to control the motion of the robots while exploring the environment. The consensus problem is also of high interest in coordination tasks, solving problems like rendezvous and flocking. In these cases, instead of considering common observations, the consensus takes into account the relative positions between neighboring robots. However, the use of conventional cameras limits the field of view of the robots, which means that the robots may not be able to see each other, specially if their commanded task is to observe the environment rather than to coordinate. Therefore, additional mechanisms are required in this setup so that the robots can estimate their relative headings and achieve a common orientation, the consensus.

The coordination of teams of robots is a problem that has received a lot of attention in the last years. Cooperative solutions to different coordination problems can be divided into leader-follower schemes [35, 54, 56, 60, 142, 147] and nearest neighbor rules [11, 33, 51, 61, 98, 132, 144, 161]. In the leader-follower approaches each robot designs its control input considering only the information provided by a single neighbor robot, the leader. In the approaches based on nearest-neighbors rules, all the robots play the same role in the formation and each robot designs its control input using the available information provided by direct neighbors in the communication graph. Within the multiple coordination problems that can be solved, we are interested in the problem of making all the robots achieve a common heading, also known as the attitude consensus problem or the problem of flocking,

Distributed solutions based on nearest neighbor rules dealing with this problem have only been focused on the coordination aspects, setting aside the additional problems of perception. If the robots need to explore the environment, as we have considered along the Thesis, then depending on their sensors they will not be able to observe each other. Solutions that consider omnidirectional range [61] or visual [99, 149] sensors do not suffer of this limitation. However, when the used sensors have field of view constraints, as is the case of monocular cameras, the observation of the neighbors may constrain the perception of the environment. If the robots are required to observe each other to coordinate their motions, the perception of the environment will be restricted to positions that satisfy this constraint.

As happened in the previous chapter with the problem of cooperative mapping, the attitude consensus problem can use computer vision methods to overcome this limitation. Specifically, the epipolar constraint [79] between pairs of images represents a very useful tool in this scenario due to its natural robustness to mismatching. A first approach using the epipoles to control the motion of a robot appeared in [12]. Non-holonomic constraints were introduced in [75, 80]. In all these approaches the goal is to control one robot and move it to a fixed position, specified by some target image. In our approach there are no fixed images, as all the robots move.

The solution presented in this Chapter assumes that each robot moves on the plane with non-holonomic motion constraints. The rest of the capabilities of the robots follow the directrices explained in Section 2.2. In order to make the robots achieve the same heading we propose a controller that computes the epipoles between the images of neighboring robots. While we focus on solving this problem, the provided solution is well suited for exploration and mapping purposes. Additionally, we have chosen to use the epipoles in the controller because their computation does not require an explicit decomposition of the fundamental matrix or knowledge about the internal parameters of the camera.

The contributions of the approach presented in the Chapter are:

- A distributed controller to align the orientations of all the robots using the visual information provided by monocular cameras. We make use of the epipolar constraint to achieve this objective.
- With our controller, the robots do not need to directly observe each other but just common features of the environment. This goes in synchrony with the rest of the algorithms presented in the Thesis, that intend to give a solution for the cooperative perception of the environment. In addition, by making all the robots to orient in the same direction, problems such as data association are simplified, reducing it to a linear translation problem [31].
- Additionally, the controller does not require a precise knowledge of the calibration of the cameras. If all the robots have the same camera we demonstrate convergence to the consensus. Otherwise we provide with error bounds in the final configuration.

This work has been partially published in [97]

7.2 Description of the system

In this section we define the dynamics of the robots and briefly review the epipolar constraint.

7.2.1 Dynamics of the robots

We consider that the team of robots is moving on the plane. The dynamics of each robot is described by the unicycle model:

$$\begin{bmatrix} \dot{x}_i \\ \dot{z}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \sin(\theta_i) & 0 \\ \cos(\theta_i) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ w_i \end{bmatrix}, \tag{7.1}$$

where $[x_i, z_i, \theta_i]^T \in \mathbb{R}^3$ is the state of robot i (position and orientation) expressed in some world reference frame and $[v_i, w_i]^T \in \mathbb{R}^2$ is the control input of the robot. Since the goal is to align the orientation of the robots, we will not pay much attention to their linear velocity and will assume that is constant for all the robots, $v_i = v \geq 0,\ \forall i$.

Now, given two robots, i and j, we make use of the polar coordinates, distance, r_{ij}, bearing angle, ψ_{ij}, and relative orientation, θ_{ij}, to describe their relative state

$$\begin{aligned} r_{ij} &= \sqrt{x_{ij}^2 + z_{ij}^2} \in \mathbb{R}_{\geq 0}, \\ \psi_{ij} &= \arctan(x_{ij}/z_{ij}) \in (-\pi/2, \pi/2], \\ \theta_{ij} &= \theta_j - \theta_i \in (-\pi, \pi], \end{aligned} \tag{7.2}$$

where $[x_{ij}, z_{ij}]^T = [x_j - x_i, z_j - z_i]^T$ are the cartesian coordinates of robot j expressed in the reference frame whose origin coincides with robot i (Fig 7.1).

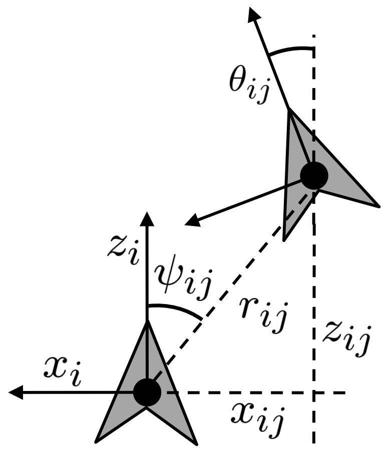


Figure 7.1: Coordinates of robot j in the reference frame of robot i.

7.2.2 Camera model and output of the system

Initially, we assume that the cameras of all the robots are the same. The calibration matrix of these cameras is unknown for the robots and equal to $\mathbf{K} = \text{diag}(\alpha, \alpha, 1)$, with $\alpha > 0$, the focal length of the camera measured in pixels. This is equivalent to say that the camera has no skew and that the origin of the image coordinates is fixed on the center of the image.

The use of monocular cameras implies that the depth of the scene will be unknown. This means that r_{ij} will not be available to the robots. If the robots are intended to explore the environment, then there will also be many situations in which they will not be able to observe each other in a direct way because of the limited field of view. To overcome this limitation the robots can exchange their images and use structure from motion techniques to estimate their neighbors positions (see Fig. 7.2). However, the lack of knowledge about the calibration of the camera means that the robots will have no direct means to estimate the exact ψ_{ij} and θ_{ij}.

For any pair of neighbor robots, i and j, the output of the system will be defined by the epipoles of the acquired images. Given a pair of images, it is possible to estimate the fundamental matrix, $\mathbf{F}_{ij}$, that relates them, provided that there are at least 7 correspondences between them [79]. After that, the epipoles, $e_{ij} = [e_{ijx}, e_{ijy}]^T$ and $e_{ji} = [e_{jix}, e_{jiy}]^T$, can be computed in a linear way as the intersection of the epipolar lines defined by $\mathbf{F}_{ij}$ and the matched features. Due to the planar motion, the y-coordinate of all the epipoles will be equal and constant for any pair of images. The x-coordinate of the epipoles satisfies

$$e_{ijx} = \alpha \tan(\psi_{ij}),\ e_{jix} = \alpha \tan(\psi_{ij} - \theta_{ij}). \tag{7.3}$$

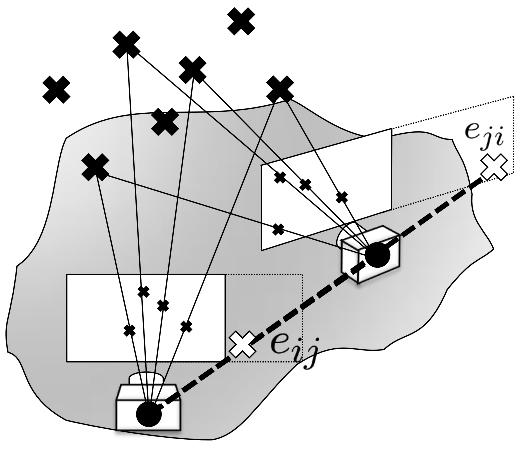


Figure 7.2: Observation of the neighbor robot with the epipoles. Both robots observe the same features in the scene and using structure from motion they can compute the epipoles without the necessity of observing each other.

Let us note that to compute the epipoles, the robots do not need to know the calibration matrix **K**. For simplicity purposes, in the following we use e_{ij} and e_{ji} to refer the x-coordinate of the epipoles.

7.3 Consensus controller

In the attitude synchronization problem, all the robots in the network shall achieve the same orientation, i.e., $\theta_{ij} \to 0$, $\forall i, j \in \mathcal{V}$, as $t \to \infty$. We propose a control law for each robot that uses the epipoles as measurements to achieve this objective.

Given a pair of neighbor robots, by eq. (7.3), a necessary condition for the attitude alignment is that their epipoles must be equal, $\theta_{ij} = 0 \Rightarrow e_{ij} = e_{ji}$. However, note that $e_{ij} = e_{ji}$ does not necessarily imply consensus in the orientation because for $\theta_{ij} = \pi$ the two epipoles are also equal. This imposes a constraint on the initial orientations of the robots. We will require that initially $\theta_{ij} < \pi/2$, $\forall i, j \in \mathcal{V}$, so that the controller is able to align the robots properly.

We define the misalignment in the epipoles as

$$w_{ij} = \begin{cases} d_{ij} & \text{if } |d_{ij}| \leq \frac{\pi}{2} \\ -\text{sign}(d_{ij})(\pi - |d_{ij}|) & \text{otherwise} \end{cases}, \tag{7.4}$$

where

$$d_{ij} = \arctan(\frac{e_{ij}}{\beta}) - \arctan(\frac{e_{ji}}{\beta}) \in (-\pi, \pi], \tag{7.5}$$

and $0 < \beta < \infty$ is some fixed positive constant to choose.

Several aspects justify this misalignment function. First of all, eq. (7.5) is a bijective mapping $(-\infty, \infty) \to (-\pi, \pi)$ that reduces the misalignment in the epipoles to quantities that represent something more similar to angular distances. This reduction also implies smaller control gains. Secondly, equation (7.4) introduces the geodesic distance in the difference between the epipoles and is used to select the closest path (clockwise or anti-clockwise) that makes both epipoles be the same. Finally, note that, if $\beta = \alpha$, then the setup is calibrated, $d_{ij} = \theta_{ij}$, and the relative orientation between the robots can be computed from the epipoles. However, for the moment we assume that this is not the case and $\beta \neq \alpha$.

The control input w_i of each robot is defined as:

$$w_i = K \sum_{j \in \mathcal{N}_i(t)} w_{ij}, \tag{7.6}$$

where $K > 0$ is the controller gain.

In order to prove the stability of the proposed controller, we will make use of the following lemma.

Lemma 7.3.1 (Properties of the controller). *The following properties hold:*

1. $w_{ij} = -w_{ji}$.
2. $\sum_{i \in \mathcal{V}} w_i = 0$.
3. *sign*(e_{ij}) = *sign*$(e_{ji}) \Rightarrow |d_{ij}| < \pi/2$.

To analyze the stability of the controller, as well as the achievement of the desired configuration, we will first assume that the communication topology is fixed.

Theorem 7.3.2 (Convergence to a common orientation). *Consider a robotic network like the one defined in section 7.2, with the robots initially oriented in such a way that* $|\theta_{ij}| \le \theta_M < \pi/2,\ \forall i, j \in \mathcal{V}$. *If the robots use the control law* (7.6) *with* β *satisfying*

$$\alpha \tan(\frac{\theta_M}{2}) < \beta < \frac{\alpha}{\tan(\frac{\theta_M}{2})}, \tag{7.7}$$

then $\lim_{t \to \infty} \theta_{ij} = 0, \forall i, j \in \mathcal{V}$, *i.e., the system will reach consensus.*

Besides the stability of the system, the theorem provides a relation between the calibration parameter and the relative orientation between the agents. Now we proceed to show the behavior of the controller when the communication topology changes over the time.

7.4 Robustness of the controller to more realistic conditions

In this section we study how the controller is affected by changes in the communication topology and the use of different cameras by the robots.

7.4.1 Changes in the communication topology

The controller presented in the previous section is only valid for a fixed communication topology. However, under real conditions it is most likely that the communication topology will change as the robots move. There are multiple reasons to study the robustness of the controller against changes in the communication topology. The most usual comes from the motion of the robots but the use of visual sensors introduce other issues that can also be modeled as changes in the topology.

- **Changes in the topology due to the motion of the robots:** The controller should take into account that the motion in the robots may introduce some changes in the graph that defines the communication topology.
- **Changes in the topology due to perception issues**: There are also perception issues that may affect the neighborhood of each robot. It is possible that two neighboring robots cannot compute their epipoles due to blurry images or temporal occlusions, which would be the same as to assume that they are not neighbors in the communication graph.
- **Changes in the topology for computational demands:** Finally, computational issues should also be considered to model the communications using a time-varying graph. The computation of the epipoles using a robust algorithm, e.g., DLT+RANSAC [79], requires some time and, although one or two fundamental matrices can be computed in a reasonable amount of time, robots with a larger number of neighbors may not be able to keep up with the rhythm of the continuous time controller.

For all these reasons it is interesting to analyze the controller in the presence of changes in the communication topology. The following result states that the proposed controller can handle changes in the communication topology, reaching the desired agreement of the headings of the robots.

Proposition 7.4.1 (Convergence to a common orientation with topology changes). *Consider a robotic network like the one defined in section 7.2, which satisfies the conditions stated in Theorem 7.3.2 and Assumptions 2.2.5 and 2.2.6. If the robots use the control law*

$$w_i = K \sum_{j \in \mathcal{N}_i(t)} w_{ij},$$

then $\lim_{t \to \infty} \theta_{ij} = 0, \forall i, j \in \mathcal{V}$.

7.4.2 Cameras with different calibrations

When modeling the team of robots in section 7.2.2 we assumed that all the robots wore the same camera with the same intrinsic parameters, i.e., the parameter α was the same for all the robots.

Unfortunately, in a real scenario, even if all the robots are equipped with the same camera (same model, same characteristics, etc.), it is nearly impossible that all the calibrations of the N cameras will be exactly the same. Let us consider that each robot is wearing a different camera with a different calibration. We denote by α_i the calibration parameter of robot i. If the parameter β in (7.5) is kept at the same value for all the robots, then the final configuration of the network will not be the desired consensus because equal epipoles do not imply equal orientations anymore. The following proposition gives the direct expression of the final error between the orientation of pairs of robots:

Proposition 7.4.2 (Error with different cameras). *The error in the orientation is equal to*

$$\tilde{\theta}_{ij} = \arctan\left(\frac{(\alpha_i - \alpha_j)\sin\psi_{ij}\cos\psi_{ij}}{\alpha_i \sin^2\psi_{ij} - \alpha_j \cos^2\psi_{ij}}\right) \tag{7.8}$$

Equation (7.8) shows that the final error in the orientation of the robots depends, not only on the difference between their calibrations but also on their relative bearing angle. This is good news because we can make the robots achieve the desired configuration, even if they do not have the same cameras.

Corollary 7.4.3. *If the robots are in parallel or leader follower formations, i.e.,* $\psi_{ij} = \pi/2$ *or* $\psi_{ij} = 0$*, then the consensus is achieved.*

The corollary is proved just by replacing these two values of ψ_{ij} in eq. (7.8) and seeing that the error value is always zero.

Nevertheless, in the experiments section we show the robustness of the controller to different cameras, even if the bearing angle between pairs of robots is different to $\pi/2$ or zero.

7.5 Experiments

In this section we show the behavior of the proposed controller in two different simulated scenarios and with a team of 3 robots moving in different environments.

7.5.1 Simulations

The properties of the proposed controller are shown in simulations. The experiments have been carried out using Matlab. We have considered a robotic network composed by ten robots with initial positions and orientations depicted in Fig. 7.3 and communications defined by the dashed lines. To simulate the vision system we have randomly generated a set of 3D features in the environment. The cameras have calibration matrix $\mathbf{K} = \text{diag}(300, 300, 1)$, and a resolution of 640×480 pixels. This implies that the robots have a limited field of view of 94 degrees. Under these conditions not all the robots can observe each other in their images. For example, robot two can only communicate with robot four and there are no other robots visible in its field of view. However, using the epipoles it can compute a control input to align its heading with the one of robot four.

The results of using the proposed controller with $\beta = 250$ are shown in Figs. 7.3 and 7.4. Since the maximum relative orientation between a pair of robots is 1.23 the bounds on β required to converge are $212 < \beta < 423$ and in this case the controller reaches the consensus. The right figure in Fig. 7.3 shows the evolution of the orientation of the robots, which converge to the same value for all of them. The left figure in Fig. 7.4 depicts the control inputs and the right figure the evolution of all the pairs of computed epipoles.

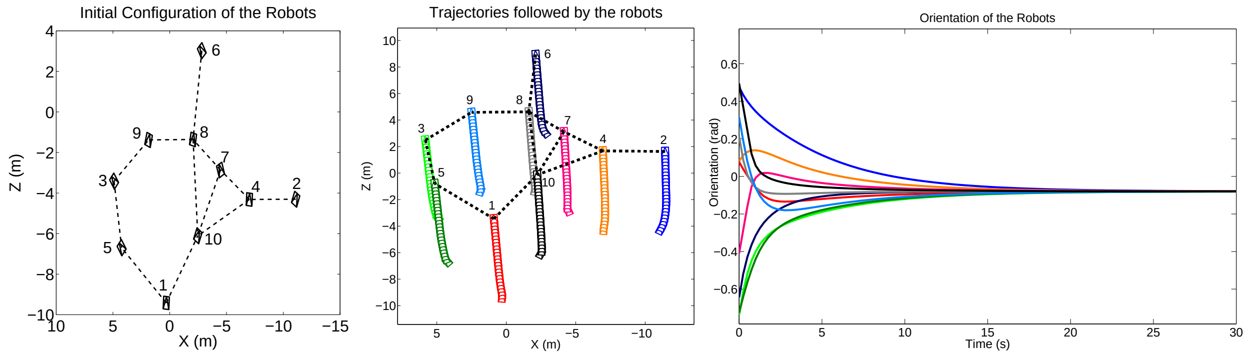


Figure 7.3: Initial positions (left) and trajectories followed by the robots (middle). Dashed lines represent direct communications between robots. In the right figure we can observe the values of the orientation of the robots, reaching the consensus.

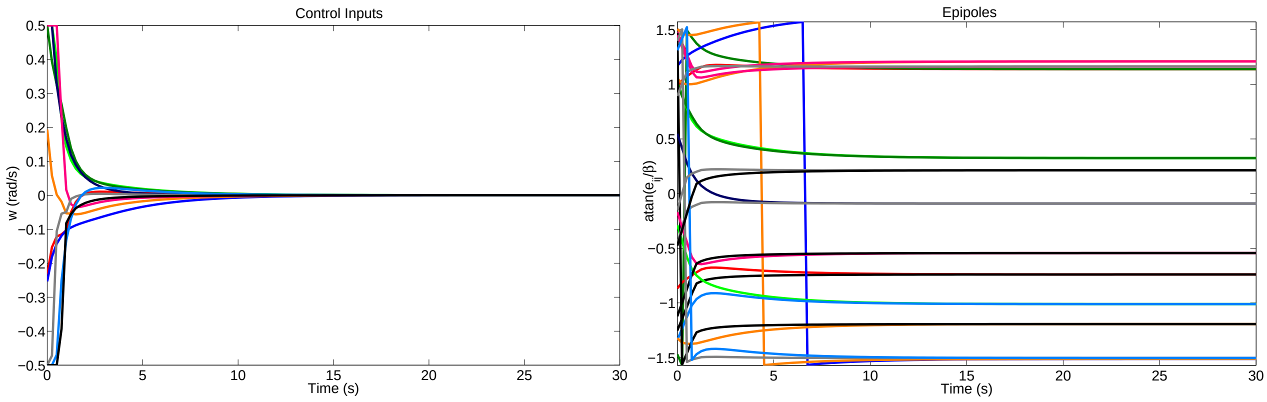


Figure 7.4: Control inputs and evolution of the pairs of computed epipoles for the robotic network in Fig. 7.3.

We have also introduced in the simulation some constraints to make it more realistic. To consider the time required for the computation of the epipoles we have discretized the controller with time step equal to 0.25 seconds. At each iteration, each robot randomly selects a subset of its neighbors to compute the epipoles. This selection generates changes in the network topology, transforming the system into a switching one. Also this selection reduces the number of fundamental matrices that the robots need to compute, improving the computational cost. The dwell time required in Assumption 2.2.5 comes from the discretization. Joint connectivity is preserved because along the time all the links of the original graph (which is connected) are selected at some point. The results of the evolution of the system are shown in Fig 7.5. We can see that the robots still achieve the attitude alignment.

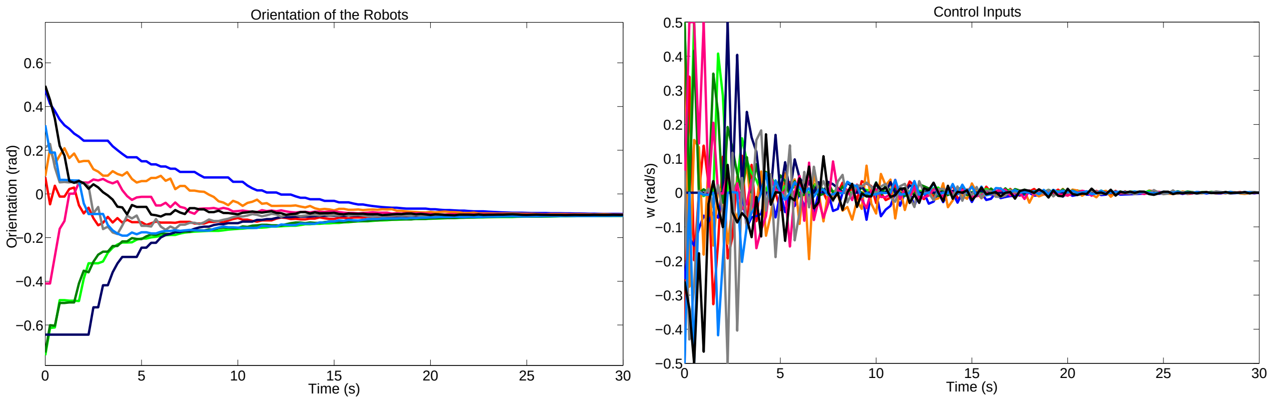


Figure 7.5: Evolution of the network with switching topologies. Orientation of the robots (left) and control inputs (right).

7.5.2 Simulations in a virtual scenario

In the second simulation we have consider a more realistic scenario. Using the virtual reality toolbox of MatLab we have created a virtual world. In this way, the robots acquire virtual images of resolution 640×480 pixels depending on their position and orientation. In this case the robots need to use real computer vision algorithms to estimate the epipoles. We have extracted SIFT [77] features from the virtual images and the 8 point algorithm with RANSAC [57] to match them in a robust way and to compute the epipoles between pairs of robots. An example of the images acquired by the robots and the features extracted and matched can be found in Figure 7.6. The results of the simulation is in Figure 7.7. Again the robots reach the desired configuration

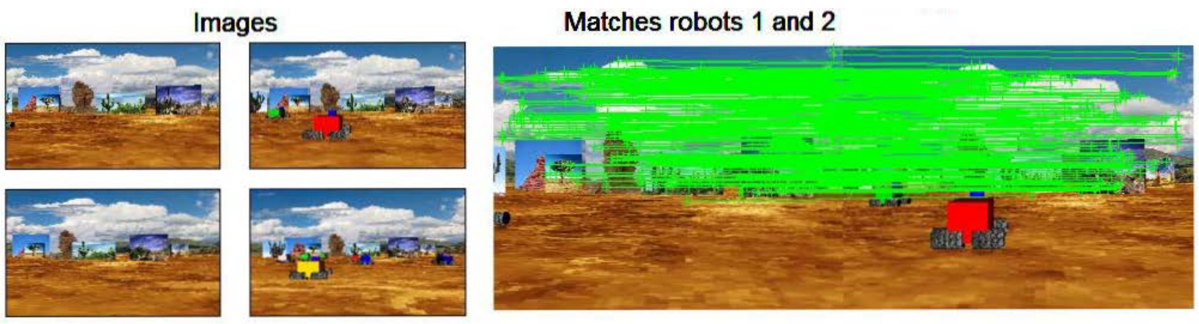


Figure 7.6: Images acquired in the virtual environment and matches between the SIFT descriptors.

7.5.3 Experiments with real robots

We have also tested our proposal in a real platform. In this section we briefly describe the whole setup, the experiments carried out and we show the results obtained. Nevertheless, for a better visualization of the results we refer the reader to the attached videos, where the whole motion of the robots can be seen.

The experiments have been carried out with three robots Pioneer 3Dx inc with non-holonomic motion constraints like the ones described in eq. (7.1). Each robot has been equipped with a laptop and a wireless antenna to communicate with the other two robots. Regarding the vision system of the robots, in most of the experiments we have used a Kinect camera onboard of each robot. In order to analyze the robustness of the controller when using different cameras in one of the experiments we have equipped one robot with a unibrain

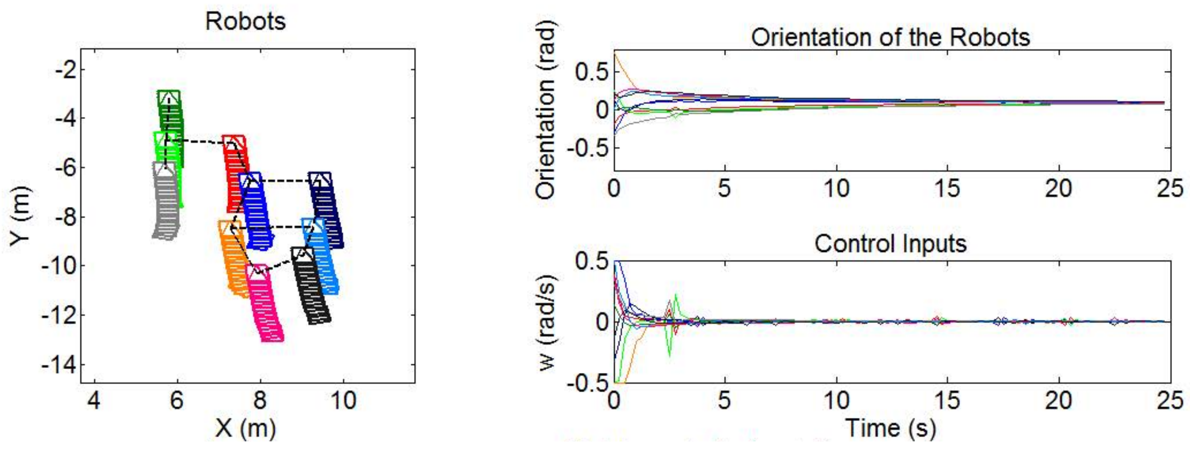


Figure 7.7: Results of the simulation in the virtual environment.

camera, which in addition to have different intrinsic parameters, it also has the disadvantage of acquiring images with radial distortion. The two cameras can be see in Figure 7.8 and an example of the images acquired by each camera is shown in Figure 7.9.

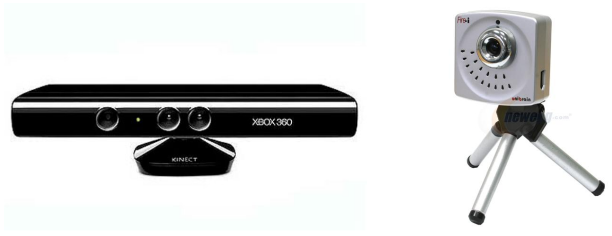


Figure 7.8: Cameras equipped by the robots in the experiments.

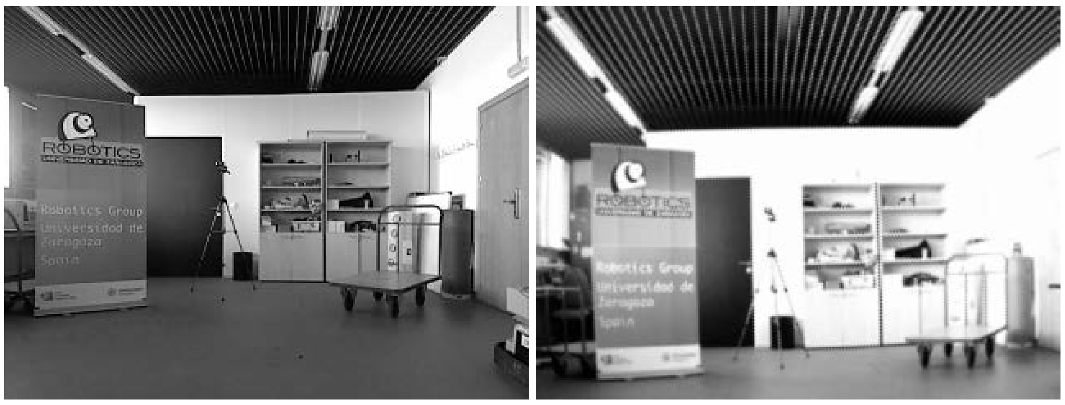


Figure 7.9: Example of images captured by the two cameras used in the experiments.

We have used SIFT features and the 8 point algorithm with RANSAC for matching and computing the epipoles. Since in this environment the loop time is important, we have used images of resolution 320×240 pixels, reducing the time required to extract the SIFT descriptors. In addition, by using smaller images, the number of features was also reduced, which implies less communications between the robots. The drawback of this reduction is that there are some iterations in which the epipoles cannot be computed due to the lack of enough good matches to obtain a robust estimation.

Since there are only three robots, in all the experiments we have considered a fixed communication topology in which every robot can communicate with each other. In order to communicate, the robots have used a real time communication protocol [142]. Let us recall that the proposed method assumes some synchronicity between the robots, in the sense that the images acquired by all of them should be acquired at the same time. In a real scenario this is difficult to achieve. For that reason we have used a barrier scheme in which the robots do not acquire a new image until the previous one has been processed. The scheme followed by the robots at each iteration is described next:

- Acquire an image and extract the SIFT descriptors.
- Communicate to the other robots the descriptors.
- Receive the SIFT extracted from the other two robots.
- Match the received features with the extracted ones and compute the epipoles.
- Compute the control velocity from the epipoles and send an ACK message to the other robots to acquire a new image.
- Wait until the reception of the ACK of the other robots and begin with the whole process again.

Although with this approach there are still some time gaps between some of the frames of the robots, they are negligible in the final results. The total time for each iteration is around 2 seconds, where most of the time is used in communicating the approximately 300 SIFT descriptors extracted per image (around 300KB). We have not dealt with packet drops, as this is handled by the communication protocol. In the following we describe the experiments carried out:

Experiment outdoors

In the first experiment we have taken the robots to the parking lot of the Engineering school of the University of Zaragoza. The robots move with constant linear velocity of $0.1m/s$. The initial and final configuration of the robots can be seen in Figure 7.10. Since there is no common frame to measure the global orientation of the robots, we cannot offer ground truth results about the error in the initial and the final orientation. Nevertheless, in the figure we can see that the robots end up in a configuration with the same orientation (up to an acceptable error). Let us remark that although the robots that are behind can see the third robot in their images, this information is not used by the control law. The robot that does not see other robots in its image is also turning with the epipoles, as can be seen comparing its initial and final orientation in Figure 7.10.

Additionally, the experiment shows that the controller presents some robustness against the planar motion assumption. When the robots left the lane, there is a bump that makes the image to be crooked (see Figure 7.11). The figure also serves as an example of the kind of images acquired by the robots in this experiments and the number of final matches computed at each iteration (around twenty per image).

Experiment indoors

We have also tested the controller in an indoor environment. In this case the robots do not move forward and only turn ($v_i = 0, \forall i$). The results of the experiment are shown in Figure 7.12. Again the robots end up in a configuration with all the orientations aligned.

Experiment using different cameras

Finally, as we have mentioned above, we have also evaluated the distributed control law equipping the robots with different cameras. In the last experiment two robots are still equipped with a Kinect sensor, but the third

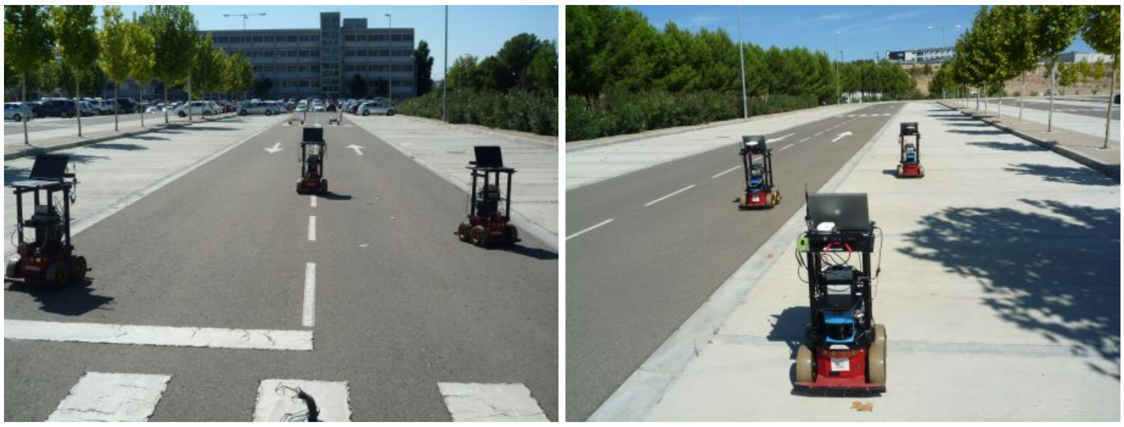


Figure 7.10: Experiment with three robots outdoors. The initial configuration of the robots is shown on the left figure and the final configuration is shown on the right figure.

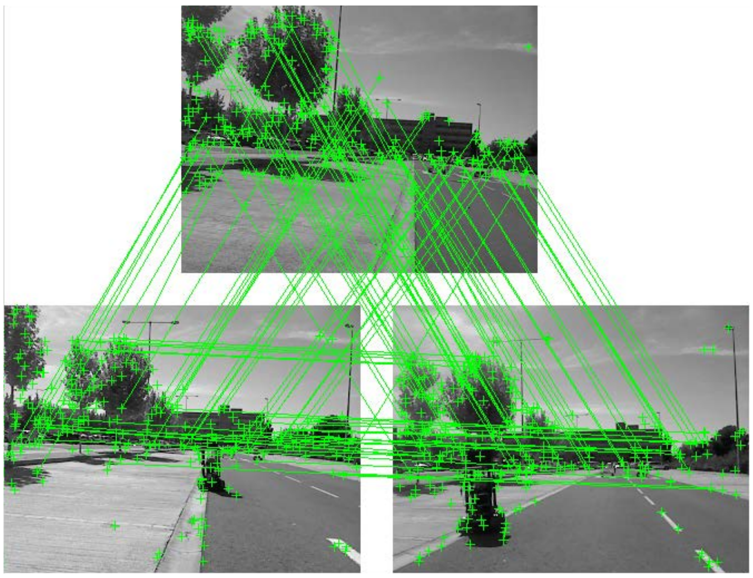


Figure 7.11: Images acquired by the robots in the parking lot with the computed matches. The top image is slightly crooked due to the bump in the road (see the robot in the other two images). The controller seems to be robust to these small inclinations in the images.

robot is equipped with a Unibrain camera. Figure 7.13 shows the initial and final configuration of the robots in this experiment. The unibrain camera has a larger value of α_i than the Kinect cameras, however, the value of β has been set to 500 for the three robots. Looking at the positions of the robots we can see that the relative bearing is neither zero nor $\pi/2$, which means that the final orientation should contain some error, as in eq. (7.8). However, the error in the final configuration is of the same magnitude as in the previous experiments, which means that in practice, the algorithm is more sensitive to the computation of the epipoles and the motion constraints of the robots than to the use of different cameras.

7.6 Discussion

Summing up, we have dealt with the consensus problem of making a team of robots with non-holonomic constraints move with the same direction, also known as the attitude consensus problem or flocking. We have

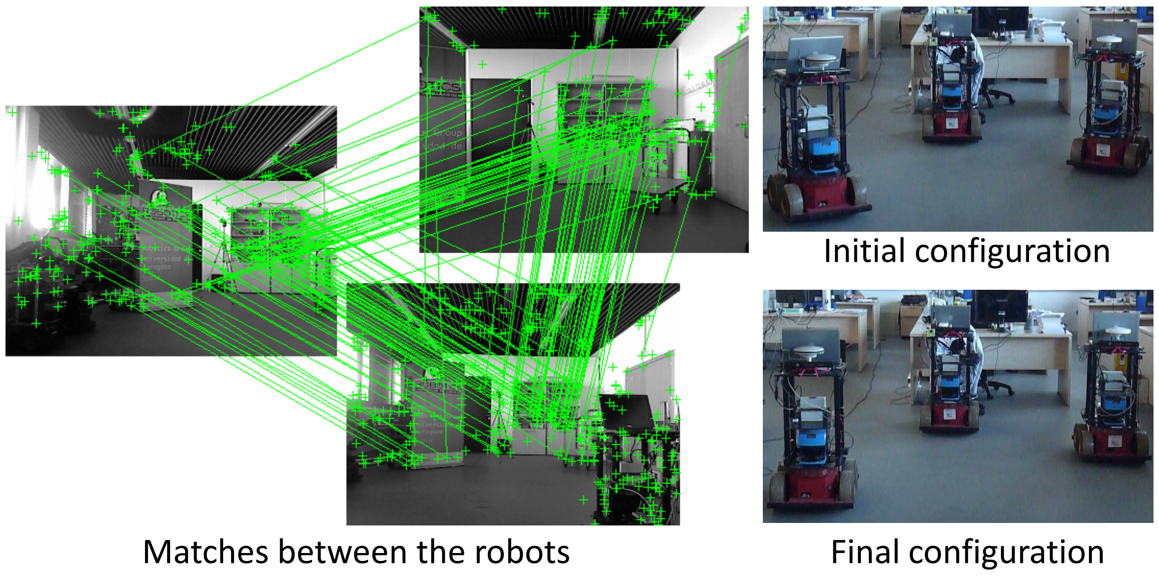


Figure 7.12: Experiment in an indoors environment. In this case the robots have zero linear velocity, $v_i = 0$, and only turn using the proposed controller.

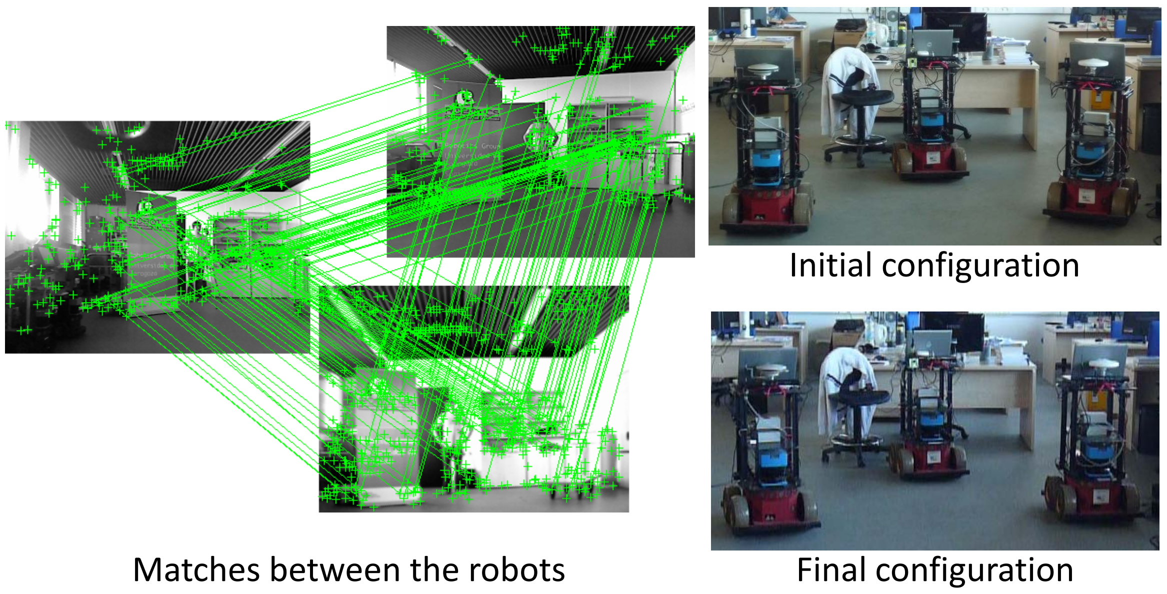


Figure 7.13: In this experiment one of the robots is equipped with a different camera (Unibrain) than the other two (Kinect). On the left figure we show the matches between the three cameras. The right figures show the initial and the final configuration of the robots, in which the three robots are aligned despite the different calibrations.

presented a new vision-based distributed controller to reach this objective that does not require the robots to directly observe each other but common features of the environment. In this way the robots can reach the consensus while focusing on the perception and mapping tasks explored in previous chapters of the Thesis.

Once again, the proposed method uses well known computer vision techniques to achieve the objective in a simple but efficient manner. We make use of the epipoles computed between the images of neighboring robots to estimate the misalignment in their orientations. The use of the epipoles presents several advantages: they do not require explicit computation of the relative motion between the robots or knowledge about the calibration of the cameras and there are well known robust techniques for their computation. When all the robots are equipped with the same camera, the controller is able to reach the exact consensus even if the calibration is poorly estimated. In the more realistic scenario where different robots wear different cameras, we have given the expression of the error in the final configuration.

In order to isolate the use of computer vision from communication and motion issues of the robots, we have tested our algorithm in a simulated virtual environment with ten robots. Using SIFT features and robust algorithms to estimate the epipoles, the robots achieve the consensus with very small errors.

We have also evaluated our proposal using real robots with cameras and limited communications using a real-time protocol to exchange messages. We have seen that the controller works well both in indoor and outdoor environments and the robots are able to reach the consensus, up to an acceptable final error. To see the effect of using different cameras, we have considered a scenario where one robot was equipped with a different camera, with the property of having a high radial distortion. We have shown that in practice the calibration of the cameras is less of an issue than the amount of information the robots require to exchange. This brings up a set of interesting questions that can lead to new lines of research in distributed robotic problems dealing with vision sensors.

Proofs

Proof of Lemma 7.3.1 (Properties of the controller)

First note that $d_{ij} = -d_{ji}$. Therefore, if $|d_{ij}| \leq \pi/2$, then $w_{ij} = -w_{ji}$. In eq. (7.4), when $|d_{ij}| > \pi/2$, $(\pi - |d_{ij}|)$ has the same sign that $(\pi - |d_{ji}|)$ because $|d_{ij}| \leq \pi$. But $\text{sign}(d_{ij}) \neq \text{sign}(d_{ji})$, which implies that $w_{ij} = -w_{ji}$ and 1) is proved.

The proof of 2) is done decomposing the sum of w_i,

$$\sum_{i\in\mathcal{V}} w_i = K \sum_{(i,j)\in\mathcal{E}} w_{ij}.$$

Taking into account that the communication graph is undirected and $w_{ij} = -w_{ji}$, then the sum is equal to zero.

To prove 3) let us consider that both epipoles have the same sign, without loss of generality, positive. The arc tangents have values in the interval $[0, \pi/2)$ and therefore, the difference in eq. (7.5) belongs to the interval $(-\pi/2, \pi/2)$. ∎

Proof of Theorem 7.3.2 (Convergence to a common orientation)

Let $\boldsymbol{\theta}(t) = (\theta_1(t), \ldots, \theta_N(t))$. The proof is done using the following Lyapunov function

$$V(\boldsymbol{\theta}) = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} \frac{1}{2}(\theta_j - \theta_i)^2 = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} \frac{1}{2}\theta_{ij}^2 \geq 0. \tag{7.9}$$

Note that due to the connectivity assumption, (7.9) is positive definite in terms of, for example, $\theta_i - \theta_1,\ i = 2, \ldots, N$. If we compute the derivative of V we obtain

$$\dot{V} = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} (\theta_j - \theta_i)(w_j - w_i). \tag{7.10}$$

We proceed to show that the derivative is negative if $\theta_{ij} \neq 0$. First, by developing (7.10) we obtain

$$\dot{V} = 2N \sum_{i\in\mathcal{V}} \theta_i w_i - \sum_{i\in\mathcal{V}} \theta_i \sum_{j\in\mathcal{V}} w_j - \sum_{i\in\mathcal{V}} w_i \sum_{j\in\mathcal{V}} \theta_j,$$

which by the second point of Lemma 7.3.1 is simplified to

$$\dot{V} = 2N \sum_{i \in \mathcal{V}} \theta_i w_i.$$

Now, regrouping the terms yields

$$\sum_{i \in \mathcal{V}} \theta_i w_i = K \sum_{i \in \mathcal{V}} \theta_i \sum_{j \in \mathcal{N}_i} w_{ij} = \frac{K}{2} \sum_{(i,j) \in \mathcal{E}} (\theta_i - \theta_j) w_{ij}.$$

Therefore, the derivative of V can be expressed as

$$\dot{V} = -KN \sum_{(i,j) \in \mathcal{E}} \theta_{ij} w_{ij}. \tag{7.11}$$

We show now that under the conditions stated in the theorem, the product $\theta_{ij} w_{ij}$ is positive for all i, j. Let us first suppose that $\theta_{ij} > 0$. We divide the analysis in four cases. The first two cases consider positive bearing angles:

- Let ψ_{ij} be positive and satisfying $\psi_{ij} \geq \theta_{ij}$. In this case $e_{ij} > e_{ji} \geq 0$. Since both epipoles have the same sign, using the third point of Lemma 7.3.1, $0 < d_{ij} < \pi/2$ and then $w_{ij} > 0$. Note that this case does not depend on the selection of β, provided that it has the same sign as α.

- If $\theta_{ij} > \psi_{ij} \geq 0$ then $e_{ji} < 0 < e_{ij}$, which implies that $d_{ij} \geq 0$. However, if $d_{ij} > \pi/2$, then $w_{ij} < 0$ and the control may not be stable. In order to have $w_{ij} > 0$ it must hold that $d_{ij} \leq \pi/2$, which is equivalent as to say that $\tan(d_{ij}) > 0$, therefore, using (7.3) and (7.5),

$$\tan(d_{ij}) = \frac{\frac{\alpha}{\beta}(\tan(\psi_{ij}) - \tan(\psi_{ij} - \theta_{ij}))}{1 + \frac{\alpha^2}{\beta^2} \tan(\psi_{ij}) \tan(\psi_{ij} - \theta_{ij})} > 0. \tag{7.12}$$

  The numerator in (7.12) is always positive due to the conditions on θ_{ij} and ψ_{ij}. Then, to satisfy (7.12) it is required that

$$1 + \frac{\alpha^2}{\beta^2} \tan(\psi_{ij}) \tan(\psi_{ij} - \theta_{ij}) > 0,$$

  thus

$$\frac{\alpha}{\beta} < \sqrt{\frac{1}{\tan(\psi_{ij}) \tan(\theta_{ij} - \psi_{ij})}}, \tag{7.13}$$

  which depends on the ratio α/β. A lower bound of the right side of eq. (7.13) is provided later in the proof.

Let us now analyze the cases of negative bearing angles:

- Let us consider first $\psi_{ij} < 0$ and $\theta_{ij} - \psi_{ij} < \pi/2$. When this situation happens $e_{ji} < e_{ij} < 0$ and $w_{ij} > 0$ because of the third point of Lemma 7.3.1. Again, when the robots are in this configuration, the control does not depend on β.

- The last case to analyze appears when $\psi_{ij} < 0$ and $\theta_{ij} - \psi_{ij} > \pi/2$. In this situation the epipoles have different sign, with $e_{ij} < 0 < e_{ji}$, which implies that, in order to have $w_{ij} > 0$, it must happen that $d_{ij} < -\pi/2$. In other words, $\tan(d_{ij}) > 0$. Now, the numerator in (7.12) is always negative, which requires

$$1 + \frac{\alpha^2}{\beta^2} \tan(\psi_{ij}) \tan(\psi_{ij} - \theta_{ij}) < 0,$$

in order to fulfill (7.12), and then

$$\frac{\alpha}{\beta} > \sqrt{\frac{1}{\tan(\psi_{ij})\tan(\theta_{ij} - \psi_{ij})}}. \tag{7.14}$$

Note that (7.13) and (7.14) are not in conflict because they are evaluated in different ranges of ψ_{ij}. An upper bound of the right side of eq. (7.14) is provided later in the proof.

The analysis when $\theta_{ij} < 0$ can be done taking into account the first point of Lemma 7.3.1. Using eq. (7.2), $-\theta_{ij} = \theta_{ji} > 0$, then $w_{ji} > 0$ and $w_{ij} < 0$. The system is in equilibrium when $\theta_{ij} = 0, \forall (i,j) \in \mathcal{E}$, but due to the fact that the communication graph is connected, then the set of equilibrium points is $\theta_{ij} = 0, \forall i,j \in \mathcal{V}$.

We compute now the bounds that satisfy (7.13) and (7.14). Let

$$\gamma(\theta_{ij}, \psi_{ij}) = \sqrt{\frac{1}{\tan(\psi_{ij})\tan(\theta_{ij} - \psi_{ij})}}. \tag{7.15}$$

We analyze (7.15) in the intervals $\mathcal{I}_1$ and $\mathcal{I}_2$

$$\mathcal{I}_1 = \{(\theta_{ij}, \psi_{ij}) \mid 0 < \psi_{ij} < \theta_{ij} < \theta_M\},$$
$$\mathcal{I}_2 = \{(\theta_{ij}, \psi_{ij}) \mid \psi_{ij} < 0 < \theta_{ij} < \theta_M, \psi_{ij} - \theta_{ij} < -\frac{\pi}{2}\},$$

The partial derivative of (7.15) with respect to θ_{ij} is equal to

$$\frac{\partial \gamma}{\partial \theta_{ij}} = \frac{-1}{2\gamma \tan(\psi_{ij})\tan^2(\theta_{ij} - \psi_{ij})\cos^2(\psi_{ij} - \theta_{ij})}. \tag{7.16}$$

We can see that the sign of (7.16) depends only on the sign of $\tan(\psi_{ij})$, which is positive on $\mathcal{I}_1$ and negative on $\mathcal{I}_2$. Therefore, the function is decreasing with θ_{ij} on $\mathcal{I}_1$ and increasing on $\mathcal{I}_2$ and in both cases the bound we are looking for will be achieved in $\theta_{ij} = \theta_M$.

If we compute the derivative of (7.15) with respect to ψ_{ij}, already considering $\theta_{ij} = \theta_M$ we obtain

$$\frac{\partial \gamma}{\partial \psi_{ij}} = \frac{\sin(\theta_M - \psi_{ij})\cos(\theta_M - \psi_{ij}) - \sin(\psi_{ij})\cos(\psi_{ij})}{2\gamma \sin^2(\theta_M - \psi_{ij})\sin^2(\psi_{ij})}. \tag{7.17}$$

The only minimum of (7.17) on the interval $\mathcal{I}_1$ is in $\psi_{ij} = \theta_M/2$. The maximum on $\mathcal{I}_2$ is found on the value $\psi_{ij} = -\pi/2 + \theta_M/2$. Using trigonometry equivalences we obtain that

$$\gamma(\theta_M, -\pi/2 + \theta_M/2) = \frac{1}{\gamma(\theta_M, \theta_M/2)}. \tag{7.18}$$

Finally, by noting that $\gamma(\theta_M, \theta_M/2) = 1/\tan(\theta_M/2)$, the condition in (7.7) is obtained.

The last point to check is the invariance of the set $|\theta_{ij}| \leq \theta_M, \ \forall i,j$. To show this, let us consider a fixed reference frame $\mathcal{F}$, and let $\theta_{\max}$ and $\theta_{\min}$ be the maximum and minimum orientation values in such frame. This means that, initially, $\max \theta_{ij} = \theta_{\max} - \theta_{\min} \leq \theta_M$ and $\theta_i \in [\theta_{\min}, \theta_{\max}]$, for all i. Now, let us note that $\theta_{\max i} = \theta_i - \theta_{\max} \leq 0$, and $\theta_{\min i} = \theta_i - \theta_{\min} \geq 0$ for all i and all t. Therefore, $w_{\max} \leq 0$ and $w_{\min} \geq 0$. Since the orientations are in a manifold, it is possible that $w_{\max} \leq -\pi$ and even when it has negative sign the difference $\theta_{\max} - \theta_{\min}$ is increased. By choosing K sufficiently small, e.g., such that $w_{\max} > -\theta_M$ and $w_{\min} < \theta_M$, we can guarantee that the extremes of the set are always pushed to the interior, proving its invariance.

We do not consider in the proof the special cases $\psi_{ij} = 0$, $\psi_{ij} = \pi/2$, $\psi_{ij} = \theta_{ij}$ and $\psi_{ij} - \theta_{ij} = \pm\pi/2$ to compute the bounds (7.13) and (7.14). However, it can be shown that the controller (7.6) is always well defined in these situations independently of β. ∎

Proof of Proposition 7.4.1 (Convergence to a common orientation with topology changes)

We use again the Lyapunov function defined in (7.9). The new derivative of V is

$$\dot{V} = -KN \sum_{(i,j)\in\mathcal{E}(t)} \theta_{ij} w_{ij} \leq 0, \tag{7.19}$$

and the function is a weak common Lyapunov function for all $\mathcal{E}(t)$, and therefore, for any network topology. Combining this with Assumption 2.2.5, we can assert that the system is stable [10].

Denote $\boldsymbol{\theta}^*$ as the set of points with all the orientations equal, i.e., $\boldsymbol{\theta}^* = \{\boldsymbol{\theta} \mid \theta_{ij} = 0, \forall i, j \in \mathcal{V}\}$. Let $\mathcal{G}_N$ be the set of graphs composed by N nodes

$$\sum_{\mathcal{G}\in\mathcal{G}_N} \dot{V}_{\mathcal{G}}(\boldsymbol{\theta}) < 0, \tag{7.20}$$

for all $\boldsymbol{\theta} \notin \boldsymbol{\theta}^*$. Therefore V is a common joint Lyapunov function of the system [25, 150], and $\boldsymbol{\theta}^*$ are the only equilibrium points that all the graphs have in common. The ergodicity requirement on the switching signal is found in Assumption 2.2.6. Therefore, using [25], we conclude that the robots will converge to some $\boldsymbol{\theta} \in \boldsymbol{\theta}^*$ and the consensus will be reached.

■

Proof of Proposition 7.4.2 (Error with different cameras)

The consensus is achieved when $e_{ij} = e_{ji}$. This implies

$$\arctan\left(\frac{\alpha_i}{\beta}\tan\psi_{ij}\right) = \arctan\left(\frac{\alpha_j}{\beta}\tan(\psi_{ij} - \theta_{ij})\right). \tag{7.21}$$

Developing the equality yields

$$\alpha_i \frac{\sin\psi_{ij}}{\cos\psi_{ij}} = \alpha_j \frac{\sin\psi_{ij}\cos\theta_{ij} - \cos\psi_{ij}\sin\theta_{ij}}{\cos\psi_{ij}\cos\theta_{ij} - \sin\psi_{ij}\sin\theta_{ij}}.$$

Rearranging the terms we obtain

$$\begin{aligned}&(\alpha_i - \alpha_j)(\sin\psi_{ij}\cos\psi_{ij})\cos\theta_{ij} = \\ &(\alpha_i \sin^2\psi_{ij} - \alpha_j\cos^2\psi_{ij})\sin\theta_{ij}.\end{aligned}$$

Thus

$$\frac{\sin\theta_{ij}}{\cos\theta_{ij}} = \frac{(\alpha_i - \alpha_j)\sin\psi_{ij}\cos\psi_{ij}}{(\alpha_i\sin^2\psi_{ij} + \alpha_j\cos^2\psi_{ij})}.$$

and taking the arc tangent (7.8) is obtained. ■

Chapter 8

Conclusions

"It's all said and done, it's real, and it's been fun." In this Thesis, we have contributed in different topics to develop a set of distributed algorithms that allow a team of robots equipped with monocular cameras achieve a consensus in different perception tasks. We have placed a great effort in two issues related with this problem, the identification and solution of the additional complications that appear because of the use of cameras and the proper representation of the visual information to simplify the achievement of the consensus. Contributions and conclusions obtained throughout this work are finally summarized in this chapter.

8.1 Conclusions

In this work we have studied the problem of achieving consensus in a decentralized way by a team of robots with limited communications and vision sensors. After a deep study of the different problems that appear in this scenario, we can extract some interesting conclusions.

In a first step, we have proposed a variety of algorithms following a linear iteration scheme, taking into account the problems derived from the use of visual information. In particular, the following contributions and conclusions were obtained.

- Firstly, we successfully have addressed the data association problem in a distributed scenario. We have proposed a distributed algorithm that makes possible for a team of robots with multiple observations to distinguish common features of the environment. The algorithm starts using the local correspondences found between direct neighbors in the communication graph and then executes two steps to find the global matches. The first step consists on propagating the local matches all over the robotic network so that every robot is aware of farther away correspondences. The second step deals with the problem of breaking the inconsistencies (different features associated by the same robot) that appear due to spurious local correspondences. After the execution of these two steps, the team of robots has the knowledge of the real number of features observed and which observations should be mixed in the consensus process. Additionally, an extensive evaluation of the proposed algorithms has shown that they can be applied with a wide variety of features and local matchers.

- Secondly, we have analyzed in detail the problem of robustness in the computation of the consensus. We have exposed the lack of robustness to outlier measurements of existing consensus methods and we have proposed a new algorithm, *De-RANSAC*, able to detect and discard outlier measurements during the computation of the average consensus. In this way, the robots are aware if they have good or bad information and can compute the real consensus value discarding the erroneous information. Following the principles of RANSAC, the proposed method generates a set of hypotheses and votes for them using distributed averaging. As the hypotheses take form, the robots are allowed to dynamically change their opinion, achieving the optimal solution in just one consensus step. For the development of the algorithm we have used homogeneous coordinates, in order to compute the average of different subsets of robots and we have presented a distributed averaging primitive to compute the number of active robots in a network.

- Thirdly, to compensate the extra communications that visual information requires, we have dealt with the speed of convergence of the consensus linear iteration. We have seen that using Chebyshev polynomials the number of iterations required to reach the consensus is considerably smaller than using existing methods. The proposed algorithm has the same requirements of a standard linear iteration, which makes it very appealing. We have characterized the main properties of the method such as the convergence speed and the parameters that optimize it. Since the knowledge of the optimal parameters requires global information of the communication topology, a distributed solution to estimate these parameters using a bisection technique has been proposed. The result has been an adaptive distributed linear iteration able to reach the maximum convergence speed and achieve the desired consensus in a small number of iterations. An empirical validation of the theoretical results with extensive simulations has shown the benefits of the proposed methods.

Afterwards, we have focused our attention on specific consensus problems and we have provided solutions to these problems exploiting well studied computer vision techniques. The use of the adequate visual information has been proved to be essential to achieve the consensus in different perception and control tasks. Specifically:

- We have studied the problem of distributed map building by a team of robots using planes as the features to represent the map. We have seen that well known computer vision techniques play a fundamental role in achieving a consensus about the visual information perceived by all the robots. The advantages of using this representation for individual robots and for the team as a unit have been highlighted. The use of homographies simplifies the data association problem, providing with a robust mechanism to detect planes in different maps. The common reference frame only requires a max-consensus and the multiplication of different homographies. The final consensus is achieved by all the robots, even if most of them have not seen a specific plane because of the homogeneous coordinate contained in the representation. In the end, all the robots manage the same global map. We have shown the performance of the method with a large data set of real images recorded in Zaragoza downtown.

- Finally, the last contribution of this work was the development of a distributed controller to make a team of robots with monocular cameras move in the consensus direction while focusing on the above mentioned exploration tasks. With the proposed controller, the robots estimate the misalignment using common features of the environment. To do so we have employed another well known computer vision property between pairs of images, the epipolar constraint. Besides the ability to explore while coordinating, a very appealing consequence of the usage of epipoles is that the controller does not require a precise knowledge of the calibration of the cameras. We have evaluated our proposal using real robots equipped with different cameras, showing that the desired configuration is achieved in different environments.

8.2 Future Work

This Thesis has shown the potential benefits of using vision sensors by multi-robot systems in perception and control tasks. However, we believe that there is still plenty of room for improvement and there is a long way to walk before these systems can actually impact in our lives as is happening nowadays with individual robots.

One of the first issues that we believe should be studied is the robust dynamic association of the features. The data association algorithm presented in the Thesis needs to be executed every time new observations are sensed. At the same time the robust consensus we have proposed requires for each robot to handle just one datum. We have the feeling that the dynamic voting version of *De-RANSAC* might be usable to allow the team of robots not only to achieve the good consensus but also to associate their observations. Instead of assuming one datum per robot, at each iteration and for each hypothesis, the robots could test their observations and plug (or unplug) into the process the suited ones. The benefits of making this possible would be of great interest. On the other hand, considering multiple data in the hypotheses increases the complexity of the problem, which will require further analysis of the properties of the algorithm.

The size of the messages is also a huge problem in the studied systems. We have been able to reduce the number of iterations to reach the consensus modifying the linear iteration. However, the size of each message has not been reduced in the Thesis, but increased in order to provide with robustness to the whole approach. In this sense, further research is required so that these systems can be used in real time with appropriate loop times. Streaming algorithms can be the solution to this problem because they are in a mature phase of development, allowing people to watch movies and tv programs in real time. Unfortunately, we are not sure that in the consensus process the variation between different iterations is enough to reduce the size of the messages using streaming but the potential benefit is worth a shot.

To finalize, we have proposed a distributed controller to coordinate the robots while exploring the environment. The development of distributed controllers that allow a team of robots to improve their perception of the environment has caught the attention of many researchers during the last decade. The use of vision sensors in this context is just starting to pop up and there is still plenty of work to do in this research line.

Bibliography

[1] K. Achour and M. Benkhelif. A new approach to 3d reconstruction without camera calibration. *Pattern Recognition*, 34(12):2467–2476, December 2001.

[2] R. P. Agarwal. *Difference equations and inequalities, Theory, Methods and Applications*. Dekker, 1992.

[3] A. Angeli, D. Filliat, S. Doncieux, and J.A. Meyer. Fast and incremental method for loop-closure detection using bag of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, October 2008.

[4] D. Angeli and P-A Bliman. Convergence Speed of Unsteady Distributed Consensus: Decay Estimate Along the Settling Spanning-Trees. *SIAM Journal on Control and Optimization*, 48(1):1–32, January 2009.

[5] R. Aragues, J. Cortes, and C. Sagues. Dynamic consensus for merging visual maps under limited communications. In *IEEE International Conference on Robotics and Automation*, pages 3032–3037, 2010.

[6] R. Aragues, J. Cortes, and C. Sagues. Distributed Consensus Algorithms for Merging Feature-Based Maps with Limited Communication. *Robotics and Autonomous Systems*, 59(3):163–180, March 2011.

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

[8] S. Avidan, Y. Moses, and Y. Moses. Centralized and distributed multi-view correspondence. *International Journal of Computer Vision*, 71(1):49–69, June 2007.

[9] T.C. Aysal, B. Oreshkin, and M.J. Coates. Accelerated distributed average consensus via localized node state prediction. *IEEE Transactions on Signal Processing*, 57(4):1563–1576, April 2009.

[10] A. Bacciotti and L. Mazzi. An invariace principle for nonlinear switched systems. *System and Control Letters*, 54(11):1109–1119, November 2005.

[11] M. Basiri, A. N. Bishop, and P. Jensfelt. Distributed control of triangular formations with angle-only constraints. *Systems and Control Letters*, 59(2):147–154, February 2010.

[12] R. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 33(2):117–137, September 1999.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[14] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):540–547, May 1996.

[15] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[16] Ian Britton. http://www.freefoto.com.

[17] A. Broder and E. Shamir. On the second eigenvalue of random regular graphs (preliminary version). In *28th Annual Symposium on Foundations of Computer Science*, pages 286–294, October 1987.

[18] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[19] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach for in-place sensor calibration. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, volume 2634 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 2003.

[20] F. Caballero, L. Merino, J. Ferruz, and A. Ollero. Homography based kalman filter for mosaic building. applications to uav position estimation. In *IEEE International Conference on Robotics and Automation*, pages 2004–2009, April 2007.

[21] M. Cao, A. S. Morse, and B. D. O. Anderson. Reaching a Consensus in a Dynamically Changing Environment A Graphical Approach: Convergence Rates, Measurement Delay and Asynchronous Events. *SIAM Journal on Control and Optimization*, 47(2):601–623, February 2008.

[22] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. *International Journal of Computer Vision*, 68(1):53–64, January 2006.

[23] F. Castanedo, M.A. Patricio, J. Garcia, and J.M. Molina. Robust data fusion in a visual sensor multi-agent architecture. In *10th International Conference on Information Fusion*, pages 1–7, 2007.

[24] J. Chen, W. E. Dixon, M. Dawson, and M. McIntyre. Homography-based visual servo tracking control of a wheeled mobile robot. *IEEE Transactions on Robotics*, 22(2):406–415, March 2006.

[25] D. Cheng, J. Wang, and X. Hu. An Extension of LaSalle's Invariance Principle and Its Application to Multi-Agent Consensus. *IEEE Transactions on Automatic Control*, 53(7):1765–1770, July 2008.

[26] O. Chum and J. Matas. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1–11, August 2008.

[27] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 55(5):372–382, October 2008.

[28] J. Civera, O. G. Grasa, A. J Davison, and J. M. M. Montiel. 1-Point RANSAC for EKF Filtering: Application to Real-Time Structure from Motion and Visual Odometry. *Journal of Field Robotics*, 27(5):609–631, October 2010.

[29] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564 – 577, May 2003.

[30] J. Cortés. Finite-time convergent gradient flows with applications to network consensus. *Automatica*, 42(11):1993–2000, November 2006.

[31] J. Cortes. Global and robust formation-shape stabilization of relative sensing networks. *Automatica*, 45(12):2754 – 2762, December 2009.

[32] J. Cortés, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in d dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, September 2006.

[33] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, March 2004.

[34] J. Courbon, Y. Mezouar, and P. Martinet. Indoor navigation of a non-holonomic mobile robot using a visual memory. *Autonomous Robots*, 25(3):253–266, October 2008.

[35] A. Das, R. Fierro, V. Kumar, J. Ostrowski, J. Spletzer, and C. J. Taylor. Vision based formation control of multiple robots. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, October 2002.

[36] R. W. Deming and L. I. Perlovsky. Concurrent multi-target localization, data association, and navigation for a swarm of flying sensors. *Information Fusion*, 8(3):316 – 330, March 2007.

[37] D. V. Dimarogonas and K. J. Kyriakopoulos. On the rendezvous problem for multiple nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(5):916–922, May 2007.

[38] L. Ding and A. M. Martine. Features versus context: An approach for precise and detailed detection and delineation of faces and facial features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):2022 – 2038, November 2010.

[39] E. Eade and T. Drummond. Scalable monocular slam. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 469–476, 2006.

[40] Y. Fang, M. Dawson, W. E. Dixon, and M. S. Queiroz. Homography-based visual servoing of wheeled mobile robots. In *IEEE International Conference on Decision and Control*, pages 2866–2871, December 2002.

[41] V. Ferrari, T. Tuytelaars, and Luc Val Gool. Wide-baseline multiple-view correspondences. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 718–725, 2003.

[42] G. Ferrari-Trecate, A. Buffa, and M. Gati. Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Transactions on Automatic Control*, 51(6):1058–1063, June 2006.

[43] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[44] N. Floudas, A. Polychronopoulos, O. Aycard, J. Burlet, and M. Ahrholdt. High level sensor data fusion approaches for object recognition in road environment. In *IEEE Intelligent Vehicles Symposium*, pages 136–141, 2007.

[45] M. Franceschelli, M Egersdedt, and A. Giua. Motion probes for fault detection and recovery in networked control systems. In *American Control Conference*, pages 4358–4363, June 2008.

[46] E. Franco, R. Olfati-Saber, T. Parisini, and M. M. Polycarpou. Distributed fault diagnosis using sensor networks and consensus-based filters. In *IEEE International Conference on Decision and Control*, pages 386–391, December 2006.

[47] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, 2007.

[48] F. Fraundorfer, K. Schindler, and H. Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and vision computing*, 24(4):395–406, June 2006.

[49] U. Frese and J. Kurlbaum. A data set for data association, June 2008.

[50] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362 – 1376, September 2009.

[51] A. Ganguli, J. Cortes, and F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, 25(2):340–352, March 2009.

[52] R. Garg, D. Ramanan, S. Seitz, and N. Snavely. Where's waldo: Matching people in images of crowds. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1793–1800, 2011.

[53] G. Gate, A. Breheret, and F. Nashashibi. Centralized fusion for fast people detection in dense environment. In *IEEE International Conference on Robotics and Automation*, pages 76–81, 2009.

[54] B.P. Gerkey, S. Thrun, and Geoff Gordon. Visibility-based Pursuit-evasion with Limited Field of View. *The International Journal of Robotics Research*, 25(4):299–315, April 2006.

[55] A. Gil, O. Reinoso, M. Ballesta, and M. Juliá. Multi-robot visual slam using a rao-blackwellized particle filter. *Robotics and Autonomous Systems*, 58(1):68–80, January 2009.

[56] T. Gustavi and X. Hu. Observer-based leader-following formation control using onboard sensor information. *IEEE Transactions on Robotics*, 24(6):1457–1462, December 2008.

[57] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2000.

[58] E. Hazan, S. Safra, and O. Schwartz. On the hardness of approximating k-dimensional matching. In *Electronic Colloquium on Computational Complexity*, 2003.

[59] Z.G. Hou, L. Cheng, and M. Tan. Decentralized robust adaptive control for the multiagent system consensus problem using neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-part B*, 39(3):636–647, June 2009.

[60] T. Ibuki, T. Hatanaka, M. Fujita, and M. W. Spong. Visual feedback attitude synchronization in leader-follower type visibility structures. In *49th IEEE Conference on Decision and Control*, pages 2486–2491, December 2010.

[61] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[62] F. Jiang and L. Wang. Finite-time information consensus for multi-agent systems with fixed and switching topologies. *Physica D*, 238(16):1550–1560, August 2009.

[63] Z. Jin and R.M. Murray. Multi-hop relay protocols for fast consensus seeking. In *IEEE International Conference on Decision and Control*, pages 1001–1006, December 2006.

[64] B. Johansson and M. Johansson. Faster linear iterations for distributed averaging. In 17^{th} *IFAC World Congress*, July 2008.

[65] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210, December 2009.

[66] Y. Kima, D.W. Gub, and I. Postlethwaite. Spectral radius minimization for optimal average consensus and output feedback stabilization. *Automatica*, 45(6):1379–1386, June 2009.

[67] C.K. Ko and X. Gao. On matrix factorization and finite-time average-consensus. In *48th IEEE International Conference on Decision and Control*, pages 5798–5803, December 2009.

[68] E. Kokiopoulou and P. Frossard. Polynomial filtering for fast convergence in distributed consensus. *IEEE Transactions on Signal Processing*, 57(1):342–354, January 2009.

[69] E. Kokiopoulou and P. Frossard. Distributed classification of multiple observation sets by consensus. *IEEE Transactions on Signal Processing*, 59(1):104–114, January 2011.

[70] K. Konolige, J. Bowman, JD. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. In *Robotics: Science and Systems*, 2009.

[71] B. Kuipers. Modeling spacial knowledge. In *International Joint Conference on Artificial Intelligence*, pages 292–298, 1978.

[72] J. Li, E. Elhamifar, I-J Wang, and R. Vidal. Consensus with robustness to outliers via distributed optimization. In *IEEE Conference on Decision and Control*, pages 2111–2117, 2010.

[73] T. Li and J-F. Zhang. Consensus Conditions of Multi-Agent Systems With Time-Varying Topologies and Stochastic Communication Noises. *IEEE Transactions on Automatic Control*, 55(9):2043–2057, September 2010.

[74] G. Lopez-Nicolas, N.R. Gans, S. Bhattacharya, C. Sagues, J. J. Guerrero, and S. Hutchinson. Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints. *IEEE Trans. on Systems, Man, and Cybernetics: Part B Cybernetics*, 40(4):1115 – 1127, April 2010.

[75] G. López-Nicolás, C. Sagues, J.J. Guerrero, D. Kragic, and P. Jensfelt. Switching visual control based on epipoles for mobile robots. *Robotics and Autonomous Systems*, 56(7):592–603, July 2008.

[76] E. Lovisari, F. Garin, and S. Zampieri. A resistance-based approach to performance analysis of the consensus algorithm. In *IEEE International Conference on Decision and Control*, pages 5714–5719, December 2010.

[77] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[78] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann publishers, 1997.

[79] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3D Vision*. SpringerVerlag, 2004.

[80] G.L. Mariottini, G. Oriolo, and D. Prattichizzo. Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. *IEEE Transactions on Robotics*, 23(1):87–100, February 2007.

[81] J. Marsden and A. Weinstein. *Calculus, Volume III, Chapter 16*. Applied Mathematics Series. Springer, 1985.

[82] S. Martínez, J. Cortés, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, August 2007.

[83] J.C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall, 2002.

[84] M. Mehyar, D. Spanos, J. Pogsajapan, S. H. Low, and R. M. Murray. Asynchronous Distributed Averaging on Communication Networks. *IEEE Transactions on Networking*, 15(3):512–520, June 2007.

[85] J. F. Menudet, J. M. Becker, T. Fournel, and C. Mennessier. Plane-based camera self-calibration by metric rectification of images. *Image and Vision Computing*, 26(7):913–934, July 2007.

[86] E. Montijano, R. Aragues, and C. Sagues. Distributed Data Association in Robotic Networks with Cameras and Limited Communications. *IEEE Transactions on Robotics*, 2012. Submitted.

[87] E. Montijano, S. Martínez, and C. Sagues. *De-RANSAC*: Robust consensus for robot formations. In *Network Science and Systems in Multi-Robot Autonomy, Workshop at the IEEE International Conference on Robotics and Automation 2010*, May 2010.

[88] E. Montijano, S. Martínez, and C. Sagues. Distributed robust data fusion based on dynamic voting. In *IEEE International Conference on Robotics and Automation*, pages 5893–5898, May 2011.

[89] E. Montijano, S. Martínez, and C. Sagues. *De-RANSAC*: Robust consensus for sensor networks. *European Journal of Control*, 2012. Submitted.

[90] E. Montijano, J. I. Montijano, and C. Sagues. Adaptive consensus and algebraic connectivity estimation in sensor networks with chebyshev polynomials. In *50th IEEE Conference on Decision and Control and European Control Conference 2011*, pages 4296–4301, December 2011.

[91] E. Montijano, J. I. Montijano, and C. Sagues. Fast distributed consensus with chebyshev polynomials. In *American Control Conference*, pages 5450–5455, June 2011.

[92] E. Montijano, J. I. Montijano, and C. Sagues. Fast distributed consensus with chebyshev polynomials. *IEEE Transactions on Signal Processing*, 2011. submitted.

[93] E. Montijano and C. Sagues. Position-based navigation using multiple homographies. In *IEEE International Conference on Emergent Technologies and Factory Automation*, pages 994–1001, September 2008.

[94] E. Montijano and C. Sagues. Fast pose estimation for visual navigation using homographies. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2704–2709, October 2009.

[95] E. Montijano and C. Sagues. Topological maps based on graphs of planar regions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1661–1666, October 2009.

[96] E. Montijano and C. Sagues. Distributed multi-camera visual mapping using topological maps of planar regions. *Pattern Recognition*, 44(7):1528–1539, July 2011.

[97] E. Montijano, J. Thunberg, X. Hu, and C. Sagues. Multi-Robot Distributed Visual Coordination using Epipoles. In *IEEE Conference on Decision and Control*, pages 2750–2655, 2011.

[98] N. Mostagh and A. Jadbabaie. Distributed geodesic control laws for flocking of nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(4):681–686, July 2007.

[99] N. Mostagh, N. Michael, A. Jadbabaie, and K. Daniilidis. Vision-based, distributed control laws for motion coordination of nonholonomic robots. *IEEE Transactions on Robotics*, 25(4):851–860, July 2009.

[100] A. C. Murillo, C. Sagues, J. J. Guerrero, T. Goedemé, T. Tuytelaars, and L. Van Gool. From omnidirectional images to hierarchical localization. *Robotics and Autonomous Systems*, 55(5):372–382, October 2007.

[101] C. Naik, H.H. Cartensen, and A. M. Dean. Reaction rate representation using chebyshev polynomials. In *Western States Section 2002 Spring Meeting of the Combustion Institute*, pages 5714–5719, March 2002.

[102] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, December 2001.

[103] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[104] D. Nister. Preemtive ransac for live structure and motion estimation. *Machine Vision and Application*, 16(5):321–329, December 2005.

[105] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, March 2006.

[106] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.

[107] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[108] R. Oliveira, J. Xavier, and J.P. Costeira. Multi-view correspondence by enforcement of rigidity constraints. *Image and Vision Computing*, 25(6):1008–1020, June 2007.

[109] A. Olshevsky and J. N. Tsitsiklis. Convergence Speed in Distributed Consensus and Averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, June 2009.

[110] A. Olshevsky and J. N. Tsitsiklis. A Lower Bound for Distributed Averaging Algorithms on the Line Graph. *IEEE Transactions on Automatic Control*, 56(11):2694–2698, November 2011.

[111] B. Oreshkin, M. J Coates, , and M. Rabbat. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, May 2010.

[112] W. Ouyang, F. Tombari, S. Mattoccia, L. D. Stefano, and W.K. Cham. Performance evaluation of full search equivalent pattern matching algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. d.o.i. 10.1109/TPAMI.2011.106.

[113] S. Ozera, C. H. Chenb, and H. A. Cirpanc. A set of new chebyshev kernel functions for support vector machine pattern classification. *Pattern Recognition*, 44(7):1435–1447, July 2011.

[114] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[115] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter 6.1 The Max-Flow, Min-Cut Theorem, pages 120–128. Dover, 1998.

[116] F. Pasqualetti, A. Bicchi, and F. Bullo. On the security of linear consensus networks. In *IEEE International Conference on Decision and Control*, pages 4894–4901, Shanghai, China, December 2009.

[117] S. Patterson, B. Bamieh, and A. E. Abbadi. Convergence rates of distributed average consensus with stochastic link failures. *IEEE Transactions on Automatic Control*, 55(4):880–892, April 2010.

[118] S. Patterson, B. Bamieh, and A. El Abbadi. Distributed average consensus with stochastic communication failures. In *46th IEEE Conference on Decision and Control*, pages 4215–4220, December 2007.

[119] Official U.S. Air Force's photostream. http://www.flickr.com/photos/usairforce/.

[120] J. Qin and N.H.C.Yung. Scene categorization via contextual visual words". *Pattern Recognition*, 43(5):1874–1888, May 2010.

[121] R. J. Radke. A survey of distributed computer vision algorithms. In *Handbook of Ambient Intelligence and Smart Environments*, pages 35–55. Springer US, 2009.

[122] P. Remagnino, A.I. Shihab, and G.A. Jones. Distributed intelligence for multi-camera visual surveillance. *Pattern Recognition*, 37(4):675–689, April 2004.

[123] A. Remazeilles and F. Chaumette. Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4):345–356, April 2007.

[124] W. Ren. Distributed cooperative attitude synchronization and tracking for multiple rigid bodies. *IEEE Transactions on Control Systems Technology*, 18(2):383–392, March 2010.

[125] W. Ren and R. W. Beard. *Distributed Consensus in Multi-vehicle Cooperative Control*. Communications and Control Engineering. Springer-Verlag, London, 2008.

[126] W. Ren, R. W. Beard, and E. M Atkins. Information consensus in multivehicle cooperative control: Collective group behavior through local interaction. *IEEE Control Systems Magazine*, 27(2):71–82, April 2007.

[127] D. L. Richardson, D. Schmidt, and J. Mitchell. Improved chebyshev methods for the numerical integration of first-order differential equations. In *Spaceflight mechanics 1998; Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, pages 1533–1544, February 1998.

[128] E. Royer, M. Lhuillier, M. Dhome, and J.M. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *Internation Journal on Computer Vision*, 74(3):237–260, 2007.

[129] B. Ghosh S. Muthukrishnan and M. H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31(4):331–354, December 1998.

[130] C. Sagues, A.C. Murillo, F. Escudero, and J.J. Guerrero. From lines to epipoles through planes in two views. *Pattern Recognition*, 39(3):384–393, March 2006.

[131] M. Schwager, P. Dames, D. Rus, and V. Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *Proceedings of the International Symposium on Robotics Research*, August 2011.

[132] M. Schwager, D. Rus, and J.J. Slotine. Decentralized, Adaptive Coverage Control for Networked Robots. *The International Journal of Robotics Research*, 28(3):357–375, March 2009.

[133] K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, January 2005.

[134] Y. A. Sheikh and M. Shah. Trajectory association across multiple airborne cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):361–367, February 2008.

[135] B. Siciliano and O. Khatib (Editors). *Handbook of Robotics*. Springer, 2008.

[136] G. Silveira, E. Malis, and P. Rives. Real-time robust detection of planar regions in a pair of images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 49–54, 2006.

[137] J. Sola, A. Monin, M. Devy, and T. Vidal-Calleja. Fusing monocular information in multicamera slam. *IEEE Transactions on Robotics*, 24(5):958–968, October 2008.

[138] S. Sundaram and C. N. Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *American Control Conference*, pages 711–716, New York, July 2007.

[139] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents - part i: Attacking the network. In *American Control Conference*, pages 1350–1356, June 2008.

[140] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents - part ii: Overcoming malicious behavior. In *American Control Conference*, pages 1357–1362, June 2008.

[141] R. Szeliski and P. H. S. Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 171–186, 1998.

[142] D. Tardioli, A.R. Mosteo, L. Riazuelo, J.L. Villarroel, and L. Montano. Enforcing Network Connectivity in Robot Team Missions. *The International Journal of Robotics Research*, 29(4):460–480, April 2010.

[143] S. A. Teulosky, W. T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The art of Scientific Computing*. Cambridge University Press, 2002.

[144] J. Thunberg, E. Montijano, and X. Hu. Distributed attitude synchronization control. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 1962–1967, December 2011.

[145] R. Tron and R. Vidal. Distributed face recognition via consensus on se(3). In *Workshop on Omnidirectional Vision*, 2008.

[146] R. Tron and R. Vidal. Distributed pose averaging in camera sensor networks via consensus on se(3). In *International Conference on Distributed Smart Cameras*, 2008.

[147] P. Urcola and L. Montano. Cooperative robot team navigation strategies based on an environment model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4577–4583, St. Louis, USA, October 2009.

[148] J.G. Verwer and B.P. Sommeijer. An implicitŰ-explicit runge–kutta–chebyshev scheme for diffusion–reaction equations. *SIAM Journal on Scientific Computing*, 25(5):1824–1835, May 2004.

[149] R. Vidal, O. Shakernia, and S. Sastry. Following the flock. *IEEE Robotics and Automation Magazine*, 11(4):14–20, December 2004.

[150] J. Wang and D. Cheng. Extensions of LaSalle's Invariance Principle for Switched Nonlinear Systems. In *Proceedings of the 17th World Congress The International Federation of Automatic Control*, pages 14397–14402, Seoul, Korea, July 2008.

[151] L. Wang and F. Xiao. Finite-time consensus problems for networks of dynamic agents. *IEEE Transactions on Automatic Control*, 55(4):950–955, April 2010.

[152] X. Wang, S. Wang, and D. Bi. Distributed visual-target-surveillance system in wireless sensor networks. *IEEE Transactions on Systems, Man and Cybernetics-part B: Cybernetics*, 39(5):1134–1146, May 2009.

[153] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(9):65–78, September 2004.

[154] L. Xiao, S. Boyd, and S. J. Kim. Distributed Average Consensus with Least-Mean-Square Deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, October 2007.

[155] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *International Conference on Information Processing in Sensor Networks*, pages 63–70, Los Angeles, April 2005.

[156] M. Xu and M. Petrou. Distributed RANSAC for 3D Reconstruction. In *Proceedings of the SPIE*, pages 1–9, January 2008.

[157] J. Yao and W. Cham. Robust multi-view feature matching from multiple unordered views. *Pattern Recognition*, 40(11):3081–3099, November 2007.

[158] D. Yuan, S. Xu, H. Zhaoa, and Y. Chub. Accelerating distributed average consensus by exploring the information of second-order neighbors. *Physica Letters A*, 37(4):2439–2445, May 2010.

[159] Y. Yuan, G. Stan, L. Shi, and J. Gonçalves. Decentralised final value theorem for discrete-time lti systems with application to minimal-time distributed consensus. In *IEEE International Conference on Decision and Control*, pages 2664–2669, December 2009.

[160] L. Zelnik-Manor and M. Irani. Multiview constraints on homographies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):214–222, February 2002.

[161] F. Zhang and N.E. Leonard. Cooperative Filters and Control for Cooperative Exploration. *IEEE Transactions on Automatic Control*, 55(3):650–663, March 2010.

[162] J. Zhou and Q. Wang. Convergence speed in distributed consensus over dynamically switching random networks. *Automatica*, 45(6):1455–1461, June 2009.

[163] M. Zhu and S. Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, February 2010.

[164] M. Zhu and S. Martínez. On the convergence time of asynchronous distributed quantized averaging algorithms. *IEEE Transactions on Automatic Control*, 56(2):386–390, February 2011.

[165] Z. Zivkovic, O. Booij, and Ben Kröse. From images to rooms. *Robotics and Autonomous Systems*, 55(5):411–418, May 2007.