Politecnico di Bari

scuola interpolitecnica di dottorato

Doctoral Program in
Information and Communication Technologies

Final Dissertation

Theory and Applications of Consensus Protocols for Distributed Estimation Algorithms

Antonio Petitti

Tutor:
Prof. Alessandro Rizzo

Co-ordinator of the Doctorate Course:
Prof. Michele Trovato

Co-tutors:
Dr. Antonio Franchi (LAAS-CNRS)
Dr. Donato Di Paola (ISSIA-CNR)

February 2015

Declaration of Authorship

I, Antonio Petitti, declare that this thesis titled, 'Theory and Applications of Consensus Protocols for Distributed Estimation Algorithms' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Politecnico di Bari.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Politecnico di Bari or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Theory is when we know everything but nothing works. Praxis is when everything works but we do not know why. We always end up by combining theory with praxis: nothing works and we do not know why."

Albert Einstein

POLITECNICO DI BARI

Abstract

Dipartimento di Ingegneria Elettrica e dell'Informazione

Doctor of Philosophy

Theory and Applications of Consensus Protocols for Distributed Estimation Algorithms

by Antonio Petitti

The research presented in this thesis has been focused on the development of distributed estimation algorithms for sensor and robotic networks. In particular, two algorithms are described. They capitalize on consensus theory to achieve a total distribution of the estimation with a low computational burden and a high tolerance to faults. The first algorithm is the Distributed Kalman filtering via Node Selection (DKNS). It is based on the max-consensus protocol and performs the distributed estimation of a measurement of interest by a network of fixed or mobile sensors. An application of the DKNS on distributed target tracking is shown and its performance is evaluated. The second algorithm is aimed at solving the problem of the estimation of the inertial parameters of an unknown payload manipulated by a team of Unmanned Ground Vehicles. The algorithm is grounded on kinematic and dynamic arguments, nonlinear observers, and average consensus protocols, and is able to work properly even in presence of noisy measurements. In order to apply the DKNS algorithm in real contexts, in the final part of the thesis, a theoretical analysis of the asynchronous max-consensus is presented. This is motivated by the little that has been done on the max-consensus protocol, which is at the base of many distributed decision-making system and, in particular, of the DKNS algorithm. Specifically, based on the theory of asynchronous algorithms, the convergence properties of the max-consensus protocol with asynchronous updates and bounded time delays have been analyzed. The main result is that the strongly connectedness of the directed communication network is a sufficient condition for the convergence of the asynchronous max-consensus protocol in finite time. Implementation issues are also taken into account with the definition of a distributed detection mechanism of the protocol convergence.

Acknowledgements

I would like to thank my advisor Prof. Alessandro Rizzo, who provided assistance in numerous ways, and Dr. Antonio Franchi, who hosted me at MPI and LAAS.

I am also grateful to my collaborator Donato Di Paola, and to my coauthors (in alphabetic order) A. Argentieri, G. Attolico, G. Cicirelli, S. Giannini, P. L. Mazzeo, A. Milella, P. Spagnolo.

Infinitely thanks to my family and my wife for supporting me in these last three years.

I also want to thank all the people who have influenced me in positive way during my PhD, in random order: G. Descisciolo, M. Tognon, L. Colasanto, N. Staub, M. Caccia, S. Lacroix, P. Stegagno, E. Vogli, C. Gerboni, D. Cazzato, R. Colella, P. Inglese, and many others I am forgetting in this moment (sorry :)).

Contents

List of Figures

List of Tables

Abbreviations

DKNS	**D**istributed **K**alman filtering via **N**ode **S**election
w.r.t.	with reference to
PCV	**P**erception **C**onfidence **V**alue
CKF	**C**entralized **K**alman **F**ilter
KCF	**K**alman **C**onsensus **F**ilter
DKF	**D**iffusion **K**alman **F**ilter
UGV	**U**nmanned **G**round **V**ehicle
MASF	**M**ulti **A**gent **S**imulation **F**ramework

to my beautiful wife Mariantonietta

Chapter 1

Introduction

Distributed estimation is one of the hottest research topics in robotics and sensor networks. The goal of distributed estimation is the agreement of a pool of sensing agents on one or more quantities of interest. Agents are distributed in space, and can be fixed or mobile, such as a fixed sensor network, or a network of mobile robots carrying sensors on board, respectively. The motivation of such an interest resides in the wide range of applications, such as search and rescue [1], environmental monitoring [2], cooperative transportation [3] and so on. Information coming from sensing agents can be centrally collected and processed, or processed onboard the network nodes, with a limited communication activity, usually with a limited subset of neighboring nodes. In the former case, centralized strategies are defined, whereas in the latter we refer to distributed ones. Decentralized strategies, which are the subject of this thesis, constitute viable and effective solutions to the problem. The advantages of distributed solutions are usually a low computational and communication burden, a good scalability, and a great tolerance to faults, due to the absence of a single point of failure. On the other hand, centralized strategies are easier to implement and usually give solutions that are closer to the optimal one than those provided by distributed strategies. Distributed strategies can be designed from scratch, as inherently distributed, or by designing a distributed version of a centralized algorithm. In both cases, the design of a distributed estimation algorithm usually requires the definition of suitable agreement procedures, able to lead all the nodes towards the production of a common estimate without the support of centralizing units, and possibly with limited communication and computation burdens. To this aim, consensus algorithms [4] [5] have been widely adopted. Remarkable applications of consensus algorithms have been successfully realized in different contexts, such as task coordination [6] [7] [8], task assignment [9], formation control [10], flocking [11] [12], rendez-vous [13], and, of course, distributed estimation [14] [15].

This thesis focuses on the design of distributed estimation algorithms for sensor and robotic networks. Specifically, a distributed Kalman filter and a distributed algorithm for the estimation of inertial parameters of an unknown load will be presented. The common feature of these strategies is the use of consensus protocols as effective means towards the distribution of the estimation algorithm. The thesis is organized as follows.

In Chapter 2, we first introduce some notation. Then, we review some basic results in graph theory and consensus protocols.

In Chapter 3, the Distributed Kalman filtering via Node Selection (DKNS) is introduced. DKNS is an iterative algorithm, where each iteration consists of two phases, namely *estimation* and *node selection*. In the estimation phase, each network node performs an estimate of the current state of the dynamical process under observation by means of a local Kalman filter. In the node selection phase, a max-consensus protocol is run on an accuracy index, called *perception confidence value*. In this way, each network node will converge in finite time to the most accurate estimate performed over the whole network at a given time. The main contribution of this chapter is the definition of a *fully distributed* Kalman filtering strategy. As we will demonstrate in Chapter 3, such a strategy is particularly effective in networks composed by sensors with limited communication and sensing ranges, where only a fraction of them can sense the state of the process under observation at a given time. Moreover, a certain degree of heterogeneity of the sensors is assumed, by considering sensors with different sensing ranges. This assumption implies that the measurement reliability changes along the space. The proposed strategy is applied toward the solution of a distributed target tracking problem.

Chapter 4 presents a distributed algorithm for the estimation of the inertial parameters of an unknown load through manipulation by a network of Unmanned Ground Vehicles. The proposed algorithm requires that each robot is able to control the force applied to the load and to measure the velocity of the contact point. The aim of the algorithm is to estimate the inertial parameters of the load, such as the mass and the moment of inertia. This will, in turn, allow the design of effective distributed and adaptive control strategies, relying on the estimate of the load parameters. To the best of our knowledge, the proposed approach is the first to be totally distributed. Moreover, the defined framework is realistic and can be easily implemented, since it relies on the availability of measurements that are easily available and not extremely noisy. Finally, we elucidate the effect of measurement noise on the quality of estimates, providing a convenient bound on their accuracy.

Many distributed estimation approaches relies on the hypothesis that information exchanges in agreement protocols are performed synchronously. This assumption is obviously not very realistic. Thus, the effect of asynchronous communication on the agreement performance is a topic that deserve accurate investigations. While asynchronous average consensus protocols have been widely investigated, this has not been the case with max-consensus ones. In this context, Chapter 5 presents the study of the convergence properties of the max-consensus protocol in presence of asynchronous updates and bounded communication delays. The contribution of Chapter 5 is threefold. First, the convergence of the asynchronous max-consensus protocol in finite time is proved, under the hypotheses of partial asynchronism and strongly connectedness of the underlying directed communication network. Second, under additional mild assumptions, an upper bound on the convergence time is derived. Third, a distributed strategy for the detection of the convergence of the protocol is presented as a valuable tool for the implementation of the protocol in real applications. To the best of our knowledge, this is the first work dealing with the analysis of the max-consensus protocol in presence of time delays and asynchronous updates.

Finally, in Chapter 6 we draw our conclusions.

Chapter 3, 4, and 5 follow the same structure. At first, a brief section introduces the related work toward the definition of the problem statement. Then, our research is presented. Finally, a numerical results section is presented, followed by our conclusions.

Chapter 2

Background Notions

In this chapter we first introduce some notation. Then, we recall some basic concepts of graph theory and we review some notions on discrete-time consensus protocols. Concepts that, instead, are specific to single chapters will be presented at the beginning of those chapters.

2.1 Notations

Throughout the thesis, we denote with $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{R}_+$ the sets of positive integers, non-negative integers, rational numbers, real numbers, and non-negative real numbers, respectively. Capital bold letters refer to matrices, while small bold letters refer to vectors. The operator $|\cdot|$ stands for set cardinality and the operator $\otimes$ stands for the Kronecker product.

2.2 Graph Theory

This thesis deals with estimation algorithms for sensor and robotic networks. The communication structure of a network is usually represented by a *graph*. A graph $\mathcal{G}$ is a pair $(\mathcal{I}, \mathcal{E})$, where $\mathcal{I} \triangleq \{1, \dots, n\}$ is a finite nonempty set of n *nodes* and $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$ is a set of ordered pairs of nodes, called *edges*. The existence of an edge $(j, i) \in \mathcal{E}$ denotes that node i can obtain information from node j, but not necessarily vice versa. Self edges $(i, i) \in \mathcal{E}$ are allowed. If the pairs of nodes are ordered the graph is said *directed* (also known as a *digraph*). In this case, for the edge (j, i), j is called *parent* node and i is called *child* node. On the other hand, if node i and j can always obtain information from each other, that is, pairs of nodes are unordered, and therefore the existence of link $(j, i) \in \mathcal{E}$

implies that of link $(i,j) \in \mathcal{E}$, the graph is said *undirected*. Given a node $i \in \mathcal{I}$, the set of *neighbors* of node i is defined by $\mathcal{N}_i = \{j \in \mathcal{I} : (j,i) \in \mathcal{E}\}$. Each link $(j,i) \in \mathcal{E}$ of a graph can be associated to a *weight,* $w^{ij} \in \mathbb{R}$, a quantity which may contain information connected to the link itself, such as for example communication cost, delay time in the propagation of information, etc. If the weights are the same for every link in the graph, then the graph is called a *non-weighted* one. Otherwise, the graph is a *weighted* one.

The connection scheme of a graph can be described in different ways. The *adjacency matrix* $\mathcal{A} = [a^{ij}] \in \mathbb{R}^{n \times n}$ of a graph with n nodes is defined as

$$a^{ij} = \begin{cases} w^{ij} & \text{if } (j,i) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

If the weights are not relevant, or the graph is not weighted, then w^{ij} is set equal to 1 for all $(j,i) \in \mathcal{E}$. In the case of undirected graphs the adjacency matrix is symmetrical. This obviously does not occur in the case of directed graphs. A *direct path* between two nodes j and i is a sequence of edges in a direct graph between two nodes, of the form (j,j_1),(j_1,j_2), ..., (j_{N-1},j_N), (j_N,i). An *undirected path* in an undirected graph is defined analogously. The number of edges in a direct path is called *distance* between two nodes along the given path. The direct path of shortest distance between two nodes is called *shortest path* and the related distance along the path is called *shortest path length.* The greatest shortest path length between any pair of nodes in a graph $\mathcal{G}$ is called *diameter* of the graph and is indicated with $\mathcal{D}$. A directed graph is *strongly connected* if there exists a direct path from every node to every other node. An undirected graph is *connected* if there exists an undirected path between every pair of distinct nodes.

2.3 Consensus Protocols

This thesis presents two different distributed estimation algorithms that capitalize on consensus protocols to achieve a total distribution of the estimation. In order to introduce a generic consensus protocol, we consider a set of n dynamical systems with state variable $x_i \in \mathbb{R}$, which are the nodes of a network described by a graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, whose dynamics can be represented, in discrete time, as

$$x_i(k+1) = f(x_i(k), u_i(k)), \quad k \in \mathbb{N}_0,\ i \in \mathcal{I}. \tag{2.2}$$

Let $\Phi : \mathbb{R}^n \to \mathbb{R}$ be a function of n variables $x_1, \ldots, x_n$. The Φ-*consensus problem* over a network consists of computing $\Phi(x(0))$, where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = [x_1 \ \ldots \ x_n]^T$, in a distributed way, by applying suitable control inputs u_i that only depend on the values of the state

of node i and on that of its neighbors. We define a *protocol* as

$$u_i = f_i^*(x_{j_1}, \ldots, x_{j_{m_i}}), \tag{2.3}$$

with $j_1, \ldots, j_{m_i} \in \{i\} \cup \mathcal{N}_i$, and we say that protocol (2.3) solves the Φ–*consensus problem* if and only if the state of each system converges asymptotically, or in finite time, to the value $\Phi(\mathbf{x}(0))$. Cases of broad applications in distributed estimation problems correspond to the choices of

$$\Phi(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.4}$$

$$\Phi(\mathbf{x}) = \max_{i\in\mathcal{I}} x_i, \tag{2.5}$$

$$\Phi(\mathbf{x}) = \min_{i\in\mathcal{I}} x_i, \tag{2.6}$$

called *average consensus, max-consensus,* and *min-consensus,* respectively. Consensus algorithms present the following convergence properties [5] [6]:

Lemma 2.1. *Given a connected network $\mathcal{G}$, an initial state $\mathbf{x}(0)$, and assuming that each node of $\mathcal{G}$ applies the following protocol,*

$$x_i(k+1) = x_i(k) + \varrho \sum_{j\in\mathcal{N}_i} \left(x_j(k) - x_i(k)\right), \tag{2.7}$$

where $0 < \varrho < 1/(\max_i \sum a^{ij})$, then all the nodes of the graph asymptotically reach a common steady state, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$, i.e.

$$\lim_{k\to\infty} \|x_i(k) - \bar{x}\| = 0,\ \forall i \in \mathcal{I}. \tag{2.8}$$

Lemma 2.2. *Given a connected network $\mathcal{G}$, an initial state $\mathbf{x}(0)$, and assuming that each node of $\mathcal{G}$ applies the following protocol,*

$$x_i(k+1) = \max_{j\in\mathcal{N}_i\cup i} x_j(k), \tag{2.9}$$

then, all the nodes of the graph reach in finite time a common steady state, $\bar{x} = \max\{x_1(0), \ldots, x_n(0)\}$. Specifically, it can be proved that

$$x_i(k) = \max\{x_1(0), \ldots, x_n(0)\},\ \forall k \geq \mathcal{D},\ \forall i \in \mathcal{I}. \tag{2.10}$$

Note that Lemma 2.2 can be easily recast for min-consensus by substituting the operator max with the operator min. For a formal proof of the previous statements the reader is referred to [6] and to [16]. Consensus protocols can be dealt with also in continuous time [5].

Chapter 3

Distributed Estimation in Limited Sensing Frameworks

In this chapter, we propose a strategy for distributed Kalman filtering over sensor networks, based on node selection rather than sensor fusion. The presented approach is particularly suitable when sensors with limited sensing capability are considered. In this case, strategies based on sensor fusion may exhibit poor performance, as several unreliable measurements may be included in the fusion process. On the other hand, our approach implements a distributed strategy able to select only the node with the most accurate estimate and propagate it through the whole network in finite time. The algorithm is based on the definition of a metric of the estimate accuracy, and on the application of an agreement protocol based on max-consensus. We prove the convergence, in finite time, of all the local estimates to the most accurate one at each discrete iteration. Moreover, we prove the equivalence with a centralized Kalman filter with multiple measurements, evolving according to a state-dependent switching dynamics. An application of the algorithm to distributed target tracking over a network of heterogeneous range-bearing sensors is shown. Simulation results and a comparison with two distributed Kalman filtering strategies based on sensor fusion confirm the suitability of the approach. Moreover, an experimental validation of the algorithm, in an ambient assisted living framework, is also shown.

The content of this chapter is grounded on a book chapter [17], a journal paper [18], and conference papers [19] [20] [21].

3.1 Distributed Kalman Filters

In the last decades, sensor networks have received significant attention by many researchers in different fields of engineering, such as robotics, control theory, and image processing [22]. As a general principle, sensor networks perform estimates of the state of dynamical processes through computation and communication among the network nodes. Recent technological innovations have made possible to deploy a large number of inexpensive and low-power sensors to cover wide areas [23], making a wide range of applications possible and affordable, such as habitat monitoring, animal tracking, environment observation, forecasting, surveillance, and domotics.

The communication scheme of a sensor network determines whether the estimation algorithm is centralized, distributed or hierarchical. Centralized algorithms make use of a fusion center, able to receive data from each node and to compute a global estimate of the system state [24] [25]. The presence of a single fusion center simplifies the process of computing a unique and optimized estimate. On the other hand, this may lead to high data transfer rates and to lack of scalability, energy efficiency, and fault tolerance, because of the presence of a single point of failure. These limitations can be overcome through *distributed* approaches [26]. Steps towards the realization of effective distributed approaches have been made at first by considering all-to-all communication schemes over the sensor network, which yield the elimination of the fusion center, yet they may introduce a heavy communication overhead [27]. Approaches based on hierarchical or hybrid architectures adopt local connection schemes, by making use of local fusion centers that only communicate with subsets of nodes [28] [29]. The transition to fully distributed estimation algorithms requires the definition of suitable agreement protocols, able to lead all the nodes towards the production of a common estimate without the support of centralizing units, possibly with limited communication and computation burdens. To this aim, consensus algorithms have been widely adopted [4] [5].

The transition from centralized to distributed algorithms has touched nearly all the techniques of estimation over networks, including the well-known Kalman filter [30] [31]. The Centralized Kalman Filter with multiple measurements (CKF) is, in fact, a centralized estimation strategy that can be implemented over sensor networks [32]. Much effort has been devoted to the implementation of distributed estimation schemes based on Kalman filtering, such as the Kalman-Consensus Filter (KCF) [33], or the Diffusion Kalman Filter (DKF) [34]. Both algorithms adopt the formalism of the information filter, which simply recasts the Kalman filter equations in terms of the information state vector and matrix [35]. However, these approaches do not guarantee the convergence of all the individual estimates to a common value. Moreover, they require a further phase of sensor fusion, even though this is performed on a subset of the network nodes, with

the aim of containing the communication and computational burden. It is well-known that special care must be taken in sensor fusion, when sensors with *limited sensing capability* are considered, as the phenomenon of *catastrophic fusion* may occur [32]. Limited sensing capability is a realistic scenario in applications, such as, for example, indoor surveillance and target tracking. In these applications, in fact, sensors have usually a limited sensing range and, at a given time instant, it may happen that the majority of sensors does not even possess a measurement. In such a scenario one can easily figure out the effect of involving unreliable sensors in the fusion operation. Thus, particular operational conditions, such as those of limited sensing capability, call for a different strategy in the computation of a unique estimate from the network data. In this perspective, we propose node selection as a valuable strategy to deal with those cases in which sensor fusion exhibits poor performance due to the potential presence of several corrupted measurements in the sensor fusion set [36]. This means that only some network nodes are selected as the owners of the best estimate that will be in turn propagated to all the network nodes.

In this chapter, we introduce a fully distributed approach to the realization of an algorithm for Distributed Kalman filtering via Node Selection (DKNS) for heterogeneous sensor networks, where nodes have limited sensing capability. Considering heterogeneous nodes has important consequences in the algorithm design, since the accuracy of the measurements performed by each sensor may change over space and time, thus involving different uncertainties on estimates to be accounted for during the agreement process over the network.

Beyond being fully distributed, another valuable property of DKNS is that it guarantees that at each iteration, every network node owns the same estimate of the process state. Node selection is achieved using a strategy based on the *max-consensus* protocol [16], performed on a measurement accuracy metric called *perception confidence value,* strictly related to the Fisher information [37] [38].

The chapter is organized as follows. Section 3.2 describes in full detail the algorithm. In Section 3.3, the performance of the proposed algorithm is assessed via numerical simulations and real-world experiments on a distributed target tracking application. Finally, our conclusions are drawn.

3.2 Distributed Kalman filtering via Node Selection

In this section, we present the Distributed Kalman filtering via Node Selection algorithm and assess its performance in limited sensing scenarios. We start by presenting the

problem setup.

We consider a sensor network modeled by an undirected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, where $\mathcal{I} \triangleq \{1, \ldots, n\}$ is the node set, and $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$ is the set of edges (links), which represent point-to-point communication channels. The neighbor set of node $i \in \mathcal{I}$ is defined as $\mathcal{N}_i = \{j \in \mathcal{I} \mid (j,i) \in \mathcal{E}\}$. Each network node $i \in \mathcal{I}$ is a sensor able to estimate the state $\mathbf{x}(k) \in \mathbb{R}^m$ of a time-discrete process with time index $k \in \mathbb{N}_0$, through a Kalman filter, and to exchange information only with its one-hop neighbors. In accordance with the classic Kalman filter theory, we assume that the process state $\mathbf{x}(k)$ evolves according to a linear model, as

$$\mathbf{x}(k) = \mathbf{A}(k)\mathbf{x}(k-1) + \boldsymbol{\upsilon}(k), \tag{3.1}$$

where $\mathbf{A}(k)$ is the state matrix and $\boldsymbol{\upsilon}(k)$ is the process noise, which is assumed to be drawn from a zero mean multivariate normal distribution with covariance matrix $\mathbf{Q}(k)$, that is, $\boldsymbol{\upsilon}(k) \sim N(\mathbf{0}, \mathbf{Q}(k))$.

Moreover, we assume that measurements $\boldsymbol{\zeta}_i(k)$ of the state $\mathbf{x}(k)$ are performed by the generic i-th node as:

$$\boldsymbol{\zeta}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \boldsymbol{\nu}_i(k), \tag{3.2}$$

where $\mathbf{H}_i(k)$ is the measurement output matrix of the i-th node and $\boldsymbol{\nu}_i(k)$ is the corresponding observation noise, which is assumed to be zero mean Gaussian white, with covariance matrix $\mathbf{R}_i(k)$, that is, $\boldsymbol{\nu}_i(k) \sim N(\mathbf{0}, \mathbf{R}_i(k))$. The noise vectors at each time step are assumed to be mutually independent.

The generic node $i \in \mathcal{I}$ is represented by the tuple:

$$\langle \hat{\mathbf{x}}_i(k),\ \mathcal{X}_i(k),\ \gamma_i(k),\ \overline{\mathbf{x}}_i(k) \rangle. \tag{3.3}$$

In (3.3), $\hat{\mathbf{x}}_i(k)$ is the local state estimate produced by the sensor i at time k. To model the limited sensing capability of each sensor, the *sensing set* $\mathcal{X}_i(k) \subset \mathbb{R}^m$ is defined, whereby node i is able to perform a measurement of the state $\mathbf{x}(k)$ if and only if $\mathbf{x}(k) \in \mathcal{X}_i(k)$. Heterogeneity of sensors is assumed, by defining individual sensing sets. We define the set of the *sensing* nodes $\Sigma(k) \triangleq \{i \in \mathcal{I} \,|\, \mathbf{x}(k) \in \mathcal{X}_i(k)\}$. The set difference $\Lambda(k) = \mathcal{I} \setminus \Sigma(k)$ is defined as the set of *predicting* nodes. The quantity $\gamma_i(k)$ is dubbed as *perception confidence value* and quantifies the estimate accuracy of node i. To work effectively, the perception confidence value must have the property of being larger as long as the estimate is more accurate. More formally, considering two state estimates $\hat{\mathbf{x}}_i(k)$ and $\hat{\mathbf{x}}_j(k)$, where i and $j \in \mathcal{I}$

$$E[||\hat{\mathbf{x}}_i(k) - \mathbf{x}(k)||^2] < E[||\hat{\mathbf{x}}_j(k) - \mathbf{x}(k)||^2] \quad \Rightarrow \quad \gamma_i(k) > \gamma_j(k) \tag{3.4}$$

Algorithm 1 Estimation phase for node i at iteration k

Input: $\overline{\mathbf{x}}_i(k-1)$, $\overline{\mathbf{P}}_i(k-1)$
Output: $\gamma_i(k)$, $\hat{\mathbf{x}}_i(k)$, $\mathbf{\Psi}_i(k)$

1: $\hat{\mathbf{x}}_i^-(k) = \mathbf{A}(k)\overline{\mathbf{x}}_i(k-1)$
2: $\mathbf{P}_i^-(k) = \mathbf{A}(k)\overline{\mathbf{P}}_i(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k)$
3: **if** $i \in \Sigma(k)$ **then**
4: $\quad [\mathbf{P}_i(k)]^{-1} = [\mathbf{P}_i^-(k)]^{-1} + \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k)$
5: $\quad \hat{\mathbf{x}}_i(k) = \mathbf{P}_i(k)\{[\mathbf{P}_i^-(k)]^{-1}\hat{\mathbf{x}}_i^-(k) + \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\boldsymbol{\zeta}_i(k)\}$
6: $\quad \mathbf{\Psi}_i(k) = \mathbf{P}_i(k)$
7: **else**
8: $\quad \hat{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i^-(k)$
9: $\quad \mathbf{\Psi}_i(k) = \mathbf{P}_i^-(k)$
10: **end if**
11: $\gamma_i(k) = 1/\text{Trace}\,[\mathbf{\Psi}_i(k)]$

must hold (where $E[\cdot]$ indicates statistical expectation). Finally, $\overline{\mathbf{x}}_i(k)$ is the value of the state estimate obtained after the procedure of node selection.

Each iteration of DKNS consists of two phases: *estimation* and *node selection.* In the former phase, each node performs an individual estimate through a local Kalman filter. In the latter one, all the nodes agree on the best available estimate in a distributed fashion. We now describe the two phases in detail at iteration k, therefore assuming that each node owns the information of the tuple (3.3) at iteration $k-1$.

3.2.1 Phase 1: Estimation Phase

The estimation phase is schematized in Algorithm 1. Each node locally runs a Kalman filter. Specifically, the local Kalman filter starts, at each iteration, from a given state estimate $\overline{\mathbf{x}}_i(k-1)$ and covariance matrix $\overline{\mathbf{P}}_i(k-1)$, which are the best ones available over the network, obtained and propagated during iteration $k-1$. If node i is a *sensing* node, at iteration k, *i.e.,* $i \in \Sigma(k)$, a full Kalman filter iteration is run (lines 1–2 and 4–6 of Algorithm 1). On the contrary, in the case of a *predicting* node, *i.e.,* $i \in \Lambda(k)$, only the prediction part of the Kalman filter is run (lines 1–2 and 8–9). Then, each node ends the estimation phase by computing the perception confidence value $\gamma_i(k)$ as:

$$\gamma_i(k) \triangleq \begin{cases} \frac{1}{\text{Trace}[\mathbf{P}_i(k)]} & \text{if} \quad i \in \Sigma(k) \\ \frac{1}{\text{Trace}\left[\mathbf{P}_i^-(k)\right]} & \text{if} \quad i \in \Lambda(k) \end{cases}, \tag{3.5}$$

if node i is a sensing one, the perception confidence value $\gamma_i(k)$ is computed on the basis of the *a posteriori* error covariance matrix, otherwise, the operation refers to the *a priori* error covariance matrix. It is well-known that minimizing the trace of the error covariance matrix is equivalent to minimize the expected value of the square of

the magnitude of the error vector [39]. Thus, $\gamma_i(k)$ grows with the accuracy of the estimation performed by node i at time k, and its definition as in Eq. (3.5) satisfies the requirements stated earlier in Eq. (3.4).

At the end of Phase 1, each node i stores the perception confidence value $\gamma_i(k)$, the local state estimate $\hat{\mathbf{x}}_i(k)$, and the local error covariance matrix $\mathbf{\Psi}_i(k)$ (output line of Algorithm 1).

3.2.2 Phase 2: Node Selection Phase

The node selection phase is run to select the node with the highest perception confidence value and to propagate its estimate and the corresponding error covariance matrix over the network in finite time. Node selection is achieved in a totally distributed fashion and its working principle is described by Algorithm 2. The core of the algorithm is a max-consensus protocol [16], described in lines 4–11 of Algorithm 2. At this stage, a new discrete-time index $\hbar$ is defined, making the max-consensus protocol run on a different sampling time (typically, shorter than the one associated with the time-discrete index k, which is the actual sampling time of DKNS).

Algorithm 2 works as follows: node i initially sets its internal variables to values obtained from Phase 1 (line 1–3 of Algorithm 2), *i.e.,* the perception confidence value (by initializing $\varsigma_i(0)$), the state estimate (by initializing $\boldsymbol{\chi}_i(0)$), and the covariance matrix (by initializing $\mathbf{\Pi}_i(0)$). Then, a protocol based on max-consensus lets the nodes' variables $\varsigma_i(\cdot)$ to converge to the maximum of all perception confidence values over the network (line 7). Correspondingly, line 8 selects the node that performed the estimate of the process state with the best accuracy (by storing the index of the selected node in μ_i), and lines 9–10 select the corresponding best estimate and error covariance matrix. It can be proved that a max-consensus protocol converges in a number of iterations equal to the communication graph diameter [40]. Thus, the node selection loop (lines 4–11) is run for exactly $\mathcal{D}$ steps, where $\mathcal{D}$ is the diameter of the graph $\mathcal{G}$, after which the convergence of all the node variables is guaranteed[1]. At the convergence of the node selection loop, after $\mathcal{D}$ time steps, $\mu_i(\mathcal{D})$ will take the index of the node which performed the best estimate (line 8), $\boldsymbol{\chi}_i(\mathcal{D})$, and $\mathbf{\Pi}_i(\mathcal{D})$ will take the corresponding estimate and covariance estimate matrices, respectively (lines 9–10). At the end of Phase 2, an agreement both on the best estimate of the process state and on its corresponding *a posteriori* covariance matrix is achieved. Each node will store these values in its variables $\overline{\mathbf{x}}_i(k)$ and $\overline{\mathbf{P}}_i(k)$ (lines 12–13), respectively. These two variables will be provided to Phase 1

[1]Convergence issues will be dealt with in detail later in this section

Algorithm 2 Node Selection phase for node i at iteration k

Input: $\gamma_i(k)$, $\hat{\mathbf{x}}_i(k)$, $\mathbf{\Psi}_i(k)$
Output: $\overline{\mathbf{x}}_i(k)$, $\overline{\mathbf{P}}_i(k)$

1: $\varsigma_i(0) = \gamma_i(k)$
2: $\boldsymbol{\chi}_i(0) = \hat{\mathbf{x}}_i(k)$
3: $\mathbf{\Pi}_i(0) = \mathbf{\Psi}_i(k)$
4: **for** $\hbar = 1$ to $\mathcal{D}$ **do**
5: SEND$(\varsigma_i(\hbar-1), \boldsymbol{\chi}_i(\hbar-1), \mathbf{\Pi}_i(\hbar-1))$
6: RECEIVE$(\varsigma_j(\hbar-1), \boldsymbol{\chi}_j(\hbar-1), \mathbf{\Pi}_j(\hbar-1)), \quad \forall j \in \mathcal{N}_i$
7: $\varsigma_i(\hbar) = \max_{j \in \mathcal{N}_i \cup i} \{\varsigma_j(\hbar-1)\}$
8: $\mu_i(\hbar) = \operatorname{argmax}_{j \in \mathcal{N}_i \cup i} \{\varsigma_j(\hbar-1)\}$
9: $\boldsymbol{\chi}_i(\hbar) = \boldsymbol{\chi}_{\mu_i}(\hbar-1)$
10: $\mathbf{\Pi}_i(\hbar) = \mathbf{\Pi}_{\mu_i}(\hbar-1)$
11: **end for**
12: $\overline{\mathbf{x}}_i(k) = \boldsymbol{\chi}_i(\mathcal{D})$
13: $\overline{\mathbf{P}}_i(k) = \mathbf{\Pi}_i(\mathcal{D})$

of the next iteration $(k+1)$. This will let the Kalman filter of each node start from the best available estimate, and therefore to improve its individual prediction performance.

Remark 3.1. Line 7 of Algorithm 2 requires a tie-break rule in the case that the maximum value for $\varsigma_j(\hbar-1)$, $j \in \mathcal{N}_i$, is allocated in more than one of the neighboring nodes. The tie-break rule must guarantee convergence and has to be performed in a distributed way. Examples of tie-break rules are: choosing an index at random among those corresponding to the maximum, or performing, on lines 9 and 10, an average of $\boldsymbol{\chi}_{\mu_i}$ and $\mathbf{\Pi}_{\mu_i}$ along the involved indices μ_i, rather than a variable assignment.

Remark 3.2. We observe that either sensing or predicting nodes can be selected as those possessing the best estimate of the process state. In the case of heterogeneous sensor networks, in fact, it is possible that a predicting node that makes a prediction starting from a very accurate past measurement performs better than a sensing node which is directly measuring with poor accuracy.

3.2.3 Convergence Properties

The DKNS algorithm makes each node converge in finite time to the estimate of the process state owned by the node with the highest perception confidence value. This is made possible thanks to the convergence property of the max-consensus protocol [16], which are here briefly recalled before providing our main result on the DKNS convergence.

Theorem 3.3. *Consider a network of n dynamical systems, connected over an undirected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$. Each dynamical system has a state variable $\phi_i(\hbar) \in \mathbb{R}$, $i \in \mathcal{I}$,*

$\bar{k} \in \mathbb{N}_0$. *The discrete-time max-consensus protocol is defined as*

$$\phi_i(\bar{k}+1) = \max_{j \in \mathcal{N}_i \bigcup i} \phi_j(\bar{k}). \tag{3.6}$$

Assume that each node of the network $\mathcal{G}$ runs the protocol in Eq.(3.6). If $\mathcal{G}$ is connected, then

$$\begin{aligned} \phi_i(\bar{k}) &= \phi_j(\bar{k}) \\ &= \max\left(\phi_1(0), \cdots, \phi_n(0)\right) \quad \forall i, j \in \mathcal{I}, \quad \forall \bar{k} \geq \mathcal{D}, \end{aligned}$$

where $\mathcal{D}$ is the graph diameter.

Proof. The proof is provided in Theorem 4.1 of [16]. □

The following theorem proves the convergence of the DKNS.

Theorem 3.4. *Given a network of n nodes modeled as an undirected connected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, and assuming that each node runs the DKNS algorithm; then, at the end of each discrete iteration $k \in \mathbb{N}_0$, the following holds: $\overline{\mathbf{x}}_i(k) = \overline{\mathbf{x}}_j(k) = \overline{\mathbf{x}}_{j^\star(k)}(k) \quad \forall\, i, j \in \mathcal{I}$, and $j^\star(k) = \underset{j \in \mathcal{I}}{\operatorname{argmax}}\, \gamma_j(k)$.*

Proof. The proof comes from the analysis of the steps of Algorithm 2. We observe that line 7 in Algorithm 2 is the update rule of the discrete-time max-consensus protocol over the perception confidence values of each node of the network (assigned to variables $\varsigma_i(\bar{k})$ in Algorithm 2). Theorem 3.3 proves that this protocol converges to the maximum of the initial states in a finite number of steps, upper bounded by the graph diameter $\mathcal{D}$, if and only if $\mathcal{G}$ is connected. Therefore, in $\mathcal{D}$ steps, it is guaranteed that

$$\varsigma_i(\mathcal{D}) = \varsigma_j(\mathcal{D}) = \max_{l \in \mathcal{I}} \varsigma_l(0) = \max_{l \in \mathcal{I}} \gamma_l(k), \quad \forall\, i, j \in \mathcal{I}.$$

Correspondingly, in $\mathcal{D}$ steps, the assignment in line 8 allows each node to store the index of the node where the maximum of the perception confidence values is located, *i.e.*, $\mu_i(\mathcal{D}) = j^\star(k) = \underset{l \in \mathcal{I}}{\operatorname{argmax}}\, \gamma_l(k)$. Thus, $\boldsymbol{\chi}_i(\mathcal{D}) = \boldsymbol{\chi}_{j^\star(k)}$ and $\boldsymbol{\Pi}_i(\mathcal{D}) = \boldsymbol{\Pi}_{j^\star(k)}$, $\forall i \in \mathcal{I}$. Finally, the assignment in line 12 concludes the proof of the theorem. □

Remark 3.5. A single iteration of DKNS consists of two cascaded phases, therefore its duration is the sum of the durations of the two phases. The estimation phase consists of a sequence of variable assignments, therefore it terminates in finite time, indicated as t_e. Taking into account the result of Theorem 3.4, the convergence of the node selection phase is guaranteed if the node selection loop (lines 4–11 in Algorithm 2) is run for at

least $\mathcal{D}$ steps. We observe that, even though the knowledge of a global quantity such as the network diameter is required, this quantity can be easily computed in a distributed way before the algorithm starts [41]. Real time applications of DKNS require that the node selection phase is completed before a new instance of the estimation phase is started. Thus, defining ε the sampling time associated to the single iteration of the DKNS algorithm (related to discrete-time index k), and τ the sampling time associated to one iteration of the node selection loop (*i.e.,* associated to the discrete-time index $\hbar$) in Algorithm 2, the inequality $t_e + \mathcal{D}\tau < \varepsilon$, must hold. We also remark that the diameter of the graph $\mathcal{D}$ is related to the number of nodes n and to the graph topology. Thus, the network structure will concur to impose constraints to the choice of the sampling time ε.

3.2.4 Analysis of the Node Selection Mechanism

In a CKF with multiple measurements [32], a fusion center collects the measurements $\boldsymbol{\zeta}_i(k)$, the measurement output matrices $\mathbf{H}_i(k)$, and the observation covariance matrices $\mathbf{R}_i(k)$, from all nodes i, $i = 1, \ldots, n$. Then, a Kalman filter is run at the fusion center level [42]. In distributed approaches, no centralization of the information is admitted. Hence, each node can communicate with a limited set of neighbors. Distributed Kalman filters have been proposed in literature to overcome the limitations of centralized approaches; yet retaining some aspects of hierarchical organization [34] [33]. Distributed versions of CKF needs to locally approximate the covariance matrix.

$$[\mathbf{P}(k)]^{-1} = [\mathbf{P}^{-}(k)]^{-1} + \sum_{j=1}^{n} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k) \tag{3.7}$$

at the node level [2]. It is straightforward to note that in CKF all the sensors contribute to the update of the *a posteriori* covariance matrix. On the other hand, in distributed approaches each node approximates locally the summation in Eq. (3.7). For example, this is done through a local summation within the neighborhood of each node, as

$$[\mathbf{P}_i(k)]^{-1} = [\mathbf{P}_i^{-}(k)]^{-1} + \sum_{j \in \mathcal{N}_i} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k), \tag{3.8}$$

where $[\mathbf{P}_i(k)]^{-1}$ is the local estimate of the centralized *a posteriori* inverse covariance matrix (3.7) carried out by node i, and $\mathcal{N}_i$ is the set of the direct neighbors of node i.

Theorem 3.6. *Given a network of n nodes modelled as an undirected connected graph* $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$*, and assuming that each node has limited sensing capability, that is, the*

[2]The update of the covariance matrix is written by using the well-known information form of the Kalman filter [42].

measurement set is given by $\boldsymbol{\zeta}_j(k) = \mathbf{H}_j(k)\mathbf{x}(k) + \boldsymbol{\nu}_j(k)$*, with* $j \in \Sigma(k)$*; then, running DKNS is equivalent to run a Centralized Kalman Filter with multiple measurements and state-dependent switching dynamics, whose evolution is dictated by:*

$$\hat{\mathbf{x}}^-(k) = \mathbf{A}(k)\hat{\mathbf{x}}(k-1), \tag{3.9}$$

$$\mathbf{P}^-(k) = \mathbf{A}(k)\mathbf{P}(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k), \tag{3.10}$$

$$[\mathbf{P}(k)]^{-1} = [\mathbf{P}^-(k)]^{-1} + \mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))[\mathbf{H}^T_{j^\star(k)}(k)\mathbf{R}^{-1}_{j^\star(k)}(k)\mathbf{H}_{j^\star(k)}(k)], \tag{3.11}$$

$$\hat{\mathbf{x}}(k) = \mathbf{P}(k)\{[\mathbf{P}^-(k)]^{-1}\hat{\mathbf{x}}^-(k) + \mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))[\mathbf{H}^T_{j^\star(k)}(k)\mathbf{R}^{-1}_{j^\star(k)}(k)\boldsymbol{\zeta}_{j^\star(k)}(k)]\}; \tag{3.12}$$

where $j^\star(k) = \underset{j\in\mathcal{I}}{\operatorname{argmax}}\ \gamma_j(k)$,

$$\gamma_j(k) \triangleq \begin{cases} \frac{1}{\mathrm{Trace}[\mathbf{P}_j(k)]} & \text{if} \quad \mathbf{x}(k) \in \mathcal{X}_j(k) \\ \frac{1}{\mathrm{Trace}[\mathbf{P}^-_j(k)]} & \text{if} \quad \mathbf{x}(k) \notin \mathcal{X}_j(k) \end{cases},$$

$\mathbf{P}^-_j(k)$, $\mathbf{P}_j(k)$ are, respectively, the *a priori* and *a posteriori* error covariance matrices associated to the measurement $\boldsymbol{\zeta}_j(k)$, $j \in \Sigma(k)$, and $\mathbb{1}_{\mathcal{X}_i(k)}(\cdot)$ is the indicator function of the set $\mathcal{X}_i(k)$, $\mathbb{1}_{\mathcal{X}_i(k)} : \mathbb{R}^m \to \{0, 1\}$, defined as

$$\mathbb{1}_{\mathcal{X}_i(k)}(\mathbf{x}(k)) \triangleq \begin{cases} 1 & \text{if} \quad \mathbf{x}(k) \in \mathcal{X}_i(k) \\ 0 & \text{if} \quad \mathbf{x}(k) \notin \mathcal{X}_i(k) \end{cases}.$$

Proof. The proof comes from the analysis of the steps of Algorithms 1 and 2. Let us focus on iteration k. Thanks to Theorem 3.4, we already proved that, at the start of Algorithm 1, all the nodes in the network own the state estimate and the covariance matrix of the node selected at the end of iteration $k-1$ of Algorithm 2, that is, $\overline{\mathbf{x}}_i(k-1) = \hat{\mathbf{x}}_{j^\star(k-1)}(k-1)$, $\overline{\mathbf{P}}_i(k-1) = \mathbf{P}_{j^\star(k-1)}(k-1)$, $\forall i \in \mathcal{I}$. Lines 1 and 2 of Algorithm 1 are common to all the network nodes, thus all the *a priori* state estimations and *a priori* error covariance matrices are identical, for any node $i \in \mathcal{I}$, and are given by

$$\begin{cases} \hat{\mathbf{x}}^-_i(k) = \mathbf{A}(k)\overline{\mathbf{x}}_i(k-1) \\ \mathbf{P}^-_i(k) = \mathbf{A}(k)\overline{\mathbf{P}}_i(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k) \end{cases}, \quad \forall i \in \mathcal{I}. \tag{3.13}$$

The computation of the *a posteriori* differs, on the other hand, between *sensing* and *predicting* nodes. Namely, for sensing nodes,

$$\begin{cases} [\mathbf{P}_i(k)]^{-1} = [\mathbf{P}^-_i(k)]^{-1} + \mathbf{H}^T_i(k)\mathbf{R}^{-1}_i(k)\mathbf{H}_i(k) \\ \hat{\mathbf{x}}_i(k) = \mathbf{P}_i(k)\{[\mathbf{P}^-_i(k)]^{-1}\hat{\mathbf{x}}^-_i(k) + \mathbf{H}^T_i(k)\mathbf{R}^{-1}_i(k)\boldsymbol{\zeta}_i(k)\} \end{cases}, \quad \forall i \in \Sigma(k),$$

is computed, while, the following holds for predicting nodes

$$\begin{cases} \mathbf{P}_i(k) = \mathbf{P}_i^-(k) \\ \hat{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i^-(k) \end{cases}, \quad \forall i \in \Lambda(k).$$

Each node possesses its own perception confidence value, so that, $\gamma_i(k)$, $\forall i \in \mathcal{I}$, is available to Algorithm 2. Theorem 3.4 proves that, in finite time, all the nodes store in $\overline{\mathbf{x}}_i(k)$ and in $\overline{\mathbf{P}}_i(k)$, in finite time, the state estimation and its corresponding error covariance matrix of a specific node, that is, node $j^\star(k) = \underset{j\in\mathcal{I}}{\operatorname{argmax}}\, \gamma_j(k)$. It can be easily verified that, if $j^\star(k)$ is a sensing node, that is, $\mathbf{x}(k) \in \mathcal{X}_{j^\star(k)}$,

$$\begin{cases} [\overline{\mathbf{P}}_i(k)]^{-1} = [\mathbf{P}_i^-(k)]^{-1} + [\mathbf{H}_{j^\star(k)}^T(k)\mathbf{R}_{j^\star(k)}^{-1}(k)\mathbf{H}_{j^\star(k)}(k)] \\ \overline{\mathbf{x}}_i(k) = \overline{\mathbf{P}}_i(k)\{[\mathbf{P}_i^-(k)]^{-1}\mathbf{x}_i^-(k) + \mathbf{H}_{j^\star(k)}^T(k)\mathbf{R}_{j^\star(k)}^{-1}(k)\boldsymbol{\zeta}_{j^\star(k)}(k)\} \end{cases}, \quad \forall i \in \mathcal{I}, \quad (3.14)$$

is obtained. On the other hand, if $j^\star(k)$ is a predicting node, that is, $\mathbf{x}(k) \notin \mathcal{X}_{j^\star(k)}$,

$$\begin{cases} \overline{\mathbf{P}}_i(k) = \mathbf{P}_i^-(k) \\ \overline{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i^-(k) \end{cases}, \quad \forall i \in \mathcal{I}, \quad (3.15)$$

holds. Noting that Eqs. (3.13)—(3.15) hold for all $i \in \mathcal{I}$, so that they can be made node-independent by dropping index i, that Eqs. (3.14)–(3.15) are mutually exclusive at any given time instant k, and that they can be unified thanks to the indicator function $\mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))$, the theorem is proved. □

3.3 Application of the Distributed Kalman filtering via Node Selection

In this section, we apply the DKNS to the *distributed tracking* of a maneuvering target performed by a network of heterogeneous sensors with limited sensing capability. To this aim, we specialize to the case of interest some of the equations concerning the sensing process. As it will be shown in the chapter, different sensing ranges characterize the heterogeneity of the sensor network. Due to the limited sensing capability, only a subset of nodes can sense the target during a given time interval, and some uncovered areas of the space may even exist. The aim of distributed target tracking is to estimate and track the target state in the environment in discrete-time, by using a distributed algorithm involving message-passing between one-hop nodes of a sensor network. We analyze the DKNS performance in the distributed target tracking scenario through numerical simulations and experimental trials.

3.3.1 Simulation Assessment

First, we formulate the details of the distributed tracking problem for the simulation setup, providing suitable models for the environment, the target, and the sensor nodes. We evaluate numerically the performance of the proposed approach, and we make a comparison between DKNS applied to target tracking, the Centralized Kalman Filter with multiple measurements (CKF), and two more target tracking algorithms based on distributed Kalman filtering [34] [33].

3.3.1.1 Environment, Target, and Network Model

The network and the target lay on a two-dimensional environment $E \triangleq [-L/2, L/2] \times [-L/2, L/2] \subset \mathbb{R}^2$, $L > 0$, that is a square field with side length L. An earth-fixed Cartesian coordinate system $\{\mathcal{W}\} = O - xy$ is used to locate points in E. The target describes a *trajectory* $\boldsymbol{\xi}(s) \in E$ where $\boldsymbol{\xi}(s)$ are the Cartesian coordinates of the target position at the continuous time $s \in \mathbb{R}^+$. The estimates of the target state are carried out in discrete-time, so that a sampling $\boldsymbol{\xi}(k) = [\xi^x(k)\ \xi^y(k)]^T \in E$, $k \in \mathbb{N}_0$, is performed by the network nodes. The discrete-time index k is associated to a constant sampling time ε. Moreover, we denote with the vector $\mathbf{v}(k) = [\mathrm{v}^x(k)\ \mathrm{v}^y(k)]^T \in \mathbb{R}^2$ the target velocity at the discrete-time k. Finally, the state vector to be estimated is defined as $\mathbf{x}(k) = [\xi^x(k)\ \mathrm{v}^x(k)\ \xi^y(k)\ \mathrm{v}^y(k)]^T$.

A network of n sensors is deployed in the environment E, so that each sensor is located at the position $\mathbf{p}_i = [p_i^x\ p_i^y]^T \in E$. Sensors are connected through an undirected communication graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, with node set $\mathcal{I} = \{1, \ldots, n\}$. The edge set $\mathcal{E}$ is determined by a communication radius, r_c, identical for each node, so that $\mathcal{E} = \{(i,j) \,|\, ||\mathbf{p}_i - \mathbf{p}_j|| \leq r_c\}$. This is also known as a r_c-disk communication graph [43]. Moreover, we assume that the graph $\mathcal{G}$ is connected and has diameter $\mathcal{D}$. Finally, the limited sensing capability of each sensor is expressed via a suitable *sensing radius,* r_{s_i}. A sensor can perform a direct measurement of the target state if and only if the target is located within its sensing range, so that the *sensing set* $\mathcal{X}_i$ of the generic node i is given by

$$\mathcal{X}_i = \left\{\mathbf{x} \in \mathbb{R}^4 \mid ||\boldsymbol{\xi} - \mathbf{p}_i|| \leq r_{s_i}, \forall\, \mathbf{v}\right\}, \quad \forall i \in \mathcal{I}. \tag{3.16}$$

Please note that the sensing radius, and consequently its related sensing set, are considered constant in time for ease of presentation. Nevertheless, as proved in Section 3.2, the approach can be extended to time-varying quantities. Figure 3.1 shows an abstract representation of the sensor network in the case of four nodes.

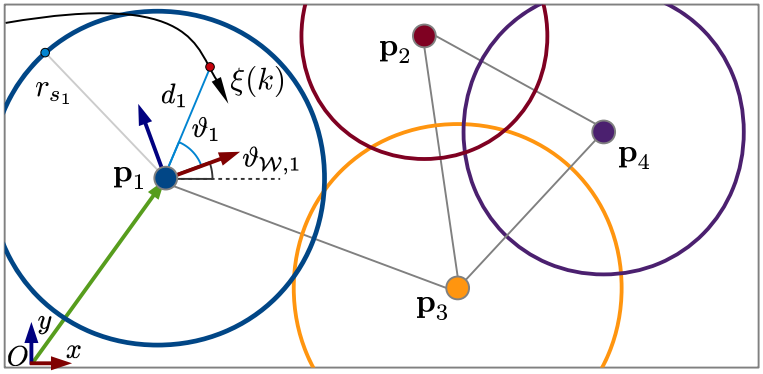


FIGURE 3.1: Abstract representation of the distributed target tracking scenario.

3.3.1.2 Sensor Network Node Model

For the simulation campaign, we consider range-bearing sensors with limited sensing capability. Range-bearing sensors are able to perform direct measurements of the target distance, $d_i(k)$, and bearing, $\vartheta_i(k)$, with respect to a sensor-fixed reference coordinate system. We assume that each sensor node is aware of its position $\mathbf{p}_i$ and orientation $\vartheta_{\mathcal{W},i}$, with respect to the earth-fixed coordinate frame, so that it is able to provide an estimate of the target position with respect to the latter frame through a simple coordinate transformation.

Range-bearing measurements are affected by noise, which we assume to be white Gaussian with zero mean and covariance matrix $\mathbf{S}_i(k)$. Assuming $\eta_i(k) = [d_i(k)\ \vartheta_i(k)]^T$, the related covariance matrix is given by [42]:

$$\mathbf{S}_i(k) = E[\eta_i(k)\eta_i(k)^T] = \begin{bmatrix} \sigma^2_{d_i}(k) & 0 \\ 0 & d_i^2(k)\sigma^2_{\vartheta_i}(k) \end{bmatrix}.$$

In range-bearing measurements, the noise grows with the distance of the target from the sensor. Therefore, to consider a realistic sensor model, a non constant variance is used. In [44], it is shown that for range-bearing sensors the variance has an almost constant trend at small distance, whereas it increases exponentially when approaching the maximum measurable value. We refer to the experimental data reported in [44], and select the interpolation functions

$$\sigma^2_{d_i}(k) = k_d \left(1 + e^{k_r\left[(d_i(k) - r_{s_i})/r_{s_i}\right]}\right), \quad d_i(k) \leq r_{s_i}, \tag{3.17}$$

and

$$\sigma^2_{\vartheta_i}(k) = k_\vartheta \frac{d_i(k)}{r_{s_i}}, \qquad d_i(k) \leq r_{s_i}, \quad k_\vartheta > 0, \tag{3.18}$$

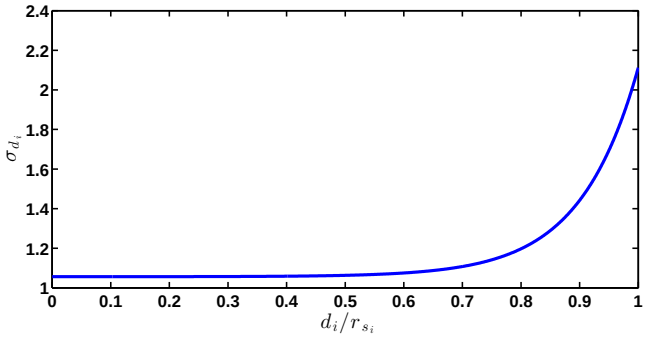


FIGURE 3.2: Trend of the standard deviation of the measurement noise σ_{d_i}, for a generic node i, versus the distance from the target, as modeled by Eq. (3.17). The values on the abscissa are normalized with respect to the value of the sensing range (r_{s_i}).

whose trends are suitable to fit the provided experimental data. It is clear that the values of $\sigma_{d_i}(k)$ and $\sigma_{\vartheta_i}(k)$ are not defined for $d_i(k) > r_{s_i}$. Figure 3.2 shows the trend of $\sigma_{d_i}(k)$, as expressed in Eq. (3.17).

According to the range-bearing model [42], we assume that a sensing node (*i.e.*, $i \in \Sigma(k)$) is able to compute the position of the target with respect to the earth-fixed Cartesian coordinate frame, as

$$\boldsymbol{\zeta}_i(k) = [p_i^x + d_i(k)\cos(\vartheta_{\mathcal{W},i} + \vartheta_i(k)),\ p_i^y + d_i(k)\sin(\vartheta_{\mathcal{W},i} + \vartheta_i(k))]^T. \tag{3.19}$$

Keeping in mind that each node runs a local Kalman filter, and assuming that sensing nodes cannot measure directly the target velocity, we formalize the observation equation (3.19) as

$$\boldsymbol{\zeta}_i(k) = \mathbf{H}\mathbf{x}(k) + \boldsymbol{\nu}_i(k), \tag{3.20}$$

where $\mathbf{H}$ is the measurement output matrix, defined as:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

and $\boldsymbol{\nu}_i(k) \in \mathbb{R}^2$ is the measurement noise, for which each component is assumed to be white Gaussian with zero mean and covariance matrix $\mathbf{R}_i(k)$, which is approximated by [42] (for higher readability, we drop the time dependence only in the following expression)

$$\mathbf{R}_i(k) \simeq \begin{bmatrix} \sigma_{d_i}^2 \sin^2(\vartheta_i) + d_i^2\sigma_{\vartheta_i}^2\cos(\vartheta_i) & c_{d\vartheta}\sin(\vartheta_i)\cos(\vartheta_i) \\ c_{d\vartheta}\sin(\vartheta_i)\cos(\vartheta_i) & \sigma_{d_i}^2\cos^2(\vartheta_i) + d_i^2\sigma_{\vartheta_i}^2\sin(\vartheta_i) \end{bmatrix},$$

where $c_{d\vartheta}(k) = \sigma^2_{d_i}(k) - d_i^2(k)\sigma^2_{\vartheta_i}(k)$. The approximation arises from the assumption that $\sigma_{d_i}(k) \ll d_i(k)$ and $\sigma_{\vartheta_i}(k) \ll 1$.

The dynamical model of the target motion, embedded in each node's Kalman filter, is defined as:

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \boldsymbol{\upsilon}(k), \tag{3.21}$$

where

$$\mathbf{A} = \mathbf{I}_2 \otimes \begin{bmatrix} 1 & \varepsilon \\ 0 & 1 \end{bmatrix},$$

ε is the time step, $\otimes$ is the Kronecker product of matrices, and $\boldsymbol{\upsilon}(k) \in \mathbb{R}^4$,

$$\boldsymbol{\upsilon}(k) = \begin{bmatrix} \upsilon_{\xi^x}(k) & \upsilon_{\mathrm{v}^x}(k) & \upsilon_{\xi^x}(k) & \upsilon_{\mathrm{v}^y}(k) \end{bmatrix}^T,$$

is a noise term that is white Gaussian with zero mean and covariance matrix $\mathbf{Q}$, *i.e.,* $\boldsymbol{\upsilon}(k) \sim N(\mathbf{0}, \mathbf{Q})$. Here, we assume

$$\upsilon_{\xi^x}(k) = \tfrac{\varepsilon}{2}\upsilon_{\mathrm{v}^x}(k), \quad \upsilon_{\xi^y}(k) = \tfrac{\varepsilon}{2}\upsilon_{\mathrm{v}^y}(k), \tag{3.22}$$

to take into account the influence of the maneuvering target on the positional coordinates [42]. Thus, matrix $\mathbf{Q}$, is given by

$$\mathbf{Q} = \begin{bmatrix} \frac{\varepsilon^2}{4}\sigma^2_{\upsilon_{\mathrm{v}^x}} & \frac{\varepsilon}{2}\sigma^2_{\upsilon_{\mathrm{v}^x}} & 0 & 0 \\ \frac{\varepsilon}{2}\sigma^2_{\upsilon_{\mathrm{v}^x}} & \sigma^2_{\upsilon_{\mathrm{v}^x}} & 0 & 0 \\ 0 & 0 & \frac{\varepsilon^2}{4}\sigma^2_{\upsilon_{\mathrm{v}^y}} & \frac{\varepsilon}{2}\sigma^2_{\upsilon_{\mathrm{v}^y}} \\ 0 & 0 & \frac{\varepsilon}{2}\sigma^2_{\upsilon_{\mathrm{v}^y}} & \sigma^2_{\upsilon_{\mathrm{v}^y}} \end{bmatrix}, \tag{3.23}$$

where $\sigma^2_{\upsilon_{\mathrm{v}^x}}$ and $\sigma^2_{\upsilon_{\mathrm{v}^y}}$ are the variances of the stochastic processes $\upsilon_{\mathrm{v}^x}(k)$ and $\upsilon_{\mathrm{v}^y}(k)$, respectively.

3.3.1.3 Tracking Setup

The performance of the DKNS algorithm is analyzed by running a campaign of Monte Carlo simulations by means of the Multi Agent Simulation Framework [45]. Each set of experiments consists of the tracking of 100 repeated random target trajectories. With the aim of comparing our approach with KCF and DKF [34] [33], we simulate a sensor network tracking the position of a maneuvering target moving inside a square field E with side length $L = 90$ and with a communication radius set as $r_c = 3\frac{L}{\lceil\sqrt{n}\rceil+1} + 2$. The nodes are placed at random positions in the environment, yet preserving connectedness of the network. In order to evaluate more in depth the performance of the DKNS, we define the percentage sensing coverage ratio ρ, that is the ratio between the sensing area

covered by all nodes, and the total area of the field E,

$$\rho = \frac{\text{area}(\bigcup_{i=1}^{n} \mathcal{X}_i)}{\text{area}(E)} \cdot 100\,, \tag{3.24}$$

where $\text{area}(E) = L^2$, is the area of the environment, and, with a little abuse of notation, $\text{area}(\cup_{i=1}^{n}\mathcal{X}_i)$ is the area of the union of all $\mathcal{X}_i$ sensing sets. Given the random distribution of the positions of the sensor nodes in the environment, we achieve a desired coverage ratio ρ by setting at first the sensing radius r_{s_i}, for each node $i \in \mathcal{I}$, to a suitable value r_ρ, computed to guarantee the coverage ratio ρ. A suitable algorithm to find r_ρ is adopted, which takes into account the existence of overlapping sensing areas. Then, heterogeneity in the sensor network is achieved by adding to the sensing radii r_{s_i}, for each node $i \in \mathcal{I}$, a Gaussian noise with zero mean and standard deviation $\sigma_\rho = 0.03 r_\rho$. The initial guess of the estimated target state is $\overline{\mathbf{x}}_i(0) = \mathbf{0}$, $\forall i \in \mathcal{I}$. Moreover, the standard deviation $\sigma_{\upsilon_{\mathrm{v}}x}$ and $\sigma_{\upsilon_{\mathrm{v}}y}$ of the process noise covariance matrix of the Kalman filter in Eq. (3.23), are chosen as $\sigma_{\upsilon_{\mathrm{v}}x} = \sigma_{\upsilon_{\mathrm{v}}y} = 3$, $\forall i \in \mathcal{I}$. The standard deviations related to the range-bearing sensing process, $\sigma_{d_i}(k)$ and $\sigma_{\vartheta_i}(k)$, are modeled by Eqs. (3.17-3.18), setting $k_d = 1.056$, $k_r = 10.07$ and $k_\vartheta = 0.1$, in order to fit the experimental dataset in [44]. Finally, again for comparison purposes we set $\mathbf{P}(0) = 10\sigma_0^2\mathbf{I}_4$, where $\sigma_0 = 5$.

Among the several aspects regarding performance assessment in target tracking applications [22], in this work we will focus on *tracking accuracy,* by evaluating the mean square error between estimated and actual target trajectory. As a metric for target tracking accuracy, the following mean square error (in norm) is computed:

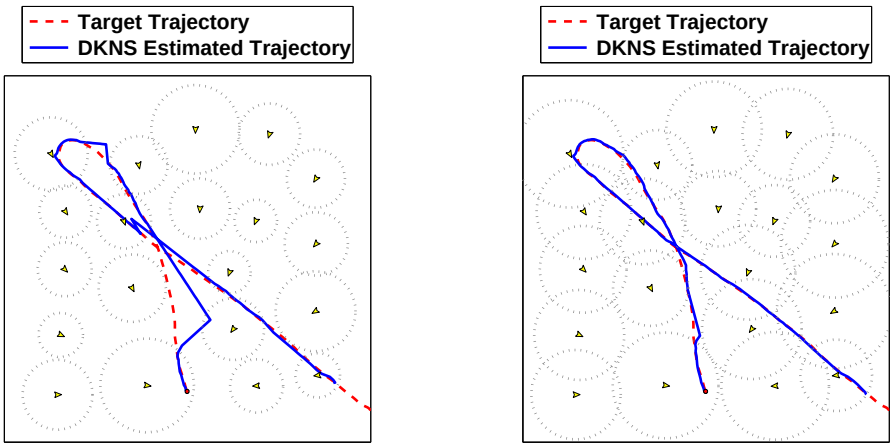
$$\alpha = \frac{1}{k_f} \sum_{k=1}^{k_f} \|\mathbf{H}\ \overline{\mathbf{x}}_i(k) - \boldsymbol{\xi}(k)\|^2 \tag{3.25}$$

where k_f is the duration (in time samples) of the target trajectory, $\boldsymbol{\xi}(k)$ is the actual target position at discrete-time k, $\mathbf{H}\ \overline{\mathbf{x}}_i(k)$ is any of the global target position estimates $\overline{\mathbf{x}}_i(k)$ at time k. We remind that, as proved in Theorem 3.4, the global target position estimates are identical for each network node. Obviously, the ideal condition of perfect tracking is $\alpha = 0$, and the tracking accuracy is better as long as α is smaller.

Furthermore, we define a third index, φ, which is the average percentage of sensing nodes during a single run. It is defined as

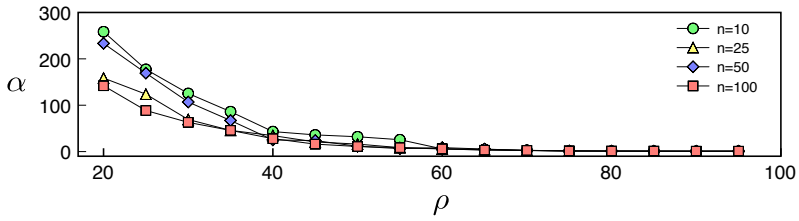
$$\varphi = \frac{1}{k_f} \sum_{k=1}^{k_f} \frac{|\Sigma(k)|}{n} \cdot 100\,, \tag{3.26}$$

where $|\cdot|$ is the set cardinality operator.

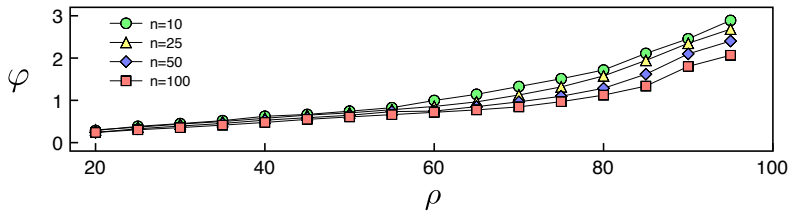


(A) (B)

FIGURE 3.3: DKNS simulations of a random generated target trajectory for a network of $n = 20$ heterogeneous nodes. The target comes from the bottom right corner of the environment, and the filled circle indicates its last position. (A) A simulation of the target tracking with $\rho = 45\%$. (B) The tracking of the same target trajectory with $\rho = 78\%$. In the latter case, due to the increasing coverage ratio ρ, the tracking accuracy α is higher than in the former.



(A)



(B)

FIGURE 3.4: Performance evaluation of DKNS algorithm. Values of α (A) and φ (B) as function of the coverage ratio ρ for networks with $n = 10$, $n = 25$, $n = 50$, and $n = 100$ nodes. Each point is computed by averaging 100 random trajectories.

3.3.1.4 DKNS Performance Evaluation

Two single experiments of the aforementioned simulation campaign are illustrated in Figure 3.3. In particular, the same trajectory is estimated via the DKNS algorithm by a network of $n = 20$ nodes, under two different coverage ratio conditions, and the estimated trajectory performed by the sensor network is plotted against the one of the target. It is important to note that, keeping constant the number of sensor nodes n, the performance improves as long as the coverage ratio ρ increases. As an example, in Figure 3.3(a), a tracking accuracy $\alpha = 42.81$ is obtained with a coverage ratio $\rho = 45\%$, while in Figure 3.3(b) a tracking accuracy $\alpha = 2.47$ is obtained with a coverage percentage of $\rho = 78\%$.

It can be noted from Figure 3.3, that when the target is not sensed by any node, the estimate is performed only through the linear model defined in Eq. (3.9). Then, when the target comes back into the sensing set of any sensor, the estimate is performed by exploiting the whole Kalman filter, thus improving the corresponding estimate.

An extensive evaluation of the performance of DKNS, made on the entire simulation campaign, is illustrated in Figure 3.4. In Figure 3.4(a), the tracking accuracy α is plotted versus the coverage ratio ρ, for different values of the number of nodes n. As can be noted, the tracking accuracy improves as long as the coverage ratio ρ increases. Furthermore, it can be observed that the performance tends to be independent from the number of nodes when the coverage ratio ρ grows (in Figure 3.4(a), the performance is practically independent from the number of sensors for $\rho > 60\%$). It can also be noted that the same tracking accuracy α can be obtained for different pairs of the number of nodes n and of the coverage ratio ρ. For example, the performance index $\alpha = 100$ is achieved with $(n = 10, \rho = 33.82\%)$, $(n = 25, \rho = 27.52\%)$, $(n = 50, \rho = 30.73\%)$, $(n = 100, \rho = 24.12\%)$. From the simulation results, it is evident that to guarantee a desired accuracy, the number of network nodes and the coverage ratio are of fundamental importance. Nevertheless, one can make up for a small number of sensor nodes via the setting of a suitable coverage ratio, obtained both an effective spatial deployment, and by setting large enough sensing radii. The last consideration can help in overcoming, in real-time applications, the growth of the convergence time of the max-consensus algorithm, which increases with the diameter $\mathcal{D}$ of the network, that can be influenced by the number of nodes n (See Section 3.2.2).

Figure 3.4(b) illustrates the average number (in percentage) of sensing nodes φ during a run of the algorithm, versus the coverage ratio ρ, for different values of the number of nodes n.

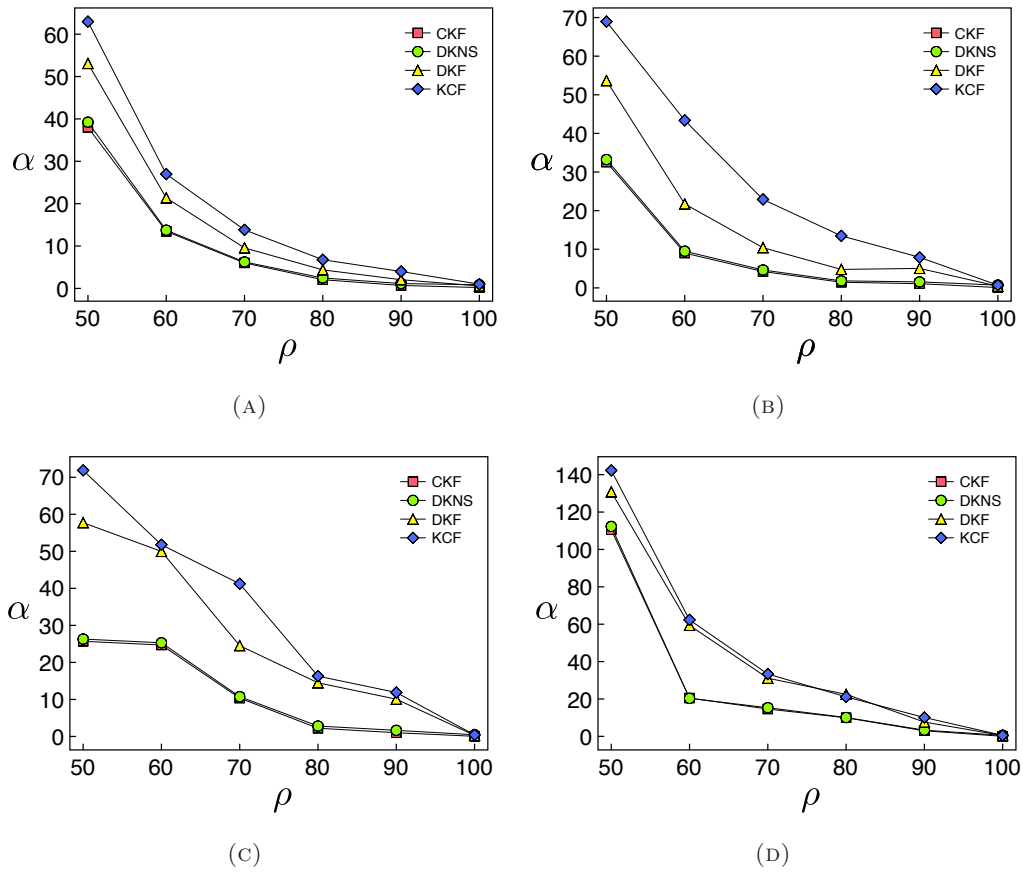
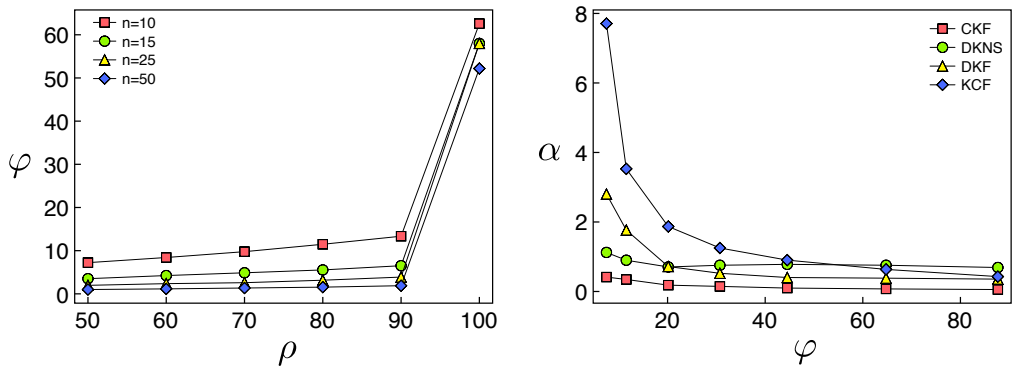


FIGURE 3.5: Comparison between DKNS and three other tracking algorithms: CKF, KCF, and DKF. Values of α as function of the coverage ratio ρ for networks with $n = 10$ (A), $n = 15$ (B), $n = 25$ (C), and $n = 50$ nodes (D). Each point is computed by averaging 100 random trajectories.

3.3.1.5 Comparison with other Target Tracking Algorithms based on Distributed Kalman Filtering

In this section, we compare the performance of the DKNS algorithm with a centralized and two distributed target tracking approaches based on Kalman filtering. The first algorithm is the Centralized Kalman filter with multiple measurements (CKF) [32], which is known to be optimal in the sense of the minimization of the error variance, thus exhibiting the best performance. The second algorithm is the Kalman Consensus Filter (KCF) with message passing (Algorithm 3 in [33]), and the third one is the Diffusion Kalman Filter (DKF), presented by [34].

The comparison is performed through 100 repeated random trajectories for different values of the coverage ratio ρ, and of the number of sensor nodes, n. More specifically, the average value of α over the 100 test trajectories has been computed for each value of ρ, and compared with the corresponding performance parameter obtained through the CKF, KCF, and DKF algorithms, run on the same test trajectories. Figure 3.5



(A) (B)

FIGURE 3.6: (a) Values of φ as function of the coverage ratio ρ for networks with $n = 10, 15, 25, 50$ nodes. (b) Values of α as function of the index φ for a network of $n = 15$ nodes.

shows that the DKNS algorithm generally outperforms KCF and DKF and, for a given value of the coverage ratio ρ, the gap in performance generally increases as long as the number of network nodes increases. Nevertheless, the gap in performance decreases as long as the coverage ratio ρ increases. It is also important to note that DKNS closely approaches the optimal performance of the CKF, as long as the number of network nodes increases. The explanation of this behavior resides in the trend of the average percentage of sensing nodes φ, illustrated in Figure 3.6(a). In fact, keeping constant the coverage ratio ρ, and increasing the number of network nodes, yields a decrease of the average number of sensing nodes φ. Hence, it can be noted that the lower is φ, the better is the approximation of the centralized tracking exhibited by DKNS. This is well explained by considering that the one-term approximation of Eq. (3.7), expressed in Eq. (3.11), is better as long as the number of sensing nodes is smaller. This statement is supported by the numerical results illustrated in Figure 3.6(b), where it can be observed that the performance of DKNS is quite independent from the value of φ, which is not the case when KCF and DKF are considered. Moreover, it is evident that the application of DKNS is not the best option in any case: in fact, when the average number of sensing nodes φ grows, KCF and DKF tend to perform better than DKNS. In our simulations, this happens for $\varphi > 20\%$ for DKF, and for $\varphi > 57\%$ for KCF.

3.3.2 Experimental Evaluation

In this section, we test the DKNS for target tracking purposes in a real world implementation. The tracking setup is part of the Distributed Ambient Assisted Living (DAAL) framework developed at ISSIA-CNR [17]. The DAAL system is a heterogeneous sensor network for distributed monitoring of people in an indoor environment. It is composed

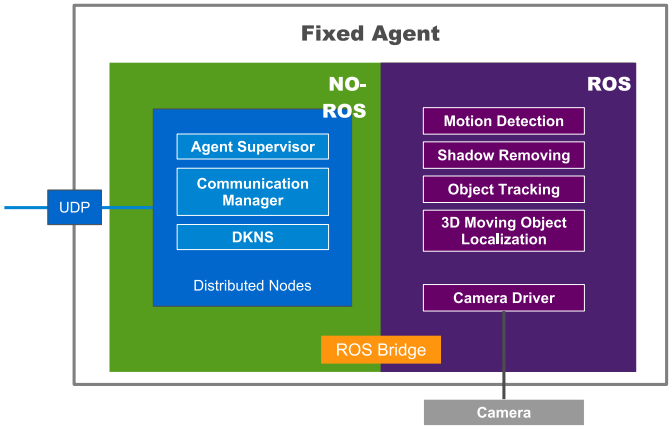


FIGURE 3.7: Structure of a Fixed Agent: ROS and NO-ROS environments with distributed and local modules.

by a network of *fixed cameras* and *mobile robots*, *i.e.*, fixed and mobile agents[3]. Integration among the various agents is performed via a distributed control architecture which uses a wireless network as a communication channel. The people tracking capability of the DAAL is implemented using the DKNS algorithm. Before analyzing the DKNS performance in a real-world scenario, we briefly introduce the components of the DAAL system.

3.3.2.1 DAAL fixed agent

The fixed agent's functionalities run on a workstation linked to the camera by a cabled connection. The schematic representation of the interconnections among the components of a fixed agent module is shown in Fig. 3.7. For each camera several threads integrated in the Robot Operating System (ROS) [4] framework are implemented.

The local ROS nodes constitute the perception part of the agent. Each agent is connected with a camera and the image flow is captured by the ROS driver, then the perception pipeline is implemented, including modules such as [21]: (a) *Motion Detection*, the binary shape of moving objects (e.g., people) is extracted; (b) *Shadow Removal*, the shadow pixels need to be removed, as they alter the real shape of objects and decrease the precision of their localization; (c) *Object Tracking*, the detected moving objects, after shadow removal, are tracked over time. Statistical (tracked object life time) and spatial information are extracted for each of them; (d) *3D Moving Object Localization*, the intersection of the central axis of the rectangular bounding box containing the moving region with its lower side provides the estimate of object position on the ground plane.

[3]Hereinafter, in this section, the network nodes will be called agents in order to avoid ambiguities in referring to ROS nodes.

[4]http://www.ros.org

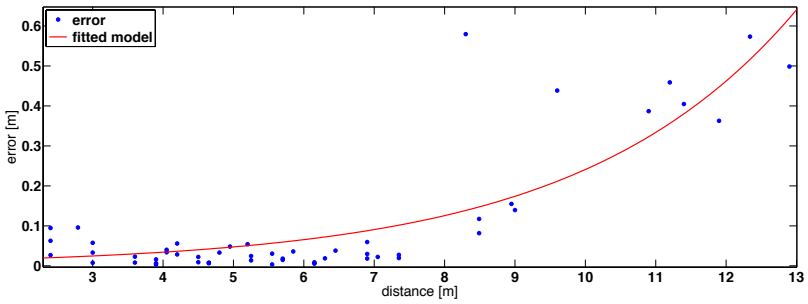


FIGURE 3.8: The measurement error model for one of the cameras: $f(x) = a\,e^{bx}$. The error is limited when the sensor-target distance is under 7-8 m, and, above that, the error increases. This suggests that the camera should be deployed ensuring a maximum distance to the object under 7-8 m

The corresponding 3D position is evaluated using a pre-calibrated homographic matrix between the image plane and the 3D ground plane. On the other hand, the distributed NO-ROS nodes are enabled to perform three different functions: to control the agent operations (Agent Supervisor), to manage the communication (Communication Manager), and to execute distributed algorithms, such as the DKNS.

As explained in previous sections, the DKNS requires a model of the measurement error of each sensor composing the network. To this aim, the measurement error of the cameras has been characterized empirically noting that it depends on the distance of the target relative to the sensor. We characterize the error model by fitting a series of measurement errors obtained by the comparison of the position measured by the camera and the real position of the target (the real position of the target is retrieved by means of a theodolite). Figure 3.8 shows the model fitted as an exponential function for one of the cameras:

$$f(x) = a\,e^{bx}, \tag{3.27}$$

where $a = 0.009269 \pm 0.0074$ and $b = 0.3258 \pm 0.0696$ are the value of the coefficients (with 95% confidence bounds) defining the actual function $f(x)$.

3.3.2.2 DAAL mobile agent

The mobile agents, *i.e.* mobile robots, are equipped with sensory devices to interact with the environment. Every mobile agent is able to localize itself in the environment and to safely navigate avoiding static and dynamic obstacles. It is also able to identify and track the position of a target in the environment.

A schematic representation of a mobile agent is shown in Fig. 3.9. ROS has been adopted for sensor acquisition and processing, and for the navigation control of the mobile robot. The navigation functionalities are provided through the Navigation Stack, a standard

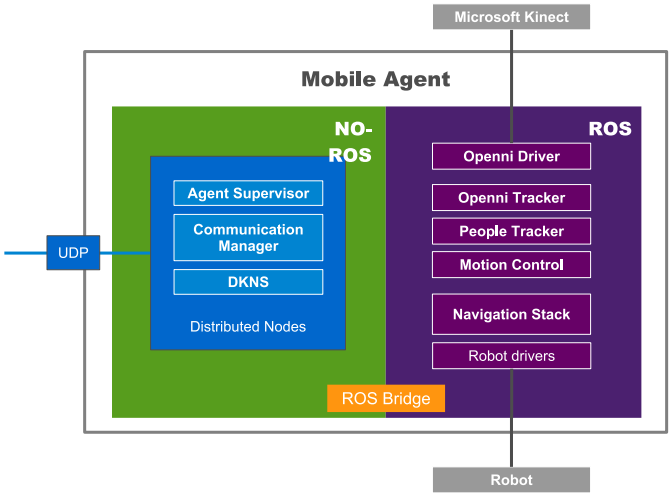


FIGURE 3.9: Structure of a Mobile Agent: ROS and NO-ROS environments with distributed and local modules.

ROS metapackage [5]. The Navigation Stack is directly connected with the robot sensors and motor drivers (robot drivers) and it enables the robot to navigate in a known environment avoiding obstacles. Moreover, the navigation stack was integrated with a novel motion controller, which takes into account motion constraints given by the people tracking and following tasks. The mobile agents are equipped with an RGB-D sensor, namely the Microsoft Kinect camera, to detect people in the environment. The Kinect sensor produces a 3D data representation, which allows the positions of a group of people in the environment and their movements to be detected. In the OpenNI Tracker, people are identified in the scene captured by the Kinect camera onboard the robot, and then a single person of interest is selected (*e.g.*, the person closest to the robot). Then, a tracking algorithm keeps track of the position of that person (People Tracker) and a control algorithm (Motion Control) allows the robot to move, avoiding obstacles in the environment, toward the person in order to improve the tracking performance. We model the measurement error of the Kinect sensor according to [46].

The distributed NO-ROS nodes, as for the fixed agent, provide the control of the whole agent (Agent Supervisor), the management of the communication (Communication Manager), and the execution of the DKNS distributed target tracking algorithm, which, in this case, uses the position of the person extracted form the ROS–based perception modules.

[5] http://wiki.ros.org/navigation

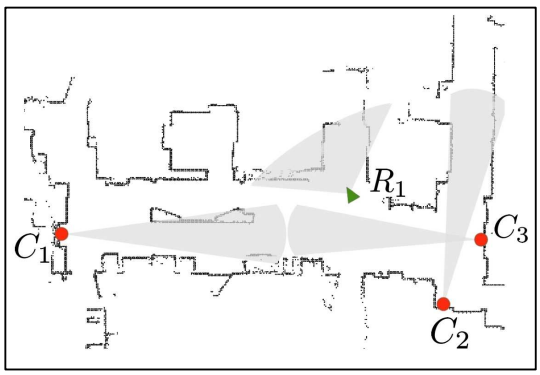


FIGURE 3.10: Map of one corridor of the office with overlaid the position of three static cameras (red circles) and one mobile Node (green triangle).

3.3.2.3 Target tracking experimental tests

We test the DKNS performance during the DAAL system validation which has been conducted in an ambient assisted living scenario.

The environment setup used for the experimentation of the system is shown in Fig. 3.10. The picture shows the map of a corridor of the ISSIA-CNR building, as it is built by the *gmapping* node, available as a ROS tool, using the laser data acquired by a mobile robot during a complete exploration of the environment. In this experimentation, three fixed cameras and one mobile robot have been employed. The positions of the fixed cameras (C_1, C_2, C_3) and of the mobile robot (R_1) are overlaid on the map. The mobile agent is able to localize itself in the environment and, using its on-board sensors, it is able to carry out surveillance tasks, such as people detection and tracking. Cameras are calibrated, therefore events detected in the image plane can be located in the real world and their positions can be communicated to the mobile agent. The mobile robot can explore areas that are unobservable by the fixed cameras and improve the accuracy in detecting events by reaching proper positions in the environment.

The fixed agents are three wireless IP cameras (C_1, C_2, C_3) with different spatial resolution, located in different points of the environment (see map in Fig. 3.10). C_2 and C_3 are Axis IP color cameras with a 640×480 pixel resolution and an acquisition frame rate of 10 frames per second. C_1 is a Mpixel Axis IP color camera with 1280×1024 pixel resolution and full frame acquisition rate of 8 frames per second (see Fig. 3.11, on the right). A calibration step to estimate intrinsic and extrinsic parameters was performed for each camera using the Matlab Calibration Toolbox[6], so that camera coordinates can be mapped to the global world reference frame provided by the map built by the mobile robot.

[6]The toolbox is available on
http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

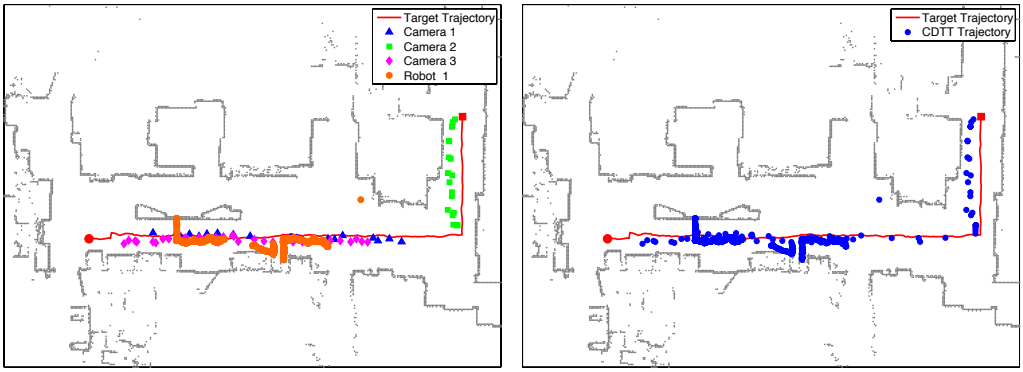
FIGURE 3.11: The agents of the network. On the left, the mobile agent PeopleBot. The robot is equipped with a laser range-finder SICK LMS200 and a Kinect. On the right, two different AXIS cameras: on the top, a Mpixel Axis IP color camera with 1280×1024. On the bottom, an Axis IP color cameras with a 640×480 pixel camera.

The mobile agent (denoted as R_1 in Fig. 3.10) consists of a PeopleBot mobile robot platform equipped with a laser range-finder, a Kinect, and an on-board laptop (see Fig. 3.11, on the left). The SICK laser is connected with the embedded robot control unit. The Kinect camera and the PeopleBot control unit are connected with the laptop, via a USB cable and a crossover cable, respectively. The laser range-finder is used to build a map of the environment and to localize the vehicle. The Kinect, whose field of view is 58 degrees horizontal, 45 degrees vertical, 70 degrees diagonal, and the operational range is between 0.8 meters (2.6 ft) and 3.5 meters (11 ft) [47], is used for both navigation (e.g., obstacle avoidance) and high-level tasks such as people detection and tracking.

The target to be tracked is a person moving in the environment. In the experimentation, the target follows different trajectories in the environment. It is interesting to visually quantify the tracking error, to this aim, in Figs. 3.12–3.13 we show two examples of those trajectories. Specifically, in Fig. 3.12(a) and Fig. 3.13(a), the target trajectory is denoted by a red line, while the target positions, as estimated by the network, are denoted by different markers. In Fig. 3.12(b) and Fig. 3.13(b) the target trajectory (red

Case	α [m]	σ_α^2 [m^2]
Trajectory 1 (Fig. 3.12)	1.15	0.86
Trajectory 2 (Fig. 3.13)	0.75	0.16

TABLE 3.1: Values of α and σ_α^2 in the tracking of a person moving in the laboratory by means of a network of 4 agents, 3 fixed and 1 mobile.



(A) (B)

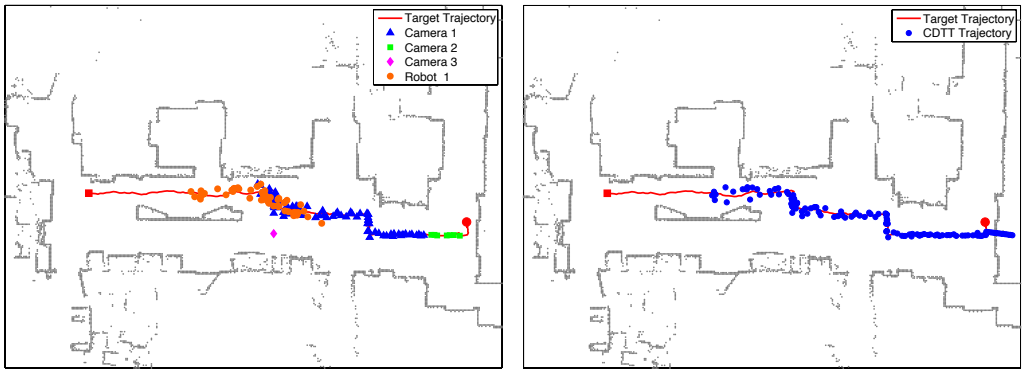
FIGURE 3.12: Trajectory 1. The measurement of the position of the target carried out by each of the sensor of the network (A) and the DKNS trajectory recovered on line and in distributed fashion by the network (B).

line) is compared with the trajectory (blue dots) as estimated by the DKNS algorithm. It should be noted that the estimated target position is the same for any agent of the network, since after convergence of the consensus step of the DKNS algorithm all the network agents share the same information about the target location. Furthermore, it should be noted how the Kalman filtering gives a prediction of the position of the target also when no sensor is measuring the target position.

In order to quantify the tracking performance, we suppose that the target is moving with a constant velocity and we calculate the mean square error, *i.e.*, the value of the index α, as done for the simulated case, together with its variance σ_α^2. Results are collected in Table 3.1, showing a mean square error of 1.15 m and 0.75 m, for Trajectory 1 and Trajectory 2, respectively. Figure 3.14 shows two frames, acquired from the Kinect camera on the robot during the tracking of the Trajectory 1 depicted in Fig. 3.12.

3.4 Conclusions

In this chapter, we have addressed the problem of distributed Kalman filtering over heterogeneous sensor networks, by introducing a novel approach, called Distributed Kalman filtering via Node Selection (DKNS). This is proved to be equivalent to a centralized Kalman filter with multiple measurements, which evolves according to a state-dependent



(A) (B)

FIGURE 3.13: Trajectory 2. The measurement of the position of the target carried out by each of the sensor of the network (A) and the DKNS trajectory recovered on line and in distributed fashion by the network (B).

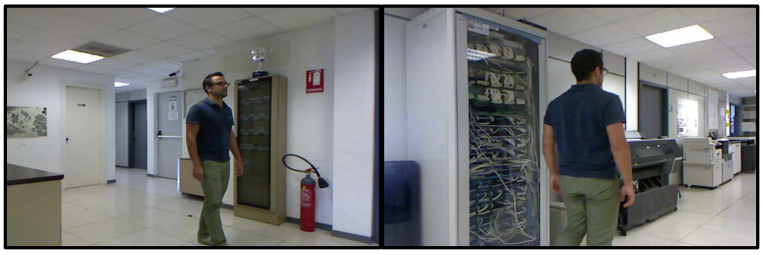


FIGURE 3.14: Two different instants of the tracking of Trajectory 1, acquired from the Kinect sensor.

switching dynamics, able to select and propagate the best estimate of the process state through the sensor network in finite time. The optimality of the estimate accuracy is assessed through the definition of a metric, called perception confidence value, which is strictly related to the Fisher information.

DKNS is particularly suitable in case of sensor networks with limited sensing capability, that is, in those cases in which only a few sensors in the network can actually perform a direct measurement of the process state. In these cases, in fact, fusing information may often lead to poor results, whereas node selection may constitute a better option. Conversely, when many reliable measurements are available, information fusion may result in better performance than node selection.

We have applied the algorithm to the discrete-time tracking of a maneuvering target, performed by a network of heterogeneous range-bearing sensors with limited sensing capability, achieving very satisfactory results. From the simulation campaign, the performance comparison with existing algorithms based on sensor fusion reflects what mentioned above in general terms, that is, DKNS is a viable and more effective option as long as limitations on the sensing capability are present. Finally, experimental tests

obtained using real sensors in a laboratory environment have shown the feasibility of the proposed algorithm in distributed target tracking scenarios.

Chapter 4

Distributed Estimation of Inertial Parameters for Cooperative Manipulation Tasks

In this chapter, we propose a distributed strategy for the estimation of the kinematic and inertial parameters of an unknown body manipulated by a team of mobile robots. This constitutes the first fundamental step toward the definition of advanced distributed control algorithms for multi-robot manipulation tasks. We assume that each robot can measure its own velocity, as well as the contact forces exerted during the body manipulation, but neither accelerations nor positions of the point of contacts between each robot and the manipulated body are directly accessible. Through kinematics and dynamics arguments, the relative positions of the contact points are estimated in a distributed fashion, also defining an observability condition. Then, the inertial parameters of the body (*i.e.*, mass, relative position of the center of mass and moment of inertia) are estimated using distributed estimation filters and a nonlinear observer in cooperation with suitable control actions that ensure the observability of the parameters. The convergence of the algorithm is proved through Lyapunov stability arguments. Finally, we provide numerical simulations, including also the study of the effect of measurement noise on the performance of the proposed approach, which corroborate our theoretical analysis.

The content of this chapter is grounded on two conference papers [48] [49].

4.1 Cooperative Manipulation

Cooperative manipulation by teams of mobile robots has been one of the hot research topics in the last decades. Many applications have been developed, especially in the fields of search and rescue [1], disaster recovering [2], cooperative transportation [3], and service robotics [50]. The objective of cooperative manipulation is to control the trajectory of a manipulated load toward a desired one with a given accuracy. At the same time, the forces exerted by the robot's end effectors should be maintained as close as possible to given reference values. The dynamics of each manipulator is usually described according to well-known dynamical models [51] [52], while the so called *Augmented Object Model* [53] is used to describe coupled dynamics of the object and the robotic system. It is important to stress that this model is based on the knowledge of the mass and the inertia of the object. Similarly, almost all of the work presented in this section is based on the assumption of the knowledge of the inertial parameters of the manipulated object, although this assumption is not always verified. In [54], a master-slave scheme is used for the manipulation of a single object. In order to lead the slave robots to follow the master, two different strategies are presented. The common feature to both strategies is that the object and the master manipulator are modeled as a unique rigid body. In the first strategy, both master and slaves use a position feedback. On the other hand, in the second strategy only the master is controlled in position, while slaves use a force-feedback control. In order to apply such control strategies, the masses of manipulators and objects must be known. Furthermore, the relative positions between the manipulators are assumed to be known. In [55], an impedance control of the manipulated object is presented. However, this strategy is based on a centralized estimation of the forces applied to the object. This is based on the knowledge of the grasp matrix [56], that is computed on the basis of the distance of the contact points relative to the center of mass of the object. Also in this case, it is assumed that the inertial parameters of the manipulated object are known. Furthermore, a distributed impedance control scheme is presented in [57], that is to say, each manipulator follows an impedance control approach where there is only an input which is the reference trajectory of the contact point. The input is derived from the desired object motion assuming the knowledge of the grasp geometry. In [3], a formation-based cooperative manipulation control is presented. The strategy is based on the knowledge of the grasp geometry and of the inertial parameters of the object. The problem is formulated as an optimal control problem which combines the classical quadratic cost function with a relaxed formation (rigidity) constraint in terms of an additional biquadratic penalty term. This relaxed rigidity constraint is justified by the use of an impedance control, by which minor deviations from the rigidity constraint result in tolerable object stress.

Path planning for cooperative manipulation is dealt with in [58], where a global path planner responsible for obstacle avoidance, and a local planner, used for manipulation tasks, are utilized. Virtual leader-follower-based control schemes are presented in [59]. Aerial applications have also been widely treated, interacting with the payload via cables or other tools [60] [61] [62]. Passivity theory and a decomposition technique are leveraged in [63] to design a control framework for cooperative planar manipulation without taking into account the payload model. In [64], the dynamical model of a non-holonomic wheeled robot team is joined with that of the payload, but its inertial parameters are supposed to be known, as well as the positions of the contact points between the robots and the object.

As seen so far, the knowledge of the inertial parameters is a prerequisite in order to be able to perform cooperative manipulation algorithms. Moreover, the design of on-line estimation techniques of inertial parameters of unknown loads makes collective manipulation tasks more effective and is beneficial in terms of reduction of the control effort. The benefits provided are at least twofold: first, effective control techniques for manipulation, like force control and pose estimation [65] [66], can be applied in order to achieve better performance with a reduced control effort. Second, manipulation of loads with time-varying characteristics can be achieved. For example, in transport application it is not rare that the payload is increased by an external cause, or that part of the load is lost during the transportation. An effective, real-time estimate of the inertial parameters would allow to implement adaptive control techniques, as well as event-driven control algorithms. We observe that, in order to benefit from the advantages of decentralized cooperative manipulation schemes, inertial parameter estimation must be designed and implemented in a totally distributed fashion [67].

Not much work has been done in the past concerning distributed estimation of inertial parameters. Moreover, the main limitations of the existing research reside in the centralization of the approaches, and in the use of acceleration and absolute positioning measurements. Notably, a strategy for the estimation of the inertial parameters of a graspless planar object is presented in [65], where robot fingers push the object and velocities, acceleration, and exchanged forces are measured and used to estimate the inertial parameters. However, it is to take into account that relying on acceleration measurements can amplify the effects of noise. An other example of estimation algorithm for inertial parameters is presented in [66], where an off-line centralized strategy for rigid loads attached to a single manipulator based on a total least-square approach is given. Clearly, such a strategy is far from being able to be used in contexts of cooperative manipulation in which distributed and real-time solutions are usually preferable.

The research presented in this chapter seeks to overcome the limitations of the existing approaches, by defining, to the best of our knowledge, the first totally distributed strategy for the estimation of the inertial parameters of an unknown load manipulated by a pool of Unmanned Ground Vehicles (UGVs). The proposed approach relies on the application of contact forces to points whose neither accelerations nor relative (and, consequently, absolute) positions are measured, and makes use of velocity measurements only. This chapter is grounded on the preliminary results presented in [48] [49], where a solution for the same estimation problem is given for noiseless [48], and noisy [49] measurements, respectively. The proposed strategy is grounded on geometrical and dynamical analysis, and leverages the theory of nonlinear observers to estimate the relative positions of the contact points, which are in turn used to derive the inertial parameters of the unknown payloads.

The chapter is structured as follows. In Section 4.2, we formalize the problem of the distributed estimation of the inertial parameters of an unknown load manipulated by a team of UGVs. In Section 4.3, we introduce the distributed estimation procedure in the noiseless case, while the noisy case is analyzed in Section 4.4. Simulation results are reported in Section 4.5, while our conclusions are drawn in Section 4.6.

4.2 Problem Statement

We consider a load modeled as a planar rigid body B, whose center of mass is denoted with C. The load is manipulated by a group of $n \in \mathbb{N}$ UGVs, where each UGV can exert a force $\mathbf{f}_i$ on a contact point C_i of B, where $i = 1, \ldots, n$. We assume that the number n is constant and known to all robots[1]. Consider a planar reference inertial frame $\{\mathcal{W}\} = O - xy$ and denote with $\mathbf{p}_C \in \mathbb{R}^2$ the position of C in $\{\mathcal{W}\}$, with $\mathbf{p}_{C_i}$ the position of C_i in $\{\mathcal{W}\}$ and with $\mathbf{f}_i$ the force applied by the i-th robot at C_i and expressed in $\{\mathcal{W}\}$. Refer to Figure 4.1 for a graphical representation of the problem setting. We also assume that friction on the load is negligible with respect to the forces exerted by robots[2].

[1]This assumption, that is the only source of centralized information needed, can be easily relaxed by implementing one of the several algorithms for the distributed estimation of a graph size [68] before the estimation algorithm starts.

[2]This can be ensured, e.g., by endowing the load with suitable wheels.

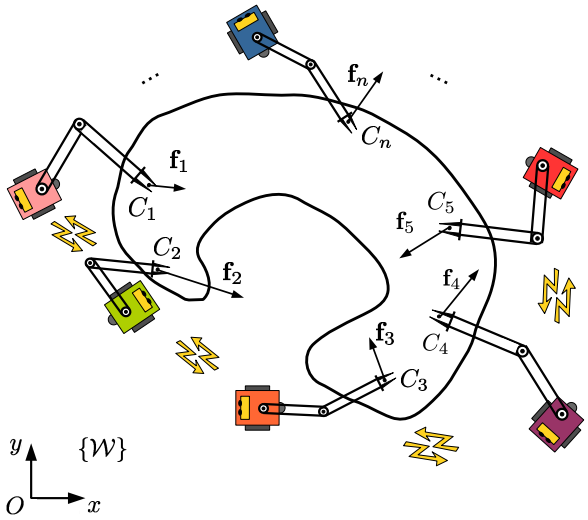


FIGURE 4.1: Set of n UGVs (six of them are shown) performing a transportation task. Each robot is able to exert a given force on the object by means of a planar manipulator. The objective of the network is the distributed estimation of the inertial parameters of the manipulated load.

Thus, the dynamical model of load B is that of a rigid body subject to n forces $\mathbf{f}_1, \ldots, \mathbf{f}_n$,

$$\ddot{\mathbf{p}}_C = \frac{1}{m} \sum_{i=1}^{n} \mathbf{f}_i \tag{4.1}$$

$$\dot{\omega} = \frac{1}{J} \sum_{i=1}^{n} (\mathbf{p}_{C_i} - \mathbf{p}_C)^{\perp T} \mathbf{f}_i, \tag{4.2}$$

where $m \in \mathbb{R}_+$ is the mass of B, $\omega \in \mathbb{R}$ is its angular velocity, $J \in \mathbb{R}_+$ is its moment of inertia, and the operator $(\cdot)^\perp$ is the linear operator that, given a generic vector $\boldsymbol{\epsilon} \in \mathbb{R}^2$, $\boldsymbol{\epsilon} = (\epsilon^x \ \epsilon^y)^T$, provides the perpendicular vector

$$\boldsymbol{\epsilon}^\perp = \boldsymbol{\Omega}\boldsymbol{\epsilon} = \underbrace{\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}}_{=\boldsymbol{\Omega}} \begin{pmatrix} \epsilon^x \\ \epsilon^y \end{pmatrix} = \begin{pmatrix} -\epsilon^y \\ \epsilon^x \end{pmatrix}. \tag{4.3}$$

We assume that robot i can control the exerted force $\mathbf{f}_i$ and can perform noisy measurements $\widetilde{\dot{\mathbf{p}}}_{C_i} \in \mathbb{R}^2$ of the velocity $\dot{\mathbf{p}}_{C_i}$ of the contact point C_i,

$$\widetilde{\dot{\mathbf{p}}}_{C_i} = \dot{\mathbf{p}}_{C_i} + \boldsymbol{\nu}_i, \tag{4.4}$$

where, $\boldsymbol{\nu}_i \in \mathbb{R}^2$, $\boldsymbol{\nu}_i \sim N(\mathbf{0}, \boldsymbol{\Sigma}_i)$ is a white gaussian noise vector with zero mean and covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{2\times 2}$. Note that we do not assume that the robot can measure

the position of the contact point, nor its acceleration[3]. Furthermore, we assume that the robot i cannot set the speed of the contact point at will.

The communication network modeled by an undirected graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, where $\mathcal{I} = \{1, \ldots, n\}$, is the node set, representing the robot set, $\mathcal{E} \subset \mathcal{I} \times \mathcal{I}$ is the edge set, representing unordered pairs of nodes through which one-hop communication channels are established. The set $\mathcal{N}_i = \{j \in \mathcal{I} : (i, j) \in \mathcal{E}\}$ represents the (communication) neighborhood of robot i. We also assume that the communication graph is connected, and that the link set $\mathcal{E}$ does not change in time.

Problem 4.1 (Inertial Parameters' Distributed Estimation). Given a network of n robots moving on a plane and manipulating a load B, with the characteristics described in Section4.2, design a distributed strategy such that each robot i is able to estimate

1. the mass m of B,
2. the moment of inertia J of B, and
3. the Grasp Matrix, which is function of the relative positions of the contact points with respect to the center of mass, that is, $\mathbf{p}_{C_i} - \mathbf{p}_C \in \mathbb{R}^2$. We observe that this quantity varies in time.

Each robot i can only

1. locally measure the velocity $\widetilde{\dot{\mathbf{p}}}_{C_i}$ of the contact point C_i;
2. locally control the applied force $\mathbf{f}_i$ acting on C_i;
3. communicate with its one-hop neighbors, contained in the set $\mathcal{N}_i$.

To focus on the effect of measurement noise on the performance of the estimation algorithm, we assume that control inputs and communications are not affected by noise.

4.3 Ideal Case: Noiseless Velocity Measurements

In this section, we analyze the ideal case where no measurement noise is present, that is to say, we assume that $\mathbf{\Sigma}_i = \mathbf{0}$, for all $i \in \mathcal{I}$. This assumption simplifies the problem and enables the study of the algorithm convergence. The noiseless assumption will be relaxed in the following section, to deal with a more realistic framework.

[3]Measuring absolute positions would require additional sensing systems such as GPSs. Measuring accelerations, on the other hand, is typically prone to noise. Velocity measurements can be instead reliably performed resorting to sensor systems usually present onboard, such as, odometry, optical flow, etc..

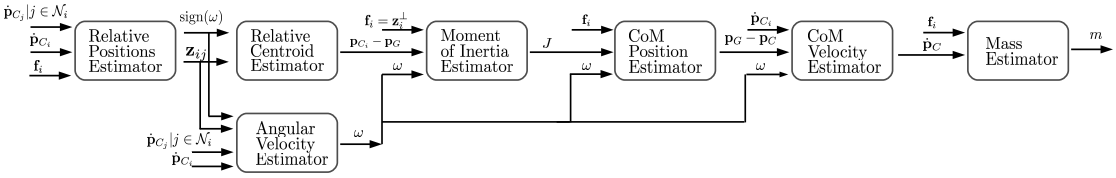


FIGURE 4.2: Block diagram showing the sequence of estimators applied to solve Problem 4.1.

4.3.1 Estimation Algorithm

Define $\mathbf{p}_G = \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_{C_i}$ as the geometric center of all the contact points and also define the relative positions $\mathbf{z}_{ij} = \mathbf{p}_{C_i} - \mathbf{p}_{C_j}$, $\mathbf{z}_i = \mathbf{p}_{C_i} - \mathbf{p}_G$, and $\mathbf{z}_C = \mathbf{p}_G - \mathbf{p}_C$. Note that $\mathbf{p}_{C_i} - \mathbf{p}_C = \mathbf{z}_i + \mathbf{z}_C$.

We first recall a simple fact that will turn useful later in our study.

Fact 4.1. Denote with $\mathbf{z}_1$ and $\mathbf{z}_2$ the relative positions of two pairs of points of B expressed in $\{\mathcal{W}\}$. Then, consider two time instants t' and t''. The following relation, based on the rigid-body constraint, provides a straightforward way to compute $\mathbf{z}_2(t'')$ from $\mathbf{z}_1(t')$, $\mathbf{z}_2(t')$, and $\mathbf{z}_1(t'')$:

$$\begin{aligned}\mathbf{z}_2(t'') &= \Gamma(\mathbf{z}_1(t'),\mathbf{z}_2(t'),\mathbf{z}_1(t'')) = \\ &= \frac{\left(\mathbf{z}_2(t')^T\mathbf{z}_1(t')\right)\mathbf{z}_1(t'') + \left(\mathbf{z}_2(t')^T\mathbf{z}_1^\perp(t')\right)\mathbf{z}_1^\perp(t'')}{\|\mathbf{z}_1(t')\|}.\end{aligned} \tag{4.5}$$

The proposed distributed estimation algorithm follows a multi-step approach, whose main parts are thoroughly described in the following of this section. Before going into the details, we sketch all the steps in chronological order (refer to Fig. 4.2 for a block diagram representation of the interconnections among steps).

The algorithm starts at time $t = t_0$. Four time instants $t_0 \le t_1 \le t_2 \le t_3 \le t_4$ are identified, which define when some crucial quantities become available due to the convergence of some of the estimation steps. After t_4, the algorithm has identified the two constant parameters m and J and will be able to observe the time-varying parameters $\mathbf{p}_{C_i} - \mathbf{p}_C$.

Step 1: $\mathbf{z}_{ij}(t)$ *becomes available after* t_1. Each robot i employs the velocity measurement $\dot{\mathbf{p}}_{C_i}$ and the velocities of its neighbors $\dot{\mathbf{p}}_{C_j}$, with $j \in \mathcal{N}_i$, to obtain, in a distributed fashion, an estimate of the relative positions $\mathbf{z}_{ij}(t)$, $j \in \mathcal{N}_i$. This estimation process needs a short convergence time to retrieve the distance $\|\mathbf{z}_{ij}(t)\|$ between contact points.

Therefore, $\mathbf{z}_{ij}(t)$ will be available for any time $t \geq t_1$, where $t_1 \geq t_0$. This step is detailed in Sec. 4.3.2;

Step 2: $\mathbf{z}_i(t)$ *becomes available after* t_1. For any time $t \geq t_1$, each robot i uses the relative position measurements $\mathbf{z}_{ij}(t)$ to compute, using the algorithm in [69], the vector $\mathbf{z}_i(t)$;

Step 3: $\omega(t)$ *becomes available after* t_1. For any time $t \geq t_1$, robot i uses the following formula to compute, locally, the angular velocity of B

$$\omega(t) = \frac{\dot{\mathbf{z}}_{ij}^T \mathbf{z}_{ij}^\perp(t)}{\|\mathbf{z}_{ij}\|^2}, \tag{4.6}$$

where j can be any possible neighbor in $\mathcal{N}_i$;

Step 4: J *becomes known after* t_2. For all $t \in (t_1, t_2)$ each robot i applies a force $\mathbf{f}_i(t) = \mathbf{z}_i^\perp(t)$ (available thanks to Step 2) and observes $\omega(t)$ (available thanks to (4.42)). This allows robot i to reach, at $t = t_2$, an estimate of the constant parameter J. Therefore, for any $t \geq t_2$, J is known by all the robots. This step is detailed in Sec. 4.3.3.1.

Step 5: $\mathbf{p}_{C_i} - \mathbf{p}_C$ *becomes available after* t_3. For any $t \in (t_2, t_3)$ each robot i applies an arbitrary nonzero force $\mathbf{f}_i(t)$ and measures $\omega(t)$. Using the nonlinear observer detailed in Sec. 4.4.3.2 together with the estimate of J computed in Step 4, each robot is able to obtain an estimate that eventually converges to the time-varying vector $\mathbf{z}_C$. For any $t \geq t_3$, each robot is then able to compute the sought vector $\mathbf{p}_{C_i} - \mathbf{p}_C$ using

$$\begin{aligned} \mathbf{p}_{C_i}(t) - \mathbf{p}_C(t) &= \mathbf{z}_i(t) + \mathbf{z}_C(t) = \\ &= \mathbf{z}_i(t) + \Gamma(\mathbf{z}_C(t_3), \mathbf{z}_{ij}(t_3), \mathbf{z}_{ij}(t)), \end{aligned} \tag{4.7}$$

where Γ is defined in (4.5) and j is any neighbor in $\mathcal{N}_i$.

Step 6: $\dot{\mathbf{p}}_C(t)$ *becomes available after* t_3. For any $t \geq t_3$, the velocity of the center of mass is computed locally by each robot i using

$$\dot{\mathbf{p}}_C(t) = \dot{\mathbf{p}}_{C_i}(t) - \omega(t)(\mathbf{p}_{C_i}(t) - \mathbf{p}_C(t))^\perp, \tag{4.8}$$

where all the quantities in the right hand side of (4.8) are known at any $t \geq t_3$ thanks to all the previous steps.

Step 7: m *becomes known after* t_4. For all $t \in (t_3, t_4)$ each robot i applies an arbitrary nonzero force $\mathbf{f}_i(t)$ and measures $\dot{\mathbf{p}}_C(t)$ thanks to (4.8). This allows each robot i to reach, at $t = t_4$, an estimate of the mass of the object m, as detailed in Sec. 4.4.3.3. Therefore, at any $t \geq t_4$ the mass m is known by all robots.

For any $t \geq t_4$, the constant parameters m and J are known by every robot (Steps 7 and 4). Furthermore, each robot i can instantaneously compute $\mathbf{p}_{C_i}(t) - \mathbf{p}_C(t)$ (Step 3). Therefore, Problem 4.1 is solved.

In the following sections we detail the algorithms used in Steps 1, 4, 3, and 7.

4.3.2 Distributed Estimation of Relative Positions of the Contact Points

In this section, we propose an algorithm to compute the relative positions $\mathbf{z}_{ij}$, for each $j \in \mathcal{N}_i$, only resorting to local sensing and 1-hop communication.

The fact that C_i and C_j belong to the same rigid body constrains their inter-distance to be constant over time, *i.e.*,

$$\mathbf{z}_{ij}^T \mathbf{z}_{ij} = \text{const.} \tag{4.9}$$

Taking the time derivative of both sides of (4.9), we obtain a constraint on the difference between the velocities of C_i and C_j, which has to be perpendicular to $\mathbf{z}_{ij}$, *i.e.*,

$$\dot{\mathbf{z}}_{ij}^T \mathbf{z}_{ij} = 0. \tag{4.10}$$

Noting that $\|\dot{\mathbf{z}}_{ij}\| > 0$ and $\|\mathbf{z}_{ij}\| > 0$, constraint (4.10) can be rewritten as

$$\frac{\mathbf{z}_{ij}}{\|\mathbf{z}_{ij}\|} = \text{sign}(\omega) \frac{\dot{\mathbf{z}}_{ij}^{\perp}}{\|\dot{\mathbf{z}}_{ij}^{\perp}\|} \quad \textit{i.e.,} \quad \mathbf{z}_{ij} = \text{sign}(\omega) \|\mathbf{z}_{ij}\| \, \mathbf{y}_{ij}, \tag{4.11}$$

where we compactly recast $\frac{\dot{\mathbf{z}}_{ij}^{\perp}}{\|\dot{\mathbf{z}}_{ij}^{\perp}\|}$ as $\mathbf{y}_{ij}$ to emphasize that quantity $\mathbf{y}_{ij}$ is available to robot i resorting to sensing and one-hop communication. Thus, the constant distance $\|\mathbf{z}_{ij}\|$ between two generic contact points represents the only unknown toward the computation of the time-varying vector $\mathbf{z}_{ij}$, except for $\text{sign}(\omega)$.

Differentiating both sides of (4.11), we obtain

$$\dot{\mathbf{z}}_{ij} = \text{sign}(\omega) \|\mathbf{z}_{ij}\| \, \dot{\mathbf{y}}_{ij}, \tag{4.12}$$

which cannot be directly used to compute $\|\mathbf{z}_{ij}\|$, since only $\dot{\mathbf{z}}_{ij}$ and $\mathbf{y}_{ij}$ are measured, but $\dot{\mathbf{y}}_{ij}$ is not.

Therefore, we apply the technique described in Appendix A, that uses the first-order low-pass version, $\dot{\mathbf{z}}_{ij}^f$ and $\mathbf{y}_{ij}^f$, of the measured quantities $\dot{\mathbf{z}}_{ij}$ and $\mathbf{y}_{ij}$, respectively. We observe that the sign of (4.46) depends on $\text{sign}(\omega)$. Therefore, in order to estimate

a positive quantity, we use the squared norm of the filtered quantities. Thus, we can summarize the previous derivations in the following result:

Proposition 4.1 (Relative position estimation from velocity measures)**.** *For each* $(i,j) \in \mathcal{E}$*, if* $\|\mathbf{z}_{ij}\| > 0$ *and* $\|\dot{\mathbf{z}}_{ij}\| > 0$*, then* $\mathbf{z}_{ij}$ *can be computed using only the velocities of* C_i *and* C_j *by means of the following estimator*

$$\mathbf{z}_{ij} = \sqrt{\frac{\|\dot{\mathbf{z}}_{ij}^f\|^2}{k_f^2 \left\|\mathbf{y}_{ij} - \mathbf{y}_{ij}^f\right\|^2}}\mathbf{y}_{ij} \tag{4.13}$$

where $(\cdot)^f$ *is a first-order low-pass filter and* k_f *is its gain.*

Finally, we obtain $\text{sign}(\omega)$ as $\text{sign}\left[\left(\dot{\mathbf{z}}_{ij}^f\right)^T (\mathbf{y}_{ij} - \mathbf{y}_{ij}^f)\right]$.

4.3.3 Estimation of Inertial Parameters

In this section, we describe in detail the estimation algorithms for the constant quantities J and m, and a nonlinear observer for $\mathbf{z}_C$, which can be used in (4.7) to have an estimate of the sought parameter $\mathbf{p}_{C_i} - \mathbf{p}_C$.

Since $\mathbf{p}_{C_i} - \mathbf{p}_C = \mathbf{z}_i + \mathbf{z}_C$, we can rewrite (4.2) as

$$\dot{\omega} = \frac{1}{J}\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\mathbf{f}_i + \frac{1}{J}\mathbf{z}_C^{\perp T}\sum_{i=1}^{n}\mathbf{f}_i. \tag{4.14}$$

Each vector $\mathbf{z}_i$ for $i = 1,\ldots,n$ can be computed in a distributed way by the i-th robot resorting to the distributed algorithm presented in [69], which can be applied since the communication graph is connected by assumption.

4.3.3.1 Estimation of the Moment of Inertia J

If each robot i applies the force

$$\mathbf{f}_i = \mathbf{z}_i^{\perp}, \quad \forall i = 1\ldots n, \forall t \geq 0 \tag{4.15}$$

at each time t, then the second summation in (4.14) vanishes

$$\sum_{i=1}^{n} \mathbf{f}_i = \sum_{i=1}^{n} \mathbf{z}_i^{\perp} = \sum_{i=1}^{n} (\mathbf{p}_{C_i} - \mathbf{p}_G)^{\perp} = \mathbf{\Omega} \sum_{i=1}^{n} (\mathbf{p}_{C_i} - \mathbf{p}_G)$$
$$= \mathbf{\Omega} \left(\sum_{i=1}^{n} \mathbf{p}_{C_i} - \sum_{i=1}^{n} \mathbf{p}_G \right) = \mathbf{\Omega} \left(n\mathbf{p}_G - n\mathbf{p}_G \right) = 0$$

and (4.14) simplifies to

$$\dot{\omega} = \frac{1}{J} \sum_{i=1}^{n} \mathbf{z}_i^{\perp T} \mathbf{z}_i^{\perp} = \frac{1}{J} \sum_{i=1}^{n} \|\mathbf{z}_i\|^2. \tag{4.16}$$

Hence, we can use the following distributed algorithm in order to compute J:

1. Distributively compute the value $w = \sum_{i=1}^{n} \|\mathbf{z}_i\|^2$ using an average consensus algorithm [6], *i.e.*, each robot sets a local state variable $\phi_i|_{t=t_0} = \|\mathbf{z}_i\|^2$ and then applies the local update rule

$$\dot{\phi}_i = \sum_{j \in \mathcal{N}_i} (\phi_j - \phi_i), \tag{4.17}$$

   which, since the communication graph is connected by assumption, leads ϕ_i to be asymptotically equal to w/n. Thus, after a consensus is reached, each robot can compute $w = n\phi_i$;

2. Each robot applies a constant force $\mathbf{f}_i = \mathbf{z}_i^{\perp}$;

3. Locally estimate J by using the filtering approach described in the Appendix A, substituting u with w, y with ω, which is known locally thanks to (4.42), and θ with $\frac{1}{J}$.

Remark 4.2. It is *not* important that each robot starts to apply the force inputs given by (4.15) at exactly the same time, so no exact time synchronization is needed. This is because we can assume that there will always exist a time when all the robot have eventually applied the input force,

Remark 4.3. When applying the constant force inputs two possible cases can occur. The first case is that the body moves of pure rotation, that is, the center of mass does not move. This is, in any case, adequate to our purposes. The second case is that the body moves with constant angular acceleration. In this case, it is clear that the control inputs can be safely applied only for a limited time, after which the movement of the body must be stopped, e.g., with a pure damping force based on a velocity feedback. However, should the time be not enough for estimation purposes, the process can be repeated several times after each stop, to ensure the acquisition of the necessary measurements.

4.3.3.2 Observer for the Center-of-Mass Relative Position $\mathbf{z}_C$

Assume that each robot applies an arbitrary force $\mathbf{f}_i(t) \neq \mathbf{0}\ \forall i = 1, \ldots, n, \forall t \geq 0$, which can be rewritten as

$$\mathbf{f}_i(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_i(t) + \Delta\mathbf{f}_i(t) = \mathbf{f}_{\text{mean}}(t) + \Delta\mathbf{f}_i(t). \tag{4.18}$$

Then, Eq. (4.2) becomes

$$\begin{aligned} \dot{\omega} =& \frac{1}{J}\left(\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\right)\mathbf{f}_{\text{mean}}(t) + \frac{n}{J}\mathbf{z}_C^{\perp T}\mathbf{f}_{\text{mean}}(t) + \\ & \frac{1}{J}\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i + \frac{1}{J}\mathbf{z}_C^{\perp T}\sum_{i=1}^{n}\Delta\mathbf{f}_i = \\ & \frac{n}{J}\mathbf{z}_C^{\perp T}\mathbf{f}_{\text{mean}}(t) + \frac{1}{J}\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i, \end{aligned} \tag{4.19}$$

where we used the facts that $\sum_{i=1}^{n} \mathbf{z}_i^{\perp T} = 0$ and $\sum_{i=1}^{n}\Delta\mathbf{f}_i = 0$. Thus, Eq. (4.19) can be written as

$$\dot{\omega} = \mathbf{z}_C^{\perp T}\check{\mathbf{f}} + \eta, \tag{4.20}$$

where we let $\check{\mathbf{f}} = \dfrac{n}{J}\mathbf{f}_{\text{mean}}$ and $\eta = \frac{1}{J}\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i$.

By means of standard dynamic consensus algorithms [70], it is possible to reach an agreement on $\check{\mathbf{f}}$. To this purpose, each robot i needs to exchange only the local quantity $\mathbf{f}_i(t)$ with its neighbors. This implies that robot i can locally compute $\Delta\mathbf{f}_i = \mathbf{f}_i - \check{\mathbf{f}}$. By exchanging the local quantity $\mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i$ with its neighbors and applying again the dynamic consensus algorithm, also η can be locally computed in a distributed way. Therefore, in the following we can safely assume that both $\check{\mathbf{f}}$ and η are locally known to each robot.

Since $\mathbf{z}_C$ is a vector of constant modulus rigidly attached to the object we have

$$\dot{\mathbf{z}}_C^{\perp} = -\mathbf{z}_C\omega. \tag{4.21}$$

Combining Eqs. (4.20) and (4.21), we obtain the following autonomous nonlinear system

$$\begin{cases} \dot{x}_1 = -x_2 x_3 \\ \dot{x}_2 = x_1 x_3 \\ \dot{x}_3 = x_1\check{f}_y - x_2\check{f}_x + \eta \\ \ \ y = x_3, \end{cases} \tag{4.22}$$

where we let $z_C^x = x_1$, $z_C^y = x_2$, $\omega = x_3$, and $\check{\mathbf{f}} = (\check{f}_x \;\; \check{f}_y)^T$. The system output is assumed as $y = x_3 = \omega$, since the angular velocity is locally estimated by each robot using Eq. (4.6). Therefore, the estimation of $\mathbf{z}_C$ is equivalent to the observation of the state of nonlinear system (4.22) with output $y = x_3 = \omega$, and where $\check{f}_y$, $\check{f}_x$, and η are known inputs.

The problem of estimating $\mathbf{z}_C$ can be dealt with as a nonlinear observability problem, as follows:

Problem 4.2. Estimating the quantity $\mathbf{z}_C$ in a distributed way is equivalent to observe the state of the autonomous nonlinear system (4.22) with output $y = x_3 = \omega$, and where $\bar{f}_y$, $\bar{f}_x$, and η are known inputs.

Before designing a suitable nonlinear observer, the observability of system (4.22) is studied.

Proposition 4.4. *If* $x_3 \not\equiv 0$ *and* $\left[\check{f}_x(t) \;\; \check{f}_y(t)\right]^T \not\equiv \mathbf{0}$*, then system* (4.22) *is locally observable in the sense of [71].*

Proof. The observability matrix [71] [72] is

$$O = \begin{pmatrix} 0 & 0 & 1 \\ \check{f}_y & -\check{f}_x & 0 \\ -\check{f}_x x_3 & -\check{f}_y x_3 & -\check{f}_x x_1 - \check{f}_y x_2 \end{pmatrix} \tag{4.23}$$

whose determinant is $\det(O) = -x_3(\check{f}_x^2 + \check{f}_y^2)$. The system (4.22) is locally observable in the sense of [71] iff O is invertible, from which the thesis follows. □

Thus, matrix $\mathbf{z}_C$ is observable from local velocity measurements if and only if the angular velocity of the object and the average vector of the applied forces are not identically zero. The nonlinear observer is designed as follows.

Since observability depends on the angular velocity $\omega = x_3$, it is important to analyze the behavior of the trajectories of (4.22). In particular, trajectories for which the angular velocity is constantly zero or asymptotically converges to zero would lead to an unobservable system. On the other hand, trajectories for which the angular velocity grows unbounded are practically unfeasible.[4]

For the particular case in which each robot applies the same force $\mathbf{f}_i(t) = \mathbf{f}(t)$, $\forall i$, we have the following interesting result. It is possible to show that, apart from a set of

[4]One cannot, in this case, stop the motion and restart as we suggested in Remark 4.3, since we are estimating a time-varying quantity and therefore we cannot restart the process each time with an improved initial estimate.

cases of zero measure, the trajectories of (4.22) yield angular velocities that are suitable both regarding the observability issue, and the practical feasibility.

Proposition 4.5. *The following facts hold for system* (4.22) *when* $\mathbf{f}_i(t) = \mathbf{f}(t)$, $\forall i$*:*

1. *the origin is a stable equilibrium point;*
2. *the angular velocity* x_3 *is bounded, and in particular:*

$$|x_3| \leq \sqrt{x_3^2(0) + 4\check{f}_y \|\mathbf{z}_C\|} \tag{4.24}$$

3. $\exists T \geq 0$ *such that* $x_3(t) = 0$ $\forall t \geq T$ *if and only if the initial angular velocity* $x_3(0)$ *is such that:*

$$x_3^2(0) = 2\check{f}_y(x_2(0) \pm \|\mathbf{z}_C\|). \tag{4.25}$$

Proof. We observe that if $\mathbf{f}_i(t) = \mathbf{f}(t)$, then $\eta = 0$, $\forall i$, holds in (4.22). To prove the stability of the origin, we consider the following candidate Lyapunov function:

$$V = \frac{1}{2}(x_1^2 + x_2^2) + \frac{1}{2}(x_3^2 - 2\check{f}_y x_2)^2. \tag{4.26}$$

It is easy to verify that $V \geq 0$ and that $V = 0$ only in the origin. Computing $\dot{V}$ we obtain

$$\begin{aligned} \dot{V} &= x_1\dot{x}_1 + x_2\dot{x}_2 + (x_3^2 - 2\check{f}_y x_2)(2x_3\dot{x}_3 - 2\check{f}_y\dot{x}_2) = \\ &= 0 + (x_3^2 - 2\check{f}_y x_2) \cdot 0 = 0, \end{aligned} \tag{4.27}$$

which implies stability.

To prove the second statement we first note that, from (4.27), the following quantity is an invariant along the trajectories:

$$x_3^2 - 2\check{f}_y x_2 = \text{const} = x_3^2(0) - 2\check{f}_y x_2(0). \tag{4.28}$$

Therefore,

$$x_3^2 = x_3^2(0) + 2\check{f}_y(x_2 - x_2(0)) \leq x_3^2(0) + 4\check{f}_y\|\mathbf{z}_C\|, \tag{4.29}$$

which proves (4.24).

We now prove the last statement. If $x_3(t) = 0$, $\forall t \geq T$, then $x_3(T) = 0$ and $\dot{x}_3(T) = 0$. Observing (4.22), it can be verified that the vanishing of the first derivative of x_3 ensures

the vanishing of all its higher order derivatives. Vice versa also holds, *i.e.*, $x_3(T) = 0$ and $\dot{x}_3(T) = 0$ imply $x_3(t) = 0 \; \forall t \geq T$. In order to prove the statement, it is then enough to show what are the initial conditions such that x_3 and its first derivative vanish at $t = T$. Considering (4.28) at time T, we have

$$\underbrace{x_3^2(T)}_{=0} - 2\check{f}_y x_2(T) = x_3^2(0) - 2\check{f}_y x_2(0), \tag{4.30}$$

which implies

$$x_2(T) = \frac{2\check{f}_y x_2(0) - x_3^2(0)}{2\check{f}_y}. \tag{4.31}$$

Moreover, it is easy to verify that another invariant quantity of system (4.22) is

$$x_1^2 + x_2^2 = \text{const} = x_1^2(0) + x_2^2(0). \tag{4.32}$$

Plugging (4.31) in (4.32) at time T we obtain

$$x_1^2(T) + \frac{(2\check{f}_y x_2(0) - x_3^2(0))^2}{4\check{f}_y^2} = x_1^2(0) + x_2^2(0). \tag{4.33}$$

In order to have $\dot{x}_3(T) = 0$, $x_1(T) = 0$ must hold (see (4.22)), therefore Eq. (4.33) becomes

$$\left(2\check{f}_y x_2(0) - x_3^2(0)\right)^2 = 4\check{f}_y^2 \left(x_1^2(0) + x_2^2(0)\right), \tag{4.34}$$

which can be rewritten as

$$x_3^4(0) - 4\check{f}_y x_2(0) x_3^2(0) - 4\check{f}_y^2 x_1^2(0) = 0, \tag{4.35}$$

that can be solved for the unknown $x_3^2(0)$, thus resulting in condition (4.25). □

Proposition 4.5 guarantees that the angular velocity x_3 does not vanish, and therefore the system remains observable for all time, except for at most four initial condition points, which represent a zero measure set that has probability zero to happen in real-world robotics applications. Moreover, the proposition shows that the angular velocity x_3 and its derivatives remain bounded and gives an upper bound that can be decreased by acting on the input. This last feature is very important for the real applicability of this method, although the application of the same force for each robot is required.

Thus, verified that vector $\mathbf{z}_C$ is observable, as stated in Proposition 4.4, the nonlinear observer is designed as follows.

Theorem 4.6. *Consider the following dynamical system*

$$\begin{aligned}\dot{\hat{x}}_1 &= -\hat{x}_2 x_3 + \check{f}_y(x_3 - \hat{x}_3)\\ \dot{\hat{x}}_2 &= \hat{x}_1 x_3 - \check{f}_x(x_3 - \hat{x}_3)\\ \dot{\hat{x}}_3 &= \hat{x}_1 \check{f}_y - \hat{x}_2 \check{f}_x + k_e(x_3 - \hat{x}_3) + \eta,\end{aligned} \tag{4.36}$$

where $k_e > 0$. System (4.36) is an asymptotic observer for system (4.22), i.e., defining $\hat{\mathbf{x}} = (\hat{x}_1\ \hat{x}_2\ \hat{x}_3)^T$ and $\mathbf{x} = (x_1\ x_2\ x_3)^T$, $\hat{\mathbf{x}} \to \mathbf{x}$ asymptotically.

Proof. Define the error vector as $\mathbf{e} = (e_1\ e_2\ e_3)^T = ((x_1 - \hat{x}_1)\ (x_2 - \hat{x}_2)\ (x_3 - \hat{x}_3))^T$, the error dynamics is given by the following nonlinear system:

$$\dot{\mathbf{e}} = \begin{pmatrix} 0 & -x_3 & -\check{f}_y \\ x_3 & 0 & \check{f}_x \\ \check{f}_y & -\check{f}_x & -k_e \end{pmatrix} \mathbf{e}. \tag{4.37}$$

Define the following candidate Lyapunov function

$$V(\mathbf{e}) = \frac{1}{2}\left(e_1^2 + e_2^2 + e_3^2\right), \tag{4.38}$$

whose time derivative along the system trajectories is

$$\begin{aligned}\dot{V}(\mathbf{e}) =& e_1(-e_2 x_3 - \check{f}_y e_3) + \\ & e_2(x_3 e_1 + \check{f}_x e_3) + e_3(\check{f}_y e_1 - \check{f}_x e_2 - k_e e_3)\\ =& -e_1 e_2 x_3 - e_1 \check{f}_y e_3 + e_2 x_3 e_1 + \\ & e_2 \check{f}_x e_3 + e_3 \check{f}_y e_1 - e_3 \check{f}_x e_2 - k_e e_3^2\\ =& -k_e e_3^2,\end{aligned} \tag{4.39}$$

which is negative semidefinite. To assess stability, we consider the set $\mathcal{V} = \left\{\mathbf{e} \text{ s.t. } \dot{V}(\mathbf{e}) = 0\right\} = \{\mathbf{e} \text{ s.t. } e_3 = 0\}$ and we study its largest invariant set $M \subset \mathcal{V}$. Given a generic vector $\check{\mathbf{e}} = (\check{e}_1\ \check{e}_2\ 0)^T \in \mathcal{V}$, by means of (4.37), it is easy to verify that the first and second time derivatives of e_3 along a trajectory containing $\check{\mathbf{e}}$ are given by

$$\left.\frac{de_3}{dt}\right|_{\check{\mathbf{e}}} = \check{f}_y \check{e}_1 - \check{f}_x \check{e}_2 \quad \left.\frac{d^2 e_3}{dt^2}\right|_{\check{\mathbf{e}}} = \left.\frac{de_1}{dt}\right|_{\check{\mathbf{e}}} = -x_3(\check{f}_y \check{e}_2 + \check{f}_x \check{e}_1). \tag{4.40}$$

Therefore, if x_3 is not vanishing, then the largest invariant set M consists of the only equilibrium point $(0\ 0\ 0)^T$. Thus, the thesis holds due to the Krasovskii–LaSalle invariance principle [73]. □

We remark that observer (4.36) can be implemented in a distributed fashion by resorting only to local information.

The estimation error of the proposed observer vanishes asymptotically in the ideal case of absence of noise. In the next section we will analyze the more realistic noise prone case.

4.3.3.3 Estimation of the Mass m

After having estimated J and observed $\mathbf{z}_C$, the quantity $\mathbf{p}_{iC}$ is known by each robot i using (4.7), and therefore each robot can estimate $\dot{\mathbf{p}}_C$ using (4.8). Thus, if each robot applies a nonzero force vector as in Sec. 4.4.3.2 and assuming that $\mathbf{f}_{\text{mean}}(t)$, defined in Eq. (4.18), is not zero, then (4.1) becomes

$$\ddot{\mathbf{p}}_C = \frac{n}{m}\mathbf{f}_{\text{mean}}, \tag{4.41}$$

i.e., a linear system with measured output $\dot{\mathbf{p}}_C$.

Thus, an approach similar to the one used to estimate J, relying on the technique recalled in the Appendix A, can be applied again to estimate m. We omit here the details for brevity. We observe that this solution is again distributed, since each robot needs only to receive the velocity $\dot{\mathbf{p}}_{C_j}$ from its neighbors.

4.4 Noisy Velocity Measurements Case

It is well-known that the presence of noisy measurements can alter the outcome of estimation and identification strategies, if not adequately treated [74]. In this section, we elucidate the effect of noise on the quality of estimates, providing a convenient bound on their accuracy.

4.4.1 Estimation Algorithm

The algorithm presented in Section 4.3 is reviewed here considering the presence of noise in the measurements, *i.e.*, $\mathbf{\Sigma}_i \neq \mathbf{0}$, for all i. The noise prone version of the algorithm consists of 4 steps that lead the estimates performed by each robot to converge to the real value of the inertial parameters. Each of the 4 steps, described in the following, converges in a finite time interval. We indicate with t_0 the starting time of the algorithm and with t_k, $k = 1, \ldots, 4$, the end time of each step. We remark that the estimation

steps are sequentially executed, *i.e.*, no information obtained at any step has to be fed back to previous steps at any time.

Step 1: *an estimation of* $\widehat{\|\mathbf{z}_{ij}\|}$ *becomes known after* t_1. Each robot i applies an arbitrary force $\mathbf{f}_i(t)$ to the body and uses the noisy measurements $\widetilde{\dot{\mathbf{p}}}_{C_i}$ and $\widetilde{\dot{\mathbf{p}}}_{C_j}$, with $j \in \mathcal{N}_i$, to estimate $\widehat{\|\mathbf{z}_{ij}\|}$ in a distributed way. This is achieved through a least square estimation that rapidly converges, at time t_1, to the inter-distances between contact points, as detailed in Sec. 4.4.2.

Step 2: *an estimation of* $\hat{J}$ *becomes known after* t_2. For $t \geq t_1$, four concurrent tasks are executed to produce an estimate $\hat{J}$ of the moment of inertia J. This implies the existence of a delay between the execution of the tasks. However, we reckon that this delay is negligible for application purposes and, in the following, we assume that all the information is instantaneously available. The first task is the estimation, by each robot, of the vectors $\hat{\mathbf{z}}_{ij}(t)$, of the sign of the angular velocity $\widehat{\text{sign}(\omega)}$, on the basis of the noisy measurement $\widetilde{\dot{\mathbf{p}}}_{C_i}$ and $\widetilde{\dot{\mathbf{p}}}_{C_j}$, with $j \in \mathcal{N}_i$, and of the inter-distances $\widehat{\|\mathbf{z}_{ij}\|}$ obtained in the previous step. This task is detailed in Sec. 4.4.2. In the second task, each robot computes the estimate $\hat{\mathbf{z}}_i(t)$ of $\mathbf{z}_i(t)$, using the position estimates $\hat{\mathbf{z}}_{ij}(t)$ and the algorithm in [69]. On the basis of the estimates $\hat{\mathbf{z}}_i(t)$, each robot applies a force $\mathbf{f}_i(t) = \hat{\mathbf{z}}_i^{\perp}(t)$ during the time interval $t_1 \leq t \leq t_2$. As a consequence, the body moves in $t_1 \leq t \leq t_2$ and each robot can compute an estimate of the angular velocity of B, $\hat{\omega}(t)$. To this aim, each robot i performs a local estimation of ω, which is a time varying quantity, as

$$\hat{\omega}_i(t) = \widehat{\text{sign}(\omega)} \frac{\|\tilde{\dot{\mathbf{z}}}_{ij}\|}{\widehat{\|\mathbf{z}_{ij}\|}}, \tag{4.42}$$

where j is any neighbor in $\mathcal{N}_i$ and update their local estimates $\hat{\omega}_i$ using a dynamic consensus algorithm [70] to converge to a common estimate $\hat{\omega}$ all over the network. Based on the estimate $\hat{\omega}$, each robot performs an estimate of the moment of inertia J, indicated as $\hat{J}_i$, as detailed in Sec. 4.4.3.1. Then, an average consensus algorithm is used also in this case to reach an agreement on a common value for $\hat{J}$ [6].

Step 3: *an estimation of* $\widehat{\mathbf{p}_{C_i} - \mathbf{p}_C}$ *becomes known after* t_3. Each robot applies an arbitrary nonzero force $\mathbf{f}_i(t)$ and estimates $\hat{\omega}(t)$, similarly to Step 2. Then, each robot can compute $\hat{\mathbf{z}}_C$, relying on the estimate of $\hat{J}$ (available thanks to Step 2) and using the nonlinear observer detailed in Sec. 4.4.3.2. After the convergence time of the observer, t_3, the unknown vector $\mathbf{p}_{C_i} - \mathbf{p}_C$ can be estimated using the following formula:

$$\begin{aligned} \widehat{(\mathbf{p}_{C_i}(t) - \mathbf{p}_C(t))} &= \hat{\mathbf{z}}_i(t) + \hat{\mathbf{z}}_C(t) = \\ &= \hat{\mathbf{z}}_i(t) + \Gamma(\hat{\mathbf{z}}_C(t_3), \hat{\mathbf{z}}_{ij}(t_3), \hat{\mathbf{z}}_{ij}(t)), \end{aligned} \tag{4.43}$$

where Γ is defined in (4.5) and $j \in \mathcal{N}_i$.

Step 4: *an estimation of $\hat{m}$ becomes available after t_4.* Each robot i estimates $\hat{\dot{\mathbf{p}}}_C(t)$, by applying any nonzero force $\mathbf{f}_i(t)$, $\forall i$, and estimates $\hat{\dot{\mathbf{p}}}_C(t)$ as

$$\hat{\dot{\mathbf{p}}}_C(t) = \widetilde{\dot{\mathbf{p}}}_{C_i}(t) - \hat{\omega}(t)(\mathbf{p}_{C_i}\widehat{(t) - \mathbf{p}}_C(t))^{\perp}. \tag{4.44}$$

We observe that all the quantities in the right hand side of (4.44) are computed in the previous steps and are locally available after $t \geq t_3$. Based on the estimate $\hat{\dot{\mathbf{p}}}_C$, each robot performs an estimate of the mass m, indicated as $\hat{m}_i$, as detailed in Sec. 4.4.3.3. An average consensus algorithm is then used to converge to a common estimate of the mass $\hat{m}$, which, for $t \geq t_4$, will be known by all robots.

4.4.2 Estimation of the Relative Positions of the Contact Points

In this section, we present a procedure for recovering the relative positions between the contact points in a distributed way, *i.e.*, using only local information dealing with noisy velocity measurements. Based on (4.11), robot i is able to recover an estimate of vector $\mathbf{z}_{ij}$ as

$$\hat{\mathbf{z}}_{ij} = \hat{\lambda}_{ij}\widetilde{\mathbf{y}}_{ij}, \tag{4.45}$$

where $\hat{\lambda}_{ij}$ is an estimate of $\lambda_{ij} = \|\mathbf{z}_{ij}\|$ and $\widetilde{\mathbf{y}}_{ij} = \frac{(\widetilde{\dot{\mathbf{p}}}_{C_i} - \widetilde{\dot{\mathbf{p}}}_{C_j})^{\perp}}{\|\widetilde{\dot{\mathbf{p}}}_{C_i} - \widetilde{\dot{\mathbf{p}}}_{C_j}\|}$ is the noisy measurement of $\mathbf{y}_{ij}$. In the following, we propose two strategies to estimate $\|\mathbf{z}_{ij}\|$, *i.e.*, the distance between the contact points C_i and C_j, $\forall (i,j) \in \mathcal{E}$.

4.4.2.1 First Strategy

The first method to estimate the distance between two contact points is based on (4.11). It is clear that the distance cannot be estimated if $\omega \equiv 0$, $\forall t$, since this implies $\dot{\mathbf{z}}_{ij} \equiv \mathbf{0}$. Thus, considering any time interval in which $\omega \neq 0$ we differentiate with respect to time both sides of (4.11), and obtain

$$\dot{\mathbf{z}}_{ij} = \text{sign}(\omega)\lambda_{ij}\ \dot{\mathbf{y}}_{ij}. \tag{4.46}$$

This method requires the knowledge of the quantity $\dot{\mathbf{y}}_{ij}$, which is not directly measurable. To overcome this limitation, we apply a first-order low-pass filter to both sides of Eq. (4.46), denoting with $\dot{\mathbf{z}}^f_{ij}$ and $\dot{\mathbf{y}}^f_{ij}$ the filtered versions of $\dot{\mathbf{z}}_{ij}$ and $\dot{\mathbf{y}}_{ij}$, respectively (refer to the Appendix A for a detailed explanation). Taking the squared norm of both

sides after filtering, we obtain

$$\|\dot{\mathbf{z}}_{ij}^f\|^2 = \lambda_{ij}^2 \; k_f^2 \|\mathbf{y}_{ij} - \mathbf{y}_{ij}^f\|^2, \tag{4.47}$$

where k_f is the gain of the low pass filter. Given the noisy measurements $\widetilde{\mathbf{y}}_{ij}$ and $\widetilde{\dot{\mathbf{z}}}_{ij}$, the estimates $\hat{\lambda}_{ij}^2$ of λ_{ij}^2 are obtained by solving the following linear least squares problem:

$$\begin{bmatrix} \|\widetilde{\dot{\mathbf{z}}}_{ij}^f(t_1)\|^2 \\ \vdots \\ \|\widetilde{\dot{\mathbf{z}}}_{ij}^f(t_q)\|^2 \end{bmatrix} = \hat{\lambda}_{ij}^2 \begin{bmatrix} k_f^2 \|\widetilde{\mathbf{y}}_{ij}(t_1) - \widetilde{\mathbf{y}}_{ij}^f(t_1)\|^2 \\ \vdots \\ k_f^2 \|\widetilde{\mathbf{y}}_{ij}(t_q) - \widetilde{\mathbf{y}}_{ij}^f(t_q)\|^2 \end{bmatrix}, \tag{4.48}$$

where $t_1 \dots t_q$ are the $q > 0$ acquisition times of the noisy measurements.

Furthermore, an estimate of $\text{sign}(\omega)$ at time t is given by

$$\widehat{\text{sign}(\omega)} = \text{sign}\left[\left(\widetilde{\dot{\mathbf{z}}}_{ij}^f(t) \right)^T \left(\widetilde{\mathbf{y}}_{ij}(t) - \widetilde{\mathbf{y}}_{ij}^f(t) \right) \right].$$

4.4.2.2 Second Strategy

Alternatively, Eq. (4.11) can be casted as

$$\|\dot{\mathbf{z}}_{ij}^{\perp}\| \mathbf{z}_{ij} = \text{sign}(\omega) \; \dot{\mathbf{z}}_{ij}^{\perp} \|\mathbf{z}_{ij}\|. \tag{4.49}$$

Differentiating with respect to time both sides of (4.49) (for $\omega \neq 0$), we obtain

$$\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt} \mathbf{z}_{ij} + \|\dot{\mathbf{z}}_{ij}^{\perp}\| \dot{\mathbf{z}}_{ij} = \text{sign}(\omega) \; \ddot{\mathbf{z}}_{ij}^{\perp} \|\mathbf{z}_{ij}\|. \tag{4.50}$$

Multiplying both sides of (4.50) by $\|\dot{\mathbf{z}}_{ij}^{\perp}\|$, we obtain

$$\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt} \|\dot{\mathbf{z}}_{ij}^{\perp}\| \mathbf{z}_{ij} + \|\dot{\mathbf{z}}_{ij}^{\perp}\|^2 \dot{\mathbf{z}_{ij}} = \text{sign}(\omega) \; \ddot{\mathbf{z}}_{ij}^{\perp} \|\dot{\mathbf{z}}_{ij}^{\perp}\| \|\mathbf{z}_{ij}\|. \tag{4.51}$$

Finally, since $\|\dot{\mathbf{z}}_{ij}^{\perp}\| \mathbf{z}_{ij} = \dot{\mathbf{z}}_{ij}^{\perp} \|\mathbf{z}_{ij}\|$, we have

$$\left(\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt} \dot{\mathbf{z}}_{ij}^{\perp} - \ddot{\mathbf{z}}_{ij}^{\perp} \|\dot{\mathbf{z}}_{ij}^{\perp}\| \right) \|\mathbf{z}_{ij}\| = -\text{sign}(\omega) \; \|\dot{\mathbf{z}}_{ij}^{\perp}\|^2 \dot{\mathbf{z}}_{ij}. \tag{4.52}$$

Computing the time derivatives $\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt}$ and $\ddot{\mathbf{z}}_{ij}$ using techniques for the differentiation of noisy signals [75] [76], and taking the squared norm of both sides, Eq. (4.52) can be solved for $\|\mathbf{z}_{ij}\|$. Since we rely on noisy measurements, we estimate $\hat{\lambda}_{ij}^2 = \widehat{\|\mathbf{z}_{ij}\|}^2$ by

solving the following linear least squares problem:

$$\begin{bmatrix} \left\| \widetilde{\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt}}(t_1)\widetilde{\mathbf{z}}_{ij}^{\perp}(t_1) - \widetilde{\ddot{\mathbf{z}}}_{ij}^{\perp}(t_1)\|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t_1)\| \right\|^2 \\ \vdots \\ \left\| \widetilde{\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt}}(t_q)\widetilde{\mathbf{z}}_{ij}^{\perp}(t_q) - \widetilde{\ddot{\mathbf{z}}}_{ij}^{\perp}(t_q)\|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t_q)\| \right\|^2 \end{bmatrix} \hat{\lambda}_{ij}^2 = \begin{bmatrix} \left\| \|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t_1)\|^2 \ \widetilde{\mathbf{z}}_{ij}(t_1) \right\|^2 \\ \vdots \\ \left\| \|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t_q)\|^2 \ \widetilde{\mathbf{z}}_{ij}(t_q) \right\|^2 \end{bmatrix} \tag{4.53}$$

Furthermore, an estimate of $\text{sign}(\omega)$ at time t is given by

$$\widehat{\text{sign}(\omega)} = \text{sign}\left[-\left(\widetilde{\frac{d\|\dot{\mathbf{z}}_{ij}^{\perp}\|}{dt}}(t)\widetilde{\mathbf{z}}_{ij}^{\perp}(t) - \widetilde{\ddot{\mathbf{z}}}_{ij}^{\perp}(t)\|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t)\| \right)^T \left(\|\widetilde{\dot{\mathbf{z}}}_{ij}^{\perp}(t)\|^2 \ \widetilde{\dot{\mathbf{z}}}_{ij}(t) \right) \right].$$

4.4.2.3 Discussion on the Two Strategies

An advantage of the first strategy is that the estimation is executed without relying on any derivative of the measurements. On the other hand, a drawback is that the unit vector computed from the velocity measurements, *i.e.*, $\widetilde{\mathbf{y}}_{ij}$, can be excessively noisy when ω is close to zero. This drawback is unavoidable, since $\omega = 0$ implies unobservability in the absence of noise terms. In the real (noisy) case, ω close to zero results in 'practical' unobservability of the inter-distances. A way to avoid this intrinsic drawback is to let the object rotate sufficiently fast during the estimation of the inter-distances.

The second method, which uses (4.52), overcomes the issue related to possible discontinuities in the unit vector $\mathbf{y}_{ij}$. However, this method assumes the knowledge of the time derivatives of the measured signals, thus, it implies the use of techniques for the differentiation of noisy signals [75] [76].

4.4.3 Estimation of the Inertial Parameters

In this section, we describe three algorithms for estimating: *(i)* the moment of inertia J, *(ii)* the time-varying vector $\mathbf{z}_C$, *i.e.*, the position of the center of mass C relative to the geometric center $\mathbf{p}_G$, and *(iii)* the mass m.

4.4.3.1 Estimation of the Moment of Inertia J

We consider the network applying the forces defined in (4.15). We note that the quantity $w = \sum_{i=1}^{n} \|\mathbf{z}_i\|^2$ in Eq. (4.16) is constant over time. Thus, the application of such forces produces a constant angular acceleration, which makes the distributed identification of J easier. In fact, let us consider the following distributed algorithm:

1. Before applying any force, each robots distributively computes the constant value $w = \sum_{i=1}^{n} \|\hat{\mathbf{z}}_i\|^2$. This can be done in a distributed way by means of a standard average consensus algorithm [6];

2. Each robot applies a constant force $\mathbf{f}_i = \hat{\mathbf{z}}_i^{\perp}$. This can be executed relying only on local information;

3. Thanks to the local estimate of ω, each robot computes a local estimate $\hat{J}_i$ of J using the approach presented in the Appendix A;

4. When the local estimates converges, the robots run a further consensus phase, thus reaching an agreement on a global estimate $\hat{J}$. This is done to average out the uncertainty in the estimates $\hat{J}_i$ caused by the noise in ω and $\mathbf{z}_i$.

Similarly to the noiseless case, we note that no perfect time synchronization is needed for the starting time of the application of forces $\mathbf{f}_i$, since each robot will eventually apply the prescribed force $\mathbf{f}_i = \hat{\mathbf{z}}_i^{\perp}$.

It may happen that the forces required for the estimation can be safely applied to the object only for a limited time interval to satisfy the practical requirement of keeping the angular velocity of the body bounded. In this case, the movement can be easily stopped every once in a while using a pure damping force based on a local velocity feedback. However, should the time be not enough for estimation purposes, the process can be repeated several times after each stop, to ensure the acquisition of the measurements necessary to identify J.

4.4.3.2 Observer for the Relative Position $\mathbf{z}_C$ of the CoM

As explained in Sec. 4.4.3.2, the estimation error of the proposed observer vanishes asymptotically in the ideal case of absence of noise. Actually, each robot sets the force $\check{\mathbf{f}}_{\mathrm{a}} = \dfrac{n}{\hat{J}} \mathbf{f}_{\mathrm{mean}}$ as input for the local observer, which is affected by noise, due to the presence of $\hat{J}$. Furthermore, the observer relies on the noisy estimate $\hat{\omega}$, computed using (4.42). Due to its asymptotic stability, we expect that in presence of noise the estimation error will remain bounded around the actual value of the parameter. In Sec. 4.5, we will numerically characterize the bound of the estimation error with respect to noise terms.

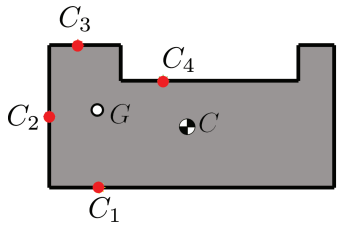


FIGURE 4.3: The simulated payload is an eight-sides polygon, obtained by a rectangular plate of sides 2 m×4 m, where a smaller rectangular of 0.5 m×2.5 m size has been cut off from the longer side. The $n = 4$ contact points, C_1, C_2, C_3, C_4, the position of the center of mass, C, and the position of geometric centroid, G, are illustrated.

4.4.3.3 Estimation of the Mass

Assume, as in Sec. 4.4.3.3, that each robot applies a force $\mathbf{f}_i(t)$ and that $\mathbf{f}_{\text{mean}}(t)$, defined in Eq. (4.18), is not zero. Thus, Eq. (4.1) becomes

$$\ddot{\mathbf{p}}_C = \frac{n}{m}\mathbf{f}_{\text{mean}}. \tag{4.54}$$

We remind that each robot can distributively compute $\mathbf{f}_{\text{mean}}$. Furthermore, at this point, each robot is able to estimate the velocity of the center of mass as

$$\hat{\dot{\mathbf{p}}}_C(t) = \widetilde{\dot{\mathbf{p}}}_{C_i}(t) + \hat{\omega}(t)\left(\hat{\mathbf{z}}_i(t) + \hat{\mathbf{z}}_C(t)\right)^{\perp}. \tag{4.55}$$

Therefore, similarly to $\hat{J}_i$, each robot can locally compute an estimate $\hat{m}_i$ using the approach detailed in Appendix A. Finally, the robots agree on a global estimation $\hat{m}$ using an average consensus algorithm that is able to average out the noise of each local mass estimator.

4.5 Numerical Simulations

In this section we validate our approach through numerical simulations. A network of $n = 4$ robots manipulates a *C-shaped* unknown planar object B, see Fig. 4.3. The object has a uniformly distributed mass, $m = 5\,\text{kg}$, and its inertia moment is $J = 8.6891\,\text{kg}\,\text{m}^2$. The robot's communication network is a line topology, *i.e.*, $\mathcal{E} = \{(1,2),(2,3),(3,4)\}$. First, in Section 4.5.1 the ideal case with no noise is considered, then in Section 4.5.2 the noisy case is analyzed.

4.5.1 Ideal Case with no Noise

If there is no noise in the measurements, the algorithm explained in Section 4.3 enables the estimation of the exact values of the inertial parameters of the manipulated body,

as we will show in the following. At the beginning, each robot estimates the relative

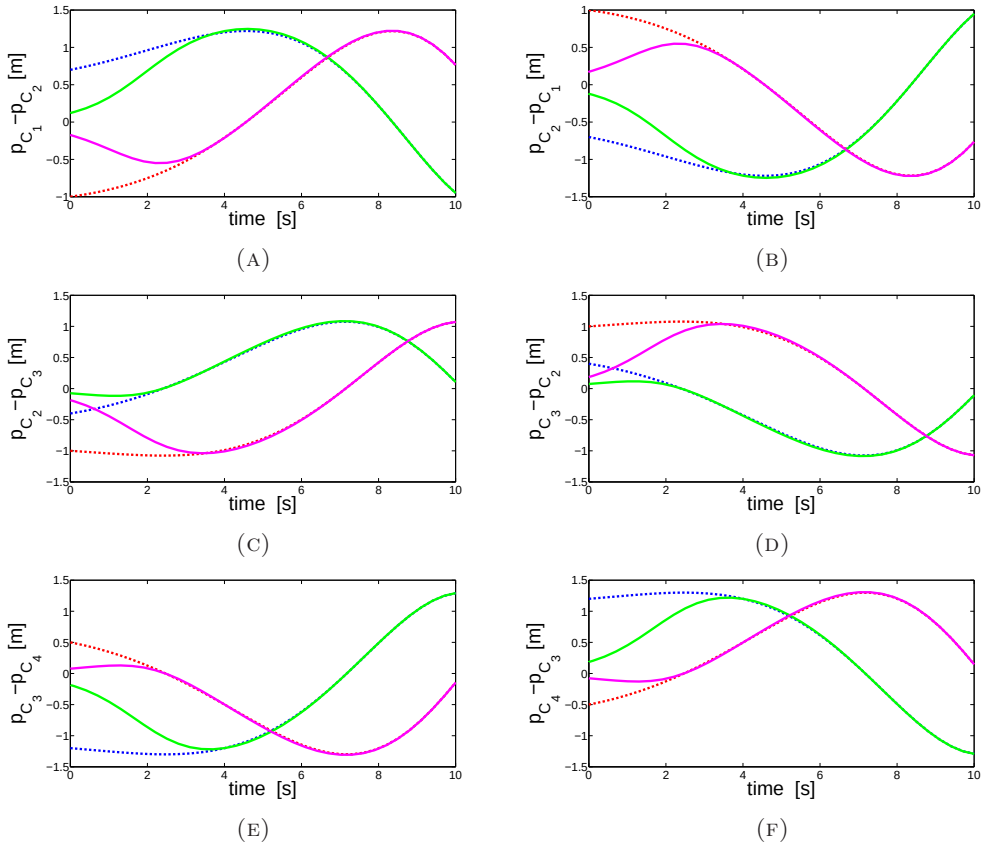
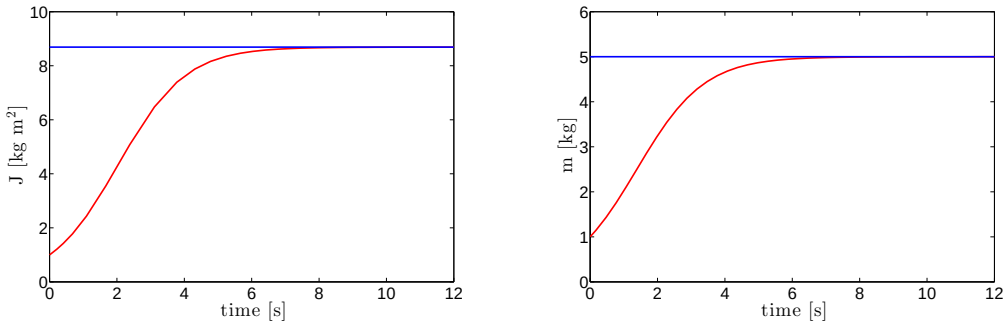


FIGURE 4.4: Estimated coordinates of the displacements between the contact points (continuous lines) vs actual coordinates of the displacements (dashed lines) for (A) $\mathbf{p}_{C_1} - \mathbf{p}_{C_2}$ (B) $\mathbf{p}_{C_2} - \mathbf{p}_{C_1}$ (C) $\mathbf{p}_{C_2} - \mathbf{p}_{C_3}$ (D) $\mathbf{p}_{C_3} - \mathbf{p}_{C_2}$ (E) $\mathbf{p}_{C_3} - \mathbf{p}_{C_4}$ (F) $\mathbf{p}_{C_4} - \mathbf{p}_{C_3}$.

positions of its neighbors by setting a random force $\mathbf{f}_i$. The results of such estimates are reported in Figs. 4.8(A)–4.8(F). Then, the moment of inertia estimation step is executed, where each robot i sets the input force as $\mathbf{f}_i = \mathbf{z}_i^{\perp}$. The initial guess of J is set to $1\,\mathrm{kg\,m^2}$. The evolution of the estimate $\hat{J}$ is illustrated in Fig. 4.5(A). After that, all the information required for the observation of the vector $\mathbf{z}_C$ are available. Since we are only estimating the inertial parameters (thus, the trajectory followed by the body is not important at this stage), the input force is simply set as $\mathbf{f}_i = \mathbf{f} = (0\ f_y)$, for all $i \in \mathcal{I}$. The convergence of the observer to $\mathbf{z}_C$ is illustrated in Fig. 4.6(A), where the value $f_y = J/n$ is used by each robot so that $\check{f}_y = 1$. Figures 4.6(B) and 4.6(C) show the trend of the error $\mathbf{e}$ and of the Lyapunov function used in the proof of the observer $V(\mathbf{e})$, respectively. Both converge quickly to zero, as predicted by our theory. At this point, each robot is able to measure the velocity of the center of mass, $\dot{\mathbf{p}}_C$, in order to estimate the mass m. Its initial guess is set at $\hat{m} = 1\,\mathrm{kg}$. Convergence of the estimated parameter is illustrated in Fig. 4.5(B).



(A) (B)

FIGURE 4.5: Estimation of the inertia moment J (A) and of the mass m (B) of the unknown payload B made by the 4 robots.

4.5.2 Simulations with Measurements' Noise and Accuracy Bounds Definition

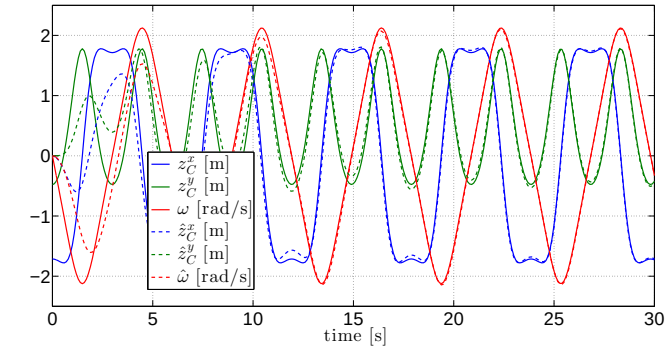
We assume that each robot is able to measure the velocity of the contact point, and that the measurement is affected by a Gaussian noise with zero mean and covariance matrix $\mathbf{\Sigma}_i = \sigma^2\mathbf{I}$, with $\sigma = 0.2$ m/s, and where $\mathbf{I} \in \mathbb{R}^{2\times 2}$ is the identity matrix.

First of all, we formalize the accuracy bound of least square estimations. Suppose to perform a least squares estimation using τ observations (υ_t, ψ_t), $t = 1, \ldots, \tau$, of the model $\psi = \theta\upsilon$, where $\theta \in \mathbb{R}$, $\psi \in \mathbb{R}^p$, and $\upsilon \in \mathbb{R}^p$. The least-squares estimate hat theta of the parameter vector theta is obtained as $\hat{\theta} = (\Upsilon^T\Upsilon)^{-1}\Upsilon^T\Psi$, where $\Upsilon = [\upsilon_1^T \ldots \upsilon_\tau^T]^T$ and $\Psi = [\psi_1^T \ldots \psi_\tau^T]^T$, with $\Upsilon, \Psi \in \mathbb{R}^{\tau\times p}$. The standard deviation $\sigma_{\hat{\theta}}$ of the estimates $\hat{\theta}$ is given by

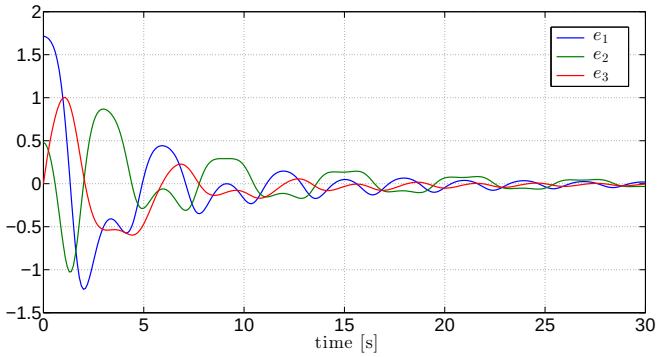
$$\sigma_{\hat{\theta}} = \sigma_\psi \frac{\tau}{\tau \sum_{t=1}^{T} \upsilon_t^2 - (\sum_{t=1}^{T} \upsilon_t)^2}, \tag{4.56}$$

where σ_ψ is the standard deviation of the observations ψ [77]. Thus, the uncertainty of the estimation of the constant parameters $\|\mathbf{z}_{ij}\|$, m, and J has the form of Eq. (4.56).

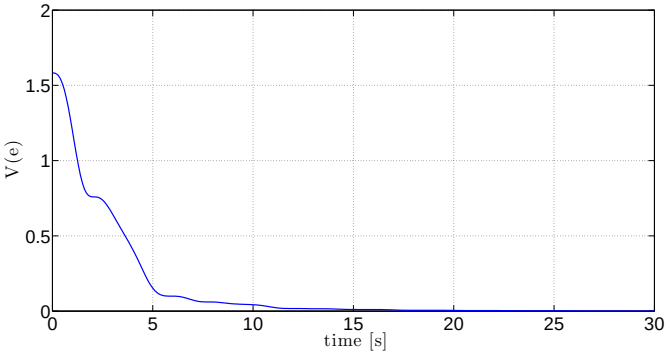
The algorithm starts with the estimation of the relative distance between the neighbors in which each robot sets an arbitrary force. We observe that the estimation must stop when the measured noisy signals $\tilde{\dot{\mathbf{z}}}_{ij}$ have a level such that the signal-to-noise ratio is too low to perform an estimate. In this case, the estimation stops when $\|\tilde{\dot{\mathbf{z}}}_{ij}\| \leq 1$ m/s (the measured $\tilde{\dot{\mathbf{z}}}_{12}$, $\tilde{\dot{\mathbf{z}}}_{23}$, and $\tilde{\dot{\mathbf{z}}}_{34}$ are illustrated in Fig. 4.7). In Fig. 4.8, the estimation of the distances using the two proposed strategies (explained in Sec. 4.4.2.1 and 4.4.2.2, respectively) is illustrated. Subsequently, the next step is executed toward the local estimation of the moment of inertia $\hat{J}_i$. The estimation process starts as soon as robot i has locally collected a sufficient number of samples, and yields the local estimates $\hat{J}_1 = 8.7053 \pm 0.0035$ kg m^2, $\hat{J}_2 = 8.7208 \pm 0.0035$ kg m^2, $\hat{J}_3 = 8.7320 \pm 0.0032$ kg m^2,



(A)



(B)



(C)

FIGURE 4.6: Observation of the center-of-mass position: (A) Cartesian coordinates of the displacement $\mathbf{z}_C = \mathbf{p}_G - \mathbf{p}_C$ and of the angular velocity ω (dashed lines) versus their real values (continuos lines), (B) estimation errors, and (C) plot of the Lyapunov function $V(\mathbf{e})$.

and $\hat{J}_4 = 8.7151 \pm 0.0035$ kg m^2. Each robot checks the convergence of the least squares estimation evaluating the variance of the estimator [77]. Then, the local estimates $\hat{J}_i$ are exchanged over the network and an average consensus is run to agree on the same estimate, notably, $\hat{J} = 8.7183 \pm 0.0004$ kg m^2. The result of such estimation is reported in Fig. 4.9. The estimate of the moment of inertia $\hat{J}$ and the observation of $\hat{\omega}(t)$, known thanks to previous step, are used for the observation of $\mathbf{z}_C$. Also in this case, since we are only estimating the inertial parameters (thus, the trajectory followed by the body is not important at this stage), each robot applies the same constant force $\mathbf{f}_i = \mathbf{f}$, for all

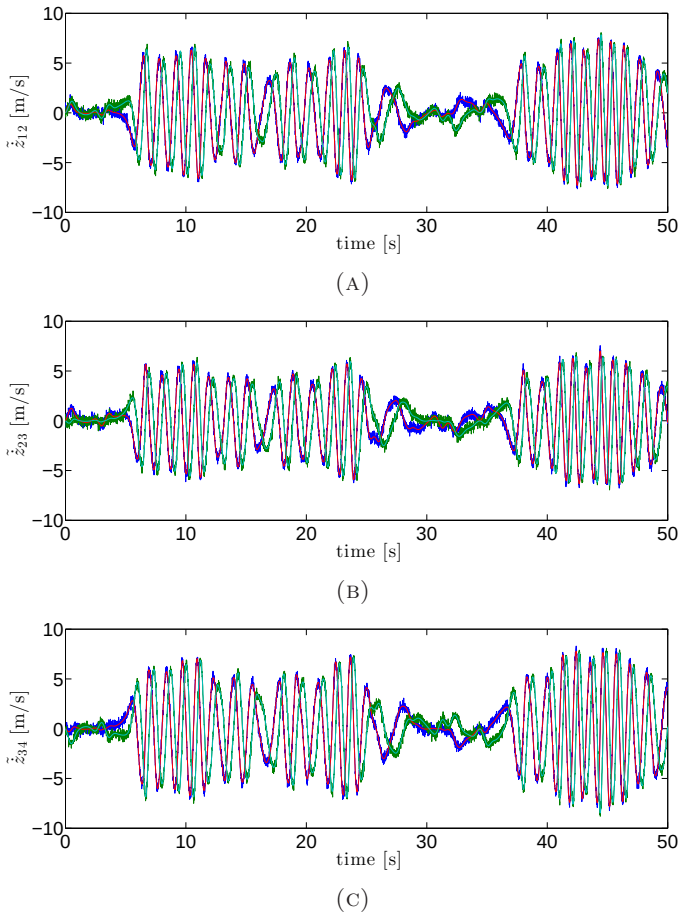


FIGURE 4.7: The measured velocity differences (A) $\tilde{\mathbf{z}}_{12}$, (B) $\tilde{\mathbf{z}}_{23}$, and (C) $\tilde{\mathbf{z}}_{34}$.

$i = 1, \ldots, 4$ in this step. In Fig. 4.10, the observations $\hat{\mathbf{z}}_C$ and ω_{obs} are illustrated. We now characterize numerically the uncertainty of the nonlinear observer in estimating $\mathbf{z}_C$, by running 1000 independent trials for different values of the variance of the noise on the angular rate, $\sigma^2_{\hat{\omega}}$. For each trial, a sinusoidal signal with random amplitude, frequency, and phase is used for the components of the force applied by each robot. In Fig. 4.12, the average, computed on the 1000 trials, of the standard deviation of the estimation error for the coordinates of $\mathbf{z}_C$ is reported. We observe that the trend is almost constant and independent from the value of $\sigma_{\hat{\omega}}$. Therefore, its mean value can be considered as a good approximation of the standard deviation, *i.e.*, $\sigma_{\mathbf{z}^x_C} = 0.075$ m and $\sigma_{\mathbf{z}^y_C} = 0.033$ m. Once the observer converges, the estimation of $\hat{\mathbf{p}}_C$ and $\hat{m}$ can be executed. The estimation of the mass, illustrated in Fig. 4.11, is carried out using the estimation of the angular rate computed by the observer, ω_{obs}. First, each robot estimates locally, respectively, $\hat{m}_1 = 4.8367 \pm 0.0451$ kg, $\hat{m}_2 = 4.8491 \pm 0.0455$ kg, $\hat{m}_3 = 4.8824 \pm 0.0453$ kg, and $\hat{m}_4 = 4.8384 \pm 0.0447$ kg and then, after the average consensus, the network agrees on the value $\hat{m} = 4.8517 \pm 0.0113$ kg.

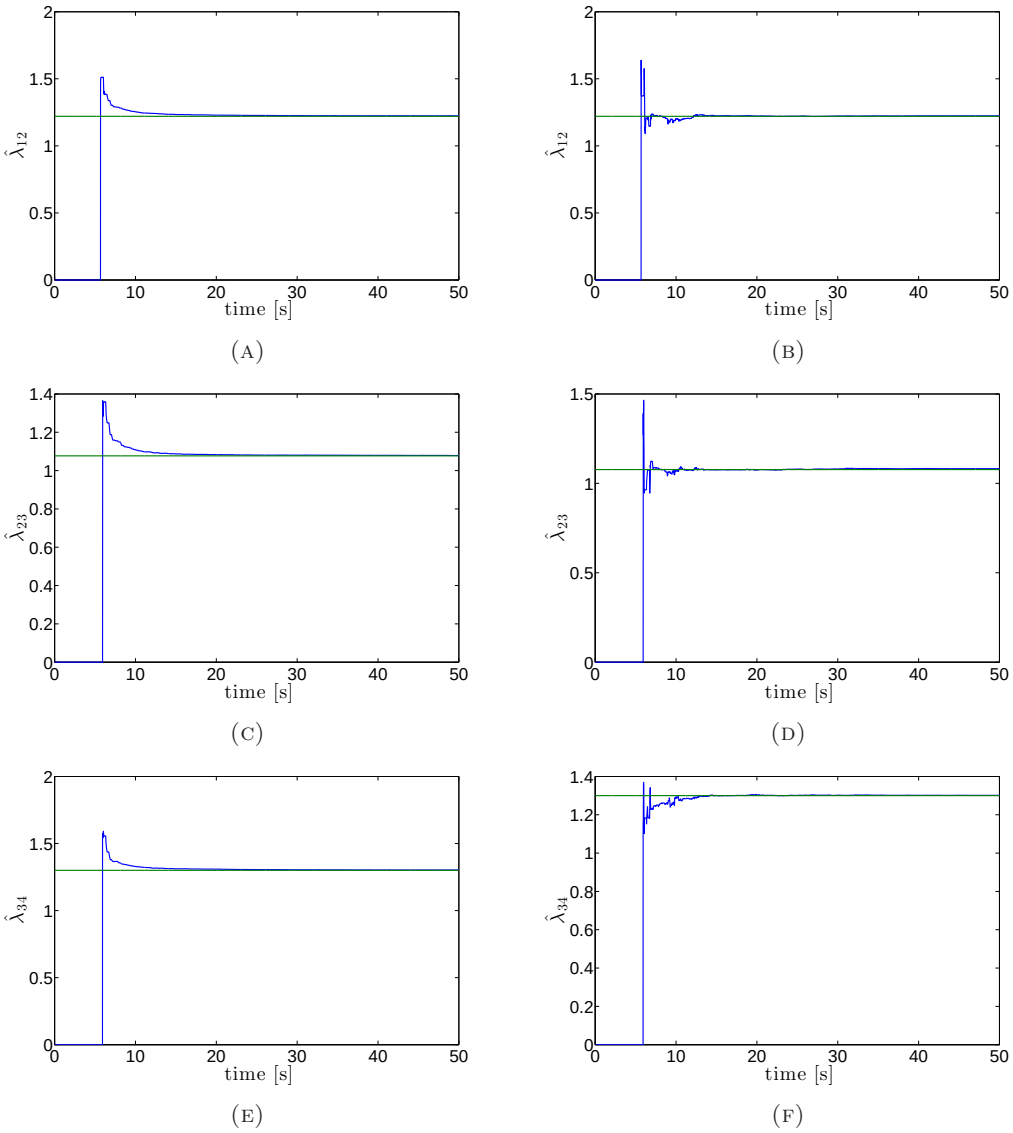
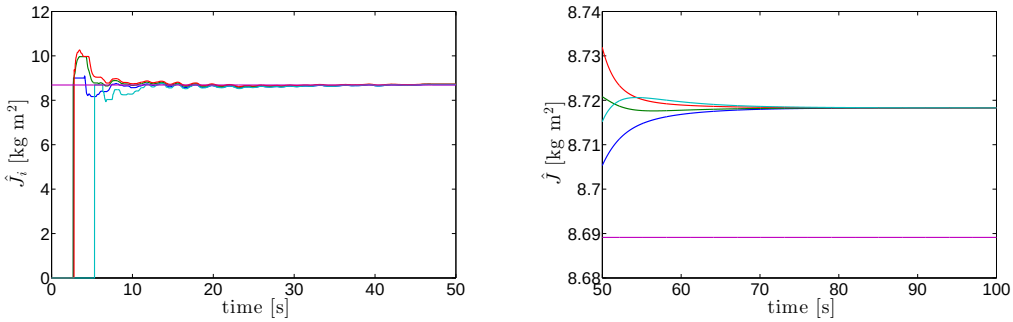


FIGURE 4.8: Estimation of the relative distance between neighbors in the network using two strategies: (A) $\hat{\lambda}_{12}$ using the first strategy (B) $\hat{\lambda}_{12}$ using the second strategy (C) $\hat{\lambda}_{23}$ using the first strategy (D) $\hat{\lambda}_{23}$ using the second strategy (E) $\hat{\lambda}_{34}$ using the first strategy (F) $\hat{\lambda}_{34}$ using the second strategy.

4.6 Conclusion

In this chapter, we have presented an effective distributed strategy for the estimation of inertial parameters of an unknown body manipulated by a team of networked mobile robots. The estimation is performed through a series of steps, that eventually yields a complete estimation in finite time. All the assumptions made are realistic. In particular, we do not assume that the robots' velocity can be controlled, yet applied forces have to be measured and controlled. Moreover, only the velocity of contact points should be measured, whereas their positions and accelerations are not needed. First, we addresses the ideal case with no noise in the measurements, then we focused on the influence of the



(A) (B)

FIGURE 4.9: Estimation of the moment of inertia of B. (A) Least Squares estimation of J: the estimate converges as soon as the number of samples is sufficient. (B) After the convergence of the estimator, the network runs an average consensus in order to reach an agreement on $\hat{J}$.

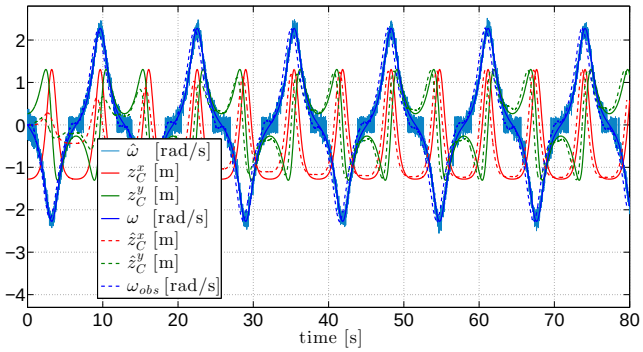
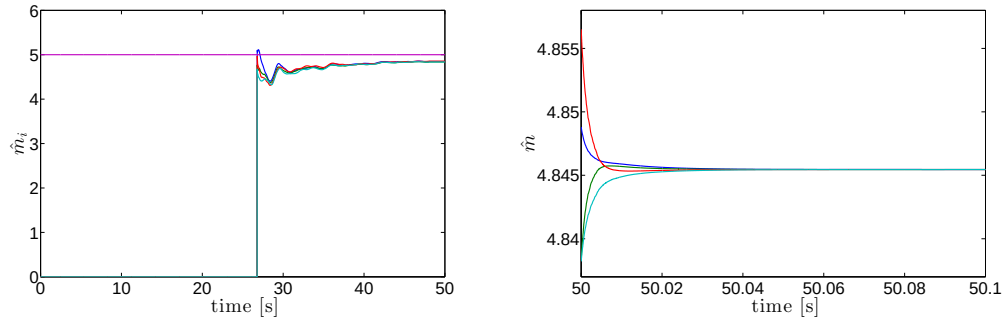


FIGURE 4.10: The observation of the vector $\mathbf{z}_C$ and of the angular rate ω: dashed lines refer to observed values, while continuos lines refer to real values. For the angular rate, the measure $\hat{\omega}$ (continuos light blue line) in input to the observer is also plotted.

measurement noise on the estimate by defining suitable solutions and confidence intervals for the estimated quantities. The proposed strategies involve low computational burden, simulation results are very satisfactory and confirm the effectiveness of our approach.



(A) (B)

FIGURE 4.11: Estimation of the mass of B. (A) Least squares estimation of m: the estimate converges as soon as the number of samples is sufficient. (B) After the convergence of the estimator, the network runs an average consensus in order to reach an agreement on $\hat{m}$.

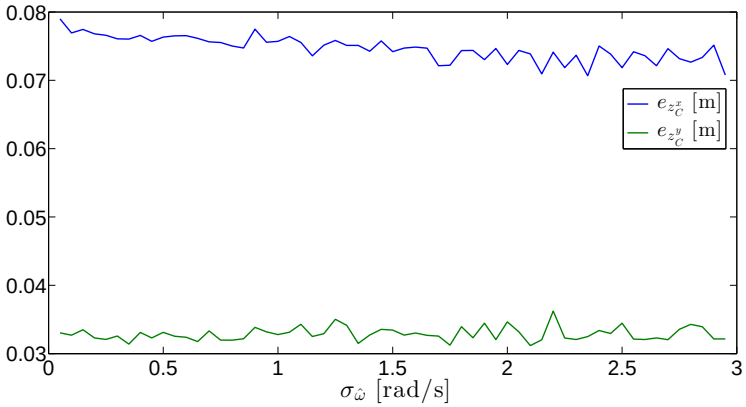


FIGURE 4.12: The average of the standard deviation of the estimation error for the coordinates of $\mathbf{z}_C$, $\mathbf{z}^x_C$ and $\mathbf{z}^y_C$ as a function of the standard deviation of the noise of the angular rate $\sigma_{\hat{\omega}}$. Results are averaged over 1000 trials.

Chapter 5

Asynchronous Max-Consensus Protocol with Time Delays

This chapter deals with the analysis of the convergence properties of the max-consensus protocol with asynchronous updates and bounded time delays on directed static networks. The work is motivated by real-world applications in distributed decision-making systems, for which max-consensus is an effective paradigm. The main result presented in this chapter is that the strongly connectedness of the directed communication network is a sufficient condition for the convergence of the asynchronous max-consensus protocol in finite time. Implementation issues are also taken into account, by complementing the theoretical analysis with the definition of a mechanism to detect the convergence of the protocol in a distributed fashion.

The content of this chapter is grounded on a journal paper and conference papers [20] [78] [79].

5.1 An Introduction to Asynchronous Consensus Protocols

Time synchronization is an important requirement for the correct design and implementation of consensus protocols. Much work in the literature assumes that consensus protocols work in synchronous frameworks, that is, all the clocks of the network's nodes are synchronized in frequency and phase. Obviously, this scenario is very often far from reality, where at least drifts and offsets in nodes' clock operations are present, up to cases involving networks of heterogeneous nodes and unreliable communication channels. In these and many other cases, asynchronism is an explicit issue to cope with. Due to strict

constraints on bandwidth and energy consumption, typically present in distributed systems applications, the adoption of suitable time synchronization strategies may not be a feasible solution [80] [81]. Thus, convergence properties of consensus protocols should be revisited in presence of asynchronous updates and delays. The convergence properties of the average consensus protocol in presence of time delays have been extensively studied in [82] [83]. On the other hand, despite of its wide range of application, only little work has been devoted to the analysis of the max-consensus protocol beyond its synchronous implementation. In particular, convergence has been studied for synchronous switching topologies [84] and for probabilistic asynchronous frameworks [85], respectively.

This chapter deals with the definition of an asynchronous discrete-time max-consensus protocol and the study of its convergence properties. The concept of asynchronism considered in this chapter encompasses several situations, from misalignments among nodes' clocks, to the presence of bounded time delays. Here, we assume that the delays in the updates of the state variables of the network nodes cannot be arbitrarily long (specifically, we adopt the partial asynchronism assumption [86]), and that the network is described by a strongly connected, directed graph. These assumptions are mild, and usually verified when dealing with real asynchronous message-passing systems. The convergence analysis is performed establishing an equivalence between the asynchronous protocol, running on a static network, and a suitable synchronous protocol running on a switching network with the same node set. Notably, the synchronous protocol can be modeled as a particular case of the asynchronous one. The convergence analysis is carried out relying on the analytic synchronization formalism [87] and on convergence results for synchronous settings [88]. It is worth to observe that the results presented in this chapter for the max-consensus protocol can be used to solve also min-consensus problems. In fact, a min-consensus protocol can be always rewritten as a max-consensus one by multiplying the nodes' state values by -1.

The chapter is organized as follows. Section 5.2 provides the necessary background. Then, a motivating example that emphasizes the problems related to the ineffective application of the synchronous max-consensus protocol in an asynchronous framework is given in Section 5.3. The asynchronous max-consensus problem, together with its equivalent synchronous model, is defined in Section 5.4. Section 5.5 presents the convergence analysis of the asynchronous, discrete-time max-consensus protocol. Convergence properties are studied relying on a global knowledge of the network. However, convergence cannot be detected at the node level without introducing additional information. This issue is crucial to implement the protocol in applications. Thus, we also define a strategy for the distributed detection of convergence. The performance of the asynchronous protocol is assessed in Section 5.6 through numerical simulations. Finally, conclusions and suggestions for future work are given in Section 5.7.

5.2 Problem Statement

In this section, we recall some background on networks of asynchronous systems and we review some useful results on the theory of synchronous max-consensus protocols.

5.2.1 Node and Network Model

We consider a network of $n > 1$ discrete-time dynamical systems, with index set $\mathcal{I} = \{1, \dots, n\}$, operating in an asynchronous communication framework. By means of asynchronism, we encompass different realistic scenarios: (i) synchronous systems with non uniform communication delays that make the delivery of relevant information asynchronous; (ii) networks of nodes with different and/or not synchronized clock periods; (iii) a combination of the previous cases. Nodes' clocks are not required to own a constant update period, making the presented framework suitable to describe also event-driven systems. We denote with $t_k^{[i]}$, $k \in \mathbb{N}_0$, the k-th update time for the i-th node, marked by its local clock. Update times for the generic node $i \in \mathcal{I}$ are collected in a suitable set, denoted $\mathcal{T}^{[i]} = \{t_0^{[i]}, t_1^{[i]}, t_2^{[i]}, \dots, t_k^{[i]}, \dots\}$. Time $t_0^{[i]}$ is the i-th node's starting time, and no assumptions are made on the synchronization of starting times across the network. Thus, the state evolution of each node must be referred to its local discrete time base. Consequently, a partial order among the update times of all the network nodes cannot be set without referring to an external, common time base. The information state of node i at time $t_k^{[i]}$ is indicated with $x_i(t_k^{[i]}) \in \mathbb{R}$. For the sake of simplicity, we consider the information state as a real value in order to ensure the proper definition of the max operator. However, more complex types of information states can be considered, provided that a proper ordering criterion is adopted.

Communication among network nodes is modeled through a static directed graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ (hereinafter digraph), where the node set $\mathcal{I}$ represents the set of dynamical systems, and the communication channels are modeled by the link set $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$. Assuming that the communication structure is represented by a digraph implies that, if an ordered pair (j, i) belongs to $\mathcal{E}$, then node i is able to receive information by node j, yet not necessary vice versa. The set of one-hop neighboring nodes from which node i can receive information is defined as $\mathcal{N}_i = \{j \in \mathcal{I} | (j, i) \in \mathcal{E}\}$. Here, we assume the existence of self-loops, that is, $(i, i) \in \mathcal{E}$, $\forall i \in \mathcal{I}$. In the following assumption, we formalize the presence of bounded communication delays.

Assumption 5.1 (Bounded communication delays). Let us define $\theta_{k_q}^{ij} \in \mathbb{R}_+$ the time at which the information produced by node $j \in \mathcal{N}_i$ at some update time $t_q^{[j]}$, $q \in \mathbb{N}_0$, is delivered to node i (here, $\theta_{k_q}^{ij}$ is measured with reference to $t_0^{[i]}$). It is assumed that, for each node $i \in \mathcal{I}$, and for each delivery time $\theta_{k_q}^{ij}$, $j \in \mathcal{N}_i$, $q \in \mathbb{N}_0$, the constraint

$0 \leq d_q^{ij} \leq \check{d}$, $\check{d} \in \mathbb{R}_+$, holds, where d_q^{ij} is the delivery delay, *i.e.,* the time interval elapsed between $t_q^{[j]}$ and $\theta_{k_q}^{ij}$.

Assumption 5.2 (Strong connectivity). The digraph $\mathcal{G}$ is assumed to be *strongly connected*, that is, there always exists a directed path between any two nodes of the digraph [40].

A path in the network is an ordered sequence of links connecting two arbitrary nodes. The length of a path is the number of links in the path. In a strongly connected digraph, the longest of the shortest paths connecting any two nodes is the graph diameter, and is indicated with $\mathcal{D}$ [40].

The adjacency matrix $\mathcal{A} = [a^{ij}]$ of the digraph $\mathcal{G}$ describes its connection scheme and is defined as

$$a^{ij} = \begin{cases} 1 & \text{if } (j,i) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \tag{5.1}$$

As previously stated, due to the existence of self-loops, we assume that $a^{ii} = 1$, $\forall i \in \mathcal{I}$.

A digraph $\mathcal{G}$ is called static if the edge set $\mathcal{E}$ is time-invariant. On the other hand, a dynamic digraph $\mathcal{G}_k = (\mathcal{I}, \mathcal{E}_k)$, where k is a discrete-time index, is a digraph with a time-varying edge set. Dynamic digraphs are useful to model communication structures that change over time and can be consistently described by time-varying adjacency matrices.

A set of dynamic digraphs $\mathcal{G}_1, ..., \mathcal{G}_r$, $r \in \mathbb{N}$, is jointly strongly connected if the union digraph $\mathcal{U} = \bigcup_{p=1}^{r} \mathcal{G}_p$, with vertex and edge set defined as $\mathcal{U} = (\mathcal{I}, \bigcup_{p=1}^{r} \mathcal{E}_p)$, is strongly connected.

5.2.2 The Synchronous Max-Consensus Protocol

Consider a set of $n > 1$ networked discrete-time dynamical systems with synchronized update times, *i.e.,* $t_k^{[i]} = t_k$, $\forall i \in \mathcal{I}$, $k \in \mathbb{N}_0$, which communicate over a digraph with no delays (*i.e.,* $d_q^{ij} = 0$, $\forall (j,i) \in \mathcal{E}$, $q \in \mathbb{N}_0$). The state variable of each system is indicated with $\breve{x}_i \in \mathbb{R}$, $i \in \mathcal{I}$ (hereinafter, variables with the superimposed symbol $\breve{}$ refer to synchronous protocols). The synchronous max-consensus protocol $\breve{\mathcal{P}}$ is defined as

$$\breve{x}_i(t_{k+1}) = \max_{j \in \mathcal{N}_i} \{\breve{x}_j(t_k)\}, \quad i \in \mathcal{I}. \tag{5.2}$$

Theorem 5.1 (Convergence of the synchronous max-consensus protocol)**.** *Consider a network of $n > 1$ dynamical systems, connected through a static digraph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ with diameter $\mathcal{D}$. Suppose that each node runs the synchronous max-consensus protocol*

expressed by Eq. (5.2). If $\mathcal{G}$ is strongly connected, then for $k \geq \mathcal{D}$

$$\begin{aligned}\breve{x}_i(t_k) &= \breve{x}_j(t_k) \\ &= \max\Big(\breve{x}_1(t_0), \cdots, \breve{x}^{[n]}(t_0)\Big) \quad \forall i, j \in \mathcal{I}\end{aligned}$$

holds.

Proof. The proof of Theorem 5.1 is reported in [16]. □

5.3 A Motivating Example

In this section, we provide an illustrative example that highlights the issues related to the ineffective application of the synchronous max-consensus protocol to asynchronous settings. The selected example concerns the application of a decentralized solution to the well-known task assignment problem, for which the execution of the max-consensus protocol is required [89]. The assignment algorithm described in the following is designed for networks of nodes that operate synchronously. Hence, its correct execution is guaranteed only in synchronous networks. In this example, we will show that the same algorithm may not work properly in asynchronous settings.

5.3.1 Synchronous Decentralized Task Assignment

Consider a network of $n > 1$ nodes, with index set $\mathcal{I} = \{1, \ldots, n\}$, as described in Section 5.2.1, where a single task has to be assigned to only one of the n nodes in a distributed fashion, according to some optimality criterion. Each node $i \in \mathcal{I}$ autonomously sets a static non-negative reward (or bid) $x_i \in \mathbb{R}_+$ related to the execution of the task. Setting the reward is also called bidding. The goal consists of performing the task assignment in a distributed fashion, so that only node $i^\star$ that bid the maximum reward $x_{i^\star}$ is granted access to the task execution (*i.e.*, a conflict free assignment).

A common strategy to the distributed solution of this task assignment problem is based on the max-consensus protocol [90].

In a synchronous setting, each node $i \in \mathcal{I}$ starts the algorithm, placing a bid x_i for the task. Then, each node shares its bid with its neighbors $\mathcal{N}_i$ in order to converge, through the application of the max-consensus protocol, towards a feasible assignment. The adopted updating rule is expressed in Eq. (5.2) where, in this case, the state information is represented by the node's reward. As stated in Theorem 5.1, the number of updates required to converge to a feasible assignment is $\mathcal{D}$, where $\mathcal{D}$ is the diameter of

FIGURE 5.1: A motivating example: the communication topology of a network of three nodes, with diameter $\mathcal{D} = 2$.

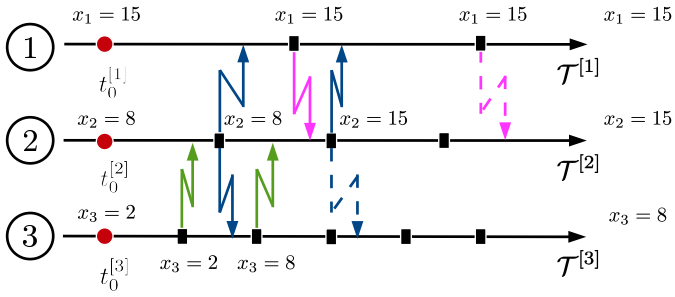


FIGURE 5.2: A motivating example: the time line of each node. Ticks indicate a max-consensus update for node i. An arrow indicates directed communication; a dashed arrow indicates a communication which is not effective for the receiver node. Beside each update, the current state of the corresponding node (maximum bid) is indicated.

the communication digraph $\mathcal{G}$. Supposing that each node is aware of the network diameter, the upper bound $\mathcal{D}$ can be used locally by each node as the termination condition for the execution of the protocol, that is to say, each node is aware that the convergence is achieved if it runs the protocol for exactly $\mathcal{D}$ updates. As said before, the crucial issue is that the max-consensus protocol requires the network nodes' operations to be synchronized. If this condition does not hold, the network nodes may process outdated bids and the consensus protocol may produce invalid assignments. To highlight this issue, in the following we analyze in detail the operations executed in this motivating example, in which a synchronous protocol is applied despite the fact that the underlying framework is actually asynchronous.

5.3.2 Decentralized Task Assignment: an Asynchronous Case

Consider a network of $n = 3$ nodes, connected in a line topology (see Fig. 5.1), where a single task has to be assigned in a distributed fashion to the node that sets the highest reward. We assume that the network diameter, that is $\mathcal{D} = 2$ in this example, is known by all nodes. The generic node i runs the max-consensus protocol at each time instant $t_k^{[i]}$, $k = 1, 2$, of its local time base, assuming to work in a synchronous setting. Now, we show that the presence of time misalignments and bounded delays leads to a wrong assignment. With reference to Fig. 5.2, we assume that the nodes' bids on the task are produced at times $t_0^{[i]}$, $i = 1, 2, 3$, and are respectively: $x_1(t_0^{[1]}) = 15$, $x_2(t_0^{[2]}) = 8$, $x_3(t_0^{[3]}) = 2$. This implies that the task should be assigned to node 1, which is the one providing the maximum bid. Then, each node starts to execute the max-consensus protocol. Node 3 computes the maximum of its received bids, $x_3(t_1^{[3]}) =$

$\max\{x_3(t_0^{[3]})\}$, and communicates its information state $x_3(t_1^{[3]})$ to node 2 at first. We note that, to ensure the proper functionality of the algorithm, an assignment $\mu^{[i]} = \underset{i\in\mathcal{I}}{\text{argmax}}\ \{x_i\}$, must be suitably updated at each node i updating time, together with the corresponding state. For the sake of clarity, this assignment is not reported in the example description. Node 2 receives the information state from node 3, updates it as $x_2(t_1^{[2]}) = \max\{x_2(t_0^{[2]}), x_3(t_1^{[3]})\} = 8$, and sends it to both nodes 1 and 3. Node 3 updates for the second and last time its state as $x_3(t_2^{[3]}) = \max\{x_3(t_1^{[3]}), x_2(t_1^{[2]})\}$. Therefore, node 3 loses the assignment of the task and considers node 2 as the winner ($x_3(t_2^{[3]}) = 8$), since node 3 has not received the state of node 1, due to time misalignment. Similar to node 3, node 1 receives the information state $x_2(t_1^{[2]})$ from node 2 and, consequently, its updated state is $x_1(t_1^{[1]}) = \max\{x_1(t_0^{[1]}), x_2(t_1^{[2]})\} = 15$, which is sent to node 2. Node 2, which has instead received all the state values in its second update, reaches the correct value $x_2(t_2^{[2]}) = 15$ (thus, it considers node 1 as winner). Finally, node 1, after having received the most recent information state $x_2(t_2^{[2]})$ from node 2, correctly assigns the task to itself ($x_1(t_2^{[1]}) = 15$). As a result, at the end of the execution of the max-consensus protocol, the task is assigned with conflict. This implies that, due to the asynchronous setting, not all the nodes are aware of the correct assignment.

Further in the chapter (specifically, in Sec. 5.6), we will show how this issue can be sorted out by applying the asynchronous max-consensus protocol, proposed hereinafter, to the same example.

5.4 Asynchronous Max-Consensus: Synchronous Equivalent Model

In this section, we introduce the asynchronous max-consensus protocol and describe the analytic synchronization procedure adopted to define an equivalent synchronous model of the asynchronous system, useful for subsequent analysis.

5.4.1 Asynchronous Max-Consensus

Suppose that node $i \in \mathcal{I}$ updates its information state at each of its local update times $t_k^{[i]} \in \mathcal{T}^{[i]}$, $k \in \mathbb{N}$, on the basis of the value of its state variable at the previous update time, and on the most recent values of the state variables received from the neighboring nodes. We assume that delays are detectable (*e.g.,* through a sequence number) and, in the following, we suppose that the most-recent-data strategy can be adopted [91].

The asynchronous max-consensus protocol $\mathcal{P}$ on the digraph $\mathcal{G}$ is defined by the following local state update rule, executed by each node $i \in \mathcal{I}$ at local update times in $\mathcal{T}^{[i]}$:

$$x_i(t_{k+1}^{[i]}) = \max_{j \in \mathcal{N}_i} \{x_j(\theta_{k_q}^{ij})\}, \quad i = 1, \ldots, n, \quad k_q \in \mathbb{N}_0. \tag{5.3}$$

As stated in Assumption 5.1, variable $\theta_{k_q}^{ij} \in \mathbb{R}_+$ indicates the time instant at which the most recent state value produced by the neighboring node $j \in \mathcal{N}_i$ at time $t_q^{[j]} \in \mathcal{T}^{[j]}$, $q \in \mathbb{N}_0$, is available to node i before time $t_{k+1}^{[i]} \in \mathcal{T}^{[i]}$. Thus, $t_0^{[i]} \leq \theta_{k_q}^{ij} < t_{k+1}^{[i]}$ holds $\forall j \in \mathcal{N}_i$ and for a given $t_q^{[j]}$. The informative value conveyed by $x_j(\theta_{k_q}^{ij})$ is the same produced by node j at time $t_q^{[j]}$, *i.e.*, $x_j(\theta_{k_q}^{ij}) = x_j(t_q^{[j]})$ (we remark that $\theta_{k_q}^{ij}$ is measured with reference to $t_0^{[i]}$). To improve readability, the dependence of $\theta_{k_q}^{ij}$ on the index q of the time instant $t_q^{[j]} \in \mathcal{T}^{[j]}$ is omitted hereinafter.

The asynchronous system under exam is defined by the triple

$$\mathcal{S} = \langle \mathcal{G}, \mathcal{P}, \mathcal{T} \rangle, \tag{5.4}$$

where $\mathcal{G}$ is the static digraph representing the communication structure, $\mathcal{P}$ is the asynchronous max-consensus protocol defined in Eq. (5.3), and $\mathcal{T}$ is the unordered set of all the update times of all the nodes in the network, obtained as $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}^{[i]}$.

Problem 5.1 (Asynchronous max-consensus problem). Given system $\mathcal{S}$ defined in triple (5.4), max-consensus is achieved in finite time if, under the execution of protocol $\mathcal{P}$ in (5.3), $\forall i \in \mathcal{I}$, there exists a $\beta_i \in \mathbb{N}$ s.t. $\forall k \geq \beta_i$, $k \in \mathbb{N}$, $x_i(t_k^{[i]}) = \max_{j \in \mathcal{I}} \{x_j(t_0^{[j]})\}$ holds.

In the following, the convergence properties of the asynchronous max-consensus protocol described by (5.3) are studied. A sufficient condition for convergence is given, as well as an upper bound on the convergence time, under mild hypotheses. Results are derived through the use of the analytic synchronization method [87], by establishing the equivalence between the asynchronous protocol $\mathcal{P}$ performed on the static digraph $\mathcal{G}$ and an equivalent synchronous protocol executed on a suitably defined switching topology. Through this technique, we prove that the strongly connectedness of the communication digraph $\mathcal{G}$ is a sufficient condition for the convergence of the asynchronous max-consensus protocol.

5.4.2 Virtual Time Base

As stated before, due to the very general assumptions made in this study, it is not possible to order the set of nodes' update times without referring them to an external, common time base. On the other hand, the definition of such a time base is needed for analysis purposes. A global ordering of the time instants in the set $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}^{[i]}$ is

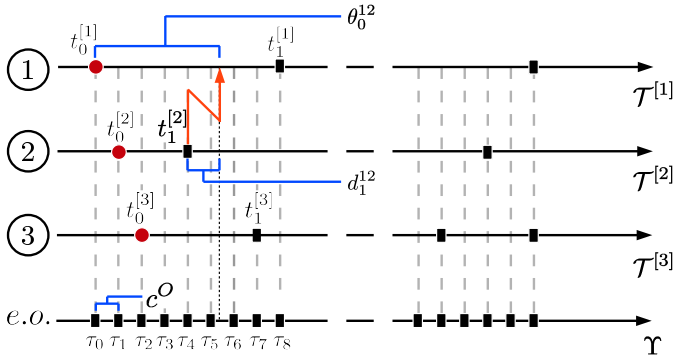


FIGURE 5.3: Update times for three different nodes with variable update times, connected as in Fig. 5.1 (the label *e.o.* means *external observer*). The time instant θ_0^{12} at which the most recent state value of node 2 is received by node 1, the delivery delay d_1^{12} *i.e.,* the time interval elapsed between $t_1^{[2]}$ and θ_0^{12}, and the constant clock c^O of the virtual time base are shown.

achievable assuming the point of view of a hypothetical external observer of the network. Let us denote with $\mathcal{T}^S$ the sorted version of the set $\mathcal{T}$. Following the ideas underlying the analytic synchronization method [86] [87], the definition of $\mathcal{T}^S$ allows for the appropriate formalization of a common discrete time base as

$$\Upsilon = \{\tau_h | \tau_h = h \cdot c^O, h \in \mathbb{N}_0, \tau_h \in \mathbb{R}_+\}, \tag{5.5}$$

where c^O is a constant clock period selected as the biggest time interval contained, an integer number of times, in every time interval occurring between two consecutive update times in the sorted set $\mathcal{T}^S$ (see Fig. 5.3). In the following, we will refer to Υ as the virtual time base, since it is not actually possessed by any node, yet it is only used for analysis purposes. Thus, a correspondence between any of the update times of the state of the nodes and a specific virtual time instant can be set. Note that this correspondence is not a bijection, that is, each element of $\mathcal{T}^S$ corresponds to an element of Υ, whereas the opposite may not be true, *i.e.,* $|\mathcal{T}^S| \leq |\Upsilon|$ holds. We formalize this relation by means of the following function:

$$f : \mathcal{T}^S \to \Upsilon. \tag{5.6}$$

The expression $f(t_k^{[i]}) = \tau_{k_i}$, $k \in \mathbb{N}_0$, means that given $t_k^{[i]} \in \mathcal{T}^{[i]}$, with $\mathcal{T}^{[i]} \subseteq \mathcal{T}^S$ and $i \in \mathcal{I}$, $\tau_{k_i} \in \Upsilon$ represents its corresponding virtual update time. Hereinafter, notation $\tau_{k_i} \in \mathcal{T}^{[i]}$ denotes that there exists a virtual time instant $\tau_{k_i} \in \Upsilon$ and a time instant $t_k^{[i]} \in \mathcal{T}^{[i]}$, such that $\tau_{k_i} = f(t_k^{[i]})$.

5.4.3 Equivalent Synchronous Model

In this section, we show that system $\mathcal{S}$, which operates with the asynchronous update rules of protocol $\mathcal{P}$ in (5.3) over a static digraph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, is equivalent to a system $\breve{\mathcal{S}}$ where a synchronous protocol $\breve{\mathcal{P}}$ is run on a switching digraph $\breve{\mathcal{G}}_h = (\mathcal{I}, \mathcal{E}_h)$, $h \in \mathbb{N}_0$, and h indexes the virtual time base instants $\tau_h \in \Upsilon$. Similarly to the virtual time base, also the switching structure can only be revealed by observing system $\mathcal{S}$ from an external, global point of view. In particular, the node sets of $\mathcal{G}$ and $\breve{\mathcal{G}}_h$ coincide for all $h \in \mathbb{N}_0$, whereas the time-varying edge sets of digraphs $\breve{\mathcal{G}}_h$ refer to the effective data exchange flows, possibly unidirectional, occurring between pairs of nodes during the virtual time windows $(\tau_{h-1}, \tau_h]$, $h \in \mathbb{N}$. In particular, given node $i \in \mathcal{I}$, a directed edge from a node $j \in \mathcal{N}_i$ to node i is switched on at τ_h if and only if $\tau_h = \tau_{k_i}$, with $\tau_{k_i} \in \mathcal{T}^{[i]}$. Since we assume that every node can access its own state, this yields $(i,i) \in \mathcal{E}_h$, $\forall \tau_h \in \Upsilon$. Thus, the digraph $\breve{\mathcal{G}}_h$, evolving along the virtual time base Υ, can be described by a time-varying adjacency matrix $\breve{\mathcal{A}}_h \in \{0,1\}^{n \times n}$, whose generic element $\breve{a}_h^{ij} \in \{0,1\}$ is defined as

$$\breve{a}_h^{ij} = \begin{cases} a^{ij} & \text{if } \tau_h = \tau_{k_i},\ j \in \mathcal{N}_i \backslash \{i\} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} . \tag{5.7}$$

We define the time-varying set of neighbors of node i at time $\tau_h \in \Upsilon$, $h \in \mathbb{N}_0$, as

$$\breve{\mathcal{N}}_i(h) = \{j \in \mathcal{I} | \breve{a}_h^{ij} = 1\}, \tag{5.8}$$

where $\breve{\mathcal{N}}_i(h) \subseteq \mathcal{N}_i$, $\forall h \in \mathbb{N}_0$, holds.

Let us consider the following synchronous max-consensus protocol $\breve{\mathcal{P}}$, running over the switching digraph $\breve{\mathcal{G}}_h$:

$$\begin{cases} \breve{x}_i(\tau_{h+1}) = \max\limits_{j \in \breve{\mathcal{N}}^{[i]}} \{\breve{x}_j(\tau_h)\} & \forall \tau_h \in \mathcal{T}^{[i]} \qquad (i) \\ \breve{x}_i(\tau_{h+1}) = \breve{x}^{[i]}(\tau_h) & \forall \tau_h \notin \mathcal{T}^{[i]} \qquad (ii) \end{cases} . \tag{5.9}$$

According to protocol $\breve{\mathcal{P}}$ in (5.9), every node $i \in \mathcal{I}$ updates its state synchronously, at time $\tau_h \in \Upsilon$. Differently from protocol $\mathcal{P}$ in (5.3), though, its neighbor set changes over the time instants τ_h. We refer to the triple

$$\breve{\mathcal{S}} = \langle \breve{\mathcal{G}}, \breve{\mathcal{P}}, \Upsilon \rangle \tag{5.10}$$

to define the synchronous system described above, where the synchronous protocol $\breve{\mathcal{P}}$ in (5.9) is run over the switching topology $\breve{\mathcal{G}}$, described by the adjacency matrix (5.7),

along the virtual time base Υ expressed by (5.5).
The synchronous max-consensus problem, related to system (5.10), is defined as:

Problem 5.2 (Synchronous max-consensus problem). Given system $\breve{\mathcal{S}}$, protocol $\breve{\mathcal{P}}$ solves the synchronous max-consensus problem if there exists an integer $\breve{\beta} \in \mathbb{N}$ s.t. $\forall \breve{k} \geq \breve{\beta}$, $\breve{k} \in \mathbb{N}$, $\breve{x}^{[i]}(\tau_{\breve{k}}) = \max_{j\in\mathcal{I}}\{\breve{x}^{[j]}(\tau_{0_j})\}$, $\forall i \in \mathcal{I}$, holds.

In the following section, relying on the equivalence between systems $\mathcal{S}$ and $\breve{\mathcal{S}}$, we prove the convergence of the asynchronous protocol $\mathcal{P}$ in $\mathcal{S}$.

5.5 Asynchronous Max-Consensus: Convergence

In this section, we provide the main result of this chapter, that is, the analysis of the convergence properties of the max-consensus protocol in presence of asynchronous updates and time delays. The study relies on a global knowledge of the network, therefore, convergence cannot be detected at local level. Thus, we also define a strategy for the distributed detection of convergence. Then, we discuss some implementation issues and, finally, we consider again the single-task assignment problem introduced in Section 5.3 in order to show the effectiveness of our approach.

5.5.1 Convergence Analysis

Consider a networked system $\mathcal{S}$ characterized by triple (5.4). After having derived the synchronous system $\breve{\mathcal{S}}$ expressed in (5.10), we now state the equivalence result.

Proposition 5.2. *If system $\breve{\mathcal{S}}$ in (5.10) reaches max-consensus in finite time, that is, if protocol $\breve{\mathcal{P}}$ solves the synchronous max-consensus problem 5.2 over the switching network $\breve{\mathcal{G}}_h$, $h \in \mathbb{N}_0$, then system $\mathcal{S}$ in (5.4) reaches max-consensus in finite time, that is, protocol $\mathcal{P}$ solves the asynchronous max-consensus problem 5.1 over the static network $\mathcal{G}$. Moreover, if the two state variable sets $\{\breve{x}_i\}_{i\in\mathcal{I}}$ and $\{x_i\}_{i\in\mathcal{I}}$, of systems $\breve{\mathcal{S}}$ and $\mathcal{S}$ respectively, assume identical initial values, i.e., $\breve{x}_i(\tau_{0_i}) = x_i(t_0^{[i]})$, $\forall i \in \mathcal{I}$, then the two systems converge to the same value, that is,* $\max_{i\in\mathcal{I}}\{\breve{x}_i(\tau_{0_i})\} = \max_{i\in\mathcal{I}}\{x_i(t_0^{[i]})\}$.

Proof. The proof is inspired by arguments presented in [91]. Consider node $i \in \mathcal{I}$ and its state variable $x_i(t_k^{[i]})$ at the update time $t_k^{[i]} \in \mathcal{T}^{[i]}$, $k \in \mathbb{N}$. Then, assume:

$$\breve{x}_i(\tau_{k_i}) = x_i(t_k^{[i]}), \quad k \in \mathbb{N}, \tag{5.11}$$

where, from Eq. (5.6), $\tau_{k_i} = f(t_k^{[i]})$ (the equivalence $\breve{x}_i(\tau_{0_i}) = x_i(t_0^{[i]})$, $\forall i \in \mathcal{I}$, is given by assumption). We now prove that the application of a single step of the asynchronous

max-consensus protocol $\mathcal{P}$, expressed by Eq. (5.3) and here rewritten as

$$x_i(t_{k+1}^{[i]}) = \max\{x_i(t_k^{[i]}), \max_{\substack{j\in\mathcal{N}_i \\ j\neq i}}\{x(\theta_k^{ij})\}\}, \tag{5.12}$$

takes the state variable x_i to a value equal to the same value obtained for $\breve{x}_i$ through the application of the synchronous protocol (5.9) at every virtual update time occurring in the interval $[\tau_{k_i}, \tau_{(k+1)_i})$.

First, consider the case $\tau_{(k+1)_i} \neq \tau_{k_i} + c^O$, *i.e.*, $\tau_{k_i} + c^O$ does not refer to an update time for node i. It follows from Eq. (5.9-(ii)) that

$$\breve{x}_i(\tau_l) = \breve{x}_i(\tau_{k_i}), \quad \forall \tau_l \in \mathbb{N} \text{ s.t. } k_i \leq l < (k+1)_i. \tag{5.13}$$

Suppose now that, for some $j \in \mathcal{N}_i$, $j \neq i$, there exists $t_k^{ij} \in \mathbb{R}$, such that: (i) $\tau_{k_i} \leq t_k^{ij} < \tau_{(k+1)_i}$; (ii) $t_k^{ij} - d_q^{ij} = \tau_{q_j}$, where τ_{q_j} is the virtual time corresponding to the local time $t_q^{[j]} \in \mathcal{T}^{[j]}$, $q \in \mathbb{N}_0$, at which the most recent value is sent by node j to its neighbours. We indicate with $k^{ij} = \left\lceil \frac{t_k^{ij}}{c^O} \right\rceil$ the index of the virtual time base at which node i detects that a new value has been received from j ($\lceil\cdot\rceil$ indicates the ceiling function). Therefore, for each $\tau_l \in \Upsilon$, $k^{ij} \leq l < (k+1)_i$, node i assumes that

$$\breve{x}_j(\tau_l) = x_j(t_q^{[j]}). \tag{5.14}$$

We remark that if there exist neighboring nodes by which no new values are received, $t_k^{ij} = \tau_{k_i}$ (or, equivalently, $k^{ij} = k_i$) and $\breve{x}_j(\tau_{k_i}) = \breve{x}_i(\tau_{k_i})$ hold. Assume now $\tau_h = \tau_{(k+1)_i} - c^O$, then in τ_{h+1}, applying Eqs. (5.9-(i)), (5.11), (5.13), the relation

$$\begin{aligned} \breve{x}_i(\tau_{h+1}) &= \max_{j\in\mathcal{N}_i}\{\breve{x}_j(\tau_h)\} = \\ &= \max\{\breve{x}_i(\tau_{k_i}), \max_{\substack{j\in\mathcal{N}_i \\ j\neq i}}\{\breve{x}_j(\tau_{k^{ij}})\} \end{aligned} \tag{5.15}$$

holds. By comparison of Eqs. (5.12) and (5.15), it is straightforward to verify that $x_i(t_{k+1}^{[i]}) = \breve{x}_i(\tau_{(k+1)_i})$. □

Once the equivalence between systems $\breve{\mathcal{S}}$ and $\mathcal{S}$ has been set, the convergence of the asynchronous max-consensus protocol is proved. To this aim, we recall a convergence theorem on synchronous max-consensus protocols in switching topologies.

Theorem 5.3. *Given a finite sequence of r adjacency matrices $\breve{A}_1, \ldots, \breve{A}_k, \ldots, \breve{A}_r$ which defines a switching topology, the synchronous system (5.10) achieves max-consensus in a*

finite time $\breve{\beta}$*, for all initial condition vectors* $\breve{\mathbf{x}}_{\mathbf{0}} = [\breve{x}(\tau_{0_1}) \dots \breve{x}(\tau_{0_n})]^T$*, if and only if the sequence of digraphs* $\breve{\mathcal{G}}(\breve{\mathcal{A}}_1), \dots, \breve{\mathcal{G}}(\breve{\mathcal{A}}_k), \dots, \breve{\mathcal{G}}(\breve{\mathcal{A}}_r)$ *is jointly strongly connected (notation* $\breve{\mathcal{G}}(\breve{\mathcal{A}}_k)$ *represents a digraph with adjacency matrix* $\breve{\mathcal{A}}_k$*).*

Proof. The proof is reported in [88]. □

On the basis of the results of Proposition 5.2 and Theorem 5.3, we shall prove in Theorem 5.5 the convergence of protocol $\mathcal{P}$ in $\mathcal{S}$. To this aim, a further definition is needed.

Definition 5.1. (*Partially asynchronous algorithm [86] [87]*) An asynchronous algorithm is called *partially asynchronous* if there exists a constant $\Delta \in \mathbb{N}$, for which the following two conditions hold:

1. each node performs an update at least once in Δ time units in the virtual time base;
2. the information used by any node is outdated by at most Δ time units in the virtual time base.

The positive constant $\Delta \in \mathbb{N}$ is called asynchronism measure.

Assumption 5.3. It is assumed that protocol $\mathcal{P}$ expressed by (5.3) is partially asynchronous with asynchronism measure Δ, that is, local state updates cannot occur arbitrarily slow, with respect to the virtual time base.

This is a realistic assumption in most applicative settings, where the greatest clock period and the greatest delay determine the value of the asynchronism measure.

Thus, the equivalent system $\mathcal{S}$ is also partially asynchronous with asynchronism measure Δ, and its partial asynchronism condition reads

Assumption 5.4. There exists a positive integer Δ s.t.

1. for every $i \in \mathcal{I}$ and for every $\tau_h \in \Upsilon$, $h \in \mathbb{N}_0$, there exists at least one value $k \in \mathbb{N}_0$ s.t. $f(t_k^{[i]}) \in \{\tau_h, \tau_{h+1}, ..., \tau_{h+\Delta-1}\}$, with $t_k^{[i]} \in \mathcal{T}^{[i]}$;
2. defining $\tau_{q_j} \in \mathcal{T}^{[j]}$ as the virtual time at which node $j \in \mathcal{N}_i$ produces the information used by node i at time $\tau_{k_i} \in \mathcal{T}^{[i]}$ through the state update rule (5.3), there holds: $\tau_{k_i} - \Delta \cdot c^O < \tau_{q_j} \leq \tau_{k_i}$, $\forall (j,i) \in \mathcal{E}$, $i \neq j$, $\forall t_k^{[i]} \in \mathcal{T}^{[i]}$.

Assumption 5.4 implies that each node updates its state at least once during any time interval of length $\Delta \cdot c^O$, and that, given any node, the state information received by its neighbors is outdated no more than Δ time units, expressed in the virtual time base.

We observe that, given Definition 5.1, the synchronous case is included as a particular case of the asynchronous one, with asynchronism measure $\Delta = 1$.

Theorem 5.4. *Given the partially asynchronous system $\mathcal{S} = \langle \mathcal{G}, \mathcal{P}, \mathcal{T} \rangle$ defined in (5.4) and its equivalent synchronous system $\breve{\mathcal{S}} = \langle \breve{\mathcal{G}}, \breve{\mathcal{P}}, \Upsilon \rangle$ defined in (5.10), if the digraph $\mathcal{G}$ is strongly connected, then $\forall \tau_h \in \Upsilon$, $\exists b_h \in \mathbb{N}$ s.t. the sequence $\{\breve{\mathcal{G}}_h, \ldots, \breve{\mathcal{G}}_k, \ldots, \breve{\mathcal{G}}_{h+b_h}\}$ is jointly strongly connected. Moreover, it results $b_h \leq \Delta - 1$, $\forall h \in \mathbb{N}_0$.*

Proof. Consider node $i \in \mathcal{I}$ and a virtual time $\tau_h \in \Upsilon$, $h \in \mathbb{N}_0$. The partial asynchronism assumption 5.4 guarantees that in at most Δ clock units expressed in the virtual time base, node i receives the values of the state variables of every neighbor $j \in \mathcal{N}_i$, that is, $\bigcup_{p=h}^{h+\Delta-1} \breve{\mathcal{N}}_i(p) = \mathcal{N}_i$, $\forall i \in \mathcal{I}$. Analogously, it can be verified that for all $\tau_h \in \Upsilon$, it results $\mathcal{G} = \bigcup_{p=h}^{h+\Delta-1} \breve{\mathcal{G}}_p$ and $\mathcal{A} = \bigvee_{p=h}^{h+\Delta-1} \breve{\mathcal{A}}_p$, where $\bigvee$ is the element-wise logical `or` of the adjacency matrices $\breve{\mathcal{A}}_p$, describing the switching digraphs $\breve{\mathcal{G}}_p$ at times $\tau_p \in \Upsilon$, $p \in \{h, \ldots, h + \Delta - 1\}$. The theorem is proved under the hypothesis of strongly connectedness of the static digraph $\mathcal{G}$ (Assumption 5.2), since it is possible to consider $b_h = \Delta - 1$, $\forall \tau_h \in \Upsilon$. □

Thus, the main result of this chapter can be summarized in the following Theorem.

Theorem 5.5. *If the digraph $\mathcal{G}$ of the partially asynchronous system $\mathcal{S}$ in (5.4) is strongly connected, then the max-consensus protocol in (5.3) converges in finite time, solving Problem 5.1, for all initial conditions $x_i(t_0^{[i]})$, $i \in \mathcal{I}$.*

Proof. The proof follows directly from the results stated in Theorems 5.3 and 5.4, and on the basis of the equivalence between systems $\mathcal{S}$ and $\breve{\mathcal{S}}$, stated in Proposition 5.2. □

Our study on the upper bound of the convergence time for the asynchronous max-consensus protocol is given in the following Corollary. It relies on the result on the finite-time convergence of its equivalent synchronous system, stated in Theorems 5.1 and 5.4.

Corollary 5.1. *(Upper bound on the convergence time)* Let $\mathcal{S}$ be the system in (5.4), where the asynchronous max-consensus protocol expressed by (5.3) is run over the static digraph $\mathcal{G}$. If $\mathcal{G}$ is strongly connected, then max-consensus is achieved, for all initial conditions, in at most $\Delta \cdot \mathcal{D}$ time steps, expressed in the virtual time base, where Δ is the asynchronism measure of $\mathcal{S}$, and $\mathcal{D}$ is the diameter of $\mathcal{G}$.

Proof. Consider the synchronous equivalent model $\breve{\mathcal{S}}$ of $\mathcal{S}$, and the following $\mathcal{D}$ sequences of Δ adjacency matrices: $S_1 \triangleq \breve{\mathcal{A}}_1, \ldots, \breve{\mathcal{A}}_\Delta$; $S_2 \triangleq \breve{\mathcal{A}}_{\Delta+1}, \ldots, \breve{\mathcal{A}}_{2\Delta}$; $\ldots$; $S_\mathcal{D} \triangleq$

$\breve{\mathcal{A}}_{(\mathcal{D}-1)\Delta+1}, \ldots, \breve{\mathcal{A}}_{\mathcal{D}\Delta}$. Each adjacency matrix models the network topology at a specific virtual time τ_h, $h \in \{1, \ldots, \mathcal{D} \cdot \Delta\}$. From Theorem 5.4, it follows that each sequence S_k, $k \in \{1, \ldots, \mathcal{D}\}$, is jointly strongly connected, and $\mathcal{G}(S_k) = \mathcal{G}$, $\forall k \in \{1, \ldots, \mathcal{D}\}$. Since $\mathcal{G}$ is strongly connected by hypothesis (Assumption 5.2), it is possible to apply Theorem 5.1. In particular, each of the $\mathcal{D}$ steps required to reach consensus is formed by Δ time instants (this property is necessary to ensure the condition of jointly strongly connectedness of each sequence S_k). Therefore, the entire process takes $\Delta \cdot \mathcal{D}$ times, referred to the virtual time base. Given the equivalence stated in Theorem 5.2, the thesis is proved. □

The proof of convergence, derived by observing the network from a global point of view, establishes a time-driven framework, in which each node has to run the protocol for a given time, necessary to ensure convergence [18]. Nevertheless, if one wants to design distributed applications relying directly on the theoretical results provided so far, some global knowledge about the network properties must be known, that is, every node should be able to estimate the asynchronism measure value Δ, the value of the diameter of the network $\mathcal{D}$, and the common virtual clock c^O. In the particular case in which each node owns a constant clock $c^{[i]} \in \mathbb{Q}$ (here, clock periods are intended to be expressed in the same measurement unit), and that all the starting times are synchronized, that is, $t_0^{[i]} = t_0^{[j]}$, $\forall \{i, j\} \subseteq \mathcal{I}$, the common virtual clock period c^O can be computed as

$$c^O = \underset{i}{\gcd}\{c^{[i]}\}, \quad i \in \mathcal{I}, \tag{5.16}$$

where gcd is the greatest common divisor operator. Under these conditions, $t_k^{[i]} = kc^{[i]}$, $\forall k \in \mathbb{N}_0, i \in \mathcal{I}$. Thus the following relation

$$k_i = \frac{kc^{[i]}}{\gcd_i\{c^{[i]}\}} \tag{5.17}$$

holds between index k of the local time base instant $t_k^{[i]}$ and index k_i of the corresponding virtual update time. It is straightforward to note that $\tau_{0_i} = \tau_0$, $\forall i \in \mathcal{I}$. Therefore, Eq. (5.16) can be combined with the result expressed by Corollary 5.1 in order to set a time-driven convergence detection mechanism (note that the values of the asynchronism measure Δ and of the diameter $\mathcal{D}$ are also required).

5.5.2 Event-based Convergence Detection Strategy

Convergence can alternatively be detected through an event-based strategy, which do not require the knowledge of global variables. This is a viable alternative, which provides the same upper bound on the convergence time of the time-driven approach. The proposed technique is inspired by [92] and [20], where it is used to solve a distributed connectivity control problem and a distributed target tracking application, respectively. Each node possesses a binary vector of tokens $\boldsymbol{\kappa}_i = [\kappa_i^1 \dots \kappa_i^n]^T \in \{0,1\}^n$, $i \in \mathcal{I}$. This is a collection of n flags, where n is the total number of network nodes, that are updated concurrently with the state of node i during the consensus process, on the basis of the token information received by its neighbors. An element κ_i^u of the token vector set to 1 indicates that node i has received the state value of node u. If the condition $\kappa_i^u = 1$ is verified for all $u \in \mathcal{I}$, then node i holds the maximum value and stops participating in the consensus process, after having sent its state value to all of its neighbors. We observe that, for the implementation of this event-based convergence detection technique, the only global information needed by all nodes is the network size n. As stated earlier in the thesis, this information can be easily computed in a distributed way [68], which allows to keep the detection technique totally distributed.

Theorem 5.6. *If every node $i \in \mathcal{I}$ updates its token vector $\boldsymbol{\kappa}_i(t_{k+1}^{[i]})$ at time $t_{k+1}^{[i]} \in \mathcal{T}^{[i]}$, according to the rule*

$$\kappa_i^u(t_{k+1}^{[i]}) = \kappa_i^u(t_k^{[i]}) \bigvee_{j \in \mathcal{N}_i} \kappa_j^u(\theta_k^{ij}), \forall u \in \mathcal{I}, \tag{5.18}$$

then condition $\bigwedge_{u \in \mathcal{I}} \kappa_i^u = 1$ is verified after at most $\Delta \cdot \mathcal{D}$ time units, expressed in the virtual time base (the operator $\bigwedge$ stands for the logical `and` *of the vector elements). This condition ensures the correct convergence of the consensus protocol to the maximum value of the initial states $x_i(t_0^{[i]})$, $i \in \mathcal{I}$.*

Proof. The proof of convergence with the token-mechanism follows the same arguments used for the proof of Corollary 5.1. The u-th element of the token vector of a generic node $i \in \mathcal{I}$, at its virtual starting time τ_{0_i}, is defined as:

$$\kappa_i^u(\tau_{0_i}) = \begin{cases} 1 & \text{if } i = u \\ 0 & \text{otherwise} \end{cases} . \tag{5.19}$$

According to the partial asynchronism assumption, after Δ virtual time units the generic node i has surely received the states of its neighbors, $j \in \mathcal{N}_i$, and has updated its state.

This implies that the token vector, at virtual time $\tau_{0_i+\Delta}$, is described by

$$\kappa_i^u(\tau_{0_i+\Delta}) = \begin{cases} 1 & \text{if } u \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} . \tag{5.20}$$

After further Δ virtual time units, it can be easily verified that

$$\kappa_i^u(\tau_{0_i+2\Delta}) = \begin{cases} 1 & \text{if } u \in \mathcal{N}_i \cup \bigcup_{p\in\mathcal{N}_i}\{\mathcal{N}_p\} \\ 0 & \text{otherwise} \end{cases} . \tag{5.21}$$

This implies that in 2Δ virtual time units, a path of length 2 has been established between node i and its 2-hop neighbors $w \in \bigcup_{p\in\mathcal{N}_i}\{\mathcal{N}_p\}$. Iteratively, after $k\Delta$ virtual time units, each node is able to establish a communication channel with its k-hop neighbors, so that the relation

$$\boldsymbol{\kappa}_i(\tau_{0_i+k\Delta}) = \boldsymbol{\kappa}_i(\tau_{0_i+(k-1)\Delta}) \vee \bigvee_{j\in\mathcal{N}_i} \boldsymbol{\kappa}_j(\tau_{0_i+(k-1)\Delta}) \tag{5.22}$$

holds, where $\vee$ is the element-wise boolean `or` operator. The generic element u of the token vector at time $k\Delta$ can be inductively computed as

$$\kappa_i^u(\tau_{0_i+k\Delta}) = \begin{cases} 1 & \text{if } u \in \{\mathcal{N}_l | \kappa_i^l(\tau_{0_i+(k-1)\Delta}) = 1, l \in \mathcal{I}\} \\ 0 & \text{otherwise} \end{cases} . \tag{5.23}$$

Since the diameter $\mathcal{D}$ of a network represents the maximum length of the shortest path between any pair of nodes, at most $\Delta \cdot \mathcal{D}$ virtual time units are required to obtain an all-one token vector $\boldsymbol{\kappa}_i = [1, 1, \ldots, 1]^T$, $\forall i \in \mathcal{I}$. Therefore, the token mechanism ensures that every node in the network receives the state information from all the other nodes in finite time. The convergence time of the procedure is bounded by the asynchronous max-consensus convergence result, while the correctness of the convergence value is ensured by the associative property of the max operator. □

5.6 Numerical Results

In this section, we show that the proposed token-based convergence detection mechanism, supported by the finite-time convergence properties of the asynchronous max-consensus protocol, provides an algorithmic framework to solve synchronization issues in max-consensus-based distributed applications [93]. As an example, we consider again the single task assignment problem, proposed in Section 5.3, showing that the token mechanism drives the network towards a correct assignment, and that the number of virtual updates to reach the convergence is upper bounded by $\Delta \cdot \mathcal{D}$.

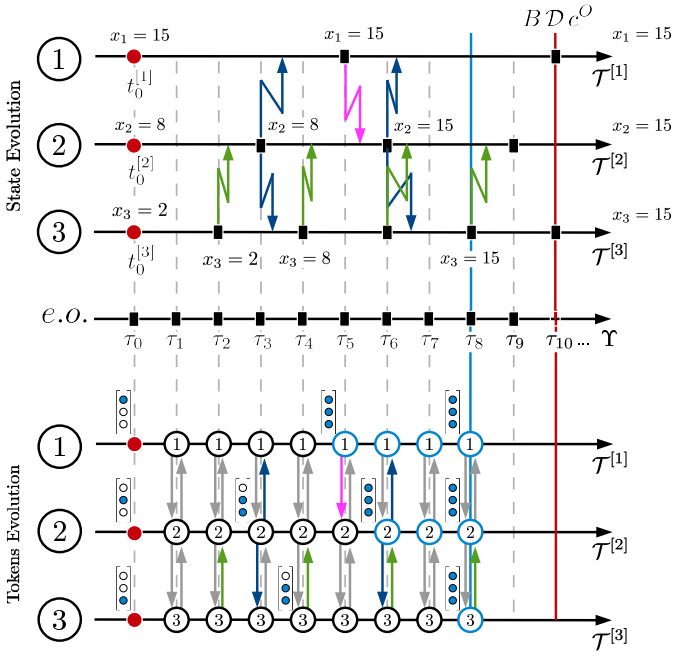


FIGURE 5.4: A numerical example: the evolution of the state and of the token vectors for a network of 3 nodes (connected as in Fig. 5.1) operating an asynchronous max-consensus with token-based convergence detection mechanism for a single task assignment problem. The vertical blue line indicates the convergence instant, while the upper bound on the convergence time is indicated by the red one.

We consider the same network of Section 5.3, composed by $n = 3$ nodes (see Fig. 5.1), with diameter $\mathcal{D} = 2$. We suppose again that there is a single task to be assigned, and that the network nodes set the following bids, $x_1(t_0^{[1]}) = 15$, $x_2(t_0^{[2]}) = 8$, and $x_3(t_0^{[3]}) = 2$, so that it should result that the task is assigned to node 1. We assume the following constant clock periods, $c^{[1]} = 0.5$ ms, $c^{[2]} = 0.3$ ms, and $c^{[3]} = 0.2$ ms, and a maximum delay $\check{d} = 0.095$ ms. Supposing all the starting times synchronized, it is easy to verify that the system is partially asynchronous with asynchronism measure $\Delta = 5$. Moreover, according to Eq. (5.16) it results $c^O = 0.1$ ms. In Fig. 5.4, the temporal sequence of send/receive messages, operated according to the asynchronous max-consensus protocol with token-based convergence detection, is illustrated. The value of the information state is also reported in correspondence of each time instant in which it evolves to a new value. As can be seen, all nodes agree on the maximum bid (*i.e.*, the network reaches consensus) after 8 time units referred to the virtual time base (node 3 is the last one that updates its state value to the maximum in $\tau_8 = 0.8$ ms). We remark that the upper bound in this case is equal to $\Delta \cdot \mathcal{D} \cdot c^O = 1$ ms, and, thus, the token-based mechanism allows the nodes to detect the convergence condition faster than the time-driven solution where, provided that the network nodes were able to estimate the global parameters of the system, they would run the protocol up to the upper bound for the convergence time.

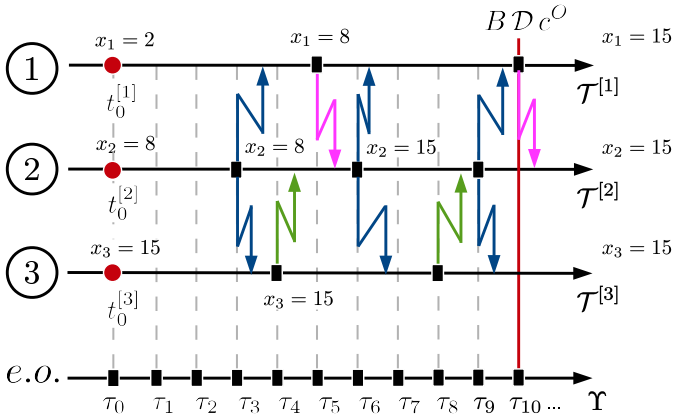


FIGURE 5.5: A numerical example: the evolution of the state for a network of 3 nodes (connected as in Fig. 5.1) operating an asynchronous max-consensus with token-based convergence detection mechanism for a single task assignment problem. The red line indicates the upper bound on the convergence time. Here, the convergence time is reached in exactly $\Delta \cdot \mathcal{D}$ time units.

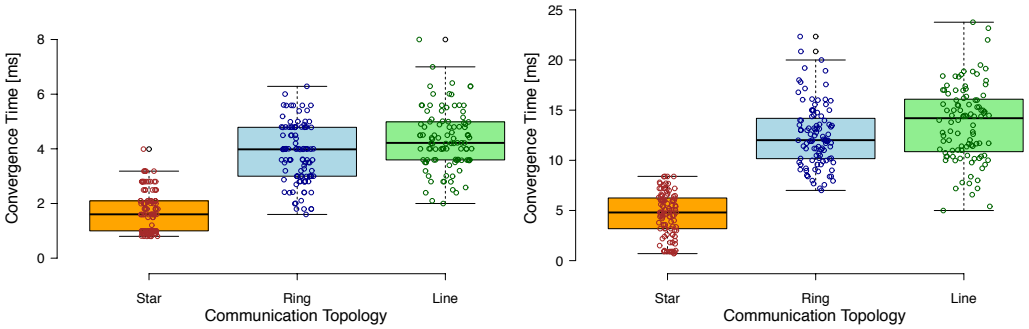


FIGURE 5.6: Box-plots of the convergence time of the max-consensus protocol for a network of 11 nodes connected through different topologies: star, ring, and line, with maximum communication delay (A) $\check{d}_1 = 0.095$ ms and (B) $\check{d}_2 = 9.5$ ms. The bottom of the box indicates the first quartile, while the top indicates the third quartile. The band inside the box is the second quartile, *i.e.*, the median. The whisker at the bottom of the box indicates the lowest data within 1.5 of the Interquartile Range (IQR, that is to say, the difference between the third quartile and the first one) of the lower quartile, while the top whisker indicates the highest data within 1.5 of the IQR of the upper quartile.

Figure 5.5 illustrates a different scenario for the same network. This setup differs from the previous one in the value of the clock period of node 3 (*i.e.*, $c^{[3]} = 0.4$ ms) and in the distribution of the initial set of nodes' bids, that is $x_1(t_0^{[1]}) = 2$, $x_2(t_0^{[2]}) = 8$, and $x_3(t_0^{[3]}) = 15$. Nevertheless, it still results $\Delta = 5$ and $c^O = 0.1$ ms. In this case, the evolution of the token-based mechanism makes the network reach the max-consensus in exactly $\Delta \cdot \mathcal{D} \cdot c^O = 1$ ms.

As stated in the previous section, the convergence time of the asynchronous max-consensus protocol depends on the diameter of the network, on the node's clock periods, and on the communication delays. An insight on the influence both of the network

topology (and, therefore, of the network diameter) and of the maximum delay on the asynchronous max-consensus convergence time is provided in Fig. 5.6. We consider a network of 11 nodes arranged in three different topologies (*i.e.,* star, ring, and line), with the following diameter values $\mathcal{D} = 2, 5, 10$, respectively. We run 100 Monte Carlo simulations of the max-consensus protocol for each topology with a virtual time base resolution fixed to 0.001 ms (*i.e.,* $c^O = 0.001$ ms). Moreover, we consider two different maximum delay values, that is, $\check{d}_1 = 0.095$ ms (Fig. 5.6(A)), and $\check{d}_2 = 9.5$ ms (Fig. 5.6(B)), respectively. We assume time-varying clock periods, *i.e.*, $c_k^{[i]} = c_{avg}^{[i]} + \chi_k^{[i]}$, $i \in \{1, \ldots, 11\}$, $k \in \mathbb{N}$, where $\chi_k^{[i]} \sim N(0, 0.001)$, that is, $\chi_k^{[i]}$ is drawn from a Gaussian distribution with zero mean and variance equal to 0.001 ms. For each simulation, the average clock periods $c_{avg}^{[i]}$, $i \in \mathcal{I}$, are set by drawing at random, with uniform probability, values from the set $C = \{0.1,\ 0.2,\ \ldots,\ 0.9\}$. From the results in Fig. 5.6, we observe that, for the same topology, the average convergence time increases as the maximum delay increases. For example, for the star topology the average convergence time is equal to 1.693 ms for the case $\check{d}_1 = 0.095$ ms, while it increases to 4.627 ms for the case $\check{d}_2 = 9.5$ ms. A similar trend can be observed for the ring topology, where the average convergence time increases from 3.812 ms in the case $\check{d}_1 = 0.095$ ms to 12.48 ms for the case $\check{d}_2 = 9.5$ ms, and also for the line topology, where the average convergence time for the case $\check{d}_1 = 0.095$ ms is equal to 4.338 ms, while it is equal to 13.633 ms for the case $\check{d}_2 = 9.5$ ms. On the other hand, Fig. 5.6 shows that, for the same maximum delay, an increase of the network diameter causes an increase of the average convergence time, as expected. In this regard, we observe that the average convergence time for a line topology and small delay (Fig. 5.6(A)) is comparable with the average convergence time for a network with star topology and large delay (Fig. 5.6(B)). This suggests that, in networks where communication delays are expected to be substantial, the performance in terms of convergence time can be improved through a sensible choice of the network topology.

5.7 Conclusion

In this chapter, we analyze the convergence properties of the asynchronous discrete-time max-consensus protocol. The work is motivated by the need to implement this protocol in applications, such as the DKNS presented in Chapter 3, where asynchronous updates and communication delays are usually present. The analysis relies on the equivalence between the asynchronous protocol running on a static network and a corresponding synchronous model running on a suitable switching topology. In particular, it is proved that, in presence of arbitrary asynchronous updates, the max-consensus protocol converges in

finite time if the underlying communication network is strongly connected, under the mild hypothesis of partial asynchronism. Moreover, the convergence time is bounded by a quantity which is function of the network diameter and of the asynchronism measure. To corroborate the theory, we propose an example where the asynchronous max-consensus theory is applied to a decentralized single task assignment problem, where a distributed mechanism for the detection of the convergence of the protocol, based on token vectors, has been proposed and implemented.

Chapter 6

Conclusion

In this thesis we have mainly focused on the design of distributed estimation algorithms for sensor and robotic networks. Specifically, a distributed Kalman filter and a distributed algorithm for the estimation of inertial parameters have been presented.

In chapter 3, we have addressed the problem of distributed Kalman filtering over heterogeneous sensor networks, by introducing a novel approach, the Distributed Kalman filtering with Node Selection (DKNS). The equivalence of the DKNS to a centralized Kalman filter with multiple measurements has been proved. The equivalent centralized filter evolves according to a state-dependent switching dynamics, able to select and propagate the best estimate of the process state through the sensor network in finite time. It is possible to define the level of accuracy of the estimate through the definition of a metric, called perception confidence value, which is related to the Fisher information. We have applied the algorithm to the discrete-time tracking of a maneuvering target, performed by a network of heterogeneous range-bearing sensors with limited sensing capability, achieving very satisfactory results. A performance comparison with existing algorithms based on sensor fusion yields the conclusion that DKNS is more effective than sensor fusion when limitations on the sensing capability are present. In this case, in fact, several inaccurate measurements could be present over the network. Since sensor fusion strategies are mostly based on some form of averaging, the presence of many inaccurate measurements can heavily affect the outcome of the fusion operation. On the other hand, through node selection, averaging is not performed and only the most accurate measurement is selected and propagated. It is evident that, when most of measurements are accurate, the application of fusion strategies is beneficial, since it contributes to filter out the noise associated to individual measurements. A real application of the DKNS in an Ambient Assisted Living framework has also been presented and validated in detail. Further work will deal with the possible presence of malicious or misbehaving nodes in

distributed estimation schemes based on node selection, such as DKNS. Due to the main characteristic of node selection algorithms, in fact, the presence of a faulty or malicious node that provides an inaccurate estimate with a high perception confidence value may heavily affect the outcome of the estimation process, since the faulty or malicious node can be wrongly selected as the one with the most accurate measurement.

In chapter 4, we have addressed the problem of the distributed estimation of inertial parameters of an unknown load manipulated by a network of robots. The estimation is performed through a sequence of steps and is achieved in finite time. We assume that each robot is able to control the exerted force and to measure the velocity of the point where the force is exerted. On the other hand, we do not assume that the robots' velocity can be controlled. An analysis of the influence of the measurement noise on the estimate has been carried out by defining suitable strategies and confidence intervals for the estimated quantities. Simulation results are very satisfactory and confirm the effectiveness of our approach, while keeping a low computational burden. Future work will deal with the extension of the proposed approach in three-dimensional applications (e.g., in aerial manipulation applications), the inclusion of non deterministic elements, such as delays, and the design and implementation of manipulation control strategies based on distributed estimation of inertial parameters.

In chapter 5, we have studied the convergence properties of the asynchronous max-consensus protocol. The work is motivated by the need to implement this kind of protocol in real-world applications, where asynchronous updates are real issues to be taken into account. The analysis is based on an equivalence between the asynchronous protocol on the given static network and a corresponding synchronous protocol run on a suitable switching topology. In particular, it is proved that, in presence of arbitrary asynchronous updates, and bounded time delays, the max-consensus protocol converges in finite time if the underlying communication network is strongly connected, under the mild hypothesis of partial asynchronism. Moreover, the convergence time is bounded by a quantity which is a function of the network diameter and of the asynchronism measure. As an example, we have applied the asynchronous max-consensus theory to a decentralized single task assignment problem, where a distributed mechanism for the detection of the convergence of the protocol, based on token vectors, has been proposed and implemented. Future work will deal with more in-depth studies on the dependence of the convergence time on network delays and topology, and on the generalization of the convergence results to a wider class of consensus protocols.

Appendix A

Linear Dynamics Filtering

In this appendix, we briefly recall a well-known parameter estimation procedure based on filtering linear dynamics [73]. This approach is used in the development of the estimation algorithms presented in Chapter 4 when dealing with constant parameters such as the distance between contact points $\|\mathbf{z}_{ij}$, the moment of inertia J of the object, and its mass m.

Given the linear system

$$\dot{\mathbf{y}} = \theta \mathbf{u}, \tag{A.1}$$

we would like to estimate the unknown constant parameter $\theta \in \mathbb{R}$ on the basis of measurements of the input $\mathbf{u} \in \mathbb{R}^l$ and the output $\mathbf{y} \in \mathbb{R}^l$, where $l \geq 1$. We assume that $\dot{\mathbf{y}}$ is unknown or not measured. To this aim, we compute low-pass filtered versions of the measured signals $\mathbf{u}$ and $\mathbf{y}$, namely $\mathbf{u}^f$ and $\mathbf{y}^f$, respectively, defined as:

$$\dot{\mathbf{u}}^f = k_f(\mathbf{u} - \mathbf{u}^f) \tag{A.2}$$

$$\dot{\mathbf{y}}^f = k_f(\mathbf{y} - \mathbf{y}^f) \tag{A.3}$$

with $k_f > 0$. Due to the linearity of (A.1)–(A.3), the following relation holds:

$$\dot{\mathbf{y}}^f = \theta \mathbf{u}^f, \tag{A.4}$$

which can be rewritten as

$$k_f(\mathbf{y} - \mathbf{y}^f) = \theta \mathbf{u}^f. \tag{A.5}$$

We remark that (A.5) holds after a transient phase due to the filter initial conditions. We observe that (A.5) relates known quantities, except for θ, which can be now estimated through a suitable estimator, generically described by the following dynamics that involves only quantities that are known, except for θ. Notably, the estimation hat theta of parameter theta is performed through the dynamical system

$$\dot{\hat{\theta}} = f_\theta(\mathbf{y}^f, \mathbf{u}^f, \mathbf{y}), \tag{A.6}$$

where f_θ is a suitable function describing the desired estimator's dynamics.

Bibliography

[1] S. Waharte and N. Trigoni. Supporting search and rescue operations with uavs. In *Proceedings of the International Conference on Emerging Security Technologies (EST)*, pages 142–147, September 2010.

[2] G. Antonelli, K. Baizid, F. Caccavale, G. Giglio, and F. Pierri. Cavis: A control software architecture for cooperative multi-unmanned aerial vehicle-manipulator systems. In *Proceedings of the 19th IFAC World Congress*, pages 24–29, Cape Town, South Africa, August 2014.

[3] D. Sieber, F. Deroo, and S. Hirche. Formation-based approach for multi-robot cooperative manipulation based on optimal control design. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 5227–5233, Tokyo, Japan, Nov 2013.

[4] Y. Cao and W. Ren. Multi-agent consensus using both current and outdated states with fixed and undirected interaction. *Journal of Intelligent and Robotic Systems*, 58:95–106, 2010. ISSN 0921-0296.

[5] W. Ren and R.W. Beard. *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications (Communications and Control Engineering)*. Springer, 1 edition, December 2007. ISBN 1848000146.

[6] R. Olfati-Saber, A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[7] W. Ren, R.W. Beard, and E.M. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the American Control Conference (ACC)*, pages 1859–1864, Portland, OR, USA, June 2005.

[8] W. Ren, R.W. Beard, and E.M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control System Magazine*, 27(2):71–82, 2007.

[9] D. Di Paola, D. Naso, and B. Turchiano. Consensus-based robust decentralized task assignment for heterogeneous robotic networks. In *Proceedings of the American Control Conference (ACC)*, pages 4711–4716, San Francisco, CA, USA, June 2011.

[10] A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[11] M.L. Hung and S. Weihua. Dynamic target tracking and observing in a mobile sensor network. *Robotics and Autonomous Systems*, 60(7):996 – 1009, 2012.

[12] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[13] S. Martinez, J. Cortes, and F. Bullo. On robust rendezvous for mobile autonomous agents. In *Proceedings of the 16th IFAC World Congress*, pages 1289–1298, Prague, Czech Republic, July 2005.

[14] P. Millán, L. Orihuela, I. Jurado, C. Vivas, and F.R. Rubio. Distributed estimation in networked systems under periodic and event-based communication policies. *International Journal of Systems Science*, 46(1):139–151, January 2015.

[15] W. Yang, X. Wang, and H. Shi. Optimal consensus-based distributed estimation with intermittent communication. *International Journal of Systems Science*, 42(9): 1521–1529, 2011.

[16] B.M. Nejad, S.A. Attia, and J. Raisch. Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies. In *XXII International Symposium on Information, Communication and Automation Technologies (ICAT)*, pages 1–7, Sarajevo, Bosnia and Herzegovina, October 2009.

[17] A. Petitti, D. Di Paola, A. Milella, P.L. Mazzeo, P. Spagnolo, G. Cicirelli, and G. Attolico. A heterogeneous robotic network for distributed ambient assisted living. In Springer, editor, *Human Behavior Understanding in Networked Sensing*. 2014.

[18] D. Di Paola, A. Petitti, and A. Rizzo. Distributed kalman filtering via node selection in heterogeneous sensor networks. *International Journal of Systems Science*, in press, 2015.

[19] A. Petitti, D. Di Paola, A. Rizzo, and G. Cicirelli. Consensus-based distributed estimation for target tracking in heterogeneous sensor networks. In *Proceedings of the 50th IEEE International Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 6648–6653, Orlando, FL, USA, December 2011.

[20] S. Giannini, A. Petitti, D. Di Paola, and A. Rizzo. Asynchronous consensus-based distributed target tracking. In *Proceedings of the 52nd IEEE Annual Conference on Decision and Control (CDC)*, pages 2006–2011, Firenze, Italy, December 2013.

[21] A. Petitti, D. Di Paola, A. Milella, P.L. Mazzeo, P. Spagnolo, G. Cicirelli, and G. Attolico. A distributed heterogeneous sensor network for tracking and monitoring. In *Proceedings of the 10th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 426–431, Aug 2013.

[22] F. Zhao and F. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. The Morgan Kaufmann Series in Networking. Morgan Kaufmann, 2004.

[23] W. Dargie and C. Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley Series on Wireless Communication and Mobile Computing. Wiley, 2010.

[24] A. Ghaffarkhah and Y. Mostofi. Communication-aware target tracking using navigation functions - centralized case. In *Proceedings of the Robot Communication and Coordination Conference*, pages 1–8, Odense, Denmark, April 2009.

[25] M. Taj and A. Cavallaro. Multi-camera track-before-detect. In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC)*, Como, IT, August 2009.

[26] M.S. Mahmoud and H.M. Khalid. Distributed kalman filtering: a bibliographic review. *Control Theory Applications, IET*, 7(4):483–501, 2013. ISSN 1751-8644.

[27] B.S.Y. Rao, H. Durrant-Whyte, and J. A. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *International Journal of Robotics Research*, 12(1):20–44, 1993. ISSN 0278-3649.

[28] S. Oruc, J. Sijs, and P.P.J. van den Bosch. Optimal decentralized kalman filter. In *Proceedings of the Mediterranean Conference on Control and Automation Conference (MED)*, pages 803–808, Thessaloniki, Greece, June 2009.

[29] W. Chin-Liang and W. Dong-Shing. Decentralized target tracking based on a weighted extended kalman filter for wireless sensor networks. In *Proceedings of the IEEE International Conference on Global Telecommunications (GLOBECOM)*, pages 1–5, New Orleans, LO, Dec. 2008.

[30] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[31] P.S. Maybeck. Stochastic models, estimation and control. *Academic Press*, I, 1979.

[32] H.B. Mitchell. *Multi-sensor Data Fusion: An Introduction*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[33] R. Olfati-Saber. Distributed kalman filtering for sensor networks. In *Proceedings of the 46th IEEE International Conference on Decision and Control (CDC)*, pages 5492 –5498, dec. 2007.

[34] F.S. Cattivelli and A.H. Sayed. Diffusion strategies for distributed kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, 55(9):2069 –2084, sept. 2010.

[35] S. Grime and H. Durrant-Whyte. Data fusion in decentralized sensor networks. *Control Engineering Practice*, 2(5):849–863, 1994.

[36] W. Yang and H. Shi. Sensor selection schemes for consensus based distributed estimation over energy constrained wireless sensor networks. *Neurocomputing*, 87: 132–137, 2012.

[37] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 1521–1526, 2003.

[38] S. Martinez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

[39] G.M. Siouris. *An Engineering Approach to Optimal Control and Estimation Theory.* Wiley-Interscience Publication, 1979.

[40] A. Bondy and U.S.R. Murty. *Graph Theory.* Graduate Texts in Mathematics. Springer, 2008.

[41] J.C.S. Cardoso, C. Baquero, and P.S. Almeida. Probabilistic estimation of network size and diameter dependable computing. In *Proceedings of the 4th Latin-American Symposium on Dependable Computing (LADC)*, pages 33–40, Seville, Spain, 2009.

[42] B.D.O. Anderson and J.B. Moore. *Optimal Filtering.* Thomas Kailath Editor, 1979.

[43] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* Applied Mathematics Series. Princeton University Press, 2009. ISBN 978-0-691-14195-4. Electronically available at http://coordinationbook.info.

[44] A. Burguera, Y. González, and G. Oliver. Sonar sensor models and their application to mobile robot localization. *Sensors*, 12(9):10217–10243, Dec 2009.

[45] A. Petitti, S. Giannini, D Di Paola, and F. Acquaviva. Masf. `http://users.ba.cnr.it/issia/iesiap59/software.html`.

[46] K. Khoshelham and S.O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[47] L. Cruz, D. Lucio, and L. Velho. Kinect and rgbd images: Challenges and applications. In *Proceedings of the 25th Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pages 36–49, Aug 2012.

[48] A. Franchi, A. Petitti, and A. Rizzo. Distributed estimation of the inertial parameters of an unknown load via multi-robot manipulation. In *Proceedings of the 52nd IEEE International Conference on Decision and Control (CDC)*, Los Angeles, CA, USA, Dec. 2014.

[49] A. Franchi, A. Petitti, and A. Rizzo. Decentralized estimation of kinematics and dynamics properties of an unknown load via multi-robot manipulation. In *submitted to: IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[50] I. Maza, F. Caballero, J. Capitan, J. R. Martinez-de Dios, and A. Ollero. Experimental results in multi-uav coordination for disaster management and civil security applications. *Journal of Intelligent & Robotic Systems*, 61(1-4):563–585, 2011. ISSN 0921-0296.

[51] J.C. John. *Introduction to Robotics: Mechanics and Control*. Boston, MA, USA, 2005. ISBN 0201095289.

[52] L. Sciavicco and B. Siciliano, editors. *Modeling and Control of Robot Manipulators*. Springer, 2000.

[53] K.-S. Chang, R. Holmberg, and O. Khatib. The augmented object model: cooperative manipulation and parallel mechanism dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 470–475, 2000.

[54] K.I. Kim and Y.F. Zheng. Two strategies of position and force control for two industrial robots handling a single object. *Robotics and Autonomous Systems*, 5(4): 395–403, 1989. ISSN 0921-8890.

[55] S.A. Schneider and R.H. Cannon. Object impedance control for cooperative manipulation: theory and experimental results. *Robotics and Automation, IEEE Transactions on*, 8(3):383–394, 1992. ISSN 1042-296X.

[56] I.D. Walker, R.A. Freeman, and S.I. Marcus. Analysis of motion and internal loading of objects grasped by multiple cooperating manipulators. *Robotics and Autonomous Systems*, 10(4):396–409, 1991.

[57] J. Szewczyk, F. Plumet, and P. Bidaud. Planning and controlling cooperating robots through distributed impedance. *Journal of Robotic Systems*, 19(6):283–297, 2002. ISSN 1097-4563.

[58] A. Yamashita, T. Arai, J. Ota, and H. Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *Proceedings of the IEEE International Conference on Robotics and Automation*, 19(2):223–237, Apr 2003. ISSN 1042-296X.

[59] A. Yufka, O. Parlaktuna, and M. Ozkan. Formation-based cooperative transportation by a group of non-holonomic mobile robots. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 3300–3307, Istanbul, Turkey, Oct 2010.

[60] G. Gioioso, A. Franchi, G. Salvietti, S. Scheggi, and D. Prattichizzo. The flying hand: a formation of uavs for cooperative aerial tele-manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[61] T. Lee, K. Sreenath, and V. Kumar. Geometric control of cooperating multiple quadrotor UAVs with a suspended payload. In *Proceedings of the IEEE 52nd International Conference on on Decision and Control (CDC)*, Florence, Italy, December 2013.

[62] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. In *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 545–558. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-32722-3.

[63] D. Lee and M.W. Spong. Bilateral teleoperation of multiple cooperative robots over delayed communication networks: Theory. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 360–365, 2005.

[64] A.C. Satici and M.W. Spong. Nonholonomic cooperative manipulation of polygonal objects in the plane. In *Proceedings of the IEEE 51st International Conference on Decision and Control (CDC)*, pages 2439–2446, 2012.

[65] Y. Yu, T. Arima, and S. Tsujio. Inertia parameter estimation of planar object in pushing operation. In *Proceedings of the IEEE International Conference on Information Acquisition*, June 2005.

[66] D. Kubus, T. Kroger, and F. M. Wahl. On-line estimation of inertial parameters using a recursive total least-squares approach. In *Proceedings of the IEEE/RJS*

International Conference on Intelligent Robots and Systems (IROS), pages 3845–3852, September 2008.

[67] N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997. ISBN 1558603484.

[68] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2nd edition, 2000.

[69] R. Aragues, L. Carlone, C. Sagues, and G. Calafiore. Distributed centroid estimation from noisy relative measurements. *System & Control Letters*, 61(7):773–779, 2012.

[70] M. Zhu and S. Martinez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322 – 329, 2010. ISSN 0005-1098.

[71] R. Hermann and A.J. Krener. Nonlinear controllability and observability. *Transactions on Automatic Control*, 22(5):728–740, 1977.

[72] A. Isidori. *Nonlinear control systems*. Springer, 1995. ISBN 978-1-84628-615-5.

[73] J.J.E. Slotine and W. Li. *Applied nonlinear control*. Prentice Hall, 1991. ISBN 9780130408907.

[74] L. Ljung, editor. *System Identification (2Nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2.

[75] A. Ditkowski, A. Bhandari, and B.W. Sheldon. Computing derivatives of noisy signals using orthogonal functions expansions. *Journal of Scientific Computing*, 36(3):333–349, 2008. ISSN 0885-7474. doi: 10.1007/s10915-008-9193-9. URL `http://dx.doi.org/10.1007/s10915-008-9193-9`.

[76] R.W. Schafer. What is a savitzky-golay filter? [lecture notes]. *Signal Processing Magazine, IEEE*, 28(4):111–117, July 2011. ISSN 1053-5888. doi: 10.1109/MSP.2011.941097.

[77] S. Van de Geer. *Least squares estimators*. Encyclopedia Statistics in The Behavioral Sciences, 2005. ISBN 0-470-86080-4.

[78] S. Giannini, D. Di Paola, A. Petitti, and A. Rizzo. On the convergence of the max-consensus protocol with asynchronous updates. In *Proceedings of the 52nd IEEE Annual Conference on Decision and Control (CDC)*, pages 2605–2610, Firenze, Italy, December 2013.

[79] S. Giannini, A. Petitti, D. Di Paola, and A. Rizzo. Asynchronous max-consensus protocol with time delays: Convergence results and applications. *Submitted to: Control Theory & Applications*.

[80] Y. Xu. Time synchronization in wireless sensor networks using max and average consensus protocol. *International Journal of Distributed Sensor Networks*, 2013: 1–10, 2013.

[81] L. Li, D.W.C. Ho, and S. Xu. A distributed event-triggered scheme for discrete-time multi-agent consensus with communication delays. *Control Theory & Applications, IET*, 8(10):830–837, July 2014.

[82] Y.G. Sun, L. Wang, and G. Xie. Average consensus in networks of dynamic agents with switching topologies and multiple time-varying delays. *Systems & Control Letters*, 57(2):175–183, February 2008.

[83] J. Qin, C. Yu, and S. Hirche. Stationary consensus of asynchronous discrete-time second-order multi-agent systems under switching topology. *IEEE Transactions on Industrial Informatics*, 8(4):986–994, November 2012.

[84] B.M. Nejad, S.A. Attia, and J. Raisch. Max-consensus in a max-plus algebraic setting: The case of switching communication topologies. In *Proceedings of the 10th International Workshop on Discrete Event Systems - Part I*, pages 173–180, Berlin, Germany, August 2010.

[85] F. Iutzeler, P. Ciblat, and J. Jakubowicz. Analysis of max-consensus algorithms in wireless channels. *IEEE Transactions on Signal Processing*, 60(11):6103–6107, November 2012.

[86] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, USA, 1997.

[87] J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. part 2: The asynchronous case. *SIAM Journal on Control and Optimization*, 46(6): 2120–2147, November 2007.

[88] J.G. Manathara, A. Dukkipati, and D. Ghose. Tropical algebraic approach to consensus over networks. *arXiv preprint arXiv:1109.0418*, pages 1–11, September 2011.

[89] D.W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, August 2007.

[90] H.L. Choi, L. Brunet, and J.P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009.

[91] F. Xiao and L. Wang. Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays. *IEEE Transactions on Automatic Control*, 53(8):1804–1816, September 2008.

[92] M.M. Zavlanos and G.J. Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, December 2008.

[93] D.P. Bertsekas and J.N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithm – a survey. *Automatica*, 27(1):3–21, January 1991.