ROYAL INSTITUTE
OF TECHNOLOGY

Event-triggered and cloud-supported control of multi-robot systems

ANTONIO ADALDO

Doctoral Thesis
Stockholm, 2018

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
SE-100 44 Stockholm
SWEDEN

TRITA-EECS-AVL-2018:41
ISBN 978-91-7729-791-8

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framläggs till offentlig granskning för avläggande av teknologie doktorsexamen i reglerteknik fredagen den 1 juni 2018 klockan 14.00 i sal Q2, Kungliga Tekniska högskolan, Osquldas väg 10, Stockholm.

Tryck: Universitetsservice US AB

Sammanfattning

I reglering av multi-robot system är syftet att uppnå ett samordnat beteende genom lokala interaktioner bland robotarna. Ett fleragentsystem är en abstrakt modell av ett multi-robot system. I denna avhandling undersöks fleragentsystem där kommunikationen mellan agenterna modelleras som tidsdiskreta händelser som utlöses av vilkor på agenternas inre tillstånd. Vi betraktar två kommunikationsmodeller. I den första modellen utbyter två agenter direkt information med varandra. I den andra modellen utbyts all information genom asynkron tillgång till ett gemensamt minne. Avhandlingens bidrag består av fyra delar.

Det första bidraget är en händelsestyrd pinningregleringsalgoritm för ett nätverk av agenter med olinjär dynamik och tidsvarierande topologi. Pinningreglering är en strategi för att styra beteendet hos ett fleragentsystem på ett önskat sätt genom att endast styra en liten del av agenterna. Vi uttrycker styrbarheten hos nätverket i form av ett medelvärde av nätverkskonnektiviteten över tiden, och vi visar att alla agenter kan drivas till en önskad referenstrajektoria.

Det andra bidraget är en regleringsalgoritm för fleragentsystem där kommunikationen mellan agenterna är ersatt av ett gemensamt minne som är installerat på ett moln. Kommunikationen mellan varje agent och molnet modelleras som en följd av händelser som planeras rekursivt av agenten. Vi kvantifierar nätverkets konnektivitet och vi visar att det är möjligt att synkronisera fleragentsystemet till samma tillståndstrajektoria och att två på varandra följande uppkopplingar till molnen av samma agent separeras av ett nedåt begränsat tidsintervall.

Det tredje bidraget är en samling av distribuerade regulatorer för täcknings- och övervakningsuppgifter med ett nätverk av mobila sensorer med anisotropa sensormönster. Vi utvecklar en abstrakt modell av den inspekterade miljön och definierar ett mått på den täckning som uppnås av sensornätverket. Vi visar att nätverket uppnår gradvis förbättrad täckning, och vi karaktäriserar nätverkets jämviktskonfigurationer.

Det fjärde bidraget är en distribuerad, molnbaserad regleringsalgoritm för inspektion av 3D-strukturer med ett nätverk av mobila sensorer, som liknar dem som betraktas i det tredje bidraget. Vi utvecklar en abstrakt modell av strukturen som ska inspekteras och kvantifierar omfattningen av inspektionen. Vi visar att nätverket enligt den föreslagna algoritmen är garanterat att slutföra inspektionen inom begränsad tid.

Alla restultat som presenteras i avhandlingen bekräftas av numeriska simuleringar och ibland av experiment med flygrobotplattformar. Experimenten visar att teorin och metoderna som utvecklas i avhandlingen är av praktisk relevans.

Abstract

In control of multi-robot systems, the aim is to obtain a coordinated behavior through local interactions among the robots. A multi-agent system is an abstract model of a multi-robot system. In this thesis, we investigate multi-agent systems where inter-agent communication is modeled by discrete events triggered by conditions on the internal state of the agents. We consider two models of communication. In the first model, two agents exchange information directly with each other. In the second model, all information is exchanged asynchronously over a shared repository. Four contributions on control algorithms for multi-agent systems are offered in the thesis.

The first contribution is an event-triggered pinning control algorithm for a network of agents with nonlinear dynamics and time-varying topology. Pinning control is a strategy to steer the behavior of the system in a desired manner by controlling only a small fraction of the agents. We express the controllability of the network in terms of an average value of the network connectivity over time, and we show that all the agents can be driven to a desired reference trajectory.

The second contribution is a control algorithm for multi-agent systems where inter-agent communication is substituted with a shared remote repository hosted on a cloud. The communication between each agent and the cloud is modeled as a sequence of events scheduled recursively by the agent. We quantify the connectivity of the network and we show that it is possible to synchronize the multi-agent system to the same state trajectory, while guaranteeing that two consecutive cloud accesses by the same agent are separated by a lower-bounded time interval.

The third contribution is a family of distributed controllers for coverage and surveillance tasks with a network of mobile agents with anisotropic sensing patterns. We develop an abstract model of the environment under inspection and define a measure of the coverage attained by the sensor network. We show that the network attains nondecreasing coverage, and we characterize the equilibrium configurations of the network.

The fourth contribution is a distributed, cloud-supported control algorithm for inspection of 3D structures with a network of mobile sensing agents, similar to those considered in the third contribution. We develop an abstract model of the structure to inspect and quantify the degree of completion of the inspection. We demonstrate that, under the proposed algorithm, the network is guaranteed to complete the inspection in finite time.

All results presented in the thesis are corroborated by numerical simulations and sometimes by experiments with aerial robotic platforms. The experiments show that the theory and methods developed in the thesis are of practical relevance.

To my family

Acknowledgements

Research is a multi-agent system, where each agent is a researcher, and the common objective of increasing human knowledge is attained by means of communication between different researchers. This communication is event-triggered, and often it occurs asynchronously over a shared repository (of papers) hosted on a cloud. However, there is no known mathematical model that may capture the richness of the interactions that have made it possible for me to write this thesis. Here is sample of my gratitude towards the agents who have maintained communication channels with me under my doctoral studies.

My advisor Karl Henrik Johansson, for your contagious enthusiasm, for inspiring me, for believing in me, for giving me responsibility, for bringing out the best of me, and for teaching me to bring out the best of people.

My advisor Dimos V. Dimarogonas, for the careful attention given to our work, for showing me that my decisions make a difference, and for teaching me how to make the most out of my resources.

My advisor Mario di Bernardo, for introducing me to research, for mentoring me, for teaching me that it is fine to take risks, and for always having my best interest at heart.

My advisor Davide Liuzza, for our nurturing cooperation, for believing in me, and for helping me fight my demons.

My advisor Guodong Shi, for the care you put in the quality of our work, for amplifying my enthusiasm for the technical challenges, and for our amazingly instructive whiteboard chats.

Prof. George Pappas and Kostantinos Gatsis, for hosting me on an instructive and fun exchange trimester.

Francesco (Kekko) Alderisio, for taking together with me the very first steps in the research world, for coping with my uninterrupted presence during our Erasmus at KTH, and for doing pretty much the same again. Mosy and Vito, for keeping in touch and exchanging tips and life hacks.

Jieqiang (Jay) Wei, for our fertile collaborations and instructive chats.

Demia, Laura, Miguel, and Riccardo, for the heartfelt advice that you give me, for sharing joy and frustration, for teaching me recipes, for laughing at my sarcastic replies to the TaMoS questions, for teaching me bouldering, for tasting small bites from my lunchbox (thus ensuring that my food is at least edible), for handling my anxiety when there is the tenancy review, and for helping me approach the bridge slowly.

Alex, Chris, and Sebastian, for sharing with me the joys and pains of being a TA, and for helping me become a better one.

Lars, and all participants in our reading group on hybrid control, for the passion and enthusiasm that you put in our technical discussions. All my colleagues in the NetCon group, for the same reasons.

My office roommates, present and past, for always making my day. And in particular: José, for being an awesome travel mate; Matias, for being an awesome neighbor; Olle, for coping with my broken Swedish; Pato and Sadegh, for your warm conversation and the countless sweet treats; Pedro, for helping me see the bright side of things; Valerio, for sharing tips and life hacks.

My colleagues in the EU project AEROWORKS, especially Pedro and my other fellow ARWs at KTH, for the six-winged adventures that we have shared, and for coping with my goofiness with ROS. Matteo, Pedro, Rui, and Xiao, for being always ready to help with a smile on. Our collaborators at NTNU and SNU, for our friendly collaboration and your warm hospitality.

All people who at some point have crossed paths with me at the department of Automatic Control at KTH, for making our office the best workplace ever. In particular, all the Administrators (resp., IT support), for patiently helping me whenever I would panic over some paperwork (resp., hardware). Special thanks to Andrea, Erik, Jay, Jonas, Manne, Matin, and Mohammed, who kindly agreed to review parts of this thesis.

The European Union, the Knut och Alice Wallenberg foundation, the Swedish Foundation for Strategic Research, and the Swedish Research Council, for funding my studies.

Alessio, Andrea, Axel, Eva, Giorgio, and Luca, for your heartfelt advice and nurturing friendship, for being my safety net and role models, for the countless SvA afternoons, for the board games, for the lunches and the picnics, and for always being encouraging.

My mother Emilia, my father Vincenzo, and my sister Giorgia, for your unconditional love and support, and because growing up with you is an unfair advantage at life.

Last, but definitely not least, Frank, for keeping up your provole no matter what.

Thank you!

Antonio Adaldo
Stockholm, April 2018

Contents

Acronyms

AUV	Autonomous Underwater Vehicle
CF	CrazyFlie
CR	CrazyRadio
D2D	Device-to-Device
GPS	Global Positioning System
HDV	Heavy-Duty Vehicle
KTH	Royal Institute of Technology
LQR	Linear Quadratic Regulation
LTU	Luleå University of Technology
P2P	Peer-to-Peer
ROS	Robot Operating System
SK	Skellefteå Kraft
UAV	Unmanned Aerial Vehicle

Symbols

$\operatorname{diag}(A)$	column vector corresponding to the diagonal of A
$\operatorname{eig}(A)$	set of the eigenvalues of A
∇	gradient operator
$\%$	integer division remainder
$\mathbb{R}_{\geq 0}$	set of the nonnegative real numbers
$\otimes$	Kronecker product
$\mathbb{N}_{>0}$	set of the positive integers
$A^{\dagger}$	left pseudoinverse of A
$\operatorname{Re}(c)$	real part of complex number c
$\mathbb{R}_{>0}$	set of the positive real numbers
$\operatorname{SE}(n)$	special Euclidean group in $\mathbb{R}^n$
Skew	Skew operator
$\operatorname{SO}(n)$	special orthogonal group in $\mathbb{R}^n$
$\mathbf{S}^n_{>0}$	space of the n-by-n symmetric and positive definite matrices
$\mathbf{S}^n_{\geq 0}$	space of the n-by-n symmetric and positive semidefinite matrices
$\operatorname{tr}(A)$	trace of A
$A^{\intercal}$	transpose of A

Chapter 1

Introduction

La donzelletta vien dalla campagna
in sul calar del sole,
col suo fascio dell'erba; e reca in mano
un mazzolin di rose e viole,
onde, siccome suole,
ornare ella si appresta
dimani, al dì di festa, il petto e il crine.

G. Leopardi,
Il sabato del villaggio, *vv 1–7.*

A multi-robot system is a network of interconnected devices, where each device is endowed with some sensing and actuation capabilities. This thesis is concerned with the design, analysis and implementation of distributed algorithms to obtain a coordinated collective behavior in a multi-robot system with only sparse communication among the devices.

Throughout the thesis, we use the formalism of a *multi-agent system* as an abstract model of a multi-robot system. A multi-agent system is composed of a set of interconnected subsystems, or *agents*, where each subsystem is the abstract model of a single robot. Roughly speaking, each agent is modeled by a set of differential equations that describe its behavior and its interactions with the external world.

This chapter is dedicated to discussing the factors that motivate the research work described in this thesis and to outlining the contents of the thesis. First, we illustrate the need for distributed algorithms for coordination under limited communication and the existence of technologies that enable the implementation of these algorithms in real systems. Then, we give three specific examples of possible

application domains for the control designs proposed in the thesis. Finally, we describe the contributions provided by the thesis, and we outline the contents of the following chapters.

The rest of this chapter is organized as follows. In Section 1.1, we illustrate the general motivations for the research work presented in the thesis. In Section 1.2, we briefly discuss the modern technologies that enable the implementation of distributed algorithms of the type presented in the thesis. In Section 1.3, we discuss three particular motivating applications for the research work described in the thesis. In Section 1.4, we formulate the research questions that are addressed in the thesis. Finally, Section 1.5 gives a detailed outline of the thesis and a list of the publications that the thesis is based on.

1.1 Motivation

Multi-agent systems are a powerful model to describe a wide array of phenomena in nature, society and technology. This thesis is concerned with multi-agent systems as a model for multi-robot systems, where each robot is modeled as a dynamical system.

In each of Chapters 3 to 7, we illustrate a desired coordinated behavior for a multi-robot system, and we design a distributed control algorithm that drives the robots to attain that behavior.

Within the broad context of modern technological solutions, networks of interconnected devices are ubiquitous. A few examples of systems that can be effectively studied as multi-agent systems are: a fleet of Autonomous Underwater Vehicles (AUVs) deployed on a seafloor mapping or exploration mission; a team of Unmanned Autonomous Vehicles (UAVs) deployed on a surveillance, inspection, or rescue mission; a platoon of autonomous heavy-duty vehicles transporting goods; a set of mobile sensing agents deployed into an environment to locally measure a specified quantity (such as temperature or humidity), and return a collective measurement to the user. Each of these examples constitutes a motivating application for the research work described in this thesis, and some of these applications are discussed in detail in Section 1.3. It is worth noting that multi-agent systems can be used to model also a variety of phenomena in biological and social sciences (such as swarming, schooling, and social networks, to name just a few). However, these applications are not directly considered in this thesis.

In most modern scenarios, communication among different devices in a network occurs over a wireless medium with limited throughput. For this reason, assuming that there is a continuous stream of information from one device to another is often unrealistic. In this thesis, communication between agents is considered an intermittent but instantaneous phenomenon. The amount of information that is

passed from one agent to another on a single communication instance is limited.

Throughout the thesis, we consider two possible communication patterns. In the first case, a device may exchange information directly with a subset of the other devices in the network. This pattern is used in Chapters 3 and 6. In the second case, a device may only exchange information with a remote repository hosted on a cloud server. Hence, a device may only withdraw information about another device if the latter has previously deposited such information in the cloud repository. This pattern is used in Chapters 4, 5 and 7.

1.2 Enabling technologies

Thanks to the recent technological developments in computation and communication, networks of interconnected devices have permeated modern society.

The growing availability of small-scale, affordable, and general-purpose electronics makes it increasingly easy to design and construct a network of interconnected devices to perform some automated task. However, in spite of Moore's law (Moore, 1965), small embedded microprocessors still have limited computational and communication capabilities with respect to complex missions, such as surveillance or inspection of a 3D structure. Therefore, it is crucial to design algorithms that make an intelligent use of the computational resources of the device, as well as of the throughput capacity of the communication medium.

There exist two main models of communication in real-world networks. In the first model, all information sent by the devices is collected in a central server, which then redistributes each piece of information to the target device or devices. In the second model, information is sent from one device directly to another device. In certain domains, such as telecommunications and signal processing, this model is sometimes referred to as Device-to-Device (D2D). In computer science, it is often referred to as Peer-to-Peer (P2P). Recently, the D2D model has been subject to keen research attention: see for example Della Penda (2018) and references therein. These two network models reflect closely the two communication patterns for multi-agent systems that we consider in this thesis.

The most prominent example of a network of interconnected devices is probably the Internet. The Internet can be seen as an infrastructure for information exchange among computers. The topology of the information exchanges through the Internet is in continuous evolution, but, in general, one may say that the Internet incorporates both the server model and the P2P model, with individual users relaying information from a central server to each other. Several widespread commercial applications, such as the Swedish music streaming service Spotify, use P2P systems for video and audio distribution.

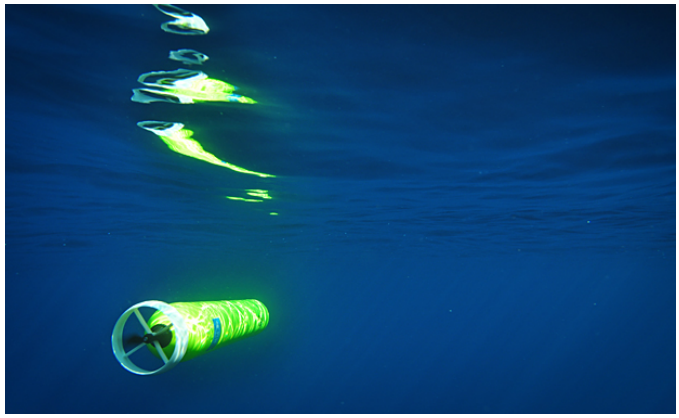

Figure 1.1: The research AUV Carl, developed at the KTH Centre for Naval Architecture. Source: courtesy of the KTH Centre for Naval Architecture.

1.3 Illustrative examples

In this section, we discuss three motivating examples for our research. In each example, we describe a multi-robot system that can be controlled by means of intermittent communication, and we describe the related possible control objectives and challenges.

Coordinated AUV navigation

AUVs are currently employed (with levels of automation varying from teleoperation to autonomous navigation) in numerous applications, such as seafloor mapping, underwater sampling, exploration, circumnavigation, search and rescue (Fiorelli et al., 2006). Figure 1.1 shows a research platform capable of autonomous underwater navigation developed by the KTH Centre for Naval Architecture.

In most realistic scenarios, it is desirable to deploy several AUVs at the same time, so that the mission at hand can be completed in a shorter time frame. Moreover, there exist some applications, such as target capturing, that structurally require the use of multiple AUVs platforms.

Consider, for example, the problem of circumnavigating a target with a fleet of AUVs. The target to circumnavigate may be a school of fish that is to be kept under observation for a biological study. The motion of each vehicle needs to be controlled by taking into account the possible motion of the school, but also the whereabouts of the other vehicles in the fleet. Hence, the vehicles need to exchange information about their positions and velocities. Moreover, the control action needs to take into account possible interferences arising from marine currents or other

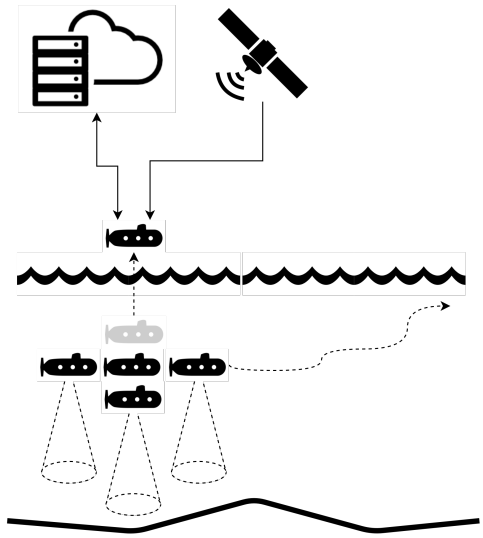

Figure 1.2: Schematic representation of a sea floor mapping mission with a fleet of AUV. In order to perform a cooperative task, such as mapping the sea floor, the vehicles have to move in a coordinated way. However, underwater communication is severely limited. Moreover, GPS is not available underwater, and the vehicles have to surface whenever they need a GPS position measurement. On the water surface, the vehicles have access to GPS and may also communicate with a base station to deposit and retrieve data.

disturbances.

In the control of a single AUV, the challenges are usually related to modeling the dynamics of the underwater navigation, compensating for the disturbances produced by the marine currents, localization, poor maneuverability, and optimization of the fuel consumption. In particular, Global Positioning System (GPS) is usually not available underwater, and localization is often obtained by means of odometry, while letting the vehicle surface every now and then to receive a GPS position fix. The challenges compound in the context of controlling a fleet. In fact, coordination among the vehicles relies on their reciprocal exchange of feedback or some other information. Underwater communication may be realized by means of acoustic modems, but these are relatively expensive, short-ranged, and power-hungry. For these reasons, it is essential to design control algorithms that require a sparse exchange of information between the vehicles. Intermittent communication should be accounted for explicitly in the control design.

Another possible approach to cooperative AUV control is to let each vehicle surface periodically to communicate with a base station. The base station may be used to obtain localization information, but also as a shared repository to asynchronously

Figure 1.3: A modern wind power plant. Source: `publicdomainpictures.net`, Public Domain.

exchange information with the other vehicles. Figure 1.2 illustrates a schematic of the envisioned scenario of a team of AUVs performing a cooperative seafloor mapping mission, and attaining coordination by periodically reaching the surface to access a shared information repository.

The coordination of a fleet of AUVs is the main motivating application to the algorithms proposed in Chapters 4 and 5. More precisely, in Chapter 4 we study the use of a cloud repository as a replacement to inter-agent communication for a generic coordination objective. In Chapter 5, we specialize the control design to a circumnavigation problem.

Surveillance and inspection of 3D structures with UAV networks

Figure 1.3 shows a modern wind power plant. Wind turbines are predicted to play a crucial role in the future of power generation, and as such, they constitute an area of growing interest for energy suppliers.

One of the most challenging aspects of power generation from wind turbines is the inspection and maintenance of the power plants. In fact, the blades, (and, more generally, all the higher parts) of a wind turbine are hardly accessible to

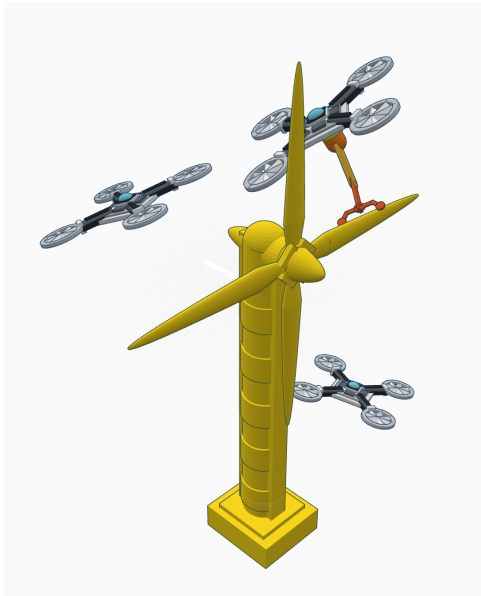

Figure 1.4: Schematic representation of the inspection of a wind turbine with a team of UAVs. The aerial robots need to inspect the whole surface of the turbine. A possible approach is to have each UAV take up one part of the surface of the turbine. To attain the desired coordination (that is, to decide which part of the turbine should be assigned to each robot) the UAVs communicate over a wireless medium, which is a shared resource with limited capacity. The robots also need to avoid collisions and counteract possible air currents. Different robots may be equipped with different sensing hardware.

direct human observation, and a manual inspection requires trained manpower and relatively dangerous maneuvers. As a consequence, the cost of performing periodic inspection and maintenance of the turbines is currently a major disadvantage of producing energy from wind. For these reasons, companies that supply energy generated from wind turbines are looking with interest at solutions for automated inspection and maintenance. UAVs, especially in the form of multi-copter robots, constitute an ideal platform for such endeavors.

Currently, the development of UAVs platforms is remarkably fast-paced compared to other branches of robotics, and new models with increased payload, autonomy and maneuverability appear on the market virtually on a daily basis.

The use of teleoperated multicopters for inspection of industrial infrastructures in hazardous contexts is already widespread: see for example the patents Williams (2010), Haffner and Venkataraman (2012). Jordan et al. (2018) gives a recent survey of available technologies for UAV inspection. Figure 1.4 illustrates the envisioned

scenario of performing the inspection of a wind turbine with a team of UAVs. A team of UAVs, ideally in the form of multicopter robots, constitutes the ideal platform for this type of mission. However, the multicopter platforms that are currently available for industrial inspection typically exhibit a limited level of autonomy beyond teleoperation, and they are meant to be used as standalone vehicles. The design and implementation of algorithms for cooperative inspection missions with multiple aerial platforms is the subject of keen research attention from the control and robotics communities. The EU Project AEROWORKS (`aeroworks2020.eu`), which has funded part of the work that this thesis is based on, is an example of a recent research effort in the direction of collaborative autonomous aerial inspection of wind turbines. The Swedish power supplier Skellefteå Kraft (SK) was a partner in the project and provided the developers with realistic inspection scenarios for development and testing.

When controlling a team of UAVs, coordination requires that the vehicles exchange information. In all realistic scenarios, such communication is handled by small embedded processors, and takes place over a wireless medium with limited throughput capacity. Therefore, it is crucial to design and implement algorithms that are able to attain the desired coordination with sparse, intermittent communication. Similarly to AUV applications, communication may occur directly between two robots, or asynchronously through a shared information repository hosted on a base station.

In this thesis, surveillance and inspection of 3D structures with a team of UAVs are the main motivating applications to the algorithms presented in Chapters 6 and 7. More precisely, the surveillance of a 3D structure with a team of autonomous mobile sensing agents is considered in Chapter 6, while the inspection of a 3D structure is considered in Chapter 7.

Automated platooning of heavy-duty vehicles

Heavy-Duty vehicles (HDVs) are responsible for a significant share of energy consumption and greenhouse gas emissions on a global scale. Since the number of active HDVs worldwide is correlated to economical growth, the environmental impact of these vehicles is predicted to grow even further in the coming years.

By letting a set of heavy-duty vehicles drive in a line, with short inter-vehicle distance, the aerodynamic drag applied to all vehicles but the first one in the line can be significantly reduced, which may lead to lowering the fuel consumption by up to ten percent. This technique is called *platooning*, and the vehicles that apply it are called a *platoon*. Figure 1.5 portrays a platoon consisting of three HDVs.

Recently, platooning has been the subject of keen research interest in the field of autonomous mobility (see for example Turri (2018) and references therein). A large number of providers of transport solutions, such as the Swedish company Scania,

Figure 1.5: A platoon of three HDVs. Source: Courtesy of Scania, license CC BY-NC-ND 3.0, `https://creativecommons.org/licenses/by-nc-nd/3.0/`.

are working to design and construct platforms that are capable of autonomous platooning (Scania AB, 2017). While autonomous platooning is promising in terms of fuel efficiency, it also presents impending challenges with respect to safety, especially in relation to external traffic. For example, platoons are expected to drive on public highways, where the presence of other vehicles cannot be neglected. Moreover, altitude changes have a significant impact on the motion of HDVs. Because of their large mass and limited engine power, HDVs experience large longitudinal forces in presence of slight slopes, which makes it hard to maintain a constant speed when moving from an uphill to a downhill segment or viceversa. Additional challenges arise if one considers applications such as adding a vehicle to the platoon, removing a vehicle from the platoon, or merging two different platoons.

A platoon of self-driving HDVs can be modeled as a multi-robot system, where the desired coordinated behavior is to safely platoon while minimizing fuel consumption. The vehicles need to exchange information about the trajectory to follow, and about their relative distances and velocities. However, the information exchange occurs over a shared wireless medium, which implies that communication may be subject to delays and packet drops. Hence, it becomes crucial to develop coordination algorithms that offer some form of robustness to this kind of communication

imperfections.

1.4 Problem formulation

This thesis is concerned with the design and implementation of algorithms that attain a desired coordinated behavior in a multi-robot system by means of intermittent communication. Communication may occur synchronously between agents, or asynchronously by exchanging data on a shared repository. More specifically, we consider benchmark coordinated behaviors such as synchronization (Chapters 3 and 4), formation (Chapter 5), coverage (Chapter 6), and inspection (Chapter 7).

We model a multi-robot system as a multi-agent system, where each agent is defined by a set of differential equations that describe quantitatively the behavior of the corresponding robot. Roughly speaking, the motion of each robot is modeled as a continuous-time phenomenon, while a communication instance is modeled as an instantaneous phenomenon.

The collective behavior of the multi-agent system is modeled in terms of an objective function, which describes succinctly and quantitatively the degree of coordination attained by the agents. The control design aims at optimizing, or at least improving, the value of this objective function. Each agent is endowed with a local controller that intermittently communicates with the controllers of the other agents. We analyze the closed-loop system mathematically to demonstrate its convergence properties, and we corroborate our theoretical results with numerical simulations. In some cases, we also provide preliminary experimental evaluations on an UAV platform.

The main challenges in attaining the coordination objectives reside in guaranteeing that the closed-loop system is well-posed (for example, that two consecutive communication instances required to attain coordination are separated by a finite time interval), and that the equilibrium configurations attained under the proposed control designs exhibit satisfactory properties in terms of stability and robustness.

The overall thesis problem is broken down in the following four research questions.

Q1 How can we obtain leader-following synchronization in a network of nonlinear agents by means of pairwise intermittent communication?

Q2 How can we coordinate a network of agents that can only exchange information asynchronously using a shared cloud repository?

Q3 How can we deploy a set of mobile sensing agents to appropriate locations for the surveillance of a 3D structure if the agents can only communicate pairwise and intermittently?

Q4 How can such mobile agents perform the inspection of a 3D structure if they can only communicate asynchronously through a shared cloud repository?

1.5 Thesis outline and contributions

This thesis is a compilation of results presented in or submitted to peer-reviewed journals and conferences. Chapter 2 illustrates some background notions and results that are used in the thesis. Chapters 3 to 7 address specific problems in the area of event-triggered or cloud-supported control of multi-robot system, and are based on one or more peer-reviewed publications by the author of the thesis. Each of these chapters is written to be relatively self-contained. Chapter 8 concludes the thesis by providing a summary of the results and offering possible directions for future work. A more detailed outline of the thesis is given as follows.

In Chapter 2, we provide a review of the background notions and results that are used in the thesis. The topics included in this chapter are: graph theory, agreement protocols, pinning control, hybrid systems, event-triggered control, coverage control, and effective coverage control. For each topic, we provide a few basic notions and we review some of the most relevant related works available in the literature. Most of the results that are mentioned in Chapter 2 are then used directly in the rest of the thesis; some of them are not used directly, but they are included to give a better picture of the topic.

In Chapter 3, we answer research question Q1. We consider the problem of synchronizing a network of nonlinear systems by using an event-triggered pinning control protocol. In particular, we consider networks with time-varying topologies, where the agents are linearly coupled. We design a model-based and event-triggered pinning control law, which drives the states of the agents to an a-priori specified, common reference trajectory. We derive a set of sufficient conditions under which the closed-loop system is well-posed, and the agents achieve exponential convergence to the reference trajectory. Networks with static topologies are studied as a special case, for which we also prove that there exists a lower bound for the inter-event times in the sequences of updates of the control signals. Different than most existing works on event-triggered multi-agent control, we envision an implementation of the control algorithm which does not require the agents to exchange state measurements at each update time. The agents exchange state measurements only when they establish their connection. When an agent updates its control signal to a new value, it is required to broadcast its value to its neighbors in the network. In this way, it is possible for neighboring agents to predict the state of each other consistently. The theoretical results are corroborated by numerical simulations.

Chapter 3 is based on the following publications.

A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Event-triggered pinning control of complex networks with switching topologies. *IEEE Conference on Decision and Control*, 2014.

A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Event-triggered pinning control of switching networks. *IEEE Transactions on Control of Network Systems*, 2(2):204–213, 2015a.

In Chapter 4, we answer research question Q2. We consider the problem of coordinating a team of second-order dynamical systems through the use of a remote information repository hosted on a cloud, which removes the need for direct inter-agent communication. Each agent schedules its own accesses independently, and does not need to be alert for information broadcast by other agents. When an agent accesses the repository, it uploads some data packets, and downloads other packets that were previously deposited by other agents. Therefore, each agent receives outdated information about the state of the other agents. The control law and the rule for scheduling the cloud accesses are designed to guarantee that the closed-loop system is well-posed and achieves a given coordination objective, even if each agent receives only outdated information about the state of the other agents. Our motivating example is a waypoint generation algorithm for AUVs, which, as described above, represent a challenging application, since underwater communication is interdicted. We demonstrate analytically that the closed-loop system is well-posed and reaches the desired coordination objective. The theoretical results are corroborated by numerical simulations.

Chapter 4 is based on the following publications.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Control of multi-agent systems with event-triggered cloud access. *European Control Conference*, 2015b.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Multi-agent trajectory tracking with event-triggered cloud access. *IEEE Conference on Decision and Control*, 2016.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Coordination of multi-agent systems with intermittent access to a cloud repository. T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, editors, *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*. Springer, 2017b.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Cloud-supported formation control of second-order multi-agent systems. *IEEE Transactions on Control of Network Systems*, Online, 2017c.

In Chapter 5, we further investigate research question Q2 with respect to a specific coordination behavior. Namely, we consider the problem of tracking and circumnavigating a target with unknown position through a network of autonomous mobile agents. The agents have access to intermittent measurements of the bearing of the target, and can also exchange data by asynchronously accessing a remote repository hosted on a cloud. First, we define mathematically the desired circumnavigation objective. Then, we design an event-triggered rule by which the agents perform the bearing measurements, and a self-triggered, recursive rule by which the agents schedule their accesses to the cloud repository. The information obtained from the measurements and from the cloud is used to steer the motion of each agent according to an appropriately designed control law. We show that, with the proposed controller, and under the proposed rules for triggering the bearing measurements and the cloud accesses, the closed-loop system is well-posed and attains the desired circumnavigation objective. We corroborate our theoretical results with a numerical simulation. We also present an experimental setup to validate the proposed algorithm.

Chapter 5 is based on the following publications.

> A. Boccia, A. Adaldo, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Tracking a mobile target by multi-robot circumnavigation using bearing measurements. *IEEE Conference on Decision and Control*, 2017.
>
> C. Cavaliere, D. Mariniello, A. Adaldo, F. Lo Iudice, D. V. Dimarogonas, K. H. Johansson, and M. di Bernardo. Cloud-supported self-triggered control for multi-agent circumnavigation. *Submitted to the IEEE Conference on Decision and Control*, 2018.

In Chapter 6, we answer research question Q3. We study a coverage problem for a network of mobile sensing agents with anisotropic and heterogeneous sensing patterns. The environment to cover is abstracted into a finite set of landmarks, where each landmark constitutes a point or small area of interest within the environment. We redefine the well-known notion of *Voronoi tessellation* to account for anisotropic patterns and discretized environments. With these premises, we define a distributed algorithm for coverage where communication is limited, pairwise, intermittent and asynchronous. We demonstrate the convergence properties of the proposed algorithm mathematically, and we corroborate the theoretical results with numerical simulations. We also illustrate two experimental implementations of the proposed algorithm that employ an AUV as a sensing agent.

Chapter 6 is based on the following publications.

A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Hybrid coverage and inspection control for anisotropic mobile sensor teams. *IFAC World Congress*, 2017a.

A. Adaldo, S. S. Mansouri, C. Kanellakis, D. V. Dimarogonas, K. H. Johansson, and G. Nikolakopoulos. Cooperative coverage for surveillance of 3D structures. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017d.

In Chapter 7, we answer research question Q4. Namely, we consider an effective coverage problem (i.e., an inspection problem) for a network of mobile sensing agents exchanging information on a cloud repository. We use a similar system model as in Chapter 6, where each agent is described in terms of its kinematics and sensing pattern, while the environment to inspect is abstracted into a finite set of landmarks. Inter-agent communication is completely replaced by communication over a cloud. The cloud is modeled as a shared information repository which receives asynchronous and partial information about the progress of the inspection, but has no computational power. As a motivating example, we consider the inspection of a 3D structure abstracted into a finite set of landmarks, where each landmark carries information about the local curvature of the surface. The algorithm is formally shown to complete the inspection in finite time. The theoretical results are corroborated by a simulation, and we also illustrate the setup for a preliminary experimental evaluation.

Chapter 7 is based on the following publication.

A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Cloud-supported effective coverage of 3D structures. *European Control Conference*, 2018.

In Chapter 8, we present a summary of the results described in the thesis, and we discuss possible directions for future research. Overall conclusions are drawn first; then, the results obtained in Chapters 3 to 7 are reviewed. Finally, possible future developments are outlined.

The following publications do not correspond directly to any technical content in this thesis, but they are relevant to the endeavors considered therein.

J. Wei, S. Zhang, A. Adaldo, X. Hu, and K. H. Johansson. Finite-time attitude synchronization with a discontinuous protocol. *IEEE International Conference on Control & Automation (ICCA)*, 2017b.

J. Wei, S. Zhang, A. Adaldo, J. Thunberg, X. Hu, and K. H. Johansson. Finite-time attitude synchronization with a distributed discontinuous protocol. *IEEE Transactions on Automatic Control*, Online, 2018.

For each of the listed publications, the order of the authors reflects their contribution, in the sense that the first authors have contributed more directly to the control design and to the writing of the paper, while the last authors have taken a supervisory role. Where listed as the first author of a publication, the author of this thesis has contributed the control design, the numerical simulations, and either the majority or all of the writing. The experiments described in Adaldo et al. (2017d) have been performed by the coauthors affiliated with Luleå University of Technology (LTU). In Wei et al. (2017b, 2018), the author of this thesis has contributed actively to the control design, but to a smaller extent to the writing of the papers. In Boccia et al. (2017), Cavaliere et al. (2018), the author of this thesis has supervised the control design and participated actively to the writing of the papers.

Chapter 2

Background

Siede con le vicine
su la scala a filar la vecchierella,
incontro là dove si perde il giorno;
e novellando vien del suo buon tempo,
quando ai dì della festa ella si ornava,
ed ancor sana e snella
solea danzar la sera intra di quei
ch'ebbe compagni nell'età più bella.

G. Leopardi,
Il sabato del villaggio*, vv 8–14.*

In this chapter, we review some of the existing research work that has offered the theoretical grounds to this thesis. For each topic that we touch, we give some fundamental notions and recall some of the most well-known research works related to that topic.

The rest of this chapter is organized as follows. In Section 2.1, we present some fundamental notions in graph theory. In Section 2.2, we describe the consensus problem. In Section 2.3, we introduce the pinning control problem. In Section 2.4, we recall some fundamental notions related to hybrid systems. In Section 2.5, we describe the main ideas related to triggered control. In Section 2.6, we describe the coverage problem for multi-agent systems in its classical formulation. Finally, in Section 2.7, we describe the effective coverage problem for multi-agent system.

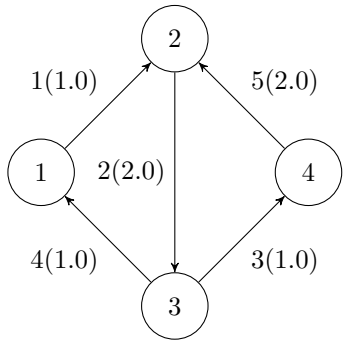

Figure 2.1: Illustration of a graph with $N = 4$ nodes and $M = 5$ edges.

2.1 Graph theory

Graph theory is an important tool in the study of multi-agent systems, because, in many cases of interest, a graph constitutes a convenient abstraction for a group of interconnected systems. The interested reader will find a comprehensive treatment in Newman (2010), Gould (2012), Dietsel (2016) among others. A more concise introduction to graph theory, with special focus on its use in the study analysis of multi-agent systems, is found in Mesbahi and Egerstedt (2010). In this section, we present only a few selected notions that we will use in Chapters 3 to 5.

In this thesis, a *graph* is defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. $\mathcal{V}$ is a finite set, and its elements are called the *vertexes* of the graph. Although any objects may be used as vertexes, in the context of multi-agent systems it is common to take $\mathcal{V} = \{1, \dots, N\}$ and let each vertex index one agent in the system. $\mathcal{E} = \{e_1, \dots, e_M\}$ is a proper subset of $\mathcal{V} \times \mathcal{V}$, with the condition $(i, i) \notin \mathcal{E}$ for all $i \in \mathcal{V}$. The elements of $\mathcal{E}$ are called the *edges* of the graph. We let M be the number of edges in the graph, and we let the edges be indexed with the integers from 1 to M in any order. Moreover, we let e_k denote the kth edge. Each edge (j, i) represents some form of information flow from agent j to agent i. Finally, $w : \mathcal{E} \to \mathbb{R}_{>0}$ is called the *weight function* of the graph, with $w(j, i)$ being the *weight* of edge (j, i). With a slight abuse of notation, the weight of edge (j, i) is also denoted w_{ji}. The weight w_{ji} of an edge (j, i) represents the strength of the influence that agent j has on agent i.

Usually, a graph is drawn by representing each node as a circle, and each edge as an arrow from one node to another. Namely, if $(j, i) \in \mathcal{E}$, then an arrow is drawn from node j to node i. Each node is labeled with its cardinality, and each edge is labeled with its cardinality and weight. For example, Figure 2.1 illustrates a graph with $N = 4$ nodes and $M = 5$ edges.

The set $\mathcal{N}_i = \{j \in \mathcal{V} : \ (j, i) \in \mathcal{E}\}$ is called the *neighborhood* of vertex i, and the

vertexes $j \in \mathcal{N}_i$ are called the *neighbors* of i. The *degree* d_i of a vertex i is defined as the sum of the weights of its neighbors, $d_i = \sum_{j \in \mathcal{N}_i} w(j,i)$. The N-by-N matrix A such that $A_{ij} = w(j,i)$ is called the *adjacency matrix* of the graph. The N-by-N diagonal matrix D such that $D_{ii} = d_i$ is called the *degree matrix* of the graph. The N-by-N matrix $L = D - A$ is called the *Laplacian* of the graph. Since all rows of the Laplacian sum to zero, 1_N is always an eigenvector of the Laplacian with eigenvalue zero. For example, for the graph in Figure 2.1, we have:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \tag{2.1}$$

$$D = \operatorname{diag}(1, 3, 2, 1); \tag{2.2}$$

$$L = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 3 & 0 & -2 \\ 0 & -2 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{2.3}$$

The N-by-M matrix B such that, for each edge $e_k = (j,i)$ we have $B_{ik} = 1$, $B_{jk} = 1$, and $B_{vk} = 0$ for all $v \notin \{i,j\}$, is called the *incidence matrix* of the graph. The N-by-M matrix W such that for each edge $e_k = (j,i)$ we have $W_{ik} = w_{ji}$ and $W_{vk} = 0$ for all $v \neq i$ is called the *weight in-incidence matrix*. For example, for the graph in Figure 2.1, we have:

$$B = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}; \tag{2.4a}$$

$$W = \begin{bmatrix} 0 & 0 & 0 & 1.0 & 0 \\ 1.0 & 0 & 0 & 0 & 2.0 \\ 0 & 2.0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \end{bmatrix}. \tag{2.4b}$$

One can verify that, for any graph, $L = WB^\intercal$.

Given two distinct vertexes j and i, a *path* from j to i is a finite sequence of distinct vertexes $v_0, v_1, \ldots, v_\ell$ such that $v_0 = j$ and $v_\ell = i$. A subset $\mathcal{T}$ of the edges is called a *spanning tree* if it has the following properties: (i) there exists a vertex r such that there exist paths from r to all other vertexes made up of edges in $\mathcal{T}$; (ii) property (i) does not hold for any proper subset of $\mathcal{T}$. The vertex r is called the *root* of the spanning tree. For example, for the graph in Figure 2.1, $\mathcal{T} = \{e_1, e_2, e_3\}$ is a spanning tree.

Proposition 2.1. *A spanning tree contains exactly $N - 1$ edges.*

Proposition 2.2. *If a graph contains a spanning tree, then the algebraic multiplicity of the zero eigenvalue of the Laplacian is* 1*, and all other eigenvalues have positive real parts.*

Suppose that a graph contains a spanning tree. Without loss of generality, we let the edges be indexed as so that the first $N-1$ edges constitute the spanning tree. Accordingly, partition the incidence matrix as $B = [B_{\mathcal{T}}\ B_{\mathcal{C}}]$ and the weight-incidence matrix as $W = [W_{\mathcal{T}}\ W_{\mathcal{C}}]$.

Proposition 2.3. *Given a spanning tree $\mathcal{T}$, the matrix $B_{\mathcal{T}}$ is full column-rank $N-1$.*

Since $B_{\mathcal{T}}$ is full column rank, it has an unique left pseudoinverse, which we denote as $B_{\mathcal{T}}^{\dagger}$. Moreover, we denote $T = B_{\mathcal{T}}^{\dagger} B_{\mathcal{C}}$. The $(N-1)$-by-$(N-1)$ matrix $R = B_{\mathcal{T}}(W_{\mathcal{T}} + W_{\mathcal{C}} T^{\mathsf{T}})$ is called the *reduced edge Laplacian* of the graph.

Proposition 2.4. *If a graph contains a spanning tree, all eigenvalues of the reduced edge Laplacian have positive real parts, and they coincide, including their multiplicities, with the nonzero eigenvalues of the Laplacian.*

A graph is called *undirected* if, for all $(i,j) \in \mathcal{V}^2$, it holds that $(j,i) \in \mathcal{E} \iff (i,j) \in \mathcal{E}$ and $w(j,i) = w(i,j)$ for all $(j,i) \in \mathcal{E}$. The Laplacian of an undirected graph is symmetric, and therefore it has real eigenvalues.

2.2 Consensus

Consensus is a benchmark problem within the broader field multi-agent systems, and may serve as an abstract model for a wide variety of coordination behaviors. Consensus is a problem of distributed computation, where a set of autonomous agents with limited communication capabilities are required to reach some form of coordination. Typical examples are: a team of mobile robots that are required to meet at the same point in space; a set of sensors that are required to average their measurements to present a single measurement to the user; a set of oscillators that are required to synchronize their oscillations. Consensus problems are usually formulated in terms of graphs, with each node corresponding to an agent, and each edge corresponding to a communication channel, or an information flow, between two agents.

The appearance of the consensus problem in the engineering literature may be traced back to DeGroot (1974), where the author analyzes a consensus-like behavior in the context of opinion dynamics in a social network. Consensus has attracted an immense amount of research in the past few decades, and an exhaustive review of the related results is out of the scope of this thesis. Let us recall the pioneering

works of Pecora and Carroll (1990), Carroll and Pecora (1991); the later studies from Pecora et al. (1997), Pecora and Carroll (1998), Watts and Strogatz (1998), Strogatz (2000), Barahona and Pecora (2002), Acebron et al. (2005) on network synchronization; the use of consensus strategies for coordination of networks of autonomous vehicles from Olfati-Saber (2006), Ren (2006, 2007), Ren and Beard (2008); and the use of gossip algorithms to reach consensus in a network system from Boyd et al. (2006), Aysal et al. (2009), Carli et al. (2010). Recently, Wei et al. (2017a) have studied consensus algorithms in networks with arbitrary sign-preserving nonlinearities.

The interested reader will find a modern overview of the main consensus problems and the corresponding resolving algorithms in Olfati-Saber et al. (2007). Here we present only a few selected definitions and results that will be used in Chapters 3 to 5.

Consider a network of N agents, and let $x_i(t) \in \mathbb{R}^n$ be the value held by the ith agent at time t. We say that the network achieves *asymptotic consensus* if

$$\lim_{t\to\infty} (x_i(t) - x_j(t)) = 0_n \quad \forall (i,j) \in \{1,\dots,N\}^2. \tag{2.5}$$

Moreover, given $\epsilon > 0$, we say that the agents achieve *practical consensus* with tolerance ϵ if

$$\limsup_{t\to\infty} \|x_i(t) - x_j(t)\| \leq \epsilon \quad \forall (i,j) \in \{1,\dots,N\}^2. \tag{2.6}$$

Diffusive coupling is a simple consensus algorithm based on updating the value held by one agent according to its difference with the values held by a subset of the other agents. Considering a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ with $\mathcal{V} = \{1,\dots,N\}$, diffusive coupling can be written as

$$\dot{x}_i(t) = \sum_{j\in\mathcal{N}_i} w_{ij}(x_j(t) - x_i(t)) \quad \forall i \in \mathcal{V}\,. \tag{2.7}$$

Denoting $x(t) = [x_1(t)^\intercal, \dots, x_N(t)^\intercal]^\intercal$, the diffusive coupling equation (2.7) can be written compactly as

$$\dot{x}(t) = -(L \otimes I_n)x(t), \tag{2.8}$$

where L is the Laplacian of the graph, and $\otimes$ denotes the Kronecker product. A fundamental result in consensus theory states that diffusive coupling attains asymptotic consensus from any initial conditions as long as the underlying graph contains a spanning tree. This result can be elicited easily if we rewrite the diffusive coupling equation in terms of the reduced edge Laplacian. Namely, we left-multiply

both sides of (2.8) by $B_{\mathcal{T}}^{\intercal} \otimes I_n$, and we recall that

$$\begin{aligned} L =& WB^{\intercal} \\ =& W_{\mathcal{T}} B_{\mathcal{T}}^{\intercal} + W_{\mathcal{C}} B_{\mathcal{C}}^{\intercal} \\ =& W_{\mathcal{T}} B_{\mathcal{T}}^{\intercal} + W_{\mathcal{C}} (B_{\mathcal{T}} T)^{\intercal} \\ =& (W_{\mathcal{T}} + W_{\mathcal{C}} T^{\intercal}) B_{\mathcal{T}}^{\intercal}. \end{aligned} \tag{2.9}$$

Then, we can rewrite (2.8) as

$$\begin{aligned} (B_{\mathcal{T}}^{\intercal} \otimes I_n)\dot{x}(t) =& -(B_{\mathcal{T}}^{\intercal} \otimes I_n)((W_{\mathcal{T}} + W_{\mathcal{C}} T^{\intercal}) B_{\mathcal{T}}^{\intercal} \otimes I_n) x(t), \\ =& -(R \otimes I_n)(B_{\mathcal{T}}^{\intercal} \otimes I_n) x(t). \end{aligned} \tag{2.10}$$

Since $-R$ is Hurwitz, from (2.10) we know that $(B_{\mathcal{T}}^{\intercal} \otimes I_n)x(t)$ must converge to the zero vector exponentially. However, since $B_{\mathcal{T}}$ refers to the edges in a spanning tree, for any two nodes i, j, the difference $x_j(t) - x_i(t)$ can be written as a linear combination of entries of $(B_{\mathcal{T}} \otimes I_n)x(t)$. Hence, for any $(i, j) \in \mathcal{V}^2$, $x_j(t) - x_i(t)$ must converge to zero exponentially. This result can be formalized as follows.

Proposition 2.5. *Consider N networked agents with dynamics (2.7). If the underlying graph contains a spanning tree, the agents attain asymptotic consensus. Moreover, for each $(i, j) \in \mathcal{V}^2$ there exists $\alpha_{ij} > 0$ such that*

$$\|x_i(t) - x_j(t)\| \leq \alpha_{ij} \, \mathrm{e}^{-\rho t}, \tag{2.11}$$

where $\rho = \min\{\mathrm{Re}(\lambda) : \ \lambda \in \mathrm{eig}(R)\}$, and R is the reduced edge Laplacian of $\mathcal{G}$. Here, $\mathrm{Re}(\lambda)$ denotes the real part of λ, which may be a complex number.

When the agents are subject to additive, bounded disturbances, a similar result to proposition 2.5 can be formulated establishing that the agents attain practical consensus.

Proposition 2.6. *Consider N networked agents with dynamics $\dot{x}_i(t) = u_i(t) + d_i(t)$, where $u_i(t)$ is a control input and $d_i(t)$ is a disturbance input. Let the agents be connected over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, and let the control inputs be computed as diffusive couplings $u_i(t) = \sum_{j \in \mathcal{N}_i} w_{ij}(x_j(t) - x_i(t))$. Let the disturbances be bounded as $\|d_i(t)\| \leq \varsigma_{i,0} \, \mathrm{e}^{-\lambda_\varsigma t} + \varsigma_{i,\infty}$, where $\varsigma_{i,0}, \varsigma_{i,\infty}$ are positive constants for each $i \in \mathcal{V}$, and λ_ς is a positive constant. If $\mathcal{G}$ contains a spanning tree, then the network attains practical consensus with a tolerance that depends only on $\mathcal{G}$ and on the parameters $\varsigma_{1,\infty}, \ldots, \varsigma_{N,\infty}$. Moreover, if $\varsigma_{i,\infty} = 0$ for all $i \in \mathcal{V}$, then, for each $(i, j) \in \mathcal{V}^2$, there exists $\alpha_{ij} > 0$ such that*

$$\|x_i(t) - x_j(t)\| \leq \alpha_{ij} \, \mathrm{e}^{-\rho t}, \tag{2.12}$$

where $\rho = \min(\{\mathrm{Re}(\lambda) : \ \lambda \in \mathrm{eig}(R)\} \cup \lambda_\varsigma)$, and R is the reduced edge Laplacian of $\mathcal{G}$.

2.3 Pinning control

Pinning control is a particular consensus problem where one wants a set of agents to synchronize onto a given reference trajectory by exploiting the interconnections among the agents, rather than controlling each individual agent. The agents that receive direct feedback from the reference are said to be *pinned*.

The origins of pinning control can be traced back to Grigoriev et al. (1997), where the authors propose a pinning strategy to synchronize a network of chaotic oscillators. Pinning control received plenty of research attention towards the turn of the century, and here we only recall a few of the related works. Wang and Chen (2002) study how the selection of the pinned agents affects the controllability of the network. Li et al. (2004) investigate stabilization of multi-agent systems via pinning control. Porfiri and di Bernardo (2008) introduce the concept of *pinning controllability* of a multi-agent system, and they introduce several criteria to assess this property. Wu et al. (2009) apply pinning control to a problem of cluster synchronization, which means that the agents are divided into subsets and each subset is required to synchronize onto a different trajectory than the others. (Song et al., 2010) apply pinning control to a leader-following problem for a network of second-order systems. (Liu et al., 2011) study how the minimum number of pinned nodes that is necessary to control the network varies depending on the network topology.

Formally, the control objective of a pinning control problem can be written as

$$\lim_{t\to\infty}(x_i(t) - r(t)) = 0_n \quad \forall i \in \{1, \dots, N\}, \tag{2.13}$$

where $r(t) \in \mathbb{R}^n$ is a given reference trajectory. Pinning control algorithms usually consist in controlling a small subset of the agents directly, while relying on the interconnections to steer the other agents. Suppose that the reference trajectory $r(t)$ satisfies $\dot{r}(t) = f(r(t))$, and that the agents have dynamics $\dot{x}_i(t) = f(x_i(t)) + u_i(t)$, where $u_i(t)$ is a control signal. Moreover, suppose that the agents are connected according to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. Then, a simple pinning control algorithm is

$$u_i(t) = p_i(r(t) - x_i(t)) + \sum_{j\in\mathcal{N}_i} w_{ij}(x_j(t) - x_i(t)), \tag{2.14}$$

where $p_i > 0$ if the ith agent is controlled directly, and $p_i = 0$ otherwise. Whether algorithm (2.14) achieves control objective (2.13) depends on the network topology and on the dynamics f of the agents.

2.4 Hybrid systems

Hybrid systems are a powerful formalism to model dynamical systems that exhibit continuous-time dynamics as well as instantaneous phenomena. The origins of this

formalism can be traced back to Witsenhausen (1966). Since then, a large number of hybrid system models have been proposed, and giving an exhaustive account is out of the scope of this thesis. Notable reference books for hybrid systems are van der Shaft and Schumacher (2000), Goebel et al. (2012).

In this thesis, we follow the hybrid-system model presented in Goebel et al. (2012). According to this model, a hybrid system is a tuple

$$\mathcal{H} = (C, F, D, G). \tag{2.15}$$

$C \subseteq \mathbb{R}^n$ is called the *flow set*, $F : C \to 2^{\mathbb{R}^n}$ is called the *flow map*, $D \subseteq \mathbb{R}^n$ is called the *jump set*, and $G : D \to 2^{\mathbb{R}^n}$ is called the *jump map*. Roughly speaking, F describes the continuous dynamics (i.e., the flow) of the system, while G describes the instantaneous phenomena (i.e., the jumps). When dealing with hybrid system, the ordinary concept of time as a scalar variable needs to be abandoned in favor of a construct that captures both flow and jumps.

A *hybrid time domain* Θ is a (finite or infinite) sequence of intervals $I_k = [t_k, t_{k+1}]$, with $k \in \{0, 1, \ldots\}$ and $t_k \leq t_{k+1}$. If the sequence is finite, then the last interval may be finite or extend to infinity to the right. The index k functions as a jump counter: the interval I_k happens between the kth and the $(k+1)$th jump. A value in a hybrid time domain is completely specified by a tuple (t, k), with $t \in I_k$ and $k \in \{0, 1, \ldots\}$. Therefore, we write with abuse of notation $(t, k) \in \Theta$ to mean that $t \in I_k$, where I_k is the kth interval of the hybrid time domain Θ. We define the following two operators on a hybrid time domain:

$$\sup_t \Theta = \sup\{t \in \mathbb{R}_{\geq 0} : \ \exists k \in \mathbb{N} : \ (t, k) \in \Theta\}; \tag{2.16a}$$

$$\sup_k \Theta = \sup\{j \in \mathbb{N} : \ \exists t \in \mathbb{R}_{\geq 0} : \ (t, k) \in \Theta\}. \tag{2.16b}$$

A hybrid time domain Θ is said to be *complete* if either $\sup_t \Theta$ or $\sup_k \Theta$ is infinite; it is said to be *Zeno* if $\sup_t \Theta$ is finite and $\sup_k \Theta$ is infinite.

Note that a hybrid time domain is completely defined by the sequence $\{t_k\}$ of its jump times. (The sequence includes infinity if the last interval extends to infinity to the right.) Therefore, we can introduce the notion of union of two hybrid time domains.

Definition 2.1. *Let Θ_1, Θ_2 be two hybrid time domains. The union of Θ_1 and Θ_2 is defined as the hybrid time domain whose jump times are the union of the jump times of Θ_1 and the jump times of Θ_2. (Infinity is counted once, if present in either of the two sequences.) The union of three or more hybrid time domains is defined associatively from the union of two time domains.*

Proposition 2.7. *The union of any finite number of hybrid time domains is Zeno if and only if one or more of the hybrid time domains is Zeno.*

A *hybrid arc* is a function $x : \Theta \to \mathbb{R}^n$, where Θ is a hybrid time domain, with the property that, in each interval I_k of Θ, the function $x(\cdot, k)$ is locally absolutely continuous. This property implies that $x(\cdot, k)$ is differentiable almost everywhere in I_k, and we denote its derivative as $\dot{x}(\cdot, k)$. A hybrid arc is said to be *complete* (respectively, *Zeno*) if its domain is complete (respectively, Zeno); it is said to be *precompact* if it is complete and its range is bounded.

A *solution* of a hybrid system is a hybrid arc $x : \Theta \to \mathbb{R}^n$ with the following properties: (i) $x(t, k) \in C$ for all $t \in \operatorname{Int} I_k$; (ii) $\dot{x}(t, k) \in F(x(t, k))$ for almost all $t \in I_k$; (iii) for all $(t, k) \in \Theta$ such that $(t, k + 1) \in \Theta$, it holds that $x(t, k) \in D$ and $x(t, k + 1) \in G(x(t, k))$.

The analysis of Zeno solutions in hybrid systems has received special research attention. When the hybrid system is a model of a closed-loop control system, Zeno solutions are considered an undesired phenomenon, because they often imply that the time interval between two consecutive control updates converges to zero. In fact, one sometimes says that a hybrid system that admits one or more Zeno solutions is not well posed. Among the numerous works that study the Zeno phenomenon, we refer the reader to Johansson et al. (1999), Heymann et al. (2005).

The following result constitutes a generalization of LaSalle's invariance principle to hybrid systems.

Proposition 2.8 (Corollary 8.4 in Goebel et al. (2012))**.** *Given a hybrid system* (2.15)*, consider a function $V : \mathbb{R}^n \to \mathbb{R}$, continuously differentiable in a neighborhood of C. Consider also the functions*

$$u_C(x) = \begin{cases} \max_{v \in F(x)} \nabla V(x)^\intercal v & \text{if } x \in C, \\ -\infty & \text{otherwise}, \end{cases} \tag{2.17a}$$

$$u_D(x) = \begin{cases} \max_{\xi \in G(x)} V(\xi) - V(x) & \text{if } x \in D, \\ -\infty & \text{otherwise}, \end{cases} \tag{2.17b}$$

where ∇V denotes the gradient of V. Suppose that, for a given set $U \subset \mathbb{R}^n$, it holds that $u_C(z) \leq 0$ and $u_D(z) \leq 0$ for all $z \in U$. Consider a precompact solution x of $\mathcal{H}$ with $\overline{\operatorname{rge} x} \subset U$. Then, for some $r \in V(U)$, the set $V^{-1}(r) \cap U \cap (\overline{u_C^{-1}(0)} \cup (u_D^{-1}(0) \cap G(u_D^{-1}(0))))$ has at least one invariant subset, and x converges to the largest such subset.

Hybrid automaton are a special class of hybrid systems. For a more detailed introduction to hybrid automata, the interested reader is referred to Lygeros et al. (2003). Since a hybrid automaton is also a hybrid system as defined by (2.15), there is a procedure to rewrite any hybrid automaton to the form (2.15). The interested reader will find the procedure in Goebel et al. (2012, Chapter 1).

A hybrid automaton is written as a tuple

$$\mathcal{H} = (Q, X, I, F, D, E, G, R), \tag{2.18}$$

where: $Q = \{q^1, q^2, \ldots\}$ is a set of discrete states; $X \subset \mathbb{R}^n$ is a continuous state space; $I \subseteq Q \times X$ is a set of possible initial states; $F : Q \times X \to X$ is a set of vector fields, with $f(q, x)$ being the dynamics of x under state q; $D : Q \to 2^X$ is a set of domains, with $D(q)$ being the domain under state q; $E :\subseteq Q \times Q$ is a set of edges, with $(q^1, q^2) \in E$ signifying a possible transition from state q^1 to state q^2; $G : E \to 2^X$ is a set of guards, meaning that $x \in G_{1,2}$ triggers a transition from q^1 to q^2; $R : E \times X \to 2^X$ is a set of reset maps, meaning that, upon a transition from q^1 to q^2, the continuous state x of the system is reset to a value in $R_{1,2}(x)$. A hybrid automaton can be represented as a graph, where each node represents a discrete state and each edge represents a transition. Each node is labeled with the corresponding vector field, while each edge is labeled with the corresponding guard and reset map. If the reset map for an edge (q^1, q^2) is not specified, it is implied that $R_{1,2}(x) = x$ for all $x \in G_{1,2}$. Initial discrete states are labeled with a *start* flag.

2.5 Event-triggered control

Event-triggered control is a control strategy where the control input is recomputed only when a specified condition is verified. Event-triggered control strategies can be used to reduce variations in the control input, which are usually associated to actuator wear, or to reduce communication between different parts of a control system. Given a controller in the form

$$u(t) = \kappa(x(t)), \tag{2.19}$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$, a possible event-triggered implementation is

$$u(t) = \kappa(x(t_k)) \quad \forall t \in [t_k, t_{k+1}), \tag{2.20a}$$

$$t_{k+1} = \inf\{t > t_{i,k} : \ \sigma_k(t) \geq 0\}, \tag{2.20b}$$

where $\sigma_k(t)$ is the function that triggers the control updates. For example, the threshold function could be chosen as

$$\sigma_k(t) = \|x(t) - x(t_k)\| - \epsilon, \tag{2.21}$$

with $\epsilon > 0$. In this case, the event-triggered design invokes a control update every time that the difference between the current state $x(t)$ and the state $x(t_k)$ used in the controller has overcome the chosen threshold ϵ.

Event-triggered control is particularly appealing in the contexts of networked control systems and multi-agent systems. In a networked control system, sensing,

control and actuation may reside in different physical locations, and need to communicate wireless: because the wireless channel has limited bandwidth, it is desirable to reduce the amount of information that needs to circulate among different devices. Similarly, in a multi-agent system it is desirable to reduce the amount of information that needs to be exchanged between different agents.

Event-triggered control is usually considered in opposition to *periodic control*, where the control updates are triggered periodically, and which has been–for several decades–the standard technique for implementing feedback control on digital platform.

It is possible to further distinguish between *event-triggered* and *self-triggered* control. We say that a controller is event-triggered when the condition that triggers the control updates is checked time-continuously. Conversely, we say that a controller is self-triggered when control updates are scheduled recursively: when the control signal is updated, the controller also schedules the following update.

The origins of event-triggered control cannot be traced back to a specific publication, but aperiodic implementations of feedback control loops have been appearing in the literature for about half a century. Among the best-known works that proposed systematic techniques for event-triggered control designs, we recall Tabuada (2007), Henningsson et al. (2008), Heemels et al. (2008), Anta and Tabuada (2010), Postoyan et al. (2013). A tutorial introduction to event-triggered and self-triggered control is offered in Heemels et al. (2012).

In this thesis, we use event-triggered control in the context of multi-agent coordination: the control updates correspond to communication events between two different agents. Such communication is event-triggered to reduce the effort put on the communication medium. Event-triggered control for multi-agent systems has received plenty of research attention in the last decade. For example, Dimarogonas et al. (2012), Seyboth et al. (2013), Dormido et al. (2013), Garcia et al. (2014) study event-triggered control for consensus-type coordination in multi-agent systems.

2.6 Coverage control

Coverage is a problem of multi-agent coordination with direct applications in mobile robotics and sensor networks. The coverage problem consists in finding a distributed algorithm that deploys a set of agents in a given domain of interest, in such a way that the collective perception of the domain attained by the agents is optimized, or at least improved. The standard reference for coverage control of a mobile sensing network is Cortes et al. (2004). Notable variations on the theme include Kwok and Martinez (2010), Martinez et al. (2007), Pavone et al. (2011), Zhong and Cassandras (2011), Bullo et al. (2012), Durham et al. (2012), Nowzari and Cortes (2012), Stergiopoulos and Tzes (2013), Le Ny and Pappas (2013), Kantaros et al.

(2015), Stergiopoulos et al. (2015), Patel et al. (2016).

A formal definition of the coverage problem can be given as follows. Let $\Omega \subset \mathbb{R}^n$ be a compact set. Consider N mobile agents indexed as $1, \ldots, N$, and let $p_i \in \mathbb{R}^n$ be the position occupied by the ith agent. Let $f : (\mathbb{R}_+)^2 \to \mathbb{R}_+$ and $\phi : \mathbb{R}^n \to \mathbb{R}_+$. The objective is to find positions $p_1, \ldots, p_N$ that minimize the following cost function:

$$V_\Omega(p) = \int_\Omega \min_{i \in \{1,\ldots,N\}} f(p_i, q)\phi(q)\mathrm{d}q, \tag{2.22}$$

where we have denoted $p = (p_1, \ldots, p_N)$. In this formalism, Ω is the environment where the agents are deployed and where they can navigate, ϕ is a measure of the importance of different points in Ω, and f is a model of how the agents perceive the surrounding environment. For example, if the agents are omnidirectional sensors, one may choose

$$f(p, q) = \|p - q\|^2, \tag{2.23}$$

to signify that perception improves when the sensing agent is close to the observed point. This is the model considered, for example, in Cortes et al. (2004). It is worth noting that the cost function V_Ω is generally nonconvex, even if the functions f and ϕ happen to be convex. Consequently, the coverage problem has no efficient general solution, and remains a challenging problem to date. Most of the related research works assume that the agents have a simple kinematic model (for example, $\dot{p}_i(t) = u_i(t)$ where $u_i(t)$ is a control signal) and focus on designing a distributed controller that drives the agents to positions that progressively improve the value of V_Ω, possibly reaching a local minimum. This behavior is usually achieved by using a version of *Lloyd's algorithm*, which was introduced by Lloyd (1982), and has a modern treatment by Du et al. (1999).

2.7 Effective coverage

Effective coverage is the problem of steering the motion of a set of sensing agents to search an assigned area, until all locations therein have been searched sufficiently well with respect to an assigned satisfaction threshold.

A formal definition of the effective coverage problem can be given as follows. Similarly to what we have done for the coverage problem, consider N mobile agents indexed as $1, \ldots, N$, and let $p_i(t) \in \mathbb{R}^n$ be the position occupied by the ith agent at time t. Let $f : (\mathbb{R}^n)^2 \to \mathbb{R}_+$ be a model of perception, in the sense that $f(p, q)$ corresponds to the perception of point q from position p. The effective coverage of a point q at time t is given by

$$C_q(t) = \int_0^t \sum_{i=0}^{N} f(p_i(\tau), q)\mathrm{d}\tau. \tag{2.24}$$

The control objective is to steer the motion of the agents in such a way that all points attain an effective coverage larger than or equal to a desired value C^* in finite time. In some cases, additional challenges are considered, such as collision avoidance among the sensing agents.

The best known work on the effective coverage problem is by Hussein and Stipanovic (2007), which also addresses collision avoidance among the sensing agents. Notable variations on the theme include Wang and Hussein (2010), where the authors consider intermittent communication among the agents, and (Panagou et al., 2016), where the authors consider agents with anisotropic sensing patterns.

Chapter 3

Event-triggered pinning control

Già tutta l'aria imbruna,
Torna azzurro il sereno, e tornan l'ombre
Giù da' colli e da' tetti,
Al biancheggiar della recente luna.

G. Leopardi,
Il sabato del villaggio*, vv 16–19.*

In this chapter, we begin our exploration of control designs for multi-agent coordination with sparse communication with an algorithm for event-triggered pinning control of switching networks.

The rest of the chapter is organized as follows. In Section 3.1, we review the existing related work and highlight the novel contributions offered in the chapter. In Section 3.2, we give a mathematical formulation of the pinning control problem under investigation, and we outline the proposed control algorithm to address such problem. In Section 3.3, we discuss a distributed and model-based implementation of the proposed algorithm that aims at reducing the necessary amount of communication among the agents. In Section 3.4, we state our main convergence result, whose proof occupies Sections 3.5 to 3.7. In Section 3.8, we specialize the general results to the case of networks with fixed topologies. In Section 3.9, we present a simulated network of nonlinear systems under the proposed algorithm, and we show that the simulation corroborates the theoretical results. Section 3.10 concludes the chapter with a summary of the results.

3.1 Introduction

Multi-agent systems constitute a suitable model for many distributed phenomena in biology, social sciences, physics, economics and engineering. Therefore, the topic of distributed control of multi-agent systems has attracted much research interest in the last few decades (Strogatz, 2001, Newman, 2003, Boccaletti et al., 2006, Olfati-Saber et al., 2007, Arenas et al., 2008).

Pinning control is a strategy to steer the collective behavior of a multi-agent system in a desired manner. In pinning control problems the goal is for a set of interconnected dynamical systems to synchronize onto a given reference trajectory. The reference trajectory is supposed to be a solution of the dynamics of the uncoupled agents, known a-priori, and corresponding to a specified control objective. A small fraction of the agents are selected in order to receive direct feedback control. Such agents are called *pins* or *pinned agents*. The remaining agents are influenced only through their connections with other agents.

Research on pinning control has been carried out from both physical and engineering perspectives. The focus is usually on the design of adaptive pinning controllers (Lu, 2007, Chen et al., 2007, Zhou et al., 2008), or on finding criteria for optimal selection of the agents to control (Grigoriev et al., 1997, Li et al., 2004, Song et al., 2010), or on finding sufficient conditions for synchronization (Wang and Chen, 2002, Chen et al., 2007, Porfiri and di Bernardo, 2008). Such conditions usually relate to the dynamics of the agents, to the network topology and to the pinning topology.

In many scenarios of multi-agent coordination, an assumption that the network topology is constant over time is unrealistic. Topology variations are due to imperfect communication between agents, or simply the existence of a proximity range beyond which communication is not possible. A large number of papers investigate synchronization (Olfati-saber and Murray, 2004, Belykh et al., 2004) or pinning control (Liu et al., 2008, Xia and Cao, 2009, Porfiri and Fiorilli, 2009) under time-varying interaction topologies. Note that communication failures can usually be regarded as switching events. Therefore, a pinning control algorithm which is intended to be robust against such failures can be designed by considering the controlled network as a switched system.

Pinning control algorithms have been traditionally designed under the hypothesis of continuous-time communication. In many realistic network systems, however, such hypothesis is not verified. On the other hand, a synchronized sampled communication is hard to obtain. Event-triggered control was introduced to limit the amount of communication instances for feedback systems (Tabuada, 2007). Recently, event-triggered control has been extended to multi-agent systems (Dimarogonas et al., 2012, Seyboth et al., 2013, Garcia and Antsaklis, 2013, Garcia et al., 2014, Liuzza et al., 2016).

In a realistic multi-agent control problem, several challenges are present at the same time: nonlinear dynamics, exogenous reference signals, limited communication capacity, and time-varying interaction topology. In this chapter, a general setup is considered, namely *weighted switching topologies with generic linear interactions* are investigated. A model-based and event-triggered pinning control law is designed, which drives the agent states to an a-priori specified common reference trajectory. We derive a set of sufficient conditions under which the closed-loop system is well posed and the agents achieve exponential convergence to the reference trajectory. Static networks are studied as a special case, for which we also prove that there exists a lower bound for the inter-event times in the sequences of updates of the control signals. Different than most existing works on event-triggered multi-agent control, we envision an implementation of the control algorithm which does not require the agents to exchange state measurements at each update time. Agents exchange state measurements only when they establish their connection. When an agent updates its control signal to a new value, it is required to broadcast its value to its neighbors in the network. In this way, it is possible for neighboring agents to predict the state of each other consistently.

The rest of the chapter is organized as follows. In Section 3.2, we give a mathematical formulation of the pinning control problem under investigation, and we outline the proposed control algorithm to address such problem. In Section 3.3, we discuss a distributed and model-based implementation of the proposed algorithm that aims at reducing the necessary amount of communication among the agents. In Section 3.4, we state our main convergence result, whose proof occupies Sections 3.5 to 3.7. In Section 3.8, we specialize the general results to the case of networks with fixed topologies. In Section 3.9, we present a simulated network of nonlinear systems under the proposed algorithm, and we show that the simulation corroborates the theoretical results. Section 3.10 concludes the chapter with a summary of the results.

3.2 System model and problem statement

Consider a multi-agent system with agents indexed as $\mathcal{V} = \{1, \dots, N\}$. Let each agent have state $x_i(t) \in \mathbb{R}^n$ that evolves according to

$$\begin{cases} \dot{x}_i(t) = f(t, x_i(t)) + u_i(t), \\ x_i(0) = x_{i,0}, \end{cases} \tag{3.1}$$

where $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is a time-varying field, $x_{i,0} \in \mathbb{R}^n$ is an initial condition, and $u_i(t) \in \mathbb{R}^n$ is a control input. We introduce the assumption that f is a globally Lipschitz function of the state, with uniform Lipschitz constant with respect to the time. This assumption is formalized as follows.

Assumption 3.1. *For each $t \geq 0$, the function $f(t, \cdot)$ is globally Lipschitz with Lipschitz constant λ_f. Namely, there exists $\lambda_f > 0$ such that, for each $t \geq 0$ and*

each $x_1, x_2 \in \mathbb{R}^n$*, we have*

$$\|f(t, x_1) - f(t, x_2)\| \leq \lambda_f \|x_1 - x_2\|. \tag{3.2}$$

A reference trajectory $r(t) \in \mathbb{R}^n$ is assigned, whose dynamics is compatible with the dynamics of the agents. Namely, we have

$$\begin{cases} \dot{r}(t) = f(t, r(t)), \\ r(0) = r_0. \end{cases} \tag{3.3}$$

The control objective is that the states of all the agents asymptotically converge to the reference trajectory, and it is formalized as

$$\lim_{t \to \infty} \|r(t) - x_i(t)\| = 0 \ \forall i \in \mathcal{V}\,. \tag{3.4}$$

To reach the control objective, we employ piecewise constant control signals,

$$u_i(t) = u_{i,k} \ \forall t \in [t_{i,k}, t_{i,k+1}), \tag{3.5}$$

with $u_{i,k} \in \mathbb{R}^n$. The sequence of the time instants $\{t_{i,k}\}_{k \in \mathbb{N}_0}$ defines a hybrid time trajectory, and corresponds to the times when the control signal $u_i(t)$ is updated to a new value. We let $t_{i,0} = 0$ for all $i \in \mathcal{V}$, so that $u_{i,0}$ is the initial control input for each agent. The control values are computed as

$$\begin{aligned} u_{i,k} = &\sum_{j=1}^{N} w_{ij}(t_{i,k}^+) C(x_j(t_{i,k}) - x_i(t_{i,k})) \\ &+ p_i(t_{i,k}^+) K(r(t_{i,k}) - x_i(t_{i,k})), \end{aligned} \tag{3.6}$$

where $w_{ij}(t) \in \mathbb{R}$ and $p_i(t) \in \mathbb{R}$ for all $t \geq 0$, and $C, K \in \mathbf{S}_{>0}^n$. Here $\mathbf{S}_{>0}^n$ denotes the set of the n-by-n symmetric and positive definite matrices. The interpretation of (3.6) is that each agent receives feedback from the other agents and from the reference trajectory to align its state with the states of the other agents and with the reference. The matrices C and K can be interpreted as a control protocol that translates a mismatch in the state space into a control action. The scalar $w_{ij}(t)$ is the weight of the feedback from agent j to agent i at time t, while $p_i(t)$ is the weight of the feedback from the reference trajectory to agent i at time t. Hence, the scalars $w_{ij}(t)$ and $p_i(t)$ with $i, j \in \mathcal{V}$ define the topology of the networked multi-agent system at each time instant $t \geq 0$. We make the assumption that the feedback between two agents is symmetric, which is formalized as follows.

Assumption 3.2. *For each* $i, j \in \mathcal{V}$*, we have* $w_{ij}(t) = w_{j,i}(t)$ *for each* $t \geq 0$*.*

We also make the assumption that the signals $w_{ij}(t)$ and $p_i(t)$ are piecewise constant and bounded. This agrees with the interpretation that changes in the values of $w_{ij}(t), p_i(t)$ correspond to changes in the topology of the network, due, for example, to communication failures. This assumption is formalized as follows.

Assumption 3.3. *The signals $w_{ij}(t)$ and $p_i(t)$ that appear in (3.6) are piecewise constant, and they are lower-bounded and upper-bounded. Namely, there exist $\underline{w}_{ij}, \bar{w}_{ij}$ and $\underline{p}_i, \bar{p}_i$ such that $\underline{w}_{ij} \leq w_{ij}(t) \leq \bar{w}_{ij}$ and $\underline{p}_i \leq p_i(t) \leq \bar{p}_i$ for all $t \geq 0$. Moreover, the hybrid time trajectory defined by the instants when a change of value occurs for some $w_{ij}(t)$ or $p_i(t)$ is not Zeno.*

In practice, $w_{ij}(t) \neq 0$ means that agents i and j are connected, and can exchange information, while $p_i(t) \neq 0$ means that agent i is connected to the reference. In most applications, we have that, at each time $t \geq 0$, $w_{ij}(t) \neq 0$ only for a small fraction of the possible pairs of agents, and $p_i(t) \neq 0$ only for a small fractions of the agents. Note that, in order to compute $u_{i,k}$ as by (3.6), agent i only needs to receive the state of agent j if $w_{ij}(t_{i,k}^+) \neq 0$, and similarly, it only needs to receive the value $r(t_{i,k})$ of the reference if $p_i(t_{i,k}^+) \neq 0$. For these reasons, (3.6) can be considered a pinning control law.

In order to completely define our control strategy, we also need to specify a rule for scheduling the control updates $t_{i,k}$ for each agent. To this aim, consider the following signals:

$$\begin{aligned} z_i(t) = &\sum_{j=1}^{N} w_{ij}(t) C(x_j(t) - x_i(t)) \\ &+ p_i(t) K(r(t) - x_i(t)). \end{aligned} \tag{3.7}$$

Note that $z_i(t)$ is similar to the control signals (3.6), but the update time $t_{i,k}$ is substituted with the current time. In other words, $z_i(t)$ would correspond to the control input $u_i(t)$ if this were to be continuously updated. An update for agent i is scheduled for each time instant when the difference between $z_i(t)$ and $u_i(t)$ has overcome an assigned threshold. The threshold is defined by the function

$$\varsigma(t) = \varsigma_0 \, e^{-\lambda_\varsigma t}, \tag{3.8}$$

where ς_0 is a positive constant and $\lambda_\varsigma > 0$ is a positive convergence rate. We refer to $\varsigma(t)$ as the *threshold function*. The threshold function is part of the control design, and it is known by all the agents. An update for agent i is also scheduled for the instants when $w_{ij}(t)$ for some j or $p_i(t)$ has changed its value. The rule for

scheduling the updates can therefore be formalized as follows:

$$\begin{aligned} t_{i,k+1} = \inf\{t \geq t_{i,k} : \\ & w_{ij}(t) \neq w_{ij}(t_{i,k}^+) \text{ for some } j, \text{ or} \\ & p_i(t) \neq p_i(t_{i,k}^+), \text{ or} \\ & \|\tilde{u}_i(t)\| \geq \varsigma(t)\}, \end{aligned} \tag{3.9}$$

where

$$\tilde{u}_i(t) = u_i(t) - z_i(t). \tag{3.10}$$

Our goal is to show that the control algorithm defined by (3.3) and (3.5) to (3.10) makes the closed-loop system well-posed and attains the control objective (3.4). Well-posedness of the closed-loop system means that the sequences $\{t_{i,k}\}_{k\in\mathbb{N}_0}$ of the control updates do not exhibit Zeno behavior.

The topology of the multi-agent system (3.1) can be loosely represented as a time-varying graph $\mathcal{G}(t)$. Each agent in the system corresponds to a node in the graph, while each couple (i, j) such that $w_{ij}(t) \neq 0$ constitutes an edge in the graph, with weight equal to $w_{ij}(t)$. Under Assumption 3.2, such graph is undirected. However, according to the definition given in Section 2.1, the weights in a graph are positive scalars, while here we only need that the weights $w_{ij}(t)$ are lower-bounded and upper-bounded. Nevertheless, interpreting the topology of the multi-agent system (3.1) as a graph allows to relate the convergence properties of the multi-agent system to the structural properties of the graph, as we will discuss in Section 3.8.

3.3 Implementation

In order to implement scheduling rule (3.9), agent i needs to know the value of the signals $w_{ij}(t)$, $p_i(t)$ and $\tilde{u}_i(t)$ at every time instant. The signals $w_{ij}(t)$ and $p_i(t)$ represent the topology of the information sources of agent i, therefore it is reasonable that agent i is aware of the value of these signals at any time instant. On the other hand, to compute $\tilde{u}_i(t)$ as by (3.7) and (3.10), agent i needs to know its own state $x_i(t)$, the reference $r(t)$, and the states $x_j(t)$ of the other agents. However, since all the agents and the reference have the same known dynamics, these signals can be predicted simply by integrating said dynamics. Namely, for the reference trajectory, (3.3) holds for all $t \geq 0$. Therefore, in order to compute $r(t)$ at all $t \in [t_{i,k}, t_{i,k+1})$, agent i only needs to know the initial value $r(t_{i,k})$. Moreover, agent i needs to compute $r(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ only if $p_i(t_{i,k}^+) \neq 0$. In fact, if $p_i(t_{i,k}^+) = 0$, $r(t)$ does not affect $\tilde{u}_i(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ (see (3.5) to (3.7) and (3.10)). Similarly, for the states $x_j(t)$ of the other agents, (3.1) holds for all $t \geq 0$. Therefore, in order to compute $x_j(t)$ for all $t \in [t_{i,k}, t_{i,k+1})$, agent i only needs to know the initial value $x_j(t_{i,k})$ and the values, say u_{j,h_j}, of the control signal that agent j uses within the interval $[t_{i,k}, t_{i,k+1})$. Moreover, agent i needs

to compute $x_j(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ only if $w_{ij}(t_{i,k}^+) \neq 0$. In fact, if $w_{ij}(t_{i,k}^+) = 0$, $x_j(t)$ does not affect $\tilde{u}_i(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ (see (3.5) to (3.7) and (3.10)). This fact implies that when an agent j updates its control input, it has to broadcast the newly computed control input, say u_{j,h_j}, to all the other agents i such that $w_{ij}(t_{j,h_j}^+) \neq 0$. These considerations lead us to propose the following Algorithm 1 as an implementation of the control algorithm (3.5) to (3.10). From Algorithm 1, it is clear that the proposed control algorithm requires inter-agent communication only when one of the agents updates its control input, and not at every time instant.

Algorithm 1 Operations executed by each agent i at a generic time instant $t \geq 0$.

compute $x_i(t)$ by prediction
if $p_i(t_{i,k}) \neq 0$, compute $r(t)$ by prediction
compute $x_j(t)$ by prediction for each j such that $w_{ij}(t_{i,k}) \neq 0$
compute $\tilde{u}_i(t)$ as by (3.10)
compute $\varsigma(t)$ as by (3.8)
if $w_{ij}(t) \neq w_{ij}(t_{i,k})$ for some j **or** $p_i(t) \neq p_i(t_{i,k})$ **or** $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ **then**
  for $j \in \mathcal{V} \setminus \{i\}$ **do**
    if $w_{ij}(t) \neq 0$ & $w_{ij}(t_{i,k}) = 0$ **then**
      acquire $x_j(t)$ from agent j
    end if
  end for
  if $p_i(t) \neq 0$ & $p_i(t_{i,k}) = 0$ **then**
    acquire $r(t)$
  end if
  $k \leftarrow k + 1$
  $t_{i,k} \leftarrow t$
  compute $u_{i,k}$ as by (3.6) and set it as the control input
  broadcast $u_{i,k}$ to each agent j such that $w_{j,i}(t_{i,k}) \neq 0$
end if

3.4 Main result

In order to state our main result, we need to introduce some further notation. Let

$$x(t) = [x_1(t)^\intercal, \ldots, x_N(t)^\intercal]^\intercal, \tag{3.11a}$$

$$F(t, x(t)) = [f(t, x_1(t))^\intercal, \ldots, f(t, x_N(t))^\intercal]^\intercal, \tag{3.11b}$$

$$u(t) = [u_1(t)^\intercal, \ldots, u_N(t)^\intercal]^\intercal, \tag{3.11c}$$

$$x_0 = [x_{1,0}^\intercal, \ldots, x_{N,0}^\intercal]^\intercal. \tag{3.11d}$$

With (3.11a) to (3.11d), the dynamics of the open-loop system (3.1) can be rewritten compactly as

$$\begin{cases} \dot{x}(t) = f(t, x(t)) + u(t), \\ x(0) = x_0. \end{cases} \tag{3.12}$$

Consider now the error signals

$$e_i(t) = r(t) - x_i(t), \tag{3.13}$$

and let

$$\begin{aligned} e(t) &= [e_1(t)^\intercal, \dots, e_N(t)^\intercal]^\intercal \\ &= 1_N \otimes r(t) - x(t). \end{aligned} \tag{3.14}$$

Note that the control objective (3.4) can be rewritten in terms of the error vector $e(t)$ as

$$\lim_{t\to\infty} e(t) = 0_{Nn}. \tag{3.15}$$

Let

$$L(t)_{ij} = \begin{cases} \sum_{j=1}^N w_{ij}(t) & \text{if } i = j, \\ -w_{ij}(t) & \text{otherwise,} \end{cases} \tag{3.16a}$$

$$P(t) = \operatorname{diag}(p_1(t)^\intercal, \dots, p_N(t)^\intercal), \tag{3.16b}$$

$$A(t) = L(t) \otimes C + P(t) \otimes K, \tag{3.16c}$$

$$\lambda(t) = \min \operatorname{eig}(A(t)). \tag{3.16d}$$

Note that, under Assumption 3.2, and with $C, K \in \mathbf{S}^N_{\succ 0}$, the matrix $A(t)$ is symmetric for any $t \geq 0$, and, therefore, its minimum eigenvalue $\lambda(t)$ is well defined.

Remark 3.1. *From* (3.16a)*, we see that $L(t)$ can be loosely interpreted as the Laplacian of the undirected graph $\mathcal{G}(t)$ that represents the topology of the multi-agent system* (3.1)*. Recall that this interpretation is not precise, since we are not requiring that $w_{ij}(t) \geq 0$.*

Our main result can now be formalized as the following theorem.

Theorem 3.1. *Consider the multi-agent system* (3.1)*, under the control algorithm defined by* (3.3) *and* (3.5) *to* (3.10)*. Let Assumptions assupmtions 3.1 to 3.3 hold. If there exist $T > 0$ and $\varphi > \lambda_f + \lambda_\varsigma$ such that, for any $t \geq 0$,*

$$\frac{1}{T} \int_t^{t+T} \lambda(\tau) \, \mathrm{d}\tau \geq \varphi, \tag{3.17}$$

then the closed-loop system is well posed and achieves the control objective (3.4)*. In particular, the error stack vector $e(t)$ defined by* (3.14) *converges to zero exponentially with a convergence rate that is lower-bounded by the convergence rate λ_ς of the threshold function; namely, there exists $\bar{\eta} > 0$ such that*

$$\|e(t)\| \leq \bar{\eta} \exp(-\lambda_\varsigma t) \quad \forall t \geq 0. \tag{3.18}$$

We recall here that $\lambda_f > 0$ is the Lipschitz constant of the agent dynamics f, and $\lambda_\varsigma > 0$ is the convergence rate of the threshold function ς.

Remark 3.2. *Condition* (3.17) *essentially requires that the connectivity between the reference trajectory and the agents in the network, parametrized by the minimum eigenvalue $\lambda(t)$ of $A(t)$, has an average over time that is large enough, compared to the Lipschitz constant of the dynamics of the agents and to the convergence rate of the threshold function. However, condition* (3.17) *does not require $\lambda(t)$ to be large at any specific time instant.*

The proof of Theorem 3.1 is given in the next three sections of the chapter. Namely, in Section 3.5, we prove that the closed-loop system achieves exponential convergence of the error vector $e(t)$, and in Section 3.6, we prove that the closed-loop system is well posed, in the sense that the sequence of the control updates of each agent does not exhibit Zeno behavior. Finally, in Section 3.7, we use the results obtained in the previous two sections two formalize the proof of Theorem 3.1.

3.5 Convergence proof

In order to analyze the convergence properties of the closed-loop system (3.1), (3.3) and (3.5) to (3.10), we write the dynamics of the open-loop system in terms of the error vector $e(t)$. Taking the time derivative of both sides in (3.14), and using (3.3) and (3.12), we can write the open-loop dynamics of the error signals as

$$\begin{cases} \dot{e}(t) = 1_N \otimes f(t, r(t)) - f(t, x(t)) - u(t), \\ e(0) = 1_N \otimes r_0 - x_0, \end{cases} \tag{3.19}$$

where $e_0 = e(0)$. Note that (3.7) can be rewritten in terms of the error signals (3.13) as

$$z_i(t) = \sum_{j=1}^{N} w_{ij}(t) C(e_i(t) - e_j(t)) + p_i(t) K e_i(t). \tag{3.20}$$

Moreover, letting

$$z(t) = [z_1(t)^\intercal, \ldots, z_N(t)^\intercal]^\intercal, \tag{3.21}$$

we can rewrite (3.20) compactly as

$$z(t) = A(t)e(t), \tag{3.22}$$

where $A(t)$ is defined in (3.16c).

Substituting (3.10) and (3.22) into (3.19), we have

$$\dot{e}(t) = 1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t). \tag{3.23}$$

From (3.23), it is clear that convergence of the error vector $e(t)$ can be related to $f(\cdot,\cdot)$ being Lipschitz, to the eigenvalues of $A(t)$, and to the boundedness of $\tilde{u}(t)$. This is formalized in the following lemma.

Lemma 3.1. *If $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \in [0,T]$ for all $i \in \mathcal{V}$, where $T > 0$, then, under Assumption 3.1, we have $\|e(t)\| \leq \eta(t)$ for all $t \in [0,T]$, where $\eta(t)$ satisfies*

$$\begin{cases} \dot{\eta}(t) = (\lambda_f - \lambda(t))\eta(t) + \sqrt{N}\,\varsigma(t), \\ \eta(0) = \eta_0, \end{cases} \tag{3.24}$$

where $\eta_0 = \|e_0\|$ and $\lambda(t)$ is defined by (3.16d)

Proof. Consider the function

$$V(t) = \frac{1}{2} e(t)^\intercal e(t). \tag{3.25}$$

Note that we shall not refer to $V(t)$ as to a candidate Lyapunov function, since we are not going to use any Lyapunov theorem. Taking the time derivative of both sides, and using (3.23), we have

$$\begin{aligned} \dot{V}(t) =& e(t)^\intercal \dot{e}(t) \\ =& e(t)^\intercal (1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t)) \\ =& \sum_{i=1}^{N} e_i(t)^\intercal (f(t, r(t)) - f(t, x_i(t))) \\ & - e(t)^\intercal A(t) e(t) - e(t)^\intercal \tilde{u}(t). \end{aligned} \tag{3.26}$$

The terms on the right-hand side of (3.26) can be bounded as follows. By Assumption 3.1, we have

$$e_i(t)^\intercal (f(t, r(t)) - f(t, x_i(t)) \leq \lambda_f \|e_i(t)\|^2. \tag{3.27}$$

Since $A(t)$ is symmetric, we have

$$-e(t)^\intercal A(t) e(t) \leq -\lambda(t)\|e(t)\|^2, \tag{3.28}$$

where $\lambda(t)$ is the smallest eigenvalue of $A(t)$. Finally, if $t \in [0,T]$, by hypothesis we have $\|\tilde{u}_i(t)\| \leq \varsigma(t)$, implying

$$e(t)^\intercal \tilde{u}(t) \leq \|e(t)\| \sqrt{N}\,\varsigma(t). \tag{3.29}$$

Substituting (3.27) to (3.29) in (3.26), we have

$$\dot{V}(t) \leq (\lambda_f - \lambda(t))\|e(t)\|^2 + \|e(t)\|\sqrt{N}\,\varsigma(t). \tag{3.30}$$

Now note that (3.25) can be written equivalently as $V(t) = 1/2\|e(t)\|^2$, which taking the time derivative of both sides yields $\dot{V}(t) = \|e(t)\|\mathrm{d}\|e(t)\|/\mathrm{d}t$, which, in turn, compared with (3.30) yields

$$\|e(t)\|\frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} \leq (\lambda_f - \lambda(t))\|e(t)\|^2 + \|e(t)\|\sqrt{N}\,\varsigma(t). \tag{3.31}$$

For any t such that $\|e(t)\| \neq 0$, (3.31) reduces to

$$\frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} \leq (\lambda_f - \lambda(t))\|e(t)\| + \sqrt{N}\,\varsigma(t). \tag{3.32}$$

On the other hand, if $e(t) = 0$, we can write, for $t \in [0, T)$,

$$\frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} = \lim_{\delta t \to 0} \frac{\|e(t+\delta t)\| - \|e(t)\|}{\delta t}. \tag{3.33}$$

where $\|e(t)\| = 0$, and

$$e(t+\delta t) = \int_t^{t+\delta t} \dot{e}(\tau)\,\mathrm{d}\tau\,. \tag{3.34}$$

Substituting (3.23) into (3.34), taking norms of both sides, using the triangular inequality and Assumption 3.1, and observing that $\tilde{u}_i(t) \leq \varsigma(t)$ for all $t \in [0, T)$, we have

$$\|e(t+\delta t)\| \leq \int_t^{t+\delta t} ((\lambda_f - \lambda(\tau))\|e(\tau)\| + \sqrt{N}\,\varsigma(\tau))\,\mathrm{d}\tau \tag{3.35}$$

Dividing both sides by δt, taking the limit for $\delta t \to 0$, using the mean value theorem, and comparing with (3.33), we have again (3.32), which therefore applies for all $t \in [0, T)$. From (3.32), and using Gronwall's lemma (Khalil, 2002), we have (3.24). □

Under the hypotheses of Lemma 3.1, we have $e(t) \to 0_{Nn}$ if $\eta(t) \to 0$. Therefore, we only need to prove that the closed-loop system is well posed and achieves $\eta(t) \to 0$ to prove Theorem 3.1. The following lemma gives a sufficient condition for convergence of $\eta(t)$.

Lemma 3.2. *Let $\eta(t)$ be defined by* (3.24)*, and let Assumption 3.3 hold. If* (3.17) *holds, then there exists $\bar{\eta} > 0$ such that*

$$\eta(t) \leq \bar{\eta}\exp(-\lambda_\varsigma t), \tag{3.36}$$

In particular, $\eta(t) \to 0$.

Proof. Condition (3.17) can be rewritten as

$$\int_t^{t+T} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \leq -T\phi \quad \forall t \geq 0, \tag{3.37}$$

where $\phi = \varphi - \lambda_f > \lambda_\varsigma$. For any $t' > t$ we can write $t' = t + \nu T + \delta t$, with $\nu \in \mathbb{N}_0$ and $0 \leq \delta t < T$. Therefore, using (3.37) repeatedly, we have

$$\begin{aligned}\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau &\leq -\nu T\phi + \int_{t+\nu T}^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \\ &= -\phi(t' - t) + \int_{t+\nu T}^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau\,.\end{aligned} \tag{3.38}$$

Under Assumption 3.3, $\lambda(\tau)$ is bounded, and therefore, the last integral in (3.38) is bounded. Hence, we can rewrite (3.38) as

$$\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \leq -\phi(t' - t) + \xi \tag{3.39}$$

for some $\xi > 0$. The Laplace solution of (3.24) in $[0, t)$ reads, using also (3.8),

$$\begin{aligned}\eta(t) = \Phi(t, 0)\eta_0 \\ + \sqrt{N}\,\varsigma_0 \int_0^t \Phi(t, \tau) \exp(-\lambda_\varsigma \tau)\,\mathrm{d}\tau\,,\end{aligned} \tag{3.40}$$

where

$$\Phi(t', t) = \exp\left(\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau\right). \tag{3.41}$$

Using (3.39) in (3.41), we have

$$\Phi(t', t) \leq \exp(-\phi(t' - t)) \exp(\xi) \tag{3.42}$$

Using (3.42) in (3.40), we have

$$\begin{aligned}\eta(t) \leq \exp(-\phi t) \exp(\xi)\eta_0 \\ + \sqrt{N}\,\varsigma_0 \exp(\xi) \int_0^t \exp(-\phi(t - \tau)) \exp(-\lambda_\varsigma \tau)\,\mathrm{d}\tau\,,\end{aligned} \tag{3.43}$$

Since $\phi > \lambda_\varsigma$, we have

$$\int_0^t \exp((\phi - \lambda_\varsigma)\tau)\,\mathrm{d}\tau = \frac{\exp((\phi - \lambda_\varsigma)t) - 1}{\phi - \lambda_\varsigma}, \tag{3.44}$$

which substituted into (3.43) yields

$$\eta(t) \leq k' \left(\eta_0 + \sqrt{N}\,\varsigma_0 \exp(\xi) \frac{\exp((\phi - \lambda_\varsigma)t) - 1}{\phi - \lambda_\varsigma}\right) \mathrm{e}^{-\phi t}\,. \tag{3.45}$$

Using again $\phi > \lambda_\varsigma$, we can further bound (3.45) as (3.36), with

$$\bar{\eta} = k' \left(\eta_0 + \frac{\sqrt{N}\,\varsigma_0}{\phi - \lambda_\varsigma}\right). \tag{3.46}$$

□

Thanks to Lemmas 3.1 and 3.2, proving that the proposed control algorithm attains the objective (3.4) reduces to proving that the algorithm makes the closed-loop system well-posed, and attains $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ as well as (3.39). This will be the subject of the following Section 3.6.

3.6 Well-posedness proof

Well-posedness of the closed-loop systems means that the hybrid time trajectory generated by the control updates $\{t_{i,k}\}$ of each agent i do not exhibit Zeno behavior. In order to study this property, first observe that $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ is automatically guaranteed by the scheduling rule (3.9). In fact, for each $k \in \mathbb{N}_0$ and each $i \in \mathcal{V}$, we have from (3.6) and (3.7) that $z_i(t_{i,k}) = u_{i,k}$, which by (3.10) implies

$$\tilde{u}_i(t_{i,k}) = 0. \tag{3.47}$$

Since a new update $t_{i,k+1}$ is triggered whenever $\|\tilde{u}_i(t)\| \geq \varsigma(t)$, it is not possible that $\|\tilde{u}_i(t)\| > \varsigma(t)$ for some $t \geq 0$, $i \in \mathcal{V}$. Well-posedness of the closed-loop system is formalized in the following lemma.

Lemma 3.3. *Consider the multi-agent system* (3.1)*, under the control algorithm defined by* (3.3) *and* (3.5) *to* (3.10)*. Under Assumptions assupmtions 3.1 to 3.3, and* (3.39)*, the closed-loop system is well posed. In particular, the sequences* $\{t_{i,k}\}_{k\in\mathbb{N}}$ *of the control updates for* $i \in \mathcal{V}$ *do not exhibit Zeno behavior.*

Proof. Let us consider a generic agent $i \in \mathcal{V}$ within the generic time interval $[t_{i,k}, t_{i,k+1})$. By (3.6), taking the time derivative of both sides in (3.10), we have

$$\dot{\tilde{u}}_i(t) = -\dot{z}_i(t). \tag{3.48}$$

Note now that, from (3.22), we have $z_i(t) = A(t)_{n(i-1)+1:in,:}e(t)$, and moreover,

$$A(t)_{n(i-1)+1:in,:} = A(t_{i,k})_{n(i-1)+1:in,:}, \tag{3.49}$$

because $w_{ij}(t)$ for all $j \in \mathcal{V}\setminus\{i\}$ and $p_i(t)$ are constant for $t \in [t_{i,k}, t_{i,k+1})$. Therefore, $\dot{z}_i(t) = A(t_{i,k})_{n(i-1)+1:in,:}\dot{e}(t)$, which substituted in (3.48) yields

$$\dot{\tilde{u}}_i(t) = -A(t_{i,k})_{n(i-1)+1:in,:}\dot{e}(t). \tag{3.50}$$

Substituting (3.23) into (3.50), we have

$$\begin{aligned}\dot{\tilde{u}}_i(t) = &- A(t_{i,k})_{n(i-1)+1:in,:}(\\ &1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t)).\end{aligned} \tag{3.51}$$

Note now that, by Assumption 3.3, we have $\|A(t)\| \leq \alpha$ for some $\alpha > 0$, since all the entries of $A(t)$ are bounded. Therefore, taking norms of both sides in (3.51),

using the triangular inequality, $\|\tilde{u}_j(t)\| \leq \varsigma(t)$ for all $j \in \mathcal{V}$, and Assumption 3.1, we have

$$\|\dot{\tilde{u}}_i(t)\| \leq \alpha((\lambda_f + \alpha)e(t) + \sqrt{N}\,\varsigma(t)). \tag{3.52}$$

Since Lemmas 3.1 and 3.2 apply, we have $\|e(t)\| \leq \bar{\eta}\,\mathrm{e}^{-\lambda_\varsigma t}$, which compared with (3.52), together with (3.8), yields

$$\|\dot{\tilde{u}}_i(t)\| \leq \alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)\,\mathrm{e}^{-\lambda_\varsigma t}\,. \tag{3.53}$$

Since $\tilde{u}_i(t_{i,k}) = 0$, we have $\tilde{u}_i(t) = \int_{t_{i,k}}^{t} \dot{\tilde{u}}_i(\tau)\,\mathrm{d}\tau$, which by taking norms of both sides, and using the triangular inequality yields

$$\|\tilde{u}_i(t)\| \leq \int_{t_{i,k}}^{t} \|\dot{\tilde{u}}_i(\tau)\|\,\mathrm{d}\tau\,. \tag{3.54}$$

Substituting (3.53) into (3.54), we have

$$\|\tilde{u}_i(t)\| \leq \alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)\frac{1 - \mathrm{e}^{-\lambda_\varsigma(t - t_{i,k})}}{\lambda_\varsigma}\,\mathrm{e}^{-\lambda_\varsigma t_{i,k}}\,. \tag{3.55}$$

Note now that (3.8) can be written as

$$\varsigma(t) = \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma t_{i,k}}\,\mathrm{e}^{-\lambda_\varsigma(t - t_{i,k})}\,. \tag{3.56}$$

Comparing (3.55) and (3.56), it is clear that a necessary condition for having $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ is

$$\alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)\frac{1 - \mathrm{e}^{-\lambda_\varsigma(t - t_{i,k})}}{\lambda_\varsigma} \geq \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma(t - t_{i,k})}, \tag{3.57}$$

which is attained if and only if $t - t_{i,k} \geq \delta > 0$, where δ satisfies

$$\alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)\frac{1 - \mathrm{e}^{-\lambda_\varsigma \delta}}{\lambda_\varsigma} = \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma \delta}, \tag{3.58}$$

or equivalently

$$\delta = \ln\left(\frac{\lambda_\varsigma + \alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)}{\alpha((\lambda_f + \alpha)\bar{\eta} + \sqrt{N}\,\varsigma_0)}\right) > 0. \tag{3.59}$$

From (3.59), it is clear that two consecutive control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ are separated by a positively lower-bounded inter-event time. Then, using Proposition 2.7, we can conclude that the hybrid time trajectory generated by the control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ is not Zeno. From Assumption 3.3, we know that the sequence of the control updates due to $w_{ij}(t) \neq w_{ij}(t_{i,k}^+)$ for some $j \in \mathcal{V}$ or $p_i(t) \neq p_i(t_{i,k}^+)$ is not Zeno either. From the scheduling law (3.9), we know that the sequence $\{t_{i,k}\}_{k \in \mathbb{N}_0}$ of the control updates of agent i is the union of the sequence of

the control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ and the sequence of the control updates due to $w_{ij}(t) \neq w_{ij}(t_{i,k}^+)$ for some $j \in \mathcal{V}$ or $p_i(t) \neq p_i(t_{i,k}^+)$. Since the union of a finite number of nonzeno sequences is not Zeno (see Proposition 2.7), we can conclude that the sequence $\{t_{i,k}\}_{k\in\mathbb{N}_0}$ is not Zeno. Finally, since this reasoning is valid for all agents $i \in \mathcal{V}$, we can conclude that the closed-loop system is well posed. □

Remark 3.3. *Lemma 3.3 does not guarantee that two consecutive control updates $t_{i,k}$ and $t_{i,k+1}$ are separated by a finite inter-event time. In fact, two events of the type $w_{ij}(t) \neq w_{ij}(t_{i,k}^+)$ or $p_i(t) \neq p_i(t_{i,k}^+)$ may occur infinitely close to each other, and also infinitely close to the events of the type $\|\tilde{u}_i(t)\| \geq \varsigma(t)$. However, a finite inter-event time is guaranteed in the particular case that the network topology is constant (that is, that the scalars $w_{ij}(t)$ and $p_i(t)$ are constant for all $i, j \in \mathcal{V}$). This case is further discussed in the following Section 3.8, which examines the particular case of networks with fixed topology.*

3.7 Proof of the main result

Using Lemma 3.3, we have that, under the scheduling rule (3.9), $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. Hence, using Lemmas 3.1 and 3.2, and taking $t \to \infty$, we can conclude that $\|e(t)\| \leq \eta(t) \leq \bar{\eta}\exp(-\lambda_\varsigma t) \to 0$. Therefore, the control objective (3.4) is achieved, and, in particular, $e(t)$ converges to zero exponentially.

3.8 Fixed network topologies

In this section, we consider the particular case that the topology of the networked multi-agent system (3.1) is constant, i.e., that the scalars $w_{ij}(t) \equiv w_{ij}$ and $p_i(t) \equiv p_i$ are constant for all $i, j \in \mathcal{V}$. In this case, condition (3.39) in Lemma 3.2 is equivalent to

$$\lambda > \lambda_f + \lambda_\varsigma, \tag{3.60}$$

where λ is the minimum eigenvalue of the (now constant) matrix A defined by (3.16c). Since the eigenvalues of A scale linearly with the matrices C and K (when C and K are scaled simultaneously), (3.60) can be satisfied by making A positive definite, and then by scaling it opportunely by scaling the matrices C and K. The following lemma relates the positive definiteness of A to the positive definiteness of $L+P$. Here $\mathbf{S}_{\geq 0}^N$ denotes the set of the N-by-N symmetric and positive semidefinite matrices.

Lemma 3.4. *Let $A, B \in \mathbf{S}_{\geq 0}^N$ and $C, D \in \mathbf{S}_{>0}^n$. Then $A \otimes C + B \otimes D \in \mathbf{S}_{\geq 0}^{Nn}$, and $A \otimes C + B \otimes D \in \mathbf{S}_{>0}^{Nn}$ if and only if $A + B \in \mathbf{S}_{>0}^N$.*

Proof. Since $A, B \in \mathbf{S}_{\geq 0}^N$, if $A + B \in \mathbf{S}_{>0}^N$, then either $A \in \mathbf{S}_{>0}^N$ or $B \in \mathbf{S}_{>0}^N$ (possibly both). Consequently, $A \otimes C, B \otimes D \in \mathbf{S}_{+}^{Nn}$, and either $A \otimes C \in \mathbf{S}_{>0}^{Nn}$ or

$B \otimes D \in \mathbf{S}^{Nn}_{>0}$. Hence, $A \otimes C + B \otimes D \in \mathbf{S}^{Nn}_{>0}$. Similarly, since $A \otimes C, B \otimes D \in \mathbf{S}^{Nn}_{+}$, if $A \otimes C + B \otimes D \in \mathbf{S}^{Nn}_{>0}$ then either $A \otimes C \in \mathbf{S}^{Nn}_{>0}$ or $B \otimes D \in \mathbf{S}^{Nn}_{>0}$ (possibly both). Therefore, either $A \in \mathbf{S}^{N}_{>0}$ or $B \in \mathbf{S}^{N}_{>0}$, which implies $A + B \in \mathbf{S}^{N}_{>0}$. □

By Lemma 3.4, A can be made positive definite by making L and P positive semidefinite and $L + P$ positive definite. A sufficient condition for making L positive semidefinite is that $w_{ij} \geq 0$ for all $i, j \in \mathcal{V}$. In fact, for $w_{ij} \geq 0$ for all $i, j \in \mathcal{V}$, L is the Laplacian matrix of a graph, which is positive semidefinite. On the other hand, a sufficient condition for making P positive semidefinite is that $p_i \geq 0$ for all $i \in \mathcal{V}$. The hypotheses $w_{ij} \geq 0$ and $p_i \geq 0$ correspond to the feedback between any two agents and from the reference to each agent being either positive ($w_{ij} > 0$ or $p_i > 0$) or absent ($w_{ij} = 0$ or $p_i = 0$). Such hypotheses are verified in most realistic settings, while negative feedback occurs in applications featuring adversarial connections between two or more agents. Given that L and P are positive semidefinite, a sufficient condition for making $L + P$ positive definite is given by the following lemma.

Lemma 3.5. *Let $w_{ij} = w_{j,i} \geq 0$ and $p_i \geq 0$ for all $i, j \in \mathcal{V}$. Let $\mathcal{G}$ be the undirected graph defined by the nodes $\mathcal{V}$ and the edges $\mathcal{E} = \{(j, i) \in \mathcal{V} \times \mathcal{V} : \ w_{ij} > 0\}$, where $w_{ij} > 0$ is also the weight of the edge (j, i). The nodes $i \in \mathcal{V}$ such that $p_i > 0$ are said to be pinned. Let L and P be defined by* (3.16a) *and* (3.16b)*, so that L is the Laplacian matrix of $\mathcal{G}$. Then $L + P$ is positive definite if and only if there is at least one pinned node in each connected component of $\mathcal{G}$.*

Proof. Without loss of generality, suppose that the nodes of $\mathcal{G}$ are ordered in such a way that the first n_1 nodes are in a first component, the following n_2 nodes are a the second component, etc. Then L and P are block-diagonal, with each block corresponding to one of the components. We divide the proof in two parts.

Part 1 (Necessity: if $L + P$ is positive definite, then there is at least one pinned node in each component of $\mathcal{G}$.)**.** Suppose that $L+P$ is positive definite, and suppose by contradiction that there is a connected component that does not contain any pinned node. Without loss of generality, suppose that this component is the first component. Then, consider the vector $v = [1^\intercal_{n_1}, 0^\intercal_{n_2}, \ldots, 0^\intercal_{n_{N_c}}]^\intercal$, where N_c is the number of components in $\mathcal{G}$. Then we have

$$v^\intercal (L + P) v = 1^\intercal_{n_1} (L_1 + P_1) 1_{n_1}, \tag{3.61}$$

where L_1 and P_1 are the blocks of L and P respectively corresponding to the first component. Since the first component does not contain any pinned node, we have $P_1 = 0_{n_1 \times n_1}$, and since L_1 is a Laplacian matrix, we have $L_1 1_{n_c} = 0$. Substituting the last two equations into (3.61), we have $v^\intercal (L + P) v = 0$, with $v \neq 0_N$, which is a contradiction.

Part 2 (Sufficiency: if there is at least one pinned node in each component of $\mathcal{G}$, then $L+P$ is positive definite.)**.** Viceversa, suppose that there is at least one pinned node in each connected component of $\mathcal{G}$, and suppose by contradiction that $L+P$ is not positive definite, i.e., that there exists a nonzero $v \in \mathbb{R}^N$ such that $v^\intercal(L+P)v = 0$. Since both L and P are block-diagonal with each block of L being the same size of the corresponding block of P, the previous equation implies that

$$\sum_{i=1}^{N_c} v_{(i)}^\intercal (L_i + P_i) v_{(i)} = 0 \tag{3.62}$$

for each $i \in \{1, \dots, N_c\}$, where N_c is the number of components in $\mathcal{G}$, $v_{(i)} \in \mathbb{R}^{n_i}$ is the restriction of v to the entries corresponding to the i-th component, and L_i and P_i are the i-th diagonal block of L and P respectively. Note that L_i and P_i are both positive semidefinite, since L_i is a Laplacian matrix and P_i is diagonal with nonnegative diagonal entries. Therefore, (3.62) implies $v_{(i)}^\intercal (L_i + P_i) v_{(i)}$ for all $i \in \{1, \dots, N_c\}$, which, in turn, implies

$$v_{(i)}^\intercal L_i v_{(i)} = 0, \tag{3.63a}$$

$$v_{(i)}^\intercal P_i v_{(i)} = 0. \tag{3.63b}$$

for all $i \in \{1, \dots, N_c\}$. Since L_i is a Laplacian matrix, (3.63a) implies $v_{(i)} = \alpha 1_{n_i}$ for some $\alpha \in \mathbb{R}$, which substituted in (3.63b) gives $\alpha \operatorname{tr}(P_i) = 0$, where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. Since the entries of P_i are nonnegative, and since there is at least one pinned node in each component, we have $\operatorname{tr}(P_i) > 0$, which means that $\alpha = 0$. Hence, $v_{(i)} = \alpha 1_{n_i} = 0_{n_i}$. Since this reasoning applies to all components $i \in \{1, \dots, N_c\}$, we conclude that $v = 0_N$, which is a contradiction. □

The intuition behind Lemma 3.5 is very simple: for the multi-agent system to converge to the reference trajectory, each agent needs to have access to information originating from the reference trajectory, either by directly receiving feedback from the reference trajectory, or by receiving feedback from other agents that are influenced by the reference trajectory. As a particular case of Lemma 3.5, we have the following corollary.

Corollary 3.1. *In Lemma 3.5, if $\mathcal{G}$ is connected, then $L+P$ is positive definite if and only if there is at least one pinned node.*

Remark 3.4. *If in Lemma 3.5 we relax the assumption that the scalars w_{ij} and p_i are nonnegative, we can still write Part 1 of the proof to show that a necessary (but, in this case, not sufficient) condition for $L+P$ to be positive definite is that there is at least one node i such that $p_i \neq 0$ in each component of the graph.*

When the topology of the network system is fixed, the proposed control algorithm comes with a guaranteed constant lower bound for the inter-event times between

two consecutive control updates $t_{i,k}$ and $t_{i,k+1}$ of the same agent. This property is immediately deduced by the proof of Lemma 3.3, observing that in this case the control updates can only be triggered by events of the kind $\|\tilde{u}_i(t)\| \geq \varsigma(t)$. In particular, the lower bound for the inter-event times is given by $\delta > 0$ defined by (3.59).

3.9 Numerical simulations

To illustrate the effectiveness of the proposed control algorithm, we apply it to a simulated network of $N = 5$ identical Chua oscillators (Matsumoto, 1984). The individual dynamics of each oscillator is described by

$$f(x) = \begin{bmatrix} a(x_2 - x_1 - \gamma(x_1)) \\ x_1 - x_2 + x_3 \\ -bx_2 \end{bmatrix}, \tag{3.64}$$

with $x = [x_1, x_2, x_3]^\intercal \in \mathbb{R}^3$, where $a, b, m_0, m_1 \in \mathbb{R}$ and $\gamma : \mathbb{R} \to \mathbb{R}$, namely,

$$\gamma(y) = m_1 y + \frac{1}{2}(m_0 - m_1)(|y + 1| - |y - 1|). \tag{3.65}$$

Choosing $a = b = 0.9$, $m_0 = -1.34$, and $m_1 = -0.73$, the oscillators are globally Lipschitz with Lipschitz constant $\lambda_f = 3.54$ (Liuzza et al., 2013). Let $C = 5I_3$ and $K = 30I_3$. All the agents are connected to each other with $w_{ij}(t) \equiv 1$. Our simulation is set on the time interval $[0, 30]$. At the beginning of the simulation, we set $p_1(0) = p_2(0) = 1$, while $p_i(0) = 0$ for $i \in \{3, 4, 5\}$, which yields $\lambda(0) = 6.14$. At $t = 0.75$, we set $p_1(t) = 0$, so that $\lambda(t) = 2.88$. At $t = 0.90s$, we set $p_2(t) = 0$, which yields $\lambda(t) = 0$. At $t = 1.0$ the original values of the signals $p_i(t)$ are restored, and the cycle is repeated for every time unit. With this setting, it is clear that Assumptions assupmtions 3.2 and 3.3 are satisfied. Also, we can verify that condition (3.39) is satisfied with $\gamma = 1.5$ and $k = 0$. Figure 3.1 provides an illustration of the graph underlying the simulated network.

For the threshold function (3.8), we choose $\varsigma_0 = 1$ and $\lambda_\varsigma = 0.3$, so that the hypotheses of Lemma 3.2 are satisfied. For each agent, the initial conditions are chosen within the domain of attraction of a Chua oscillator with the chosen parameters a, b, m_0, m_1.

Some results of the simulation are illustrated in Figures figs. 3.2 to 3.4 and Table table 3.1. Namely, Figure fig. 3.2 illustrates the evolution of the second state variable $x_i^{(2)}$ for each agent $i \in \{1, \ldots, 5\}$ over the whole simulation. This result confirms that the control objective (3.4) is achieved, i.e., the state of each agent converges asymptotically to the reference trajectory. As a term of comparison, Figure fig. 3.3 illustrates the evolution of the same state variables, with the same initial conditions, when no control input is applied ($u_i(t) \equiv 0_3$). Figure fig. 3.4 illustrates the

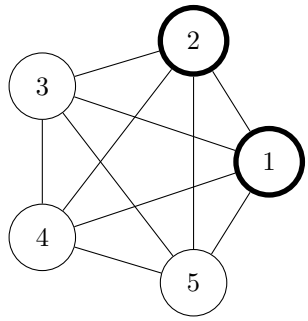

Figure 3.1: An illustration of the graph underlying the simulated network. Each node in the graph represents a Chua oscillator. The nodes with thicker contour represent the oscillators that receive feedback from the reference trajectory during part of the simulation.

Table 3.1: Average inter-event time for each Chua oscillator over the time interval $[0.0, 30.0]$, with the proposed control algorithm applied.

NODE	AVERAGE $t^i_{i,k+1} - t_{i,k}$
1	0.061
2	0.054
3	0.115
4	0.123
5	0.115

time instants when each agent updates its control input within the interval $[0, 1]$, and Table table 3.1 illustrates the average inter-event time for each agent over the whole simulation. These results confirm that the closed-loop system is not Zeno.

3.10 Summary

In this chapter, we have proposed an algorithm for event-triggered pinning synchronization of complex networks of nonlinear agents with switching topologies. We have found sufficient conditions under which Zeno behavior of the closed-loop system is excluded, and the synchronization objective is achieved. We have also shown that the error stack vector that represents the global distance of the system from the synchronization vanishes exponentially. A constant lower bound on the inter-event times has been provided for the case of networks with time-invariant topologies. Numerical simulations have been presented to validate the theoretical results.

Some viable extensions of this work include the application of the proposed algorithm to more general classes of networks, such as networks with asymmetric couplings among the agents, and networks where errors in the communication can

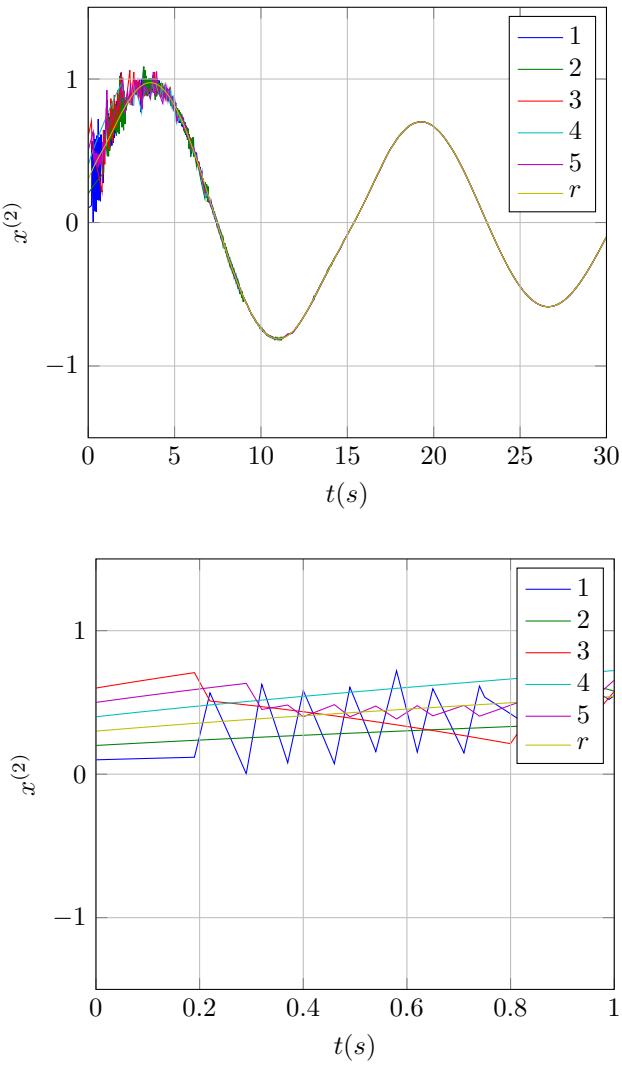

Figure 3.2: Evolution of the state variable $x_i^{(2)}(t)$ for each Chua oscillator $i \in \{1, \ldots, 5\}$ and of $r^{(2)}(t)$ for the reference trajectory, over $t \in [0, 30]$ (above) and $t \in [0, 1]$ (below), with the proposed control algorithm applied. As predicted by Theorem 3.1, the state of each agent converges to the reference trajectory.

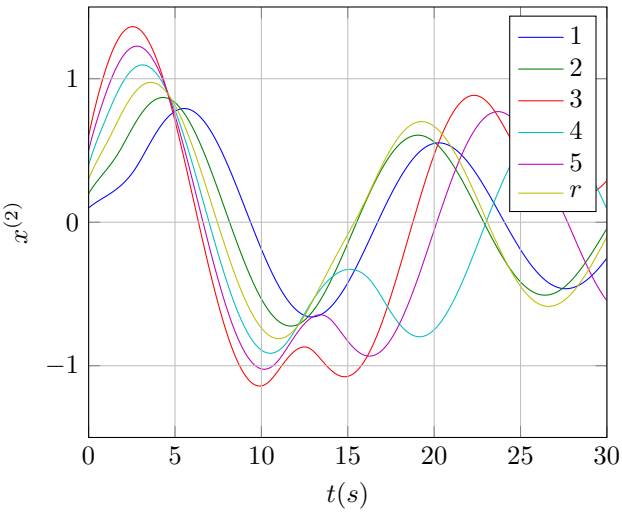

Figure 3.3: Evolution of the state variable $x_i^{(2)}$ for each Chua oscillator $i \in \{1, \ldots, 5\}$ and of $r^{(2)}$ for the reference trajectory, over $t \in [0, 30]$, with no control input applied ($u_i(t) \equiv 0_3$). The states of the agents do not converge to the reference trajectory.

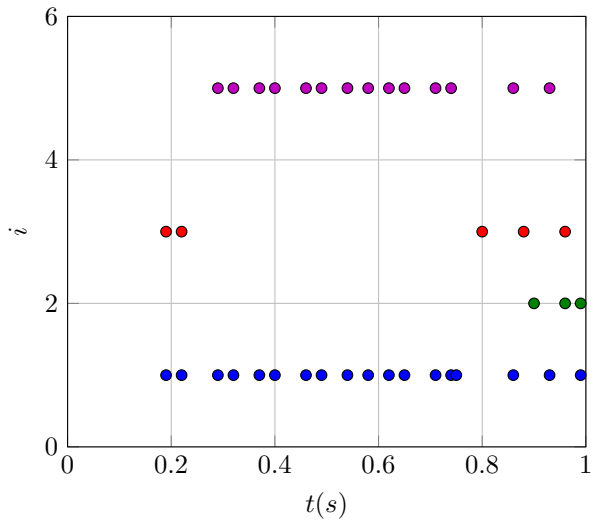

Figure 3.4: Control updates for each Chua oscillator $i \in \{1, \ldots, 5\}$ over the time interval $[0, 1]$, with the proposed control algorithm applied.

occur, such as delays and packet drops.

Chapter 4

Cloud-supported formation control

Or la squilla dà segno
Della festa che viene;
Ed a quel suon diresti
Che il cor si riconforta.

G. Leopardi,
Il sabato del villaggio*, vv. 20–23.*

In this chapter, we encounter cloud-supported coordination for the first time in the thesis. We consider a similar coordination objective as in Chapter 3, but we propose a different, novel communication scheme where all information passes through a shared repository hosted on a cloud.

The rest of this chapter is organized as follows. In Section 4.1, we review the existing related work and highlight the novel contributions offered in the chapter. In Sections 4.2 and 4.3, we present the proposed system model, problem statement, and control strategy. In Section 4.4, we state our main result, whose proof is given in Sections 4.5 to 4.7. Section 4.8 corroborates the theoretical results by presenting two numerical simulations of the proposed control strategy. Section 4.9 concludes the chapter with a summary of the results.

4.1 Introduction

In some realistic applications of multi-agent systems, inter-agent communication is completely or almost completely interdicted. This challenge arises, for example, in the coordination of a fleet of autonomous underwater vehicles (AUVs) (Teixeira

et al., 2011). Because of their severely limited communication, sensing, and localization capabilities, underwater vehicles are virtually isolated systems. Underwater communication and positioning may be implemented by means of battery-powered acoustic modems, but such devices are expensive, limited in range, and power-hungry. Inertial sensor for underwater positioning are prohibitively expensive in most practical scenarios. Moreover, GPS is not available underwater, and a vehicle needs to surface whenever it needs to get a position fix (Paull et al., 2014).

When such limitations arise, coordination strategies that rely on continuous information exchanges among the agents cannot be implemented. To address this challenge, the idea of *triggered control* has been tailored to multi-agent systems. Triggered control is a control technique where the control input is only updated at discrete time instants, when some condition is satisfied (Heemels et al., 2012). Triggered control was introduced in relation to networked control systems, to limit the amount of communication within the different parts of the system (i.e., the plant, the sensors, the actuators). In the context of multi-agent systems, triggered control is used to limit the communication among different agents. Various flavors of triggered control have been applied to multi-agent systems: with *event-triggered control*, inter-agent communication is triggered when a given state condition is satisfied; with *self-triggered control*, the agents schedule when to exchange data in a recursive fashion, so that there is no need to monitor a condition between consecutive communication instances. However, even these triggered control schemes require that the agents exchange information, and, therefore, are not well-suited for those scenarios where direct inter-agent communication is interdicted.

In the control architecture described in this chapter, inter-agent communication is substituted by the use of a shared information repository hosted on a cloud. Each agent schedules its own accesses independently, and does not need to be alert for information broadcast by other agents. When an agent accesses the repository, it uploads some data packets, and downloads other packets that were previously deposited by other agents. Therefore, each agent receives outdated information about the state of the other agents. The control law and the rule for scheduling the cloud accesses are designed to guarantee that the closed-loop system is well-posed and achieves a given coordination objective, even if each agent receives only outdated information about the state of the other agents. Our motivating example is a waypoint generation algorithm for AUVs, which, as described above, represent a challenging application, since underwater communication is interdicted.

The use of a shared information repository in multi-agent control tasks is subject to recent, but growing, research attention. For example, Patel et al. (2016), employ asynchronous communication with a base station to address a multi-agent coverage control problem; Hale and Egerstedt (2015) present a cloud-supported approach to multi-agent optimization; Bowman et al. (2016) present a cloud-supported control algorithm to consensus of first-order integrators.

In this chapter, we introduce cloud support for multi-agent systems with second-order dynamics. We consider both persistent and vanishing disturbances, which lead to approximate and asymptotic coordination, respectively. In both cases, we show that the closed-loop system is well posed (meaning that the sequence of the cloud accesses does not exhibit Zeno behavior (Johansson et al., 1999, Heymann et al., 2005)), and achieves the control objective. Our analysis extends the use of the edge Laplacian (Zelazo and Mesbahi, 2011, Zeng et al., 2015) to second-order multi-agent systems on directed graphs, which allows us to consider control tasks with asymmetric information flow among the agents, such as leader-following tasks.

4.2 System model and problem statement

In this section, we describe our control architecture, by defining the dynamical model of the agents, the communication between the agents and the cloud repository, the control inputs to the agents, and the control objective.

Agent Model

We consider a set $\mathcal{V} = \{1, \dots, N\}$ of N agents. The position and velocity of agent i are denoted respectively as $p_i, v_i \in \mathbb{R}^n$. For the sake of generality, we consider the generic agent dimension $n \in \mathbb{N}$. However, in our motivating example of planar AUV coordination, we have $n = 2$. The agents move according to the following equations:

$$\dot{p}_i(t) = v_i(t), \tag{4.1a}$$
$$\dot{v}_i(t) = u_i(t) + d_i(t), \tag{4.1b}$$

for $i = 1, \dots, N$, where $u_i(t)$ is a control input and $d_i(t)$ is a disturbance signal. We denote $p(t) = [p_1(t)^\intercal, \dots, p_N(t)^\intercal]^\intercal$, and similarly for $v(t)$, $u(t)$ and $d(t)$, so that (4.1) can be rewritten as

$$\dot{p}(t) = v(t), \tag{4.2a}$$
$$\dot{v}(t) = u(t) + d(t). \tag{4.2b}$$

The control objective is for all the agents to converge to the same positions and velocities within a given tolerance. Such objective is formalized mathematically later in this section.

Assumption 4.1. *The disturbance signals $d_i(t)$ in (4.1b) satisfy $\|d_i(t)\| \leq \delta(t)$, where*

$$\delta(t) = (\delta_0 - \delta_\infty)e^{-\lambda_\delta t} + \delta_\infty, \tag{4.3}$$

for some $0 \leq \delta_\infty \leq \delta_0$ and $\lambda_\delta > 0$.

Assumption 4.1 allows to consider both scenarios where only a constant upper bound is known ($\delta_0 = \delta_\infty$) and scenarios where the disturbances slowly vanish ($\delta_\infty = 0$), which makes it possible to reach asymptotic convergence.

Cloud Repository

The agents cannot exchange any information directly, but can only upload and download information on a shared repository hosted on a cloud. The cloud is accessed intermittently by each agent and asynchronously by different agents. A motivating application is a group of AUVs that can only communicate with a remote repository when they are on the water surface, while they are isolated when they are underwater. When an agent accesses the cloud, it also has access to a sampled measurement of its own position and velocity. In our motivating application, this corresponds to the underwater vehicles being able to access GPS while they are on the water surface. The time instants when agent i accesses the cloud are denoted $t_{i,k}$, $k \in \mathbb{N}$, and by convention $t_{i,0} = 0$ for all the agents. For convenience, we denote $l_i(t)$ the index of the most recent access time of agent i before time t; that is,

$$l_i(t) = \max\{k \in \mathbb{N} : \ t_{i,k} \leq t\}. \tag{4.4}$$

The position and velocity measurement obtained by agent i upon the time instant $t_{i,k}$ are denoted $p_{i,k}$ and $v_{i,k}$ respectively. The control signals $u_i(t)$ are held constant between two consecutive cloud accesses:

$$u_i(t) = u_{i,k} \quad \forall t \in [t_{i,k}, t_{i,k+1}). \tag{4.5}$$

When an agent accesses the cloud, it uploads data that other agents may download later, when they, in turn, access the cloud. Namely, when agent i accesses the cloud at time $t_{i,k}$, it uploads a packet containing the following information: the current time $t_{i,k}$, the position and velocity measurements $p_{i,k}$ and $v_{i,k}$, the value $u_{i,k}$ of the control input that is going to be applied in the time interval $[t_{i,k}, t_{i,k+1})$, and the time $t_{i,k+1}$ of the next access. The data packet may overwrite the packet that was uploaded on the previous access, avoiding that the amount of data contained in the cloud grow over time, since, at each time instant, the cloud only contains the data that each agent has uploaded upon its latest access. The data contained in the cloud at a generic time instant is represented in Table table 4.1.

To achieve inter-agent coordination, each agent needs to download information about a subset of the other agents. For each agent i, we denote as $\mathcal{N}_i \subseteq \mathcal{V} \setminus \{i\}$ the subset of the agents whose information is downloaded by agent i. Namely, when agent i accesses the cloud at time $t_{i,k}$, it downloads and stores the latest packet uploaded by each agent $j \in \mathcal{N}_i$. This information, together with the measurements $p_{i,k}$ and $v_{i,k}$, is used by agent i to compute its control input $u_{i,k}$ for the upcoming time interval $[t_{i,k}, t_{i,k+1})$, and to schedule the next cloud access $t_{i,k+1}$. The number $\mathcal{N}_i$ of other agents whose information is downloaded by agent i may be chosen

agent	1	2	...	N
last access	$t_{1,l_1(t)}$	$t_{2,l_2(t)}$	...	$t_{N,l_N(t)}$
position	$p_{1,l_1(t)}$	$p_{2,l_2(t)}$	...	$p_{N,l_N(t)}$
velocity	$v_{1,l_1(t)}$	$v_{2,l_2(t)}$	...	$v_{N,l_N(t)}$
control	$u_{1,l_1(t)}$	$u_{2,l_2(t)}$	...	$u_{N,l_N(t)}$
next access	$t_{1,l_1(t)+1}$	$t_{2,l_2(t)+1}$	...	$t_{N,l_N(t)+1}$

Table 4.1: Data contained in the cloud at a generic time instant $t \geq 0$. The i-th column corresponds to the latest packet uploaded by agent i.

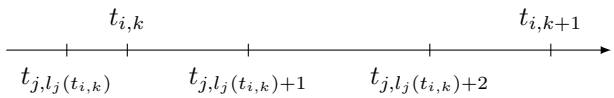

Figure 4.1: Excerpt of a possible sequence of cloud accesses on the time line. Recall that $t_{j,l_j(t)}$ denotes the most recent cloud access of agent j with respect to the time t. Note that there can be more than one access of agent j between two consecutive accesses of agent i.

according to the available bandwidth or to the computational capabilities of the agent.

In order to better illustrate the access sequence and the corresponding notation, Figure 4.1 illustrates a possible sequence of cloud accesses on the time line. Note that, in the scenario depicted in Figure 4.1, within the interval $[t_{i,k}, t_{i,k+1})$, while agent i is underwater and, therefore, isolated, agent j surfaces and changes its control input more than one time. Agent i does not know the control input that agent j will apply after t_{j,h_j+1}, nor it knows whether agent j will surface additional times after t_{j,h_j+1}. The scheduling algorithm is able to guarantee the overall system's convergence in spite of these limitations.

The cloud uses the packets that it is storing to compute information about the global state of the system. Such information can be downloaded by the agents when they access the cloud, and used to improve the coordination performance. Here, we consider the following case: when agent i accesses the cloud at time $t_{i,k}$, it also receives the positive scalar $\hat{\eta}(t_{i,k})$ representing an estimate of how far the system is from reaching the control objective. This estimate is formally defined in Section 4.5. The operations that each agent i performs upon each cloud access $t_{i,k}$ are summarized in the following Algorithm 2.

Remark 4.1. *In most existing self-triggered control protocols for multi-agent coordination (De Persis and Frasca, 2013, Fan et al., 2015), when one agent updates its*

Algorithm 2 Operations executed by agent i at $t_{i,k}$.

measure position $p_{i,k}$
measure velocity $v_{i,k}$
for $j \in \mathcal{N}_i$ **do**
  download packet $\{t_{j,l_j}, p_{j,l_j}, v_{j,l_j}, u_{j,l_j}, t_{j,l_j+1}\}$
end for
receive $\hat{\eta}(t_{i,k})$ from the cloud
compute control input $u_{i,k}$
schedule next access $t_{i,k+1}$
upload packet $\{t_{i,k}, p_{i,k}, v_{i,k}, u_{i,k}, t_{i,k+1}\}$

control input, such information is broadcast immediately to that agent's neighbors, which requires the neighbors to always be alert for possibly coming information. This requirement is relaxed in the proposed cloud-supported framework.

Controller

The controls $u_{i,k}$, with $i \in \mathcal{V}$, are computed as follows:

$$u_{i,k} = \sum_{j \in \mathcal{N}_i} w_{ij}(k_p(\hat{p}_j(t_{i,k}) - p_{i,k}) + k_v(\hat{v}_j(t_{i,k}) - v_{i,k})), \tag{4.6a}$$

$$\hat{v}_j(t) = v_{j,l_j(t)} + u_{j,l_j(t)}(t - t_{j,l_j(t)}), \tag{4.6b}$$

$$\hat{p}_j(t) = p_{j,l_j(t)} + v_{j,l_j(t)}(t - t_{j,l_j(t)}) + \frac{1}{2}u_{j,l_j(t)}(t - t_{j,l_j(t)})^2, \tag{4.6c}$$

where $k_p, k_v > 0$ are control gains and $w_{ij} > 0$ represents the strength of the influence of agent j on agent i. The values $\hat{p}_j(t_{(i,k)})$ and $\hat{v}_j(t_{(i,k)})$ represent the estimates of, respectively, the position and the velocity of agent j at time $t_{i,k}$. Note that, in order to compute such estimates, agent i only needs the data downloaded from the cloud at time $t_{i,k}$, and it is not necessary to communicate directly with agent j.

The sets $\mathcal{N}_1, \ldots, \mathcal{N}_N$ and the scalars w_{ij} induce a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ over the set $\mathcal{V}$ of the agents, where $\mathcal{N}_i$ is the set of the neighbors of i and w_{ij} is the weight of edge (j, i). We are going to refer to this graph as the *network graph*. Throughout the chapter, we assume that the network graph contains a spanning tree.

Assumption 4.2. *The network graph contains a spanning tree.*

Control Objective

As we have anticipated, the control objective is that the agents synchronize their positions and velocities. This objective can be formalized in a convenient way if

we exploit Assumption 4.2. Denote the spanning tree of the network graph as $\mathcal{T}$ and let $\mathcal{C} = \mathcal{E} \setminus \mathcal{T}$. Without loss of generality, let the edges that are in $\mathcal{T}$ be indexed from 1 to $N-1$ and the edges that are in $\mathcal{C}$ be indexed from N to M, and partition the incidence matrix and weight matrix accordingly as $B = [B_{\mathcal{T}}, B_{\mathcal{C}}]$ and $W = [W_{\mathcal{T}}, W_{\mathcal{C}}]$. Denote $x(t) = (B_{\mathcal{T}}^{\intercal} \otimes I_n)p(t)$ and $y(t) = (B_{\mathcal{T}}^{\intercal} \otimes I_n)v(t)$. In other words, $x(t) = [x_1(t)^{\intercal}, \dots, x_{N-1}(t)^{\intercal}]^{\intercal}$, where each $x_\iota(t) = p_{\mathrm{head}(\iota)}(t) - p_{\mathrm{tail}(\iota)}$ is the difference between the positions of two agents whose indexes constitute an edge in $\mathcal{T}$. Similarly, $y(t) = [y_1(t)^{\intercal}, \dots, y_{N-1}(t)^{\intercal}]^{\intercal}$, where $y_\iota(t) = v_{\mathrm{head}(\iota)}(t) - v_{\mathrm{tail}(\iota)}$. Let $\xi(t) = [x(t)^{\intercal}, y(t)^{\intercal}]^{\intercal}$. Note that, for each $i, j \in \mathcal{V}$, $p_j(t) - p_i(t)$ can be written as a linear combination of the variables $x_\iota(t)$ for $\iota \in \{1, \dots, M\}$, and, similarly, $v_j(t) - v_i(t)$ can be written as a linear combination of the variables $y_\iota(t)$. Therefore, the control objective can be formalized in terms of $\xi(t)$ as follows.

Definition 4.1. *Consider the multi-agent system* (4.2) *and the associated network graph* $\mathcal{G}$*. We say that the multi-agent system achieves* practical convergence *with tolerance* $\epsilon \geq 0$ *if*

$$\limsup_{t \to \infty} \|\xi(t)\| \leq \epsilon. \tag{4.7}$$

In particular, if the system achieves practical convergence with tolerance $\epsilon = 0$*, we say that the system achieves* asymptotic convergence*.*

4.3 Self-triggered cloud access scheduling

Each agent schedules its own access to the cloud recursively, i.e., agent i schedules the access $t_{i,k+1}$ when it accesses the cloud at time $t_{i,k}$. The scheduling is based on comparing two time-varying functions of the data downloaded from the cloud with a given threshold function. The threshold function is chosen as

$$\varsigma(t) = \varsigma_\infty + (\varsigma_0 - \varsigma_\infty)e^{-\lambda_\varsigma t}, \tag{4.8}$$

with $\lambda_\varsigma > 0$ and $0 \leq \varsigma_\infty < \varsigma_0$. To define the scheduling rule, we need to introduce some additional notation. Let $\hat{p}(t) = [\hat{p}_1(t)^{\intercal}, \dots, \hat{p}_N(t)^{\intercal}]^{\intercal}$, where $\hat{p}_i(t)$ is defined in (4.6c), and similarly for $\hat{v}(t)$. Let

$$\hat{x}(t) = (B_{\mathcal{T}} \otimes I_n)\hat{p}(t), \tag{4.9a}$$
$$\hat{y}(t) = (B_{\mathcal{T}} \otimes I_n)\hat{v}(t), \tag{4.9b}$$
$$\hat{\xi}(t) = [\hat{x}(t)^{\intercal}, \hat{y}(t)^{\intercal}]^{\intercal}. \tag{4.9c}$$

Moreover, let

$$\Delta_i(t) = \int_{t_{i,l_i(t)}}^{t} \int_{t_{i,l_i(t)}}^{\tau} \delta(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau + \int_{t_{i,l_i(t)}}^{t} \delta(\tau)\,\mathrm{d}\tau\,, \tag{4.10a}$$
$$\Delta(t) = [\Delta_1(t), \dots, \Delta_N(t)]^{\intercal}, \tag{4.10b}$$
$$\hat{\eta}(t) = \|\hat{\xi}(t)\| + \|B_{\mathcal{T}}\| \cdot \|\Delta(t)\|, \tag{4.10c}$$

where $\delta(\cdot)$ is defined in Assumption 4.1. Note that (4.10c), evaluated for $t = t_{i,k}$, defines the estimate $\hat{\eta}(t_{i,k})$ that agent i receives from the cloud at time $t_{i,k}$. Moreover, let

$$F_{e,r} = \begin{bmatrix} 0_{(N-1)\times(N-1)} & I_{N-1} \\ -k_p R & -k_v R \end{bmatrix}, \tag{4.11}$$

where k_p and k_v are the control gains in (4.6a), R is the reduced edge Laplacian of the network graph, and

$$\lambda = -\max\{\mathrm{Re}(s) : s \in \mathrm{eig}(F_{e,r})\}. \tag{4.12}$$

Consider the function

$$\eta(t, \eta_0) = e^{-\lambda(t-t_0)}\eta_0 + \sqrt{N}\|B_{\mathcal{T}}\| \int_{t_0}^{t} e^{-\lambda(t-\tau)}(\varsigma(\tau) + \delta(\tau))\,\mathrm{d}\tau\,, \tag{4.13}$$

and the coefficients

$$\beta_i = \left(\sqrt{k_p^2 + k_v^2}\right)\left\|(W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})_i\right\|, \tag{4.14a}$$

$$\nu_i = \max_{j:i\in\mathcal{N}_j}\left\{\sum_{q\in\mathcal{N}_j} w_{qj}\right\}, \tag{4.14b}$$

where $(W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})_i$ denotes the i-th row of $(W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})$. Finally, choose α such that $0 < \alpha < 1$. Then, each agent schedules the cloud accesses as follows:

$$t_{i,k+1} = \inf\left\{t > t_{i,k} : \sigma_{i,k}(t) \geq \varsigma(t) \vee \Omega_{i,k}(t) \geq \frac{\alpha}{\nu_i}\varsigma(t)\right\}, \tag{4.15}$$

where

$$\Omega_{i,k}(t) = k_p \int_{t_{i,k}}^{t}\int_{t_{i,k}}^{\tau} \delta(\sigma)\mathrm{d}\sigma\,\mathrm{d}\tau + k_v \int_{t_{i,k}}^{t} \delta(\tau)\,\mathrm{d}\tau\,, \tag{4.16a}$$

$$
\begin{aligned}
\sigma_{i,k}(t) = \Bigg\| & \Bigg(\sum_{j\in\mathcal{N}} w_{ij}\Bigg)\big(k_v(t-t_{i,k})u_{i,k} \\
& + k_p((t-t_{i,k})v_{i,k} + (1/2)(t-t_{i,k})^2 u_{i,k})\big) \\
& + \sum_{j\in\mathcal{N}_i} w_{ij}(k_v(t'_{j,h_j} - t_{i,k})u_{j,h_j} \\
& + k_p((t-t_{i,k})v_{j,h_j} \\
& + (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k})u_{j,h_j} \\
& + (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j})u_{j,h})\Bigg\| \\
& + \sum_{j\in\mathcal{N}_i} w_{ij}\Bigg(\int_{t_{j,h_j+1}}^{t''_{j,h_j}} \mu_j^{i,k}(\tau)\,\mathrm{d}\tau \\
& + \int_{t_{j,h_j+1}}^{t''_{j,h_j}}\int_{t_{j,h_j+1}}^{\tau} \mu_j^{i,k}(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau\Bigg) \\
& + \Bigg(\sum_{j\in\mathcal{N}_i} w_{ij}\Bigg)\Omega_{i,k}(t) + \sum_{j\in\mathcal{N}_i} w_{ij}\Omega_{j,h_j}(t),
\end{aligned}
\tag{4.16b}
$$

$$
\mu_j^{i,k}(t) = \beta_j \eta(t, \hat{\eta}(t_{i,k})) + \varsigma(t), \tag{4.16c}
$$

$$
t'_{j,h_j} = \min\{t, t_{j,h_j+1}\}, \tag{4.16d}
$$

$$
t''_{j,h_j} = \max\{t, t_{j,h_j+1}\}, \tag{4.16e}
$$

and where we have denoted $h_j = l_j(t_{i,k})$ for brevity. Recall here that $l_i(t)$ is defined by (4.4). The scheduling rule (4.15) and (4.16) can be interpreted as follows. After the access $t_{i,k}$, $\Omega_{i,k}(t)$ represents an upper bound on the part of $\|\tilde{u}_i(t)\|$ that is due to the disturbances that have acted on agent i in the interval $[t_{i,k}, t)$. This upper bound is kept under a threshold (here set to $(\alpha/\nu_i)\varsigma(t)$), so that the information deposited by agent i in the cloud can be used to predict its position and velocity within a certain error bound. Similarly, $\sigma_{i,k}(t)$ represents an upper bound on the mismatch $\|\tilde{u}_i(t)\|$, which takes into account both the effect of the disturbances $d_j(t)$ with $j \in \mathcal{N}_i \cup \{i\}$, the piecewise-constant nature of the control signals $u_j(t)$ with $j \in \mathbb{N}_i \cup \{i\}$, and the fact that, for $t > t_{j,h_j+1}$, the current value of u_{j,h_j+1} of $u_j(t)$ is not known by agent i (agent i has downloaded u_{j,h_j} in the latest packet, but has no information on the choice of the control that agent j will adopt at time t_{j,h_j+1}). Indeed, for $t > t_{j,h_j+1}$, agent i uses $\mu_j^{i,k}(t)$ as an upper-bound for $u_j(t)$, hence exploiting the estimate $\hat{\eta}(t_{i,k})$. The upper bound $\sigma_{i,k}(t)$ is kept under $\varsigma(t)$, so that the hypotheses of Lemma 4.2 are satisfied. A new cloud access is triggered when either of $\Omega_{i,k}(t)$ or $\sigma_{i,k}(t)$ are about to cross the assigned threshold. In fact, under the scheduling rule (4.15), since $\Omega_{i,k}(t)$ and $\sigma_{i,k}(t)$ are continuous functions of time, we have $\Omega_{i,k}(t) < \alpha/\nu_i\,\varsigma(t)$ and $\sigma_{i,k}(t) < \varsigma(t)$ for all $t \in (t_{i,k}, t_{i,k+1})$. Note that (4.15) and (4.16) can be evaluated by agent i when it accesses the cloud

(i.e., at time $t_{i,k}$) and do not require communication with the other agents. Under such assumption, it is easy to prove that the multi-agent system (4.1) achieves the desired control objective. Then, we show that said assumption is satisfied at all times if the accesses of the agents to the cloud are scheduled according to (4.15) and (4.16).

Remark 4.2. *The requirement that the cloud should compute $\hat{\eta}(t_{i,k})$ upon each agent connection introduces a form of centralized computation in the algorithm. However, as we shall see later in the chapter, $\hat{\eta}(t_{i,k})$ is only used in the scheduling as an upper bound, and it can be substituted by a more conservative bound (for example, a bound depending on the initial conditions) if it is desired to avoid this centralized computation.*

4.4 Main result

Our main result is formalized as the following theorem.

Theorem 4.1. *Consider the multi-agent system* (4.1)*, with control law* (4.6a) *to* (4.6c) *and cloud accesses scheduled by* (4.15) *and* (4.16)*. Let Assumptions 4.1 and 4.2 hold, and let k_p and k_v be such that $F_{e,r}$ is Hurwitz. If $\varsigma_\infty > 0$, the closed-loop system does not exhibit Zeno behavior and achieves practical convergence with radius*

$$\epsilon = \frac{\sqrt{N}\|B_{\mathcal{T}}\|(\varsigma_\infty + \delta_\infty)}{\lambda}, \tag{4.17}$$

where ς_∞ is the asymptotic value of the threshold function (4.8)*, δ_∞ is the asymptotic value of the disturbance bound* (4.3)*, and λ is defined in* (4.12)*. If $\delta_\infty = 0$, $\varsigma_\infty = 0$ and $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$, then the closed-loop system does not exhibit Zeno behavior and achieves asymptotic convergence.*

Remark 4.3. *Note that our convergence result* (4.17) *is similar to the convergence results obtained in classic papers about event-triggered coordination of multi-agent system, such as the references presented in Section 2.5. Here, however, convergence is obtained by means of a remote repository accessed asynchronously by the agents, and not by direct inter-agent communication.*

Remark 4.4. *Note that, under Assumption 4.2, we can always choose k_p and k_v such that $F_{e,r}$ is Hurwitz. The proof of this remark is formalized as follows.*

The proof of Theorem 4.1 is given in the following three sections of the chapter. Namely, in Section 4.5, we study the convergence properties of the closed-loop system, while, in Section 4.6, we show that the closed-loop system does not exhibit Zeno behavior (Johansson et al., 1999). Finally, in Section 4.7 we put the results of Sections 4.5 and 4.6 together to state a formal proof of Theorem 4.1.

4.5 Convergence of the closed-loop system

Our first step in the analysis of the closed-loop system is to rewrite the system dynamics in terms of the error vector $\xi(t)$. First, compare the control signals $u_{i,k}$ defined by (4.6a) with

$$z_i(t) = \sum_{j \in \mathcal{N}_i} w_{ij}(k_p(p_j(t) - p_i(t)) + k_v(v_j(t) - v_i(t))). \tag{4.18}$$

We can write $z_i(t)$ in terms of the incidence matrix and the weight matrix of the network graph as

$$z_i(t) = ((W_i B^\intercal) \otimes I_n)(k_p p(t) + k_v v(t)), \tag{4.19}$$

where W_i denotes the i-th row of W. Letting $z(t) = [z_1(t)^\intercal, \dots, z_N(t)^\intercal]^\intercal$, we can rewrite (4.18) as

$$z(t) = ((W B^\intercal) \otimes I_n)(k_p p(t) + k_v v(t)). \tag{4.20}$$

Moreover, substituting $W = [W_\mathcal{T}\ W_\mathcal{C}]$ and $B = [B_\mathcal{T}\ B_\mathcal{C}] = B_\mathcal{T}[I\ T]$ in (4.20), we have

$$z(t) = ((W_\mathcal{T} + W_\mathcal{C} T^\intercal) B_\mathcal{T}^\intercal \otimes I_n)(k_p p(t) + k_v v(t)). \tag{4.21}$$

Using the properties of the Kronecker product, and recalling that $x(t) = (B_\mathcal{T}^\intercal \otimes I_n)p(t)$ and $y(t) = (B_\mathcal{T}^\intercal \otimes I_n)v(t)$, we can rewrite (4.21) as

$$z(t) = ((W_\mathcal{T} + W_\mathcal{C} T^\intercal) \otimes I_n)(k_p x(t) + k_v y(t)). \tag{4.22}$$

Left-multiplying both sides of (4.22) by $B_\mathcal{T}^\intercal \otimes I_n$, using again the properties of the Kronecker product, and denoting the reduced edge Laplacian of the network graph as R, we have

$$(B_\mathcal{T}^\intercal \otimes I_n) z(t) = (R \otimes I_n)(k_p x(t) + k_v y(t)). \tag{4.23}$$

Let $\tilde{u}_i(t)$ be the mismatch between the control input of agent i and $z_i(t)$, namely,

$$\tilde{u}_i(t) = u_i(t) - z_i(t). \tag{4.24}$$

We denote $\tilde{u}(t) = [\tilde{u}_1(t)^\intercal, \dots, \tilde{u}_N(t)^\intercal]^\intercal$, so that we can rewrite (4.24) as

$$\tilde{u}(t) = u(t) - z(t). \tag{4.25}$$

Left-multiplying both sides of (4.2a) and (4.2b) by $B_\mathcal{T}^\intercal \otimes I_n$, we have

$$\dot{x}(t) = y(t), \tag{4.26a}$$
$$\dot{y}(t) = (B_\mathcal{T}^\intercal \otimes I_n)(u(t) + d(t)). \tag{4.26b}$$

Substituting (4.23) and (4.25) in (4.26a) and (4.26b), we have

$$\dot{x}(t) = y(t), \tag{4.27a}$$

$$\begin{aligned}\dot{y}(t) = &-(R \otimes I_n)(k_p x(t) + k_v y(t)) \\ &+ (B_{\mathcal{T}}^{\intercal} \otimes I_n)(\tilde{u}(t) + d(t)),\end{aligned} \tag{4.27b}$$

which, recalling that $\xi(t) = [x(t)^{\intercal}\ y(t)^{\intercal}]^{\intercal}$, can be rewritten as

$$\dot{\xi}(t) = (F_{e,r} \otimes I_n)\xi(t) + (G \otimes I_n)(\tilde{u}(t) + d(t)), \tag{4.28}$$

where $F_{e,r}$ is defined in (4.11) and $G = [0_{(N-1)\times N}^{\intercal}\ B_{\mathcal{T}}^{\intercal}]^{\intercal}$.

The following Lemma 4.1 shows that $\hat{\eta}(t)$ defined by (4.10c) constitutes an upper bound for the state error vector $\xi(t)$.

Lemma 4.1. *Under Assumption 4.1, we have $\|\xi(t)\| \leq \hat{\eta}(t)$ for all $t \geq 0$, where $\hat{\eta}(t)$ is defined by* (4.10c).

Proof. Denote $D_{v,i}(t) = \int_{t_{i,l_i(t)}}^{t} d_i(\tau)\,\mathrm{d}\tau$ and $D_{p,i}(t) = \int_{t_{i,l_i(t)}}^{t}\int_{t_{i,l_i(t)}}^{\tau} d_i(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau$, let $D_p(t) = [D_{p,1}(t), \ldots, D_{p,N}(t)]$, and similarly for $D_v(t)$. Using (4.1), (4.6b) and (4.6c), we have

$$p(t) = \hat{p}(t) + D_p(t), \tag{4.29a}$$
$$v(t) = \hat{v}(t) + D_v(t), \tag{4.29b}$$

Left multiplying (4.29a) and (4.29b) by $(B_{\mathcal{T}} \otimes I_n)$, and using (4.9a) to (4.9c), we have

$$\xi(t) = \hat{\xi}(t) + ((B_{\mathcal{T}} \otimes I_n) \otimes I_2)D(t), \tag{4.30}$$

where we have denoted $D(t) = [D_p(t)^{\intercal}, D_v(t)^{\intercal}]^{\intercal}$. Taking norms of both sides, and using the triangular inequality, the properties of the Kronecker product, and Assumption 4.1, we have

$$\|\xi(t)\| \leq \|\hat{\xi}(t)\| + \|B_{\mathcal{T}}\|\|D(t)\|. \tag{4.31}$$

Under Assumption 4.1, we have $\|D(t)\| \leq \|\Delta(t)\|$, which substituted in (4.31) yields the desired result. □

Note that $\hat{\eta}(t)$ as defined in (4.10c) can be computed by the cloud at any time instant. However, the cloud does not need to compute $\hat{\eta}(t)$ at all time instants, but only when an agent connects to download $\hat{\eta}(t_{i,k})$. As a consequence of Lemma 4.1, we have, in particular,

$$\|\xi(t_{i,k})\| \leq \hat{\eta}(t_{i,k}). \tag{4.32}$$

The following lemma relates a bound on the control errors $\tilde{u}_i(t)$ to a bound on the state error vector $\xi(t)$ and on the control signals $u_i(t)$.

Lemma 4.2. *Consider the multi-agent system* (4.1)*, and let Assumption 4.1 hold. Suppose that*

$$\|\tilde{u}_i(t)\| \leq \varsigma(t) \tag{4.33}$$

for all $t \in [t_0, t_f)$ *and all* $i \in \mathcal{V}$*, where* $\tilde{u}_i(t)$ *is defined by* (4.24) *and* $\varsigma(t)$ *is the threshold function* (4.8)*. Let* $\eta_0 \geq \|\xi(t_0)\|$*. Then, for all* $t \in [t_0, t_f)$*, we have*

$$\|\xi(t)\| \leq \eta(t, \eta_0), \tag{4.34}$$

where $\eta(\cdot, \cdot)$ *is defined by* (4.13)*. Moreover, we have*

$$\|u_i(t)\| \leq \beta_i \eta(t, \eta_0) + \varsigma(t) \tag{4.35}$$

for all $t \in [t_0, t_f)$*, and all* $i \in \mathcal{V}$*, where* β_i *is defined by* (4.14a)*.*

Proof. The Laplace solution of (4.28) reads

$$\xi(t) = e^{F_{e,r}(t-t_0)}\xi(t_0) + \int_{t_0}^{t} e^{F_{e,r}(t-\tau)}(G \otimes I_n)(\tilde{u}(\tau) + d(\tau))\,\mathrm{d}\tau\,. \tag{4.36}$$

Taking norms of both sides in (4.36), and using (4.33), Assumption 4.1, the properties of the Kronecker product, and the triangular inequality, and observing that $\|e^{F_{e,r}(t-t_0)}\| \leq e^{-\lambda(t-t_0)}$, and that $\|G\| = \|B_{\mathcal{T}}\|$, we have (4.34). Moreover, from (4.24), we have $u_i(t) = z_i(t) + \tilde{u}_i(t)$. Taking norms of both sides, and using the triangular inequality, we have $\|u_i(t)\| \leq \|z_i(t)\| + \|\tilde{u}_i(t)\|$. Selecting the rows corresponding to the i-th agent in (4.22), we have $z_i(t) = ((W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})_i \otimes I_n)(k_p x(t) + k_v y(t))$, where $(W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})_i$ denotes the i-th row of $(W_{\mathcal{T}} + W_{\mathcal{C}}T^{\intercal})$. Taking norms of both sides, and substituting the result in the previous inequality, we have $\|u_i(t)\| \leq \beta_i\|\xi(t)\| + \|\tilde{u}_i(t)\|$. Using (4.33) and (4.34), we obtain (4.35). □

Since (4.32) holds, we can invoke Lemma 4.2 with $t_0 = t_{i,k}$ and $\eta_0 = \hat{\eta}(t_{i,k})$, which leads to the implication

$$\begin{aligned} &\|\tilde{u}_j(t)\| \leq \varsigma(t) \quad \forall t \in [t_{i,k}, t_f), j \in \mathcal{V} \\ &\quad \implies \|u_j(t)\| \leq \mu_j^{i,k}(t) \quad \forall t \in [t_{i,k}, t_f), j \in \mathcal{V}, \end{aligned} \tag{4.37}$$

where $\mu_j^{i,k}(t)$ is defined by (4.16c).

The following Lemma 4.3 shows that, under the scheduling rule (4.15) and (4.16), we can guarantee that $\|\tilde{u}_i(t)\| \leq \|\varsigma(t)\|$ for all agents, thus satisfying the hypotheses of Lemma 4.2.

Lemma 4.3. *Consider the multi-agent system* (4.2) *under the control law* (4.5) *and* (4.6a) *to* (4.6c) *and the scheduling rule* (4.15) *and* (4.16)*. Then, under Assumption 4.1, we have* $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ *for all* $t \geq 0$ *and* $i \in \mathcal{V}$*.*

Proof. Since (4.15) guarantees $\sigma_{i,l_i(t)}(t) \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$, we only need to show that $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. Without loss of generality, let $l_i(t) = k$, and consider $t \in [t_{i,k}, t_{i,k+1})$. Substituting (4.6a) and (4.18) in (4.24), we have

$$\begin{aligned}\tilde{u}_i(t) = \sum_{j \in \mathcal{N}_i} w_{ij}(k_p((\hat{p}_j^{(i,k)} - p_j(t)) - (p_{i,k} - p_i(t))) \\ + k_v((\hat{v}_j^{(i,k)} - v_j(t)) - (v_{i,k} - v_i(t))))\,.\end{aligned} \tag{4.38}$$

First, consider the term $v_i(t)$ in (4.38). Integrating (4.1b) in $(t_{i,k}, t)$, we have, for $t \in (t_{i,k}, t_{i,k+1})$,

$$v_i(t) = v_{i,k} + (t - t_{i,k})u_{i,k} + \int_{t_{i,k}}^{t} d_i(t)\,\mathrm{d}\tau\,. \tag{4.39}$$

Now consider the term $v_j(t)$ in (4.38). Integrating (4.1b) for agent j in (t_{j,h_j}, t), and using (4.6b), we have

$$v_j(t) = \hat{v}_j^{(i,k)} + \int_{t_{i,k}}^{t} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \tag{4.40}$$

Now we need to distinguish two cases. If $t \leq t_{j,h_j+1}$, then (4.40) can be rewritten as

$$v_j(t) = \hat{v}_j^{(i,k)} + (t - t_{i,k})u_{j,h_j} + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \tag{4.41}$$

Conversely, if $t > t_{j,h_j+1}$, (4.40) can be rewritten as

$$\begin{aligned}v_j(t) = \hat{v}_j^{(i,k)} + (t_{j,h_j+1} - t_{i,k})u_{j,h_j} \\ + \int_{t_{j,h_j+1}}^{t} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,.\end{aligned} \tag{4.42}$$

Using (4.16d) and (4.16e), we can write (4.41) and (4.42) compactly as

$$\begin{aligned}v_j(t) = \hat{v}_j^{(i,k)} + (t'_{j,h_j} - t_{i,k})u_{j,h_j} \\ + \int_{t_{j,h_j+1}}^{t''_{j,h_j}} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,.\end{aligned} \tag{4.43}$$

Now consider the term $p_i(t)$ in (4.38). Integrating (4.1a) in $(t_{i,k}, t)$, and using (4.39), we have, for $t \in (t_{i,k}, t_{i,k+1})$,

$$\begin{aligned}p_i(t) = p_{i,k} + v_{i,k}(t - t_{i,k}) + (1/2)(t - t_{i,k})^2 u_{i,k} \\ + \int_{t_{i,k}}^{t}\int_{t_{i,k}}^{\tau} d_i(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau\,.\end{aligned} \tag{4.44}$$

Finally, consider the term $p_j(t)$ in (4.38). Integrating (4.1a) for agent j in $[t_{j,h_j}, t)$, and using (4.6c), we have

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k}) v_{j,h_j} \\ &+ \int_{t_{i,k}}^{t} \int_{t_{j,h_j}}^{\tau} u_j(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau + \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau \,. \end{aligned} \tag{4.45}$$

Similarly as we did for (4.40), we need to distinguish two cases. If $t \leq t_{j,h_j+1}$, then (4.45) can be rewritten as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k}) v_{j,h_j} \\ &+ (1/2)(t - t_{i,k})(t + t_{i,k} - 2t_{j,h_j}) u_{j,h_j} \\ &+ \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau \,. \end{aligned} \tag{4.46}$$

Conversely, if $t > t_{j,h_j+1}$, (4.45) can be rewritten as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k}) v_{j,h_j} \\ &+ (1/2)(t_{j,h_j+1} - t_{i,k})(t_{j,h_j+1} + t_{i,k} - 2t_{j,h_j}) u_{j,h_j} \\ &+ (t - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j}) u_{j,h_j} \\ &+ \int_{t_{j,h_j+1}}^{t} \int_{t_{j,h_j+1}}^{\tau} u_j(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau + \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau \,. \end{aligned} \tag{4.47}$$

Using (4.16d) and (4.16e), we can write (4.46) and (4.47) compactly as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k}) v_{j,h_j} \\ &+ (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k}) u_{j,h_j} \\ &+ (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j}) u_{j,h} \\ &+ \int_{t_{j,h_j+1}}^{t''_{j,h_j}} \int_{t_{j,h_j+1}}^{\tau} u_j(\theta) \, \mathrm{d}\theta \, \mathrm{d}\tau + \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\theta) \, \mathrm{d}\theta \, \mathrm{d}\tau \,. \end{aligned} \tag{4.48}$$

Substituting (4.39), (4.43), (4.44) and (4.48) in (4.38), taking norms of both sides,

and using the triangular inequality and Assumption 4.1, we have

$$\begin{aligned}
\|\tilde{u}_i(t)\| \leq & \left\| \left(\sum_{j \in \mathcal{N}} w_{ij} \right) (k_v(t - t_{i,k}) u_{i,k} \right. \\
& + k_p((t - t_{i,k}) v_{i,k} + (1/2)(t - t_{i,k})^2 u_{i,k})) \\
& + \sum_{j \in \mathcal{N}_i} w_{ij} (k_v(t'_{j,h_j} - t_{i,k}) u_{j,h_j} \\
& + k_p((t - t_{i,k}) v_{j,h_j} \\
& + (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k}) u_{j,h_j} \\
& \left. + (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j}) u_{j,h}) \right\| \\
& + \sum_{j \in \mathcal{N}_i} w_{ij} \left(\int_{t_{j,h_j+1}}^{t''_{j,h_j}} \|u_j(\tau)\| \, \mathrm{d}\tau \right. \\
& \quad \left. + \int_{t_{j,h_j+1}}^{t''_{j,h_j}} \int_{t_{j,h_j+1}}^{\tau} \|u_j(\theta)\| \, \mathrm{d}\theta \, \mathrm{d}\tau \right) \\
& + \left(\sum_{j \in \mathcal{N}_i} w_{ij} \right) \Omega_{i,k}(t) + \sum_{j \in \mathcal{N}_i} w_{ij} \Omega_{j,h_j}(t).
\end{aligned} \tag{4.49}$$

Comparing (4.49) with (4.16b), we have that, if $\|\tilde{u}_i(t)\| > \sigma_{i,k}(t)$ for some $t \in (t_{i,k}, t_{i,k+1})$, then it must be $\|u_j(\tau)\| > \mu_j^{i,k}(\tau)$ for some $j \in \mathcal{V}$ and some $\tau \in (t_{j,h_j+1}, t)$. But, by (4.37), the previous inequality implies $\|\tilde{u}_j(\tau')\| > \varsigma(\tau')$ for some $\tau' \in (t_{j,h_j+1}, t)$, which, by (4.15), implies in turn $\|\tilde{u}_j(\tau')\| > \sigma_{j,l_j(\tau')}(\tau')$. Therefore, the condition $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ cannot be violated by any of the agents if it has not been previously violated by another agent. Since we have $\tilde{u}(0) = 0_{Nn}$, this condition holds for all the agents at time zero, and, therefore, cannot be violated by any of the agents. Hence, we have $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$, which, by (4.15), implies $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. □

4.6 Well-posedness of the closed-loop system

The second step in our analysis is to prove that the closed-loop system is well posed, in the sense that the sequence of the updates $t_{i,k}$ for $k \in \mathbb{N}_0$ does not present Zeno behavior for any of the agents. We are going to distinguish two cases, namely $\varsigma_\infty > 0$ and $\varsigma_\infty = 0$, where ς_∞ is the asymptotic value of the threshold function (4.8).

Lemma 4.4. *Consider the multi-agent system* (4.2)*, with control law* (4.5) *and* (4.6a) *to* (4.6c) *and cloud accesses scheduled by* (4.15) *and* (4.16)*. Let k_p and k_v be such that $F_{e,r}$ is Hurwitz. Suppose $\varsigma_\infty > 0$. Then, under Assumptions 4.1 and 4.2, the closed-loop system does not exhibit Zeno behavior.*

Proof. First, note that $\Omega_{i,k}(t)$, for $t \in [t_{i,k}, t_{i,k+1})$, can be computed explicitly as

$$\begin{aligned}\Omega_{i,k}(t) = k_p\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}e^{-\lambda_\delta t_{i,k}}\Bigg((t-t_{i,k}) \\ - \frac{1-e^{-\lambda_\delta(t-t_{i,k})}}{\lambda_\delta}\Bigg) + \frac{1}{2}\delta_\infty(t-t_{i,k})^2\Bigg) \\ + k_v\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}e^{-\lambda_\delta t_{i,k}}(1-e^{-\lambda_\delta(t-t_{i,k})}) + \delta_\infty(t-t_{i,k})\Bigg).\end{aligned} \tag{4.50}$$

Observing that $e^{-\lambda_\delta t_{i,k}} \leq 1$, we can bound (4.50) as

$$\begin{aligned}\Omega_{i,k}(t) \leq k_p\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}(t-t_{i,k}) + \frac{1}{2}\delta_\infty(t-t_{i,k})^2\Bigg) \\ + k_v\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}(1-e^{-\lambda_\delta(t-t_{i,k})}) + \delta_\infty(t-t_{i,k})\Bigg),\end{aligned} \tag{4.51}$$

while $\varsigma(t) \geq \varsigma_\infty$. Therefore, for $\Omega_{i,k}(t)$ to be larger than $\alpha\,\varsigma(t)/\nu_i$, the right-hand side of (4.51) must be greater or equal than $\alpha\,\varsigma_\infty/\nu_i$, which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\Omega_{i,k}(t) \geq \alpha\,\varsigma(t)/\nu_i$ cannot be responsible for Zeno behavior. Now we produce a similar argument for the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$. First, note that, thanks to (4.14b) and (4.15), the last two addends of $\sigma_{i,k}(t)$ in (4.16b) can be bounded by

$$\Bigg(\sum_{j\in\mathcal{N}} w_{ij}\Bigg)\Omega_{i,k}(t) + \sum_{j\in\mathcal{N}} w_{ij}\Omega_{j,h_j}(t) \leq \alpha\,\varsigma(t). \tag{4.52}$$

Moreover, for $t \in [t_{i,k}, t_{i,k+1})$, letting $h_j = l_j(t_{i,k})$, we have, from Lemma 4.2,

$$\|u_{i,k}\| \leq \beta_i\eta(t_{i,k}, \|\xi(0)\|) + \varsigma(t_{i,k}), \tag{4.53a}$$
$$\|u_{j,h_j}\| \leq \beta_j\eta(t_{i,k}, \|\xi(0)\|) + \varsigma(t_{i,k}). \tag{4.53b}$$

Since $\eta(t, \eta_0)$ is an upper-bounded function of t for any η_0, we can denote as $\bar{\eta}$ the upper bound of $\eta(\cdot, \|\xi(0)\|)$, which, by observing also that $\varsigma(t_{i,k}) \leq \varsigma_0$, allows us to further bound (4.53a) and (4.53b) as

$$\|u_{i,k}\| \leq \beta_i\bar{\eta} + \varsigma_0, \tag{4.54a}$$
$$\|u_{j,h_j}\| \leq \beta_j\bar{\eta} + \varsigma_0\,. \tag{4.54b}$$

Reasoning similarly for $\mu_j(t)$ for $t \in [t_{j,h_j+1})$, we have

$$\mu_j(t) \leq \beta_j\bar{\eta} + \varsigma_0\,. \tag{4.55}$$

Also, note that

$$\|v_{j,h_j} - v_{i,k}\| \leq \|\xi(t_{i,k})\| \leq \bar{\eta}. \tag{4.56}$$

Substituting (4.52), (4.54a), (4.54b), (4.55) and (4.56) into (4.16b), and using the triangular inequality, we have

$$\begin{aligned}\sigma_{i,k}(t) \leq &\left(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j\bar{\eta}+\varsigma_0+\beta_i\bar{\eta}+\varsigma_0)\right)\\ &(k_v(t-t_{i,k})+(1/2)k_p(t-t_{i,k})^2)\\ &+\left(\sum_{j\in\mathcal{N}} w_{ij}\right)\bar{\eta}k_p(t-t_{i,k})+\alpha\,\varsigma(t).\end{aligned} \tag{4.57}$$

Noticing also that $\varsigma(t) \geq \varsigma_\infty$ for all $t \geq 0$, the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ implies that the first two addends of (4.57) are larger than $(1-\alpha)\,\varsigma_\infty$, which requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ cannot be responsible for Zeno behavior either. □

When the disturbances eventually vanish, the particular case of asymptotic convergence may be considered with the specific choice of $\varsigma_\infty = 0$ and a convergence rate λ_ς slower than λ, which is defined by (4.12) and can be interpreted as the natural convergence rate of the network. This result is formally stated in the following lemma.

Lemma 4.5. *Consider the multi-agent system* (4.1)*, with control law* (4.5) *and* (4.6a) *to* (4.6c) *and cloud accesses scheduled by* (4.15) *and* (4.16)*. Let Assumptions 4.1 and 4.2 hold, with* $\delta_\infty = 0$ *in Assumption 4.1. Choose* k_p *and* k_v *such that* $F_{e,r}$ *is Hurwitz, and choose* $\varsigma_\infty = 0$ *and* $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$*. Then, the closed-loop system does not exhibit Zeno behavior.*

Proof. We write the proof for the case $\lambda > \lambda_\delta$. The structure of the proof is the same for $\lambda < \lambda_\delta$ and $\lambda = \lambda_\delta$, with only the expression (4.65) of the coefficient $\bar{\bar{\eta}}$ taking a slightly different form. Under Assumption 4.1 with $\delta_\infty = 0$ and $\lambda_\varsigma < \lambda_\delta$, we can write

$$\delta(t) \leq \delta_0 e^{-\lambda_\varsigma t}. \tag{4.58}$$

Substituting (4.58) in (4.16a), and solving the integrals explicitly, we have

$$\begin{aligned}\Omega_{i,k}(t) \leq &\left(k_p\frac{\delta_0}{\lambda_\varsigma}\left((t-t_{i,k})-\frac{1-e^{-\lambda_\varsigma(t-t_{i,k})}}{\lambda_\varsigma}\right)\right.\\ &\left.+\,k_v\frac{\delta_0}{\lambda_\varsigma}(1-e^{-\lambda_\varsigma(t-t_{i,k})})\right)e^{-\lambda_\varsigma t_{i,k}}.\end{aligned} \tag{4.59}$$

Since by hypothesis $\varsigma_\infty = 0$, we have

$$\varsigma(t) = \varsigma_0\, e^{-\lambda_\varsigma t} = \varsigma_0\, e^{-\lambda_\varsigma t_{i,k}} e^{-\lambda_\varsigma(t-t_{i,k})}. \tag{4.60}$$

Comparing (4.59) and (4.60), after dividing both sides by $e^{-\lambda_\varsigma t_{i,k}}$, we have that the triggering condition $\Omega_{i,k}(t) \geq \alpha/\nu_i\,\varsigma(t)$ requires

$$k_p \frac{\delta_0}{\lambda_\varsigma}\left((t-t_{i,k}) - \frac{1-e^{-\lambda_\varsigma(t-t_{i,k})}}{\lambda_\varsigma}\right) \tag{4.61a}$$

$$+ k_v \frac{\delta_0}{\lambda_\varsigma}(1-e^{-\lambda_\varsigma(t-t_{i,k})}) \geq \frac{\alpha}{\nu_i}\,\varsigma_0\,e^{-\lambda_\varsigma(t-t_{i,k})}, \tag{4.61b}$$

which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\Omega_{i,k}(t) \geq \alpha/\nu_i\,\varsigma(t)$ cannot be responsible for Zeno behavior. Now we produce a similar argument for the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$. Under Assumption 4.1 with $\delta_\infty = 0$, Lemma 4.2 holds, and (4.13) can be written as

$$\eta(t,\eta_0) = e^{-\lambda(t-t_0)}\eta_0 + \sqrt{N}\|B_{\mathcal{T}}\|e^{-\lambda t} \left(\frac{\varsigma_0(e^{(\lambda-\lambda_\varsigma)t} - e^{(\lambda-\lambda_\varsigma)t_0})}{\lambda-\lambda_\varsigma} + \frac{\delta_0(e^{(\lambda-\lambda_\delta)t} - e^{(\lambda-\lambda_\delta)t_0})}{\lambda-\lambda_\delta}\right) \tag{4.62}$$

for all $t \geq t_0$. Choosing $t_0 = 0$ and $\eta_0 = \|\xi(0)\|$, from (4.62) we have

$$\eta(t,\|\xi(0)\|) \leq e^{-\lambda t}\|\xi(0)\| + \sqrt{N}\|B_{\mathcal{T}}\|\left(\frac{\varsigma_0\,e^{-\lambda_\varsigma t}}{\lambda-\lambda_\varsigma} + \frac{\delta_0 e^{-\lambda_\delta t}}{\lambda-\lambda_\delta}\right) \tag{4.63}$$

for all $t \geq 0$. Since, by hypothesis, $\lambda_\varsigma < \lambda$, (4.63) can be further bounded by

$$\eta(t,\|\xi(0)\|) \leq \bar{\bar{\eta}}e^{-\lambda_\varsigma t}, \tag{4.64}$$

where we have denoted

$$\bar{\bar{\eta}} = \left(\|\xi(0)\| + \sqrt{N}\|B_{\mathcal{T}}\|\left(\frac{\varsigma_0}{\lambda-\lambda_\varsigma} + \frac{\delta_0}{\lambda-\lambda_\delta}\right)\right). \tag{4.65}$$

Substituting (4.64) in (4.53a) and (4.53b), and observing that $\varsigma(t_{i,k}) = \varsigma_0\,e^{-\lambda_\varsigma t}$, we have

$$\|u_{i,k}\| \leq (\beta_i\bar{\bar{\eta}} + \varsigma_0)e^{-\lambda_\varsigma t_{i,k}}, \tag{4.66a}$$

$$\|u_{j,h_j}\| \leq (\beta_j\bar{\bar{\eta}} + \varsigma_0)e^{-\lambda_\varsigma t_{i,k}}. \tag{4.66b}$$

Reasoning similarly for $\mu_j(t)$ for $t > t_{j,h_j+1}$, we have

$$\mu_j(t) \leq (\beta_j\bar{\bar{\eta}} + \varsigma_0)e^{-\lambda_\varsigma t_{i,k}}. \tag{4.67}$$

Also, note that

$$\|v_{j,h_j} - v_{i,k}\| \leq \|\xi(t_{i,k})\| \leq \bar{\bar{\eta}}e^{-\lambda_\varsigma t_{i,k}}. \tag{4.68}$$

Finally, note that (4.52) remains valid under the hypotheses of this lemma. Therefore, substituting (4.52), (4.66a), (4.66b), (4.67) and (4.68) into (4.16b), observing

that $\varsigma(t) = \varsigma_0\, e^{-\lambda_\varsigma t_{i,k}}$ and using the triangular inequality, we have

$$\begin{aligned}\sigma_{i,k}(t) \leq \Bigg(\Bigg(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j \bar{\bar{\eta}} + \varsigma_0 + \beta_i \bar{\bar{\eta}} + \varsigma_0) \Bigg) \\ (k_v(t - t_{i,k}) + (1/2)k_p(t - t_{i,k})^2) \\ + \Bigg(\sum_{j\in\mathcal{N}} w_{ij} \Bigg) \bar{\bar{\eta}} k_p(t - t_{i,k}) + \alpha\, \varsigma_0 \Bigg) e^{-\lambda_\varsigma t_{i,k}}.\end{aligned} \tag{4.69}$$

From (4.60) and (4.69), after dividing both sides by $e^{-\lambda_\varsigma t_{i,k}}$, we can see that the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$, implies

$$\begin{aligned}\Bigg(\Bigg(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j \bar{\bar{\eta}} + \varsigma_0 + \beta_i \bar{\bar{\eta}} + \varsigma_0) \Bigg) \\ (k_v(t - t_{i,k}) + (1/2)k_p(t - t_{i,k})^2) \\ + \Bigg(\sum_{j\in\mathcal{N}} w_{ij} \Bigg) \bar{\bar{\eta}} k_p(t - t_{i,k}) \Bigg) \geq (1 - \alpha)\, \varsigma_0\, e^{-\lambda_\varsigma (t - t_{i,k})},\end{aligned} \tag{4.70}$$

which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ cannot be responsible for Zeno behavior either. □

4.7 Proof of the main result

In this section, we use the partial results that we have developed in this chapter so far to establish a formal proof of Theorem 4.1.

Proof of Theorem 4.1. From Lemma 4.3, we know that, under the control law (4.5) and (4.6a) to (4.6c) and the scheduling rule (4.15) and (4.16) the hypotheses of Lemma 4.2 are satisfied.

If $\delta_\infty > 0$, we know from Lemma 4.4 that the closed-loop system does not exhibit Zeno behavior. Therefore, we can take $t \to \infty$ in (4.13) in Lemma 4.2, obtaining $\limsup_{t\to\infty} \|\xi(t)\| \leq \epsilon$, with ϵ given by (4.17).

If $\delta_\infty = 0$, $\varsigma_\infty = 0$ and $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$, we know from Lemma 4.5 that the closed-loop system does not exhibit Zeno behavior. Therefore, we can take again $t \to \infty$ in (4.13), obtaining $\limsup_{t\to\infty} \|\xi(t)\| \leq \epsilon$. But since $\delta_\infty = \varsigma_\infty = 0$, (4.17) evaluates to zero, and therefore $\lim_{t\to\infty} \xi(t) = 0$. □

4.8 Numerical simulations

In this section, two numerical simulations of the proposed control algorithm are presented, one for a scenario where practical convergence is reached, and one for

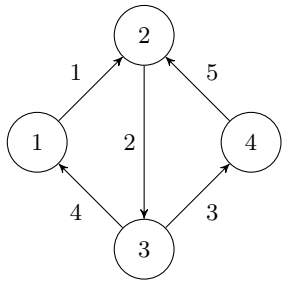

Figure 4.2: A graph with 4 nodes and 5 edges. The nodes and the edges are labeled with their indexes.

a scenario where asymptotic convergence is reached. For both simulations, we consider a multi-agent system made up of $N = 4$ agents with state in $\mathbb{R}^2$, which exchange information through a cloud repository according to the graph $\mathcal{G}$ illustrated in Figure 4.2, where all the edges are assigned unitary weights.

The assigned graph contains a spanning tree $\mathcal{T}$ made up of the first three edges. The corresponding reduced edge Laplacian is

$$R = \begin{bmatrix} 2 & 0 & -1 \\ -1 & 2 & 1 \\ 0 & -1 & 1 \end{bmatrix}. \tag{4.71}$$

The control gains are chosen as $k_p = 0.5$ and $k_d = 1.0$, which leads to $\lambda = -\max\{\mathrm{Re}(s) : \ s \in \mathrm{eig}(F_{e,r})\} = 0.5$ and $\|B_{\mathcal{T}}\| \simeq 2.45$. The disturbances are chosen as

$$d_i(t) = \delta(t) \begin{bmatrix} \cos(2\pi(i/N)t + 2\pi((N-i)/N)) \\ \sin(2\pi(i/N)t + 2\pi((N-i)/N)) \end{bmatrix}, \tag{4.72}$$

where $\delta(t)$ is defined by (4.3) with $\delta_0 = 0.2$, $\lambda_\delta = 0.45$, $\delta_\infty = 0.02$ in the first simulation, and $\delta_\infty = 0$ in the second simulation. It is easy to see that, with these parameters, Assumption 4.1 is satisfied. The threshold function is chosen as (4.8), with $\varsigma_0 = 5.0$, $\lambda_\varsigma = 0.4$, $\varsigma_\infty = 0.5$ for the first simulation, and $\varsigma_\infty = 0$ for the second simulation. Note that, with these choices, the first simulation scenario satisfies the hypotheses of Theorem theorem 4.1 for practical coordination, and the second simulation scenario satisfies the hypotheses of Theorem theorem 4.1 for asymptotic convergence. For the coefficient α that appears in (4.15), we choose $\alpha = 0.05$.

The results of the first simulation are illustrated in Figure 4.3. From Figure 4.3 it looks clear that the multi-agent system only achieves practical convergence, but the norm of the disagreement vector is significantly reduced. From Figure 4.3, we can also see that the cloud accesses do not accumulate; on the contrary, they seem

to become less frequent over time, which corroborates the result that the closed-loop system does not exhibit Zeno behavior. The results of the second simulation are illustrated in Figure 4.4. From Figure 4.4, we can see clearly that $\xi(t) \to 0$, which means that asymptotic convergence is reached. From Figure 4.4, we can also see that the cloud accesses do not accumulate even if the threshold function is converging to zero, which again corroborates the result that the closed-loop system does not exhibit Zeno behavior.

4.9 Summary

This chapter has addressed a cloud-supported self-triggered control problem for multi-agent coordination of a team of agents with second-order dynamics. Coordination has been achieved by having the agents asynchronously deposit and retrieve data on a cloud repository, rather than by inter-agent communication. Two control objectives have been considered, namely practical and asymptotic convergence. It has been shown that the proposed control strategy achieves practical convergence in the presence of unknown bounded persistent disturbances, and asymptotic convergence in the presence of unknown disturbances if they slowly vanish. Well-posedness of the closed-loop system has been proved by showing that there is a lower bound for the time interval between two consecutive accesses to the cloud. The proposed scheme can be adopted in all cases when direct communication among agents is interdicted, as illustrated in our motivating example of controlling a fleet of AUVs.

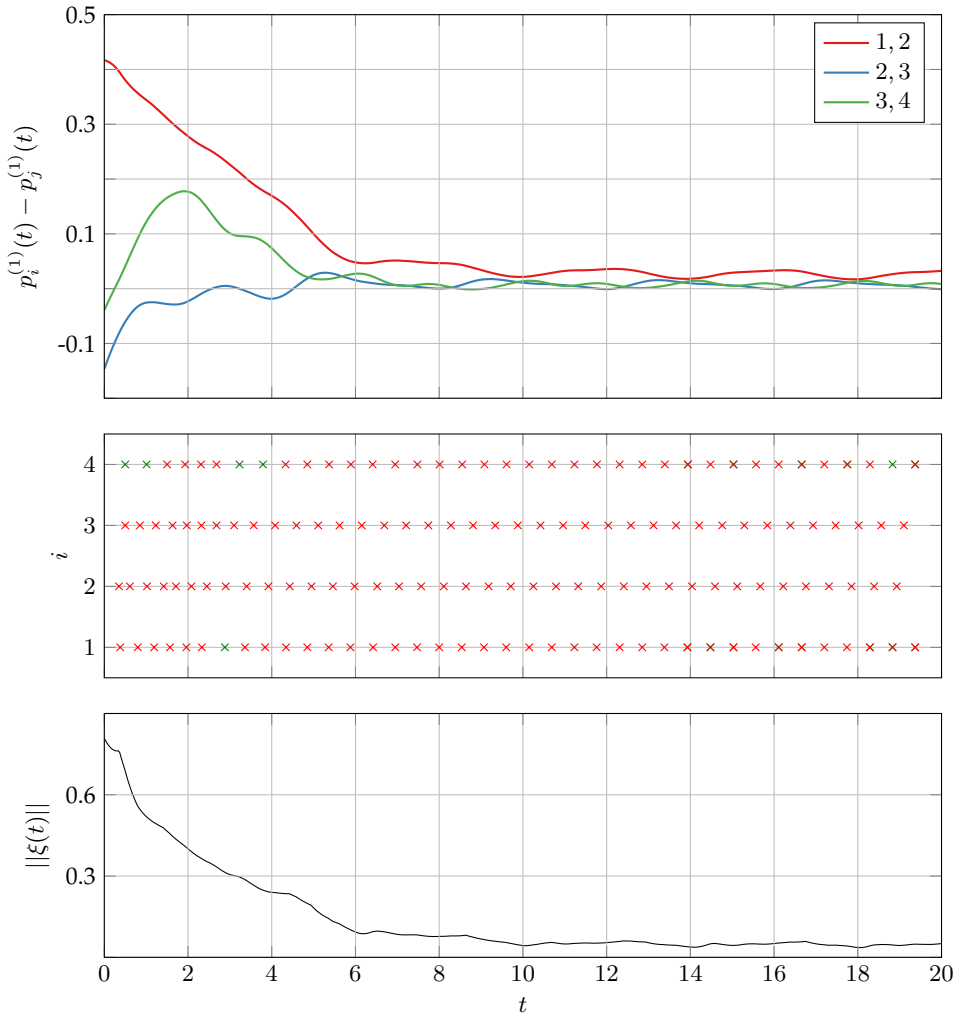

Figure 4.3: Simulation with persistent disturbances. Top: position mismatches across the edges (j, i) in the spanning tree over time. Middle: time instants when each agent accesses the cloud; a green cross denotes an access triggered by $\Omega_{i,k}(t) \geq \alpha/\nu_i\, \varsigma(t)$; a red cross denotes an access triggered by $\sigma_{i,k}(t) \geq \varsigma(t)$. Bottom: norm of the global disagreement vector $\xi(t)$ over time.

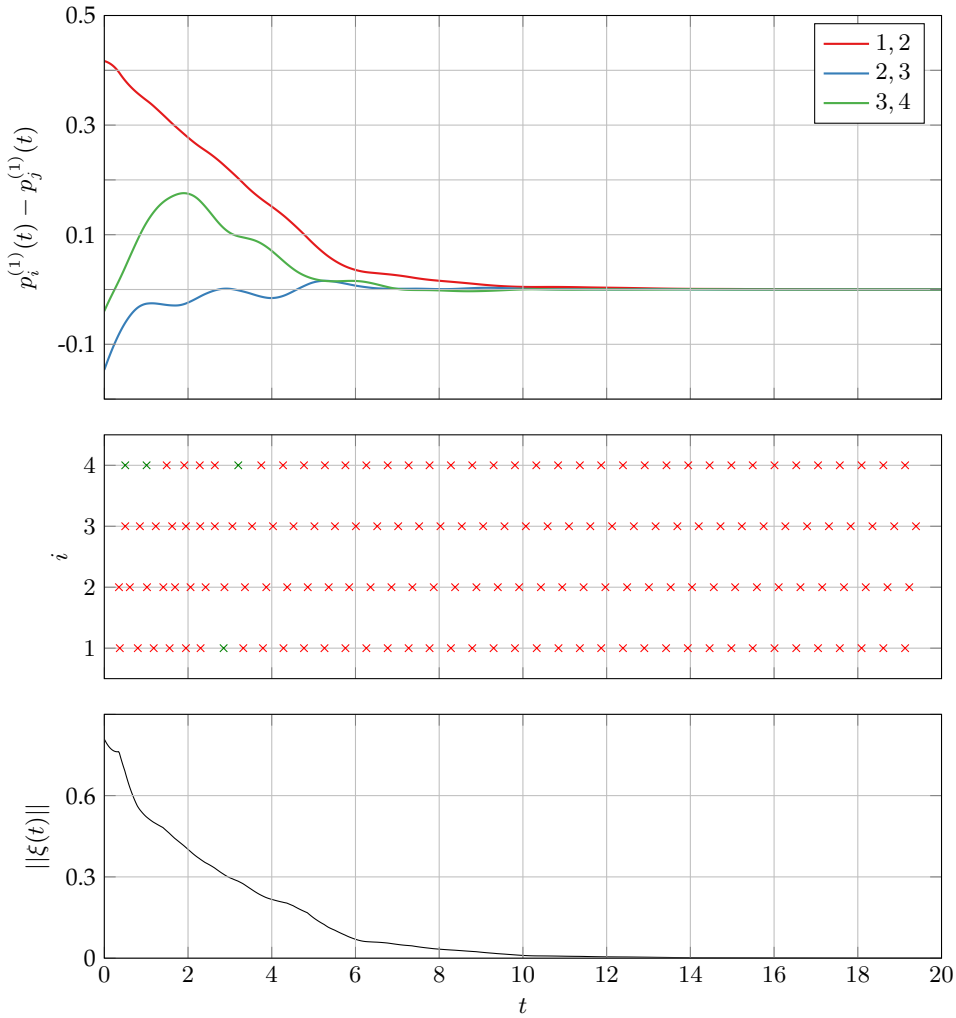

Figure 4.4: Simulation with asymptotically vanishing disturbances. Same subplots as in Figure 4.3.

Chapter 5

Cloud-supported circumnavigation

I fanciulli gridando
Su la piazzuola in frotta,
E qua e là saltando,
Fanno un lieto romore:
E intanto riede alla sua parca mensa,
Fischiando, il zappatore,
E seco pensa al dì del suo riposo.

G. Leopardi,
Il sabato del villaggio*, vv. 24–30.*

In this chapter, we study a particular application for the cloud-supported control design introduced in Chapter 4. Namely, we apply cloud-supported coordination to a problem of target circumnavigation.

The rest of this chapter is organized as follows. In Section 5.1, we review the existing related work and highlight the novel contributions offered in the chapter. In Section 5.2, we define the mathematical model used to describe the network of agents that are asked to perform the circumnavigation. In Section 5.3, we design an event-based rule to trigger the measurement of the bearing of the target, and we demonstrate that, under this rule, each agent converges to a circle centered at the target, with a desired distance from the target. In Section 5.4, we design a self-triggered control law to schedule the accesses of each agent to the cloud repository, and we demonstrate that, under this rule, the agents converge to a regular-polygon formation around the target. In Section 5.5, we validate our results by numerical simulation. In Section 5.6, we describe a preliminary experimental evaluation of the proposed algorithm. Section 5.7 concludes the chapter with a summary of the

results.

5.1 Introduction

The problem of tracking and circumnavigating a target with a network of autonomous agents finds numerous applications in mobile robotics. A first application is the surveillance of a building or structure with a team of aerial robots. In fact, in many cases, having the robots circumnavigate the target to survey yields a better coverage than deploying the robots to fixed positions around the target. Another application is monitoring an organism by surrounding it with a set of autonomous mobile sensing agents.

Circumnavigation of a target with a single agent has been studied, for example, in (Shames et al., 2012, Deghat et al., 2014), where the agent measures the bearing of the target and its distance from the target, respectively. The problem of steering a network of autonomous agents to a circling formation is the subject of a vast body of work, including, for example, (Marshall et al., 2004, 2006, Sepulchre et al., 2007, Kim and Sugie, 2007, Shames et al., 2011). Recently, these two problems have been merged, and the scenario where a network of autonomous agents is required to locate and circumnavigate a target with unknown location has been considered (Swartling et al., 2014).

The vast majority of the existing works on circumnavigation are based on the underlying assumption that each agent may perform measurements and/or exchange information with the other agents in a continuous-time fashion. However, in reality, both the measurements and the exchange of information happen through wireless communication channels with limited bandwidth capacity. Therefore, the frequency with which the agents may perform these actions is limited. To address these concerns, in this work we propose a control framework for multi-agent circumnavigation where bearing measurements and communication are event-triggered and self-triggered, respectively. Event-triggered and self-triggered control are used to reduce the amount of communication necessary to achieve a control task (Heemels et al., 2012). These schemes are applied to multi-agent systems to achieve the desired coordination while reducing the amount of information exchanged among different agents (Dimarogonas et al., 2012).

Moreover, instead of letting the agents communicate directly with each other, we let them transfer data over a shared information repository hosted on a cloud. The accesses of the agents to the cloud are scheduled according to a recursive rule such that, when an agent accesses the cloud, it also computes when it will access the cloud next. The use of a cloud repository in multi-agent systems has recently gained much research attention, because it presents several advantages with respect to direct communication among the agents. Several classical multi-agent coordination problems have recently been studied in a context where a cloud repository replaces

direct communication (Hale and Egerstedt, 2015, Bowman et al., 2016, Patel et al., 2016).

The effectiveness of the proposed algorithm is verified analytically and by numerical simulation. We also illustrate a setup for preliminary experimental evaluation of the proposed algorithm.

5.2 System model and problem statement

In this work, we consider a network of N autonomous vehicles modeled as planar, first-order integrators, described by

$$\dot{y}_i(t) = u_i(t),\ i \in \mathcal{N}, \tag{5.1}$$

where $y_i(t) \in \mathbb{R}^2$ is the position of the agent, and $u_i(t) \in \mathbb{R}^2$ is the decentralized control action exerted on the agent, thus having the dimensions of a velocity, and $\mathcal{N} = \{1, \dots, N\}$. The agents are required to locate and circumnavigate a target, whose position is denoted x, while forming a balanced circular formation around the target. We define the *counterclockwise angle* between two agents i and j as the angle $\beta_i^j(t)$ subtended at x by $y_i(t)$ and $y_j(t)$, evaluated counterclockwise from the $y_i(t)$ to $y_j(t)$. To simplify the notation, we let

$$\beta_i(t) := \beta_i^{i\%N+1}(t), \tag{5.2}$$

where $a\%b$, with $a \in \mathbb{N}$ and $b \in \mathbb{N}_{>0}$, denotes the remainder of the integer division of a by b. In other words, $\beta_i(t)$ denotes the counterclockwise angle between agent i and the agent with the consecutive index $(i+1)$ in a circular fashion. Finally, we denote $\varphi_i(t)$ the bearing of the target with respect to the position $y_i(t)$ of agent i. Namely, we set

$$\varphi_i(t) := \frac{x - y_i(t)}{\|x - y_i(t)\|}. \tag{5.3}$$

Note that, for any two agents i and j, we have $\beta_i^j(t) = \angle(\varphi_i(t), \varphi_j(t))$. The bearing vector is well defined if and only if $y_i(t) \neq x$. Therefore, we need to make sure that our control law guarantees that the agents do not travel indefinitely close to the target. Also, the initial positions of the agents must not coincide with the position of the target, and this is formalized in the following assumption.

Assumption 5.1. *For all agents $i \in \mathcal{N}$, we have $y_i(0) \neq x$.*

Our objective is to design a decentralized control action $u_i(t)$, with $i \in \{1, \dots, N\}$ such that

$$\lim_{t\to\infty} \|x - y_i(t)\| = D^\star,\ i \in \mathcal{N}, \tag{5.4a}$$

$$\lim_{t\to\infty}(\beta_i(t)-\beta_j(t))=0,\ \forall(i,j)\in\mathcal{N}\times\mathcal{N}, \tag{5.4b}$$

where $D^\star>0$ is a desired distance.

Lemma 5.1. *Let N distinct vectors $\varphi_1,\dots,\varphi_N\in\mathbb{R}^2$ be such that $\angle(\varphi_i,\varphi_{i+1})=\angle(\varphi_j,\varphi_{j+1})$ for any two $(i,j)\in\mathcal{N}\times\mathcal{N}$, where we have denoted $\varphi_{N+1}=\varphi_1$. Then, for all $i\in\mathcal{N}$, we have $\min_{j\neq i}\angle(\varphi_i,\varphi_j)=2\pi/N$.*

Proof. Let $\bar{\beta}=\angle(\varphi_i,\varphi_{i+1})$ and $\theta_i=\min_{j\neq i}\angle(\varphi_i,\varphi_j)$. Denote as ν_i the index j that attains the minimum $\angle(\varphi_i,\varphi_j)$. Then, we have $\angle(\varphi_i,\varphi_{i+1})=\bar{\beta}=\angle(\varphi_{\nu_i},\varphi_{\nu_i+1})$, and, consequently, $\angle(\varphi_{i+1},\varphi_{\nu_i+1})=\angle(\varphi_i,\varphi_{\nu_i})=\theta_i$. However, since $\varphi_{\nu_{i+1}}$ must precede φ_{ν_i+1} when proceeding counterclockwise from φ_i, we have $\theta_{i+1}=\angle(\varphi_{i+1},\varphi_{\nu_{i+1}})\leq\angle(\varphi_i,\varphi_{\nu_i})=\theta_i$, where we have denoted $\theta_{N+1}=\theta_1$. Since the indexes are circular, we must conclude that $\theta_i=\theta_j$ for all $(i,j)\in\mathcal{N}\times\mathcal{N}$. □

Note that, by Lemma 5.1, the control objective (5.4) implies that the agents tend to become equally spaced on the circle with center x and radius D^*.

To reach the control objective (5.4), we assume that the agents can measure the bearing of the target and that they can exchange data over a shared repository hosted on a cloud server. However, since the bearing measurements and the exchange of information over the cloud rely on wireless communication, we do not assume that they can be executed continuously. Instead, we model these communication instances as instantaneous events, that are triggered by appropriately designed conditions. We let $t_{i,k}$ denote the time when agent i measures the bearing of the target for the k-th time. Similarly, we let $\tau_{i,k}$ denote the time when agent i accesses the cloud repository for the k-th time.

The distributed control law that we propose takes the following form:

$$\omega_{i,k}=\kappa(\alpha+\hat{\beta}_{i,k}), \tag{5.5a}$$

$$u_i(t)=D^*\omega_{i,k}\hat{\varphi}_i(t)^\perp,\ t\in(\tau_{i,k},\tau_{i,k+1}), \tag{5.5b}$$

where κ and α are positive constants, while $\hat{\varphi}_i(t)$ and $\hat{\beta}_{i,k}$ are local estimates of $\varphi_i(t)$ and $\beta_i(t)$, respectively. As we shall see in the following, $\omega_{i,k}>0$ represents the angular speed with which agent i rotates around its current estimate of the position of the target.

For the control law (5.5) to be completely defined, we need to specify how the estimates $\hat{\varphi}_i(t)$ and $\hat{\beta}_{i,k}$ are computed. Roughly speaking, the estimates $\hat{\varphi}_i(t)$ of the bearing vectors are obtained from the bearing measurements, while the estimates $\hat{\beta}_{i,k}$ of the counterclockwise angles are obtained from the information exchanged over the cloud.

agent	1	2	...	N
last access	$\tau_{1,l_1(t)}$	$\tau_{2,l_2(t)}$	...	$\tau_{N,l_N(t)}$
bearing	$\hat{\varphi}_{1,l_1(t)}$	$\hat{\varphi}_{2,l_2(t)}$	...	$\hat{\varphi}_{N,l_N(t)}$
speed	$\omega_{1,l_1(t)}$	$\omega_{2,l_2(t)}$	...	$\omega_{N,l_N(t)}$
next access	$\tau_{1,l_1(t)+1}$	$\tau_{2,l_2(t)+1}$	...	$\tau_{N,l_N(t)+1}$

Table 5.1: Data contained in the cloud at a generic time instant $t \geq 0$. The i-th column corresponds to the latest packet uploaded by agent i.

Estimate of the bearing vectors

The estimates $\hat{\varphi}_i(t)$ of the bearing vectors are obtained as follows. For all $t \in (t_{i,k}, t_{i,k+1})$, we let

$$\hat{x}_i(t) = y_i(t_{i,k}) + D^* \varphi_i(t_{i,k}), \tag{5.6a}$$

$$\hat{\varphi}_i(t) = \frac{\hat{x}_i(t) - y_i(t)}{\|\hat{x}_i(t) - y_i(t)\|}. \tag{5.6b}$$

The estimation law (5.6) can be interpreted as follows: between two consecutive bearing measurements $\varphi_i(t_{i,k})$ and $\varphi_i(t_{i,k+1})$, agent i assumes that the target is located on the direction defined by the most recent bearing measurement $\varphi_i(t_{i,k})$, at a distance D^* from the position $y_i(t_{i,k})$ of the agent at the measurement time. Differentiating (5.6b) and using (5.1) and (5.5), we can write the dynamics of $\hat{\varphi}_i(t)$ as

$$\dot{\hat{\varphi}}_i(t) = \omega_{i,k} \hat{\varphi}_i(t)^\perp, \ t \in (\tau_{i,k}, \tau_{i,k+1}), \ t \notin \{t_{i,h}\}_{h \in \mathbb{N}}. \tag{5.7}$$

Note that (5.7) simply means that $\hat{\varphi}_i(t)$ rotates counterclockwise with constant angular speed $\omega_{i,k}$ during the interval $(\tau_{i,k}, \tau_{i,k+1})$. The time instants $t_{i,h}$ with $h \in \mathbb{N}$ are excluded because $\hat{\varphi}_i(t)$ is discontinuous over such instants.

Estimate of the counterclockwise angles

To define the estimates of the counterclockwise angles, we need first to define the pattern by which the agents exchange information over the cloud repository. To this aim, let $l_i(t)$ denote the cardinality of the most recent access to the cloud of agent i before time t. In other words, let

$$l_i(t) = \max\{k \in \mathbb{N} : \ \tau_{i,k} < t\}. \tag{5.8}$$

The information contained in the cloud repository at a generic time instant is illustrated in Table 5.1. From Table 5.1, we can see that each column contains

information about one agent. Namely, the i-th column contains: the time $\tau_{i,l_i(t)}$ of the most recent access of agent i to the cloud; the estimated bearing vector $\hat{\varphi}_i(\tau_{i,l_i(t)})$ of agent i at said time; the angular velocity $\omega_i(\tau_{i,l_i(t)})$ applied to agent i at said time, and; the time of the following access $\tau_{i,l_i(t)+1}$. Whenever agent i accesses the cloud repository, it downloads the information relative to agent $i\%N+1$, and computes the angular speed $\omega_{i,k}$ and the time $\tau_{i,k+1}$ of its next access; then, it uploads the quadruple $(\tau_{i,k}, \hat{\varphi}_i(\tau_{i,k}), \omega_i(\tau_{i,k}), \tau_{i,k+1})$. This quadruple overwrites the corresponding row in the repository, so that the cloud contains updated information about agent i. In this way, the amount of information contained in the repository does not grow over time, and the capacity of the repository can be proportional to the number of agents in the network. The estimates of the counterclockwise angles are generated as follows. Let

$$\hat{\beta}^j_{i,k} = \angle(\hat{\varphi}_i(\tau_{i,k}),\ \mathrm{rot}(\varphi_j(\tau_{j,l_j(\tau_{i,k})}), \omega_j(\tau_{j,l_j(\tau_{i,k})})(\tau_{i,k} - \tau_{j,l_j(\tau_{i,k})}))), \tag{5.9}$$

where $\mathrm{rot}(\phi, \theta)$ denotes the vector obtained rotating the vector ϕ clockwise by an angle θ. In particular, let $\hat{\beta}_{i,k} = \hat{\beta}^{i\%N+1}_{i,k}$.

Simply put, $\hat{\beta}_{i,k}$ is an estimate of $\beta_i(\tau_{i,k})$ based on the data available in the cloud. (In fact, $\hat{\beta}_{i,k} = \beta_i(\tau_{i,k})$ if the estimates of the bearing vectors contained in the cloud coincide with the actual bearing vectors at the access times.) Note that $\hat{\beta}_{i,k}$ can be computed by agent i using only information downloaded from the cloud at time $\tau_{i,k}$.

Remark 5.1. *In order to compute $\hat{\beta}_{i,k}$, agent i needs to download only the information related to agent $i\%N+1$. Therefore, even though the cloud stores information about all the agents, the pattern of the information exchange among the agents is distributed. More specifically, the information is exchanged as if the agents were connected on a graph where the edges are $\{(i\%N+1, i),\ i \in \mathcal{N}\}$. This type of graph is called a* directed ring. *A directed ring with N vertexes contains exactly N spanning trees, each thereof is obtained by removing a single edge from the ring.*

To conclude the definition of our control law, we must now give the rules that trigger the measurements of the bearing vector and the accesses to the cloud. In the following Sections 5.3 and 5.4, we illustrate appropriate scheduling rules to attain the control objective (5.4).

5.3 Triggering of the bearing measurements

Our control strategy prescribes that the generic agent i performs a measurement of the bearing vector whenever the estimated bearing vector $\hat{\varphi}_i(t)$ is orthogonal to the most recent bearing measurement $\varphi_i(t_{i,k})$. In other words, the bearing measurements are scheduled according to the following recursive rule:

$$t_{i,k+1} = \inf\{t \geq t_{i,k} :\ \angle(\hat{\varphi}_i(t), \varphi(t_{i,k})) \geq \pi/2\}. \tag{5.10}$$

In the following Theorem 5.1, we show that, under the scheduling rule (5.10), the position of the target $\hat{x}_i(t)$ estimated by each agent converges to the real position x of the target.

Theorem 5.1. *Consider a generic agent i with kinematics* (5.1) *and under the control law* (5.5)*. Let the bearing measurements be scheduled as prescribed by* (5.10)*. Then, under Assumption 5.1, we have $y_i(t) \neq x$ for all $t \geq 0$ and $\lim_{t\to\infty} \hat{x}_i(t) = x$. Moreover, the interval $t_{i,k+1} - t_{i,k}$ between two consecutive bearing measurements is lower-bounded by $(\pi/2)/(\alpha + 2\pi)$.*

Proof. First note that $\hat{\varphi}_i(t)$ rotates with angular speed $\omega_{i,k}$, which is upper-bounded by $\kappa(\alpha + 2\pi)$. Hence, the interval between two consecutive measurements is lower-bounded by $(\pi/2)/(\alpha + 2\pi)$. Let us consider the discrete time system obtained by integrating (5.1) over the time interval between two consecutive measurements $t_{i,k}$ and $t_{i,k+1}$ of the bearing vector $\varphi_i(t)$. Substituting in (5.1) the expression of $u_i(t)$ given in (5.5), and noting that $\alpha > 0$ by design, and $\beta_i(t)$ is nonnegative by definition, we have that, at any time instant, agent i rotates about its current estimate of the target with strictly positive (counterclockwise) angular velocity, and thus $t_{i,k+1}$ is finite for all $k \in \mathbb{N}$. Hence,

$$y_i(t_{i,k+1}) = \hat{x}_i(t_{i,k}) + (y_i(t_{i,k}) - \hat{x}_i(t_{i,k}))^{\perp}. \tag{5.11}$$

Substituting in (5.11) the expressions of $\hat{x}_i(t)$ and $\hat{\varphi}_i(t)$ in (5.6), we obtain

$$y_i(t_{i,k+1}) = y_i(t_{i,k}) + \varphi_i(t_{i,k})D^\star - \varphi(t_{i,k})^{\perp}D^\star. \tag{5.12}$$

Now, consider the estimation error

$$\tilde{x}_{i,k} := \hat{x}_i(t_{i,k+1}) - x, \tag{5.13}$$

and the distance between the agent and the target

$$z_{i,k} = y_i(t_{i,k}) - x. \tag{5.14}$$

From Figure 5.1, and leveraging (5.12), we can see that

$$\|z_{i,k+1}\| = \sqrt{\|\tilde{x}_{i,k}\|^2 + D^{\star 2}} \tag{5.15a}$$

$$\|\tilde{x}_{i,k}\| = \sqrt{\|\tilde{x}_{i,k}\|^2 + D^{\star 2}} - D^\star \tag{5.15b}$$

From (5.15a), we can see that, as long as $z_{i,0} \neq 0$, we have $z_{i,k} \neq 0$ for all $k \in \mathbb{N}$. Since under Assumption 5.1 we have $z_{i,0} = y_i(0) - x \neq 0$, we can conclude that $z_{i,k} = y_i(t_{i,k}) - x \neq 0$ for all $k \in \mathbb{N}$. Moreover, subtracting $\|\tilde{x}_{i,k}\|$ from both sides in (5.15b), we have

$$\|\tilde{x}_{i,k+1}\| - \|\tilde{x}_{i,k}\| = \sqrt{\|\tilde{x}_{i,k}\|^2 + D^{\star 2}} - (\|\tilde{x}_{i,k}\| + D^\star). \tag{5.16}$$

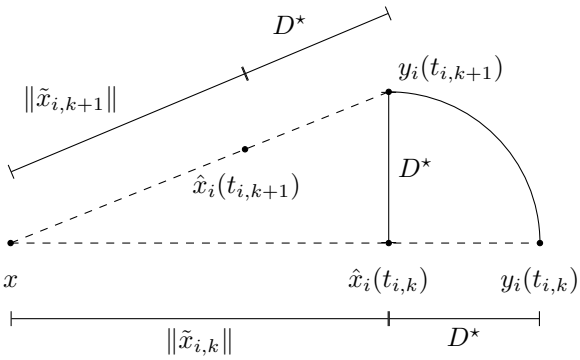

Figure 5.1: Illustration of (5.15).

From (5.16), we can see that $\|\tilde{x}_{i,k+1}\| - \|\tilde{x}_{i,k}\| \leq 0$, and that $\|\tilde{x}_{i,k+1}\| - \|\tilde{x}_{i,k}\| = 0$ if and only if $\|\tilde{x}_{i,k}\| = 0$. From LaSalle's invariance principle for discrete systems (LaSalle, 1986), we know that $\|\tilde{x}_{i,k}\|$ must converge to zero for $k \to \infty$. Hence, $\hat{x}_i(t)$ must converge to x, which concludes the proof. □

Theorem 5.1 ensures the control law in (5.5) is capable of achieving the control goal in (5.4a), that is, ensuring the agents circumnavigate the target. This is evident if the reader recalls that the agents are modeled as simple integrators, and thus $u_i(t)$ is the velocity of agent i. Then, if $\hat{x}_i(t)$ converges to x, from (5.3) and (5.6b) we have that $\hat{\varphi}_i$ tends to φ_i and thus from (5.5) we have that agent i rotates about x at a distance $D^\star$ with positive (counterclockwise) angular velocity. Note that this result is completely independent from the task of achieving a balanced circular formation about the target and thus from the estimates of $\beta_i(t)$, as Theorem 5.1 only assumes such estimates are nonnegative, which is true by definition.

5.4 Triggering of the access to the cloud

Having taken care of the circumnavigation task, we can now turn our attention to ensuring that the agents achieve a balanced circular formation; that is, the control goal in (5.4b). Achieving such goal is intrinsically tied to the selection of the rule that triggers accesses to the cloud.

We propose two different scheduling rules to trigger the communication between each agent and the cloud repository. The first rule is based on a similar rationale as the one developed in Chapter 4. Under this rule, it is possible to demonstrate analytically that the agents converge to the desired polygonal formation around the target, but the at the cost of performing possibly more communication than necessary. The second rule is heuristic, and its soundness is corroborated by sim-

ulation results. This rule has the advantage of resulting in a smaller number of communication instances with the cloud.

Rule A: Analytic scheduling

We start by noting that accessing the cloud to download the information to compute $\hat{\beta}_{i,k}$ only affects the goal of achieving a balanced circular formation. By Theorem 5.1, we know all agents converge to the circle of radius D^* centered in the position of the target, regardless of the particular values assumed by the angular speed $\omega_{i,k}$. Therefore, we can as well study the triggering of the cloud accesses assuming that all agents have already converged to the desired circle. In this case, from (5.5) we have that the planar velocities $u_i(t)$ of the agents are substantially tangential to the circle. Hence, we can reason directly on the angular speeds, which we have denoted $\omega_{i,k}$. Note that, since $\hat{\beta}_{i,k} \geq 0$ by definition, we have that each agent circumnavigates the target counterclockwise.

Without loss of generality, let $t \in (\tau_{i,k}, \tau_{i,k+1})$, and let $j = i\%N + 1$, $h = l_j(\tau_{i,k})$. With this notation, when agent i accesses the cloud at time $\tau_{i,k}$, it downloads the quadruple $(\tau_{j,h},\ \hat{\varphi}_j(\tau_{j,h}),\ \omega_{j,h},\ \tau_{j,h+1})$.

Note that, when all agents have reached the desired circle, we have $\hat{\varphi}_i(t) = \varphi_i(t)$ for all $i \in \mathcal{N}$. Hence, from (5.7), we have

$$\dot{\beta}_i(t) = \omega_{j,l_j(t)} - \omega_{i,k}, \tag{5.17}$$

which, using (5.5a), becomes

$$\dot{\beta}_i(t) = \kappa(\hat{\beta}_{j,l_j(t)} - \hat{\beta}_{i,k}). \tag{5.18}$$

Denoting $e_i(t) = \hat{\beta}_{i,l_i(t)} - \beta_i(t)$, (5.18) becomes

$$\dot{\beta}_i(t) = \kappa(\beta_j(t) - \beta_i(t) + e_j(t) - e_i(t)), \tag{5.19}$$

From (5.19), it is clear that $\beta_i(t)$ evolves according to a diffusive coupling (over a ring graph) with additive disturbances $e_i(t)$. In fact, (5.19) can be rewritten in vectorial form as

$$\dot{\beta}(t) = -\kappa L(\beta(t) + e(t)), \tag{5.20}$$

where we have denoted $\beta(t) = [\beta_1(t), \dots, \beta_N(t)]^\intercal$ and $e(t) = [e_1(t), \dots, e_N(t)]^\intercal$, and where L is the Laplacian of a ring graph with N nodes. Since a ring graph has a spanning tree, the variables $\beta_i(t)$ will reach consensus as long as the additive disturbances vanish quickly enough. To formalize this result, let $y_i(t) = \beta_{i+1}(t) - \beta_i(t)$ for $i \in \mathcal{N}\setminus\{N\}$, and let $y(t) = [y_1(t), \dots, y_{N-1}(t)]^\intercal$. In this way, we can write $y(t) = B_{\mathcal{T}}^\intercal \beta(t)$, where $B_{\mathcal{T}}$ is the part of the incidence matrix of the ring graph relative to the edges $(2,1), \dots, (N, N-1)$, and we can rewrite (5.20) as

$$\begin{aligned} \dot{y}(t) &= -\kappa B_{\mathcal{T}}^\intercal L(\beta(t) + e(t)) \\ &= -\kappa R y(t) - \kappa S e(t), \end{aligned} \tag{5.21}$$

where R denotes the reduced edge Laplacian of the ring graph and $S = B_{\mathcal{T}}^{\dagger} L$. The Laplace solution of (5.21) reads

$$y(t) = \mathrm{e}^{-\kappa R t}\, y(0) - \kappa S \int_0^t \mathrm{e}^{-\kappa R(t-\tau)}\, e(\tau) \mathrm{d}\tau. \tag{5.22}$$

Let us suppose that we find a scheduling that makes $e_i(t)$ vanish exponentially: $|e_i(t)| \leq \varsigma_i(t)$, where $\varsigma_i(t) = \varsigma_{i,0} e^{-\lambda_\varsigma t}$. In this case, we can take norms of both sides in (5.22) and use the triangular inequality to write

$$\|y(t)\| \leq \|\mathrm{e}^{-\kappa R t}\| \cdot \|y(0)\| + \kappa \|S\| \cdot \|\varsigma_0\| \int_0^t \|\mathrm{e}^{-\kappa R(t-\tau)}\|\, \mathrm{e}^{-\lambda_\varsigma \tau}\, \mathrm{d}\tau, \tag{5.23}$$

where $\varsigma_0 = [\varsigma_{1,0}, \ldots, \varsigma_{N,0}]^\intercal$. Since $-R$ is Hurwitz, (5.23) implies that $y(t)$ converges to zero, which means that all $\beta_i(t)$ converge to the same value. Our conclusion can be formalized as the following Lemma.

Lemma 5.2. *Let $\beta(t)$ evolve as by (5.20), and let $|e_i(t)| \leq \varsigma_i(t)$, where $\varsigma_i(t) = \varsigma_{i,0}\, \mathrm{e}^{-\lambda_\varsigma t}$ and $\varsigma_{i,0}$, λ_ς are positive constants. Then, $\beta_i(t) - \beta_j(t)$ converges to zero for any pair $(i, j) \in \mathcal{N} \times \mathcal{N}$.*

Proof. Follows from (5.23) observing that $y_i(t) = B_{\mathcal{T}}^{\intercal} \beta(t)$. □

Remark 5.2. *In order to evaluate the right-hand side of (5.23), the agents need to have access to an upper bound of $\|y(0)\|$. Such bound could be given, for example, in terms of a bounding box for the initial positions of the agents.*

Now we need to formulate a scheduling rule that guarantees that $|e_i(t)| \leq \varsigma_i(t)$. To this aim, we can find an upper bound for $|e_i(t)|$ and schedule a cloud access whenever that upper bound is about to reach $\varsigma_i(t)$. The recursive rule to schedule the accesses to the cloud takes into account that part of the error $e_i(t)$ arises from the possible existence, in the time interval $(\tau_{i,k}\ \tau_{i,k+1})$, of a smaller interval $(\tau_{j,h+1}\ \tau_{i,k+1})$ in which the angular speed of agent j is different from $\omega_{j,h}$ downloaded by agent i from the cloud. This indeed happens if $\tau_{j,h+1} \in (\tau_{i,k},\ \tau_{i,k+1})$.

Recalling that $e_i(t) = \beta_{i,k} - \beta_i(t)$, and integrating (5.18), we have

$$\begin{aligned} e_i(t) &= \int_{t_{i,k}}^{t} (\omega_{i,k} - \omega_{j,l_j(\tau)}) \mathrm{d}\tau \\ &= (t - \tau_{i,k})\omega_{i,k} + (\min\{t, \tau_{j,h+1}\} - \tau_{i,k})\omega_{j,h} + \int_{\tau_{j,h+1}}^{\max\{t, \tau_{j,h+1}\}} \omega_{j,l_j(\theta)} \mathrm{d}\theta. \end{aligned} \tag{5.24}$$

Observing that $\omega_{j,l_j(\theta)} = \kappa\hat{\beta}_{j,l_j(\theta)} = \beta_j(\theta) + e_j(\theta)$, we can use (5.24) to write an upper bound for $e_i(t)$ as

$$|e_i(t)| \leq (t - \tau_{i,k})\omega_{i,k} + (\min\{t, \tau_{j,h+1}\} - \tau_{i,k})\omega_{j,h} + \int_{\tau_{j,h+1}}^{\max\{t,\tau_{j,h+1}\}} \mu_j(\theta)\mathrm{d}\theta, \tag{5.25}$$

where

$$\mu_j(\theta) = \kappa(\eta(t) + \varsigma_j(t)), \tag{5.26}$$

with $\eta(t)$ denoting the right-hand side of (5.23). Note that the right-hand side of (5.25) can be computed by agent i when it accesses the cloud at time $\tau_{i,k}$, by using the data downloaded from the cloud together with some limited knowledge about the initial conditions of the network. Hence, the desired upper bound for $e_i(t)$ is given by (5.25). If we denote as $\sigma_{i,k}(t)$ the right-hand side of (5.25),

$$\sigma_{i,k}(t) = (t - \tau_{i,k})\omega_{i,k} + (\min\{t, \tau_{j,h+1}\} - \tau_{i,k})\omega_{j,h} + \int_{\tau_{j,h+1}}^{\max\{t,\tau_{j,h+1}\}} \mu_j(\theta)\mathrm{d}\theta, \tag{5.27}$$

then, the rule to schedule the accesses to the cloud is defined by:

$$\tau_{i,k+1} = \inf\{t > \tau_{i,k} : \sigma_{i,k}(t) \geq \varsigma_i(t)\}. \tag{5.28}$$

Broadly speaking, (5.28) indicates that the next access to the cloud performed by agent i is scheduled as soon as $\sigma_{i,k}(t)$ reaches the threshold $\varsigma_i(t)$. The rationale behind (5.28) is that $\sigma_{i,k}(t)$ constitutes an upper bound for $e_i(t)$ that agent i can compute at time $\tau_{i,k}$. Hence, by ensuring that $\sigma_{i,k}(t) \leq \varsigma_i(t)$, one automatically ensures that $e_i(t)$ converges to zero exponentially, with a rate of at least λ_ς. This result is formalized in the following Lemma 5.3.

Lemma 5.3. *Let $\beta_i(t)$ evolve as by (5.18), and let the cloud accesses $\tau_{i,k}$ be scheduled as per (5.28). Then, $e_i(t) = \hat{\beta}_{i,k} - \beta_i(t)$ converges to zero for all $i \in \mathcal{N}$, and, consequently, $\beta_i(t) - \beta_j(t)$ converges to zero for all $(i, j) \in \mathcal{N} \times \mathcal{N}$. Moreover, the interval $\tau_{i,k+1} - \tau_{i,k}$ between two consecutive accesses to the cloud performed by the same agent is lower-bounded by a positive constant.*

Proof. Follows by observing that $\sigma_{i,k}(t)$ is an upper bound for $|e_i(t)|$, and that under (5.28) one has $\sigma_{i,k}(t) \leq \varsigma_i(t)$. The result that the interval $\tau_{i,k+1} - \tau_{i,k}$ is lower-bounded by a positive constant is proven in a similar way as in Lemma 4.5, and therefore a formal proof is omitted here. □

Rule B: Heuristic scheduling

To reduce the communication between the agents and the cloud, we propose the following heuristic rule to trigger the cloud accesses:

$$\tau_{i,k+1} := \inf\{t > \tau_{i,k} : \ (t - \tau_{i,k})\omega_{i,k} = \hat{\beta}_{i,k}\} \tag{5.29}$$

Roughly speaking, this rule prescribes that each agent i accesses the cloud whenever it sweeps, around its current estimate of the target $\hat{x}_i(t)$, an angle equal to $\hat{\beta}_{i,k}$.

5.5 Numerical simulations

Comparison between Rule A and Rule B

To demonstrate the effectiveness of our approach, for each of the two rules proposed in Section 5.4, we carried out extensive numerical simulations involving a set of $N = 5$ agents. Specifically, we performed 100 simulations of the duration of $T = 25s$ each. As the asymptotic convergence of the agents to the circle of radius $D^\star$ centered in x is guaranteed regardless of the rule triggering the accesses to the cloud, in this set of numerical experiments, we assumed the agents have already converged onto the circle surrounding the target and started all simulations accordingly. Therefore, we focused on their angular speed $\omega_{i,k} = \kappa(\alpha + \hat{\beta}_{i,k})$ and, throughout our numerical campaign, we have set $\kappa = 1$ and $\alpha = 0.5$. To ensure the comparison between Rule A and Rule B is independent of the initial conditions, we have randomly selected a set of 100 different values of the vector $y(0)$ and assigned the i-th initial condition to both the i-th simulation performed under Rule A, and the i-th simulation performed under Rule B. As for the parameters of Rule A, we have $\gamma_i = 1$ $i = 1, \ldots, 4$, $\gamma_5 = 2$, and $\lambda = \min(\mathrm{Re}(\mathrm{eig}(R))) = 0.69$. Moreover, we have set $\lambda_\varsigma = 0.68$, and $\varsigma_{i,0} = 10$ for all $i \in \mathcal{N}$.

In all simulations performed under Rule B we have observed convergence of the angles β_i $i = 1, \ldots, n$, in a time t_c if t_c computed as the first time instant such that an agent accesses the cloud and $|\beta_i(t) - 2\pi/N| \leq 0.01(2\pi/N)$. Having provided numerical evidence demonstrating the effectiveness of Rule B in allowing the agents to achieve the control goal in (5.4b), we next compare the performances of the two rules regulating the accesses to the cloud. We start by highlighting that, as is the case for the two simulations with identical initial conditions shown in Figures 5.2 and 5.3, Rule A ensures faster convergence than Rule B. Namely, we find that the observed average convergence time under Rule A is $6.32s$, while under Rule B it is 14.98. These results confirm that the observed difference in the convergence time is statistically significant. Conversely, we find that under Rule A, the total number of accesses to the cloud before convergence is achieved averages at 316 accesses against the average of 103 observed under Rule B. Again, we find that this difference is statistically significant. Moreover, we observe that the frequency of the total accesses to the cloud under Rule A averages to 50Hz, while the frequency under Rule B to 7Hz. Finally, as expected, we observe a substantial difference in the frequency of the accesses after convergence is achieved. Namely, under Rule A such frequency averages at 203Hz, while under Rule B it averages at 7Hz.

To summarize, our numerical investigation confirms that

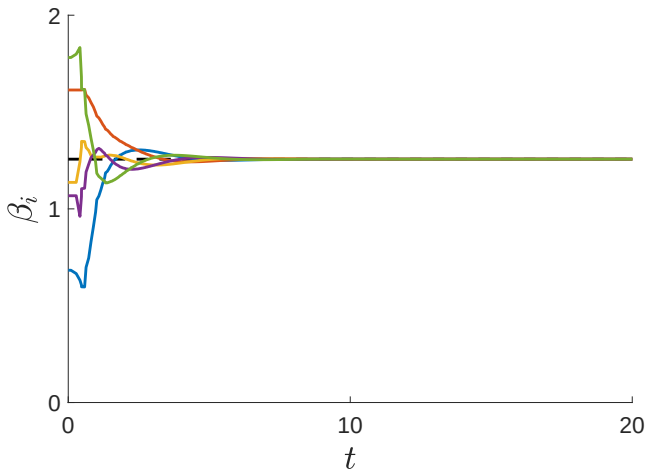

Figure 5.2: Plot of $\beta_i(t)$ $i = 1, \ldots, 5$ for a simulation performed under Rule A. The black dashed line denotes the target value for $\beta_i(t)$, that is $2\pi/5$.

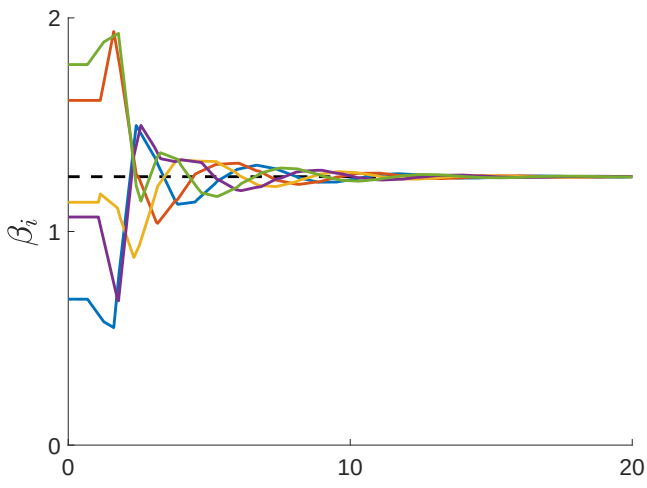

Figure 5.3: Plot of $\beta_i(t)$ $i = 1, \ldots, 5$ for a simulation performed under Rule B. The black dashed line denotes the target value for $\beta_i(t)$, that is $2\pi/5$.

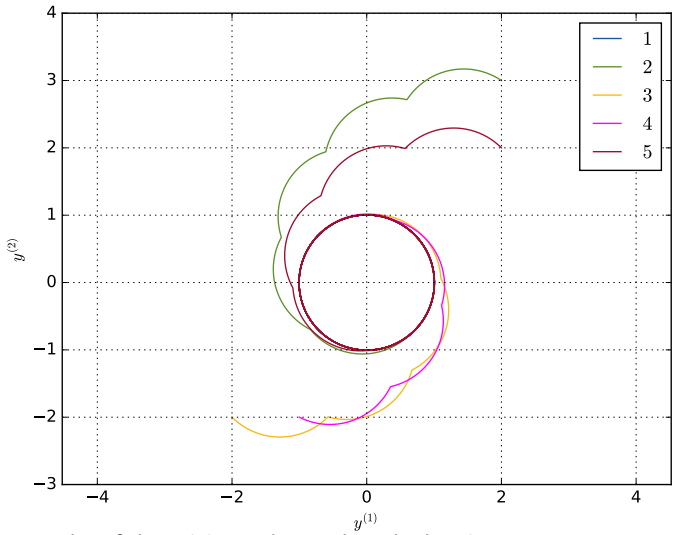

Figure 5.4: Results of the ROS simulation described in Section 5.5: trajectories of the agents.

(a) Rule A allows to achieve convergence faster than Rule B;

(b) Rule B allows to substantially reduce the frequency at which the agents access the cloud, thus proving more efficient in achieving the control goal in (5.4b) .

Robot Operating System (ROS) implementation

To demonstrate the use of the proposed control algorithm when the agents have to approach the target, we have run a simulation in the ROS environment, where each agent is simulated as a different ROS node. The interested reader may find a description of ROS in Quigley et al. (2009).

Each agent interacts with an additional ROS node that represents the cloud repository. The communication with the cloud occurs by means of ROS messages and services. In this simulation, we consider $N = 5$ agents, with $\alpha = 0.4$ and $\kappa = 0.2$, and the cloud accesses are scheduled according to Rule B. The initial conditions are $y_1(0) = [1, 0]^\intercal$, $y_2(0) = [2, 3]^\intercal$, $y_3(0) = [-2, -2]^\intercal$, $y_4(0) = [-1, -2]^\intercal$, and $y_5(0) = [2, 2]^\intercal$. The results of the simulation are summarized in Figures 5.4 to 5.6, where we can see that the agents converge to the desired circle around the target while reaching a regular formation, and that each agent accesses the cloud at approximately 0.3Hz. Although the ROS implementation requires some overhead, it presents the following significant advantages: as each agent is implemented as a ROS node, the distributed nature of the algorithm is reproduced in the simulation; the code used to implement the controller in the simulation can be re-used tout-court in the experimental implementations.

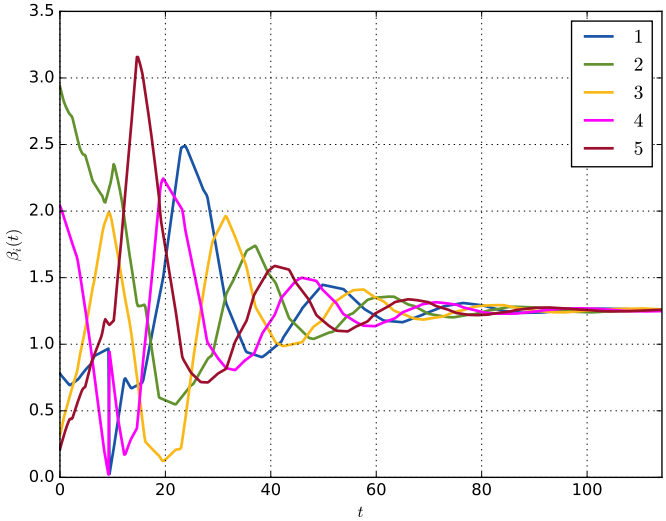

Figure 5.5: Results of the ROS simulation described in Section 5.5: angles $\beta_i(t)$.

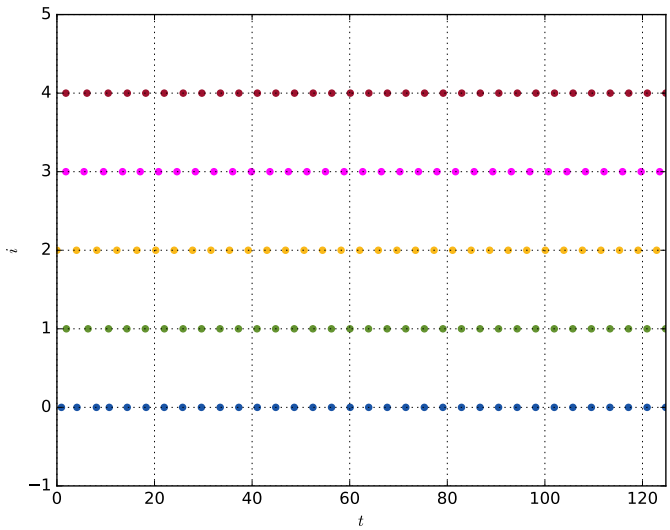

Figure 5.6: Results of the ROS simulation described in Section 5.5: cloud access times for each agent.

5.6 Preliminary experimental evaluation

In order to validate the proposed control framework, we have performed some experiments in the flying arena of the Smart Mobility Lab at KTH. In the experiments, the networked agents are three Bitcraze CrazyFlie (CF) (`bitcraze.io`), which are open-source and open-hardware nanoquadcopters. We have implemented the control and communication features in ROS, exploiting its modularity and robot-aimed tools. We have used a custom USB radio dongle called CrazyRadio (CR) through which we sent setpoints consisting in thrust and attitude (roll, pitch and yaw angles); these were then transformed into commands for the four propellers by an on-board micro-controller. The real-time positions and orientations of quadcopters are tracked by a motion capture system with an update rate of 100Hz, capable of detecting the position of some reflective spherical markers that we applied on the quadcopters.

The logical structure of the control architecture is described in Figure 5.7. As a preliminary implementation, we integrate the velocity command $u_i(t)$ generated by the controller according to (5.5), producing a certain number of intermediate goal positions that are fed to a PID flight controller, as shown in Figure 5.7.

In this experiments, different scheduling rules than Rule A and Rule B were used to determine the times when the agents would access new information about the other agents. In fact, this setup should be considered a preliminary testbed for the proposed control infrastructure, rather than an exact implementation of the proposed control algorithm.

Convergence was achieved in spite of the limitations of the experimental setup, whose improvement will be the subject of future work. In particular, the control strategy designed so far uses predictions of the angular speed of the agents, causing convergence to be affected by any real-world deviation of the trajectories from the predicted ones. Moreover, the PID controller used on-board as a flight controller could not always cope with the unmodeled dynamics leading to possible instabilities that rendered the experimental results worse that the numerical ones.

A video of the execution of a representative experiment is reachable on Zenodo (Mariniello et al., 2017), and a snapshot is shown in Figure 5.8.

5.7 Summary

In this chapter, we have proposed a cloud-supported control framework for multi-agent circumnavigation missions. We have proposed a scenario where the agents have limited communication capabilities, and, to attain the desired circumnavigation, they may intermittently measure the bearing of the target to circumnavigate and intermittently access a shared information repository hosted on a cloud. We

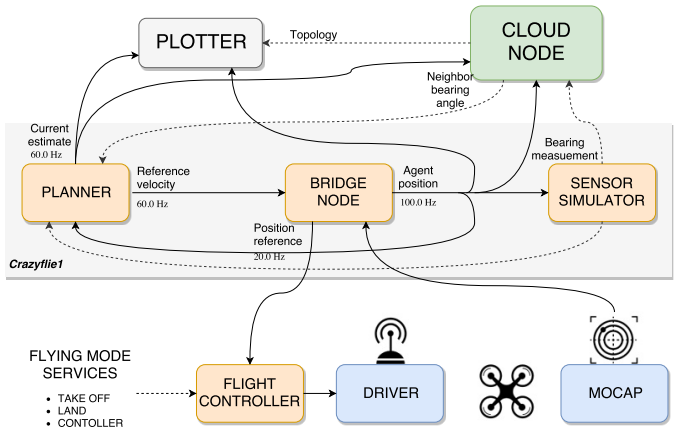

Figure 5.7: Logical structure of the implemented control solutions for experiments. The dashed arrows correspond to ROS services, the continue ones correspond to ROS topics. The colors are used as follows: the orange nodes represent features that are replicated for each agent of the network; the green node is the cloud node that is able to get informations from every agent of the network; the plotter, in white color, gets informations both about the agents and about the designed task, and represents the interface for the human user; the blue nodes are related to the actuation of the control and to the acquisition of the feedback from the ground.

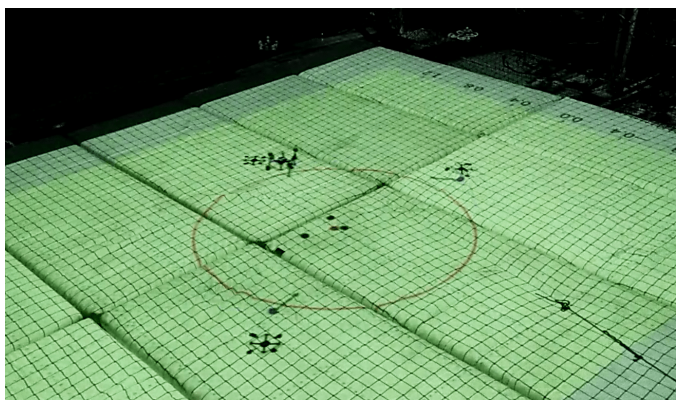

Figure 5.8: Execution of an experiment consisting in a localization and circumnavigation task made by a network of three CF in the Smart Mobility Lab. On the ground, we can see the projection of: the target position, as a red dot, in the center of the arena; the agents position, represented as blue spots; the agent-neighbor relations consisting in green arrows pointing from the agent towards the neighbor; the desired circumference, plotted as a red circle.

have designed event-triggered and self-triggered rules to schedule the bearing measurements and the cloud accesses performed by the agents. We have investigated how the proposed scheduling rules attain the desired circumnavigation objective, both analytically and with simulations.

Chapter 6

Event-triggred coverage control

Poi quando intorno è spenta ogni altra face,
E tutto l'altro tace,
Odi il martel picchiare, odi la sega
Del legnaiuol, che veglia
Nella chiusa bottega alla lucerna,
E s'affretta, e s'adopra
Di fornir l'opra anzi il chiarir dell'alba.

G. Leopardi,
Il sabato del villaggio, *vv. 31–37.*

In this chapter, we move on to study a substantially different coordination objective than those considered in Chapters 3 to 5. Namely, we consider the objective to deploy a network of autonomous sensing agents within an assigned environment in such a way that the collective perception of the environment attained by the agents is improved according to a specified measure.

The rest of this chapter is organized as follows. In Section 6.1, we review the existing related work and highlight the novel contributions offered in the chapter. In Sections 6.2 and 6.3, we define the abstract model that is used to describe the surveillance scenario under investigation. In Section 6.4, we give our generalized definition of Voronoi tessellation, that is later used to describe formally the convergence properties of the proposed coverage algorithm. In Section 6.5, we describe the controller used to steer the motion of a single agent, when no communication with the other agents occurs. In Section 6.6, we describe the information exchanged by different agents, and we formalize the event-triggered mechanism by which the communication instances are initiated. In Section 6.7, we describe the distributed

controller that implements the proposed coverage algorithm, and we derive the convergence properties of this controller mathematically. In Section 6.9, we describe two numerical simulations that validate the proposed coverage algorithm. In Section 6.10, we describe two preliminary experimental evaluations of the proposed algorithm. Section 6.11, concludes the chapter with a summary of the results.

6.1 Introduction

A wide variety of applications involve collecting information in hazardous environments, which makes it desirable to delegate such missions to a team of autonomous agents with sensing capabilities. Therefore, in the last few decades, a lot of research interest has been devoted to the problem of autonomous deployment of robot teams in an assigned space (Cortes et al., 2004, Martinez et al., 2007, Kwok and Martinez, 2010, Zhong and Cassandras, 2011, Le Ny and Pappas, 2013). Typically, the goal is to design a distributed algorithm that gradually drives the agents to a spatial configuration such that the team's collective perception of the environment is optimized according to some criterion. This problem is commonly known as the *coverage* problem, and it is often approached using Voronoi tessellations and the Lloyd algorithm. The classical reference for the coverage problem is Cortes et al. (2004). The original formulation of the Lloyd algorithm is found in Lloyd (1982), while a modern treatment can be found in Du et al. (1999).

The majority of the existing work on coverage considers agents with omnidirectional perception of the surrounding environment. Recently, agents with anisotropic sensing patterns (Stergiopoulos and Tzes, 2013, Stergiopoulos et al., 2015) as well as vision-based sensing patterns (Kantaros et al., 2015) have been considered.

Dynamic versions of the coverage problem have also been studied, where the agents do not converge to fixed positions, but navigate the environment until a satisfactory level of coverage has been reached. This problem is commonly known as *effective* or *dynamic coverage*. One of the first references for the effective coverage problem is Hussein and Stipanovic (2007). A vision-based version of effective coverage is studied in Panagou et al. (2016). We shall consider a problem similar to effective coverage in Chapter 7, where the described formalism is employed to model the mission of inspecting a 3D structure with a network of sensing agents.

Information exchange among the sensing agents constitutes one of the major challenges in the real-world implementation of coverage algorithms. For this reason, a gossip-based communication strategy for coverage is studied in Bullo et al. (2012). In some coverage missions, it is convenient to abstract the environment into a finite set of points (Durham et al., 2012), which may either correspond to a sparse set of points of interest, or to a discretized approximation of the environment itself.

In this chapter, we consider a coverage problem where the idea of discretization of

the environment under observation (Durham et al., 2012) is conjugated with the use of generalized sensing patterns for the agents (Panagou et al., 2016), as well as with the idea of Voronoi tessellations, which is opportunely redefined according to the considered sensing patterns. To the best of our knowledge, Voronoi tessellations for anisotropic sensing have only been considered in Gusrialdi et al. (2008), but only for elliptical footprints and under continuous environments. Instead, we apply vision-based sensing patterns over discretized environments. Using our notion of Voronoi tessellation, we are able to define a novel distributed algorithm for coverage, where communication is limited, pairwise, intermittent and asynchronous.

Unlike in the classical works on coverage control, continuous computations to partition the environment into time-varying Voronoi cells are not necessary in this framework. In fact, inspired by (Durham et al., 2012), we abstract the structure to survey into a finite set of landmarks, so that the area assigned to one agent is defined by the set of the landmarks assigned to that agent.

Using the hybrid system model developed in Goebel et al. (2012), we show formally that our algorithm makes the sensing agents converge to a configuration corresponding to a Voronoi tessellation, while a measure of the coverage attained by the team increases monotonically. We also propose an extension of the algorithm to avoid collisions among the agents as well as with other objects nearby. The algorithm is validated by means of numerical simulations and experiments with aerial robotic platforms.

6.2 System model and problem statement

We consider a set of N mobile sensors, indexed as $1, \dots, N$. The ith sensor has pose $T_i(t) \in \mathrm{SE}(3)$. The position and orientation of the ith sensor are denoted $p_i(t) \in \mathbb{R}^3$ and $R_i(t) \in \mathrm{SO}(3)$ respectively. If the orientation is expressed as a rotation matrix, and the pose is expressed as a homogeneous transformation matrix, then one has

$$R_i(t) = \begin{bmatrix} x_i(t) & y_i(t) & z_i(t) \end{bmatrix}, \tag{6.1}$$

$$T_i(t) = \begin{bmatrix} R_i(t) & p_i(t) \\ 0_3^\intercal & 1 \end{bmatrix}, \tag{6.2}$$

where $x_i(t)$, $y_i(t)$ and $z_i(t)$ form an orthonormal right-handed frame. However, the specific representation of the pose is not relevant to our purposes, and we write simply $T_i(t) = (p_i(t), x_i(t), y_i(t), z_i(t))$ to mean that the pose of a sensor aggregates its position and orientation. We assume that, for each sensor, it is possible to control the position and the orientation independently. Namely, we let

$$\dot{p}_i(t) = v_i(t), \tag{6.3a}$$

$$\dot{\xi}_i(t) = -\operatorname{Skew}(\xi_i(t))\omega_i(t), \quad \xi_i \in \{x_i, y_i, z_i\}, \tag{6.3b}$$

where $v_i(t)$ and $\omega_i(t)$ are control variables and are called, respectively, the *linear velocity* and the *angular velocity* of the sensor. Here, Skew denotes the *skew operator*. We recall that the skew operator corresponds to the cross product, in the sense that, for any $u, v \in \mathbb{R}^3$, we have $u \times v = \mathrm{Skew}(u)v$.

The environment that the sensors are required to cover is abstracted into a finite set of *landmarks*. A landmark represents a point or small area of interest within a domain. We consider a set of M landmarks indexed as $1, \ldots, M$. Like a sensor, a landmark is characterized by its pose, with the difference that, in this work, the pose of a landmark is assumed to be constant. The pose of the jth landmark is denoted $L_j \in \mathrm{SE}(3)$. The position and orientation are denoted q_j and Q_j respectively, with $Q_j = \begin{bmatrix}\phi_j & \chi_j & \psi_j\end{bmatrix}$, and we write $L_j = (q_j, \phi_j, \chi_j, \psi_j)$.

Our objective is to find a distributed controller that makes the sensors arrange themselves in a configuration such that their collective perception of the landmarks is improved. To this aim, we need to derive a model of the perception of the landmarks attained by the sensors.

We let the sensing capabilities of the sensors be described by a function $f : \mathrm{SE}(3)^2 \to \mathbb{R}$. This function is called the *footprint* of the sensors. Namely, $f(T_i(t), L_j)$ is a measure of how well the ith sensor perceives the jth landmark, with higher values corresponding to a better perception. For simplicity, in this work we assume that all the sensors have the same footprint; however, the results generalize easily to the case that different sensors have different footprints. A footprint needs to satisfy the following technical assumptions.

Assumption 6.1. *The sensor footprint f is continuously differentiable with respect to p_i, x_i, y_i and z_i. Moreover, the footprint is radially unbounded from below; i.e.,*

$$\lim_{\rho \to \infty} \sup_{\|p_i\| = \rho} f(T_i, L) = -\infty \tag{6.4}$$

for any $L \in \mathrm{SE}(3)$.

This formalism is more general than most of those found in the literature, because different sensor models can be captured by choosing the sensor footprint opportunely. For example, the sensing agents considered in Cortes et al. (2004) have omnidirectional perception, and their sensing capabilities degrade proportionally to the squared distance between the sensor and the perceived point, regardless of the orientations. This model can be captured with our framework by setting $f(T_i(t), L_j) = -\|p_i(t) - q_j\|^2$.

It is reasonable to assume that the sensor footprint only depends on the relative pose of the landmark with respect to the sensor. In this case, we have $f(T_i(t), L_j) = \tilde{f}(T_i(t)^{-1} L_j)$, where $T_i(t)^{-1} L_j$ is the pose of the landmark in the reference frame

of the sensor, and $\tilde{f} : \mathrm{SE}(3) \to \mathbb{R}$. However, for the sake of generality, we let the footprint depend separately on the pose of the sensor and that of the landmark. In this way, the footprint may capture, if desired, environmental features such as lighting and occlusion.

Once a sensor footprint is given, the coverage of the environment can be defined as a function of the poses of the sensors. To this aim, we let each sensor be assigned a subset of the landmarks. We let $\mathcal{L}_i(t)$ denote the subset assigned to the ith sensor at time t. We require that the subsets $\mathcal{L}_i(t)$ constitute a partition of the set $\{1, \dots, M\}$ of the landmarks; namely, we require that, at all times $t \geq 0$,

$$\mathcal{L}_i(t) \cap \mathcal{L}_j(t) = \emptyset \quad \forall (i,j) \in \{1, \dots, N\}^2, \tag{6.5}$$

$$\mathcal{L}_1(t) \cup \cdots \cup \mathcal{L}_N(t) = \{1, \dots, M\}. \tag{6.6}$$

Given a partition $\mathcal{P}(t) = (\mathcal{L}_1(t), \dots, \mathcal{L}_N(t))$ of the landmarks, the global coverage attained by the sensors is defined as

$$\Gamma(T(t), \mathcal{P}(t)) = \sum_{i=1}^{N} \sum_{j \in \mathcal{L}_i(t)} f(T_i(t), L_j). \tag{6.7}$$

where we have denoted $T(t) = \{T_1(t), \dots, T_N(t)\}$. In other words, the global coverage is defined as the sum of the perceptions of the landmarks, each attained by the sensor that the landmark is assigned to. For convenience, we denote the contribution given by the ith sensor to the coverage function (6.7) as $\Gamma_i(T_i(t), \mathcal{L}_i(t))$. Namely, we let

$$\Gamma_i(T_i(t), \mathcal{L}_i(t)) = \sum_{j \in \mathcal{L}_i(t)} f(T_i(t), L_j), \tag{6.8}$$

so that (6.7) can be rewritten as

$$\Gamma(T(t), \mathcal{P}(t)) = \sum_{i=1}^{N} \Gamma_i(T_i(t), \mathcal{L}_i(t)). \tag{6.9}$$

Our control objective is to find a distributed controller that steers the poses of the sensors and the partition of the landmarks in such a way that the global coverage is progressively improved. Note that finding an optimal set of poses and an optimal partition is a challenging problem, because the global coverage function depends jointly on the poses $T(t)$, which are constrained to the continuous set $\mathrm{SE}(3)$, and on the partition $\mathcal{P}(t)$ which belongs to the discrete set of the partitions of $\{1, \dots, M\}$.

6.3 Footprint design for surveillance of 3D structures

Before we move on to the design of our distributed controller, we describe a footprint design that is particularly well-suited for applications in surveillance of 3D structures with a team of mobile robots.

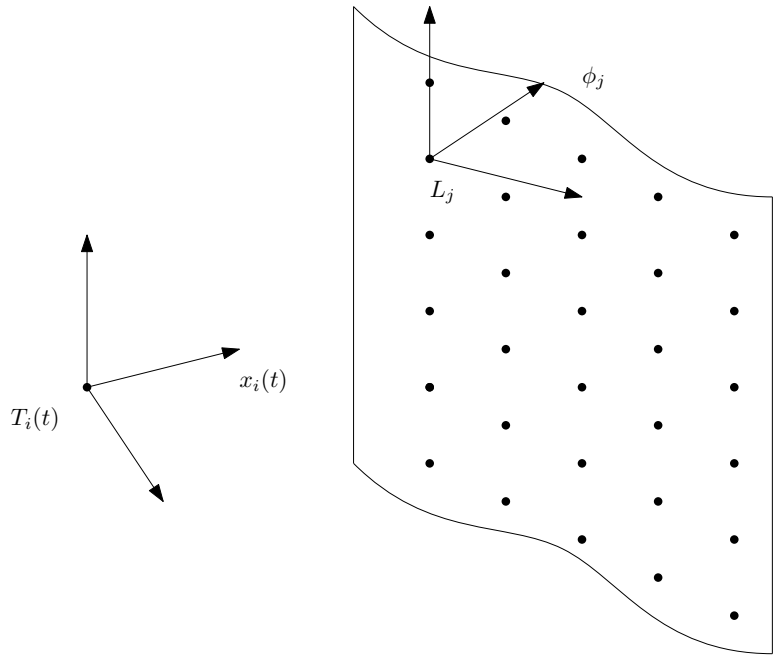

Figure 6.1: Abstraction of a mobile camera as a sensing agent and of a 3D structure as a set of landmarks. The frame $T_i(t)$ is attached to the camera, with $x_i(t)$ corresponding to the direction that the camera is looking at. M points of interest are selected on the surface, and the frame L_j is attached to the jth point, with ϕ_j corresponding to the inward normal to the surface at that location.

In this application domain, each sensor is a monodirectional camera, and the vector $x_i(t)$ denotes the direction that the ith sensor is looking at. A landmark is a point or small area of interest on a 3D structure that the sensors are asked to survey, and ϕ_j denotes the inward normal to the surface of the structure at the position q_j of the jth landmark. This scenario is illustrated in Figure 6.1.

The perception of a landmark is best when the landmark lies on the line of sight of the sensor, at a specific distance D. This condition can be written as $q_j = p_i(t) + Dx_i(t)$. Moreover, we want the inward normal to the surface to be parallel to the line of sight of the sensor, which means that the surface is exposed to the sensor frontally, rather than sideways. This condition can be expressed simply as $\phi_j = x_i(t)$. A footprint that captures these properties can be expressed as follows:

$$f(T_i(t), L_j) = -g(p_i(t), x_i(t), q_j) - \gamma g(p_i(t), \phi_j, q_j), \tag{6.10a}$$

$$g(p, \phi, q) = \|p + D\phi - q\|^2 \left(\alpha + \beta \frac{v^\intercal (p + D\phi - q)}{\|p + D\phi - q\|}\right), \tag{6.10b}$$

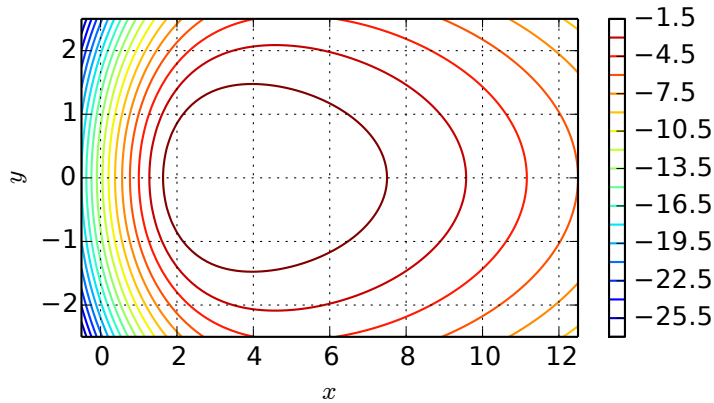

Figure 6.2: Contour plot of footprint (6.10) as a function of $q_j^{(1)}$ and $q_j^{(2)}$, with $p_i(t) = 0_3$, $x_i(t) = [1\ 0\ 0]^\intercal$, $q_j^{(3)} = 0$, $\alpha = 2.1$, $\beta = 1.9$, and $\gamma = 0$.

where α, β and γ are positive weights, with $\alpha > \beta$. It is implied that $g(p, \phi, q)$ is given a continuous extension in all values of (p, ϕ, q) such that $p + D\phi - q = 0$. Naturally, for such values one has $g(p, \phi, q) = 0$.

Note that footprint (6.10) has the following properties. First, $f(T_i(t), L_j) \leq 0$ for all $T_i(t), L_j \in \mathrm{SE}(3)$, so that the values that are closer to zero correspond to a better perception. Second, $f(T_i(t), L_j) = 0$ if and only if $p_i(t) + Dx_i(t) = q_j$ and $x_i(t) = \phi_j$, which means that the best possible perception is attained if and only if the landmark lies on the line of sight of the agent at the desired distance (i.e., $p_i(t) + Dx_i(t) = q_j$), and the surface is exposed to the agent frontally (i.e., $x_i(t) = \phi_j$). Finally, f is continuously differentiable if one considers the continuous extension of g described in the previous paragraph.

Footprint (6.10) is a function of numerous independent scalar parameters, and it is not possible to capture its variations with respect to all these parameters in a single plot. The rationale behind (6.10) is as follows. When the landmark lies on the line of sight of the agent (i.e., $x_i(t)$ and $q_j - p_i(t)$ have the same direction), then $g(p_i(t), x_i(t), q_j) = (\alpha - \beta)\|p_i(t) + Dx_i(t) - q_j\|^2$, which is the smallest possible value of $g(p_i(t), x_i(t), q_j)$ for a given value of $\|p_i(t) + Dx_i(t) - q_j\|$. Conversely, when the landmark lies behind the line of sight of the sensor (i.e., $x_i(t)$ and $q_j - p_i(t)$ have opposite directions), then $g(p_i(t), x_i(t), q_j) = (\alpha + \beta)\|p_i(t) + Dx_i(t) - q_j\|^2$, which is the largest possible value of $g(p_i(t), x_i(t), q_j)$ for a given value of $\|p_i(t) + Dx_i(t) - q_j\|$. A contour plot of Footprint (6.10) as a function of $q_j^{(1)}$ and $q_j^{(2)}$ is given in Figure 6.2.

One further advantage of Footprint (6.10) is that it can be simplified to model the perception of a point in the Euclidean space or in the Euclidean plane, without the point belonging to a specific surface. In fact, it is sufficient to set $\gamma = 0$ to make Footprint (6.10) oblivious of the orientation of the landmarks. In this way, there is no need to associate an orientation to the landmarks at all, and each landmark may simply represent a point in space.

We are going to exemplify the different uses of Footprint (6.10) in Section 6.9.

6.4 Generalized Voronoi tessellations

Our controller is based on the idea of Voronoi tessellations (Du et al., 1999). However, we generalize the definition of Voronoi tessellations to suit our system model.

Definition 6.1. *Let a sensor footprint* $f : \mathrm{SE}(3)^2 \to [0,1]$ *be assigned. Suppose that* f *is continuously differentiable in its first argument. Consider a tuple* $(T, \mathcal{P})$*, where* $T = (T_1, \dots, T_N) \in \mathrm{SE}(3)^N$ *and* $\mathcal{P} = (\mathcal{L}_1, \dots, \mathcal{L}_N)$ *is a partition of* $\{1, \dots, M\}$*. The tuple* $(T, \mathcal{P})$ *is called a* Voronoi tessellation *if the following properties are satisfied:*

$$\frac{\partial \Gamma}{\partial p_i}(T, \mathcal{P}) = 0_3 \quad \forall i \in \{1, \dots, N\}, \tag{6.11a}$$

$$\sum_{\xi_i \in \{x_i, y_i, z_i\}} \mathrm{Skew}(\xi_i) \frac{\partial \Gamma}{\partial \xi_i}(T, \mathcal{P}) = 0_3 \quad \forall i \in \{1, \dots, N\}, \tag{6.11b}$$

$$f(T_j, L_h) \leq f(T_i, L_h) \quad \forall h \in \mathcal{L}_i \quad \forall (i,j) \in \{1, \dots, N\}^2. \tag{6.11c}$$

Definition 6.1 can be interpreted as follows. A tuple $(T, \mathcal{P})$ is a Voronoi tessellation if: the gradient of the coverage with respect to the position of each single sensor evaluates to zero; the gradient of the coverage with respect to the orientation of each single sensor evaluates to zero; each landmark is assigned to the sensor that attains the best perception of that landmark.

Being a Voronoi tessellation is a necessary condition for a tuple $(T, \mathcal{P})$ to constitute a maximizer for the coverage function Γ. However, not all Voronoi tessellations are also optimal solutions. This property is formalized as follows.

Proposition 6.1. *Suppose that* $(T^*, \mathcal{P}^*)$ *is an optimal solution to*

$$\begin{aligned} &\textit{maximize} && \Gamma(T, \mathcal{P}), \\ &\textit{subject to} && T \in \mathrm{SE}(3)^N, \\ & && \mathcal{P} \textit{ is an } N\textit{-partition of } \{1, \dots, M\}. \end{aligned} \tag{6.12}$$

Then $(T^*, \mathcal{P}^*)$ *is also a Voronoi tessellation.*

Proof. Suppose by contradiction that $(T^*, \mathcal{P}^*)$ is not a Voronoi tessellation. Then one of conditions (6.11) is violated. Denote $T^* = (T_1^*, \dots, T_N^*)$, $T_i^* = (p_i^*, R_i^*)$ and $\mathcal{P}^* = (\mathcal{L}_1^*, \dots, \mathcal{L}_N^*)$.

- If (6.11a) is violated, let $v_i = \partial\Gamma(T^*, \mathcal{P}^*)/\partial p_i \neq 0_3$. Suppose that we translate p_i by a small positive quantity $\mathrm{d}\rho$ along v_i. The increment in the value of Γ is

$$\mathrm{d}\Gamma = \left(\frac{\partial\Gamma}{\partial p_i}(T^*, \mathcal{P}^*)\right)^{\mathsf{T}} v_i \mathrm{d}\rho = \|v_i\|^2 \mathrm{d}\rho > 0. \tag{6.13}$$

  Hence, there exists p_i' in a neighborhood of p_i^* such that, denoting $T' = (T_1^*, \dots, (p_i', R_i^*), \dots, T_N^*)$, we have $\Gamma(T', \mathcal{P}^*) > \Gamma(\mathcal{T}^*, \mathcal{P}^*)$. Therefore, $(T^*, \mathcal{P}^*)$ is not an optimal solution to (6.12), which is a contradiction.

- If (6.11b) is violated, let $\omega_i = \sum_{\xi_i \in \{x_i, y_i, z_i\}} \mathrm{Skew}(\xi_i)\partial\Gamma(T^*, \mathcal{P}^*)/\partial\xi_i \neq 0_3$. Suppose that we rotate R_i^* by a small positive angle $\mathrm{d}\theta$ along the axis ω_i (i.e., $\mathrm{d}\xi_i = -\mathrm{Skew}(\xi_i^*)\omega_i \mathrm{d}\theta$ for $\xi_i \in \{x_i, y_i, z_i\}$). Then, the increment in the value of Γ is

$$\begin{aligned}
\mathrm{d}\Gamma =& \sum_{\xi_i \in \{x_i, y_i, z_i\}} \left(\frac{\partial\Gamma}{\partial\xi_i}(T^*, \mathcal{P}^*)\right)^{\mathsf{T}} \mathrm{d}\xi_i \\
=& -\sum_{\xi_i \in \{x_i, y_i, z_i\}} \left(\frac{\partial\Gamma}{\partial\xi_i}(T^*, \mathcal{P}^*)\right)^{\mathsf{T}} \mathrm{Skew}(\xi_i^*)\omega_i \mathrm{d}\theta \\
=& \left(\sum_{\xi_i \in \{x_i, y_i, z_i\}} \mathrm{Skew}(\xi_i^*)\frac{\partial\Gamma}{\partial\xi_i}(T^*, \mathcal{P}^*)\right)^{\mathsf{T}} \omega_i \mathrm{d}\theta \\
=& \|\omega_i\|^2 \mathrm{d}\theta > 0.
\end{aligned} \tag{6.14}$$

  Hence, there exists $R_i' \in \mathrm{SO}(3)$ in a neighborhood of R_i^* such that, denoting $T' = (T_1^*, \dots, (p_i^*, R_i'), \dots, T_N^*)$, we have $\Gamma(T', \mathcal{P}^*) > \Gamma(T^*, \mathcal{P}^*)$. Therefore, $(T^*, \mathcal{P}^*)$ is not an optimal solution to (6.12), which is a contradiction.

- If (6.11c) is violated, let $h \in \mathcal{L}_i^*$ and $f(T_j^*, L_h^*) > f(T_i^*, L_h^*)$. Denote $\mathcal{L}_i' = \mathcal{L}_i^* \backslash \{h\}$ and $\mathcal{L}_j' = \mathcal{L}_j^* \cup \{h\}$. Use the notation P' for what is obtained from $\mathcal{P}^*$ by substituting $\mathcal{L}_i^*$ with $\mathcal{L}_i'$ and $\mathcal{L}_j^*$ with $\mathcal{L}_j'$. Then, we have

$$\Gamma(T^*, \mathcal{P}') = \Gamma(T^*, \mathcal{P}^*) + f(T_j, L_h) - f(T_i, L_h) > \Gamma(T^*, \mathcal{P}^*). \tag{6.15}$$

  Therefore, $\Gamma(T^*, \mathcal{P}^*)$ is not optimal which is a contradiction.

We thus conclude that $\Gamma(T^*, \mathcal{P}^*)$ is a Voronoi tessellation. □

6.5 Control of the motion of a single sensor

In this section, we shall design a local controller to steer the motion of a single sensor which has a fixed set of landmarks assigned to it. This controller will constitute a stepping stone in the design of a distributed controller for the whole sensor team, taking into account also the possible changes in the assignment of the landmarks. Without loss of generality, we consider the ith sensor, with pose $T_i(t)$, and we let the landmarks $\mathcal{L}_i \subseteq \{1, \dots, M\}$ be assigned to it.

The proposed controller is simply a gradient climb of the contribution given by the sensor to the coverage function. Namely, we let

$$v_i(t) = \frac{\partial \Gamma_i}{\partial p_i}(T_i(t), \mathcal{L}_i), \tag{6.16a}$$

$$\omega_i(t) = \sum_{\xi_i \in \{x_i, y_i, z_i\}} \mathrm{Skew}(\xi_i) \frac{\partial \Gamma_i}{\partial \xi_i}(T_i(t), \mathcal{L}_i). \tag{6.16b}$$

For the proposed controller, we have the following convergence result.

Lemma 6.1. *Consider a sensing agent with pose $T_i(t)$ and a fixed set $\mathcal{L}_i$ of landmarks. Let the pose of the sensor be controlled as in* (6.16)*. Suppose that the sensor footprint f is continuously differentiable in its first argument. Then, the function $\Gamma_i(T_i(t), \mathcal{L}_i)$ is nondecreasing. Moreover, there exists $T_i^* \in \mathrm{SE}(3)$ such that $T_i(t) \to T_i^*$ and, denoting $T_i^* = (p_i^*, x_i^*, y_i^*, z_i^*)$,*

$$\frac{\partial \Gamma_i}{\partial p_i}(T_i^*, \mathcal{L}_i) = 0_3, \tag{6.17a}$$

$$\sum_{\xi_i \in \{x_i, y_i, z_i\}} \mathrm{Skew}(\xi_i^*) \frac{\partial \Gamma_i}{\partial \xi_i}(T_i^*, \mathcal{L}_i) = 0_3. \tag{6.17b}$$

Proof. Recalling that the sensor footprint f is bounded and continuously differentiable, we can see from (6.8) that also Γ_i is bounded and continuously differentiable. The time-derivative of Γ_i can be written as follows:

$$\dot{\Gamma}_i(T_i(t), \mathcal{L}_i) = \frac{\partial \Gamma_i}{\partial p_i}(T_i(t), \mathcal{L}_i)\dot{p}_i(t) + \sum_{\xi_i \in \{x_i, y_i, z_i\}} \frac{\partial \Gamma_i}{\partial \xi_i}(T_i(t), \mathcal{L}_i)\dot{\xi}_i(t). \tag{6.18}$$

Using (6.3) and (6.16) in (6.18), we have

$$\dot{\Gamma}_i(T_i(t), \mathcal{L}_i) = \|v_i(t)\|^2 + \|\omega_i(t)\|^2 \geq 0, \tag{6.19}$$

which proves that $\Gamma_i(T_i(t), \mathcal{L}_i)$ is nondecreasing. Denote $\Omega = \{T_i \in \mathrm{SE}(3) : \Gamma_i(T_i, \mathcal{L}_i) \leq \Gamma_i(T_i(0), \mathcal{L}_i)\}$. Note that Ω is compact and positively invariant to

the system dynamics. Moreover, since $\Gamma_i(T_i(t), \mathcal{L}_i)$ is bounded, from LaSalle's invariance principle we know that $T_i(t)$ must converge to the largest invariant subset of $\{T_i \in \mathrm{SE}(3) : \dot{\Gamma}_i(T_i(t), \mathcal{L}_i) = 0\}$. However, in this set we have $v_i(t) = 0_3$ and $\omega_i(t) = 0_3$, which implies that $T_i(t)$ must converge to some $T_i^* \in \mathrm{SE}(3)$. Finally, evaluating (6.8) for $T_i(t) = T_i^*$, we have (6.17). □

It is worth noting that, in general, there may exist more than one (possibly infinite) poses T_i^* that satisfy (6.17). Lemma 6.1 only guarantees that $T_i(t)$ converges to one such pose. Even if the sensor footprint and the landmark $\mathcal{L}_i$ are given, the specific pose that $T_i(t)$ converges to may depend on the initial conditions.

6.6 Transfer of the landmarks

In this section, we are going to describe the mechanism that governs the distribution of the landmark among the agents.

The network is initialized with an arbitrary distribution $\mathcal{P}(0) = (\mathcal{L}_1(0), \ldots, \mathcal{L}_N(0))$ of the landmarks. Variations in the landmark assignments are considered instantaneous events, and they occur when an agent initiates a communication instance with another agent. When an agent i initiates a communication with another agent j, agent i yields to agent j all the landmarks that are currently assigned to agent i, but that agent j can perceive better. A more formal description of the policy for the landmark transfers is given as follows. We denote as $t_{ij}^{(k)}$ the time instant when agent i initiates a communication instance with agent j for the kth time. Conventionally, we let $t_{ij}^{(0)} = 0$ for all i, j. Upon the time instant $t_{ij}^{(k)}$, the landmark assignment is updated as follows:

$$\Delta_{ij}^{(k)} = \{h \in \mathcal{L}_i(t_{ij}^{(k)}) : \ f(T_j(t_{ij}^{(k)}), L_h) > f(T_i(t_{ij}^{(k)}), L_h)\}, \tag{6.20a}$$

$$\mathcal{L}_i(t_{ij}^{(k)^+}) = \mathcal{L}_i(t_{ij}^{(k)}) \setminus \Delta_{ij}^{(k)}, \tag{6.20b}$$

$$\mathcal{L}_j(t_{ij}^{(k)^+}) = \mathcal{L}_j(t_{ij}^{(k)}) \cup \Delta_{ij}^{(k)}. \tag{6.20c}$$

By design, each landmark transfer produces an instantaneous improvement of the total coverage attained by the network, because a landmark is transfered only if the receiving agent has a better perception of that landmark than the yielding agent. This property is formalized in the following lemma.

Lemma 6.2. *Consider a network of sensing agents with poses $T(t) = (T_1(t), \ldots, T_N(t))$, and let $\mathcal{P}(t) = (\mathcal{L}_1(t), \ldots, \mathcal{L}_N(t))$, where $\mathcal{L}_i(t)$ is the subset of the landmarks assigned to the ith agent at time t. Let the landmark assignment be updated according to* (6.20)*. Then, $\Gamma(T(t_{ij}^{(k)^+}), \mathcal{P}(t_{ij}^{(k)^+})) \geq \Gamma(T(t_{ij}^{(k)}), \mathcal{P}(t_{ij}^{(k)}))$.*

Proof. The coverage attained by the network before and after the update are related by the following equation:

$$\begin{aligned}\Gamma(T(t_{ij}^{(k)^+}),\mathcal{P}(t_{ij}^{(k)^+})) =&\Gamma(T(t_{ij}^{(k)}),\mathcal{P}(t_{ij}^{(k)})) \\ &+ \sum_{L\in\Delta_{ij}^{(k)}} (f(T_j(t_{ij}^{(k)}),L) - f(T_i(t_{ij}^{(k)}),L)),\end{aligned} \tag{6.21}$$

where $\Delta_{ij}^{(k)}$ is the set of the landmarks transfered from the ith agent to the jth agent, as given by (6.20a). However, from (6.20a), we can see that any landmark in $\Delta_{ij}^{(k)}$ satisfies $f(T_j(t_{ij}^{(k)}),L) > f(T_i(t_{ij}^{(k)}),L)$. Therefore, we have

$$\sum_{L\in\Delta_{ij}^{(k)}} (f(T_j(t_{ij}^{(k)}),L) - f(T_i(t_{ij}^{(k)}),L)) \geq 0, \tag{6.22}$$

and by (6.21), we conclude $\Gamma(T(t_{ij}^{(k)^+}),\mathcal{P}(t_{ij}^{(k)^+})) \geq \Gamma(T(t_{ij}^{(k)}),\mathcal{P}(t_{ij}^{(k)}))$. □

To complete the description of the algorithm, we only need to specify a scheduling rule to determine when the communication instances $t_{ij}^{(k)}$ shall take place. However, the proposed algorithm does not need to enforce a specific scheduling rule. To prove that the agents converge to a Voronoi tessellation, it is sufficient to assume that the update times are distinct and that the interval between two consecutive communication instances involving the same pair (i,j) of agents $t_{ij}^{(k)}$ and $t_{ij}^{(k+1)}$ is upper-bounded. These requirements are formalized as the following assumption.

Assumption 6.2. *(i) For any $i,i',j,j' \in \{1,\dots,N\}$, and any $k,k' \in \mathbb{N}\cup\{0\}$, such that $(i,j,k) \neq (i',j',k')$, we have $t_{ij}^{(k)} \neq t_{i'j'}^{(k')}$. (ii) There exists $\tau_{\max} > 0$ such that, for all agents $i,j \in \{1,\dots,N\}$, and all $k \in \mathbb{N}\cup\{0\}$, we have $t_{ij}^{(k+1)} \leq t_{ij}^{(k)} + \tau_{\max}$.*

For practical reasons, it is convenient to enforce also a lower bound on the interval between two consecutive communication instances. Enforcing such lower bound allows one to exclude that the solutions of the closed-loop system are Zeno.

Remark 6.1. *We note that, in order to satisfy Assumption 6.2, each agent needs to be able to initiate a communication with every other agent in the system. However, all communication links are only pairwise and intermittent, and no continuous all-to-all communication is required.*

Before we move on to demonstrate the properties of the proposed coverage algorithm, we propose here one particular strategy to schedule the communication instances $t_{ij}^{(k)}$ for each agent.

To simplify the notation, consider, for each agent i, the sequence $\{t_i^{(k)}\}_{k\in\mathbb{N}\cup\{0\}}$ that collects the communication instances initiated by agent i with any other agent in the network. With this notation, consider the threshold functions

$$\varsigma_i^{(k)}(t, v_i, \omega_i) = \min\{t - \epsilon_t - t_i^{(k)},\ \epsilon_v - \|v_i\|,\ \epsilon_\omega - \|\omega_i\|\}, \tag{6.23}$$

where ϵ_v, ϵ_t and ϵ_ω are positive constants. Then, agent i initiates a communication instance according to the following rule:

$$t_i^{(k+1)} = \inf\{t > t_i^{(k)} :\ \varsigma_i^{(k)}(t, v_i(t), \omega_i(t)) > 0\}. \tag{6.24}$$

The agent that is going to be the communication recipient is selected in a cyclic fashion among the other agents in the network. When the scheduling rule (6.23) and (6.24), is used in conjunction with controller (6.16), Assumption 6.2 is satisfied. In fact, the linear (respectively, angular) velocity of each agent is bound to drop below ϵ_v (respectively, ϵ_ω) within a finite time; Otherwise, from (6.19), we would have that the coverage attained by the corresponding agent would grow unbounded. However, since the footprint of the agents are upper-bounded by definition, we know that the coverage attained by an agent must be upper-bounded as well. Finally, note that the scheduling rule (6.23) and (6.24) also enforces a lower bound ϵ_t between two consecutive communication instances, thus leading to the exclusion of the undesired Zeno behavior.

6.7 Modeling the closed-loop sensor network as a hybrid system

In this section, we shall model the closed-loop system (6.3), (6.16) and (6.20) as a hybrid system. The obtained model will be used to prove formally that the network converges to a Voronoi tessellation. Note that this modeling process is undertaken for purely analytical purposes. None of the controllers are required to keep track of the global state of the hybrid system.

Roughly speaking, the system flow is given by the kinematics (6.3) of the agents and the control law (6.16), while the system jumps are given by the landmark transfers (6.20). The state of the hybrid system is a function of the hybrid time (i.e., of the time elapsed and the number of jumps occurred). However, in the rest of this section we are going to omit the dependency of the state on the hybrid time to keep the notation agile.

The state of the hybrid system comprises the poses of the sensors, the landmark assignment, and the current time. The state also contains the number ν_{ij} of communication instances elapsed between any pair of agents i and j until the current time. We denote the state of the hybrid system as X, and we write $X = (T, \mathcal{P}, \nu, t)$.

The flow set is given by all values of X such that a communication instance is not

triggered. This set can be written as follows:

$$C = \{X : \ t \leq t_{ij}^{(\nu_{ij}+1)} \quad \forall (i,j) \in \{1,\dots,N\}\}. \tag{6.25}$$

Conversely, the jump set is given by all values of X that trigger a communication instance. This set can be written as follows:

$$D = \{X : \ t \geq t_{ij}^{(\nu_{ij}+1)} \quad \text{for some } (i,j) \in \{1,\dots,N\}\}. \tag{6.26}$$

The flow map $F(X)$ is implicitly specified by the kinematics (6.3) of the agents under the control law (6.16). Note that, within two consecutive jumps, the landmark sets $\mathcal{L}_i$ are constant, and therefore the control law (6.16) is well-defined. The jump map $G(X)$ is implicitly specified by the updates (6.20), with the constraint that the update is triggered only for the pair (i,j) such that $t \geq t_{ij}^{(k)}$. For the same pair, the variable ν_{ij} is incremented to keep track of the number of communications elapsed:

$$\nu_{ij}(t^+) = \nu_{ij}(t) + 1. \tag{6.27}$$

In this model, the flow map and the jump maps are single valued, With these notations, we can prove our main result, which is formalized as the following theorem.

Theorem 6.1. *Consider a network of N sensing agents indexed as $1,\dots,N$ and a set $\mathcal{L}$ of M landmarks indexed as $1,\dots,M$. Let $T_i(t) = (p_i(t), x_i(t), y_i(t), z_i(t))$ be the pose of the ith agent, with kinematics* (6.3)*. Let $\mathcal{L}_i(t) \subseteq \mathcal{L}$ be the set of landmarks assigned to the ith agent at time t, and let the agents be controlled as by* (6.16) *and* (6.20)*. Let the communication times $t_{ij}^{(k)}$ satisfy Assumption 6.2. Then, denoting $T(t) = (T_1(t),\dots,T_N(t))$ and $\mathcal{P}(t) = (\mathcal{L}_1(t),\dots,\mathcal{L}_N(t))$, the tuple $(T(t), P(t))$ converges to a Voronoi tessellation, while the coverage $\Gamma(T(t), P(t))$ attained by the network is nondecreasing.*

Proof. From Lemma 6.1, we know that the network coverage $\Gamma(T(t), \mathcal{P}(t))$ is nondecreasing along the system flow. From Lemma 6.2, we know that the network coverage is nondecreasing across the system jumps. Hence, we can conclude that the network coverage is nondecreasing along any solution of the hybrid system. Next, we need to prove that any complete solution of the hybrid system converges to a state that corresponds to a Voronoi tessellation (i.e., that satisfies the properties of Definition 6.1). Consider the function $V(X) = -\Gamma(T, \mathcal{P})$ that measures the coverage attained collectively by the networked agents. First, consider a time instant t such that the state of the system belongs to the flow set C. For such t, the values of $T(t)$ and $\mathcal{P}(t)$ are well defined, as well as the kinematics of the sensing agents. Therefore, we can write

$$\begin{aligned} \nabla V(X)^\intercal F(X) &= -\sum_{i=1}^{N} \sum_{\xi_i \in \{p_i, x_i, y_i, z_i\}} \frac{\partial \Gamma(T(t), \mathcal{P}(t))}{\partial \xi_i} \dot{\xi}_i \\ &= -\sum_{i=1}^{N} \sum_{\xi_i \in \{p_i, x_i, y_i, z_i\}} \frac{\partial \Gamma_i(T_i(t), \mathcal{L}_i(t))}{\partial \xi_i} \dot{\xi}_i. \end{aligned} \tag{6.28}$$

From (6.18) and (6.19) in Lemma 6.1, we know that the right-hand side can be rewritten as

$$\nabla V(X)^{\intercal} F(X) = -\sum_{i=1}^{N} (\|v_i(t)\|^2 + \|\omega_i(t)\|^2) \leq 0. \tag{6.29}$$

Now, consider instead a time instant t such that the state belongs to the jump set D. For such t, there exist (i, j, k) such that $t = t_{ij}^{(k)}$, and a communication instance is triggered. From Lemma 6.2, we know that the communication instance produces an increment in the value of the coverage function. Therefore, we can write

$$V(G(X)) - V(X) = -\Gamma(T(t^+), \mathcal{P}(t^+)) + \Gamma(T(t), \mathcal{P}(t)) \leq 0. \tag{6.30}$$

Consider the functions $u_C(X)$ and $u_D(X)$ defined as in Proposition 2.8. Using (6.28) and (6.30) we can see that $u_C(X) \leq 0$ and $u_D(X) \leq 0$ for all possible states X. Hence, from Proposition 2.8, we know that, for any precompact solution ϕ^* of the hybrid system, there exists $r \in \mathbb{R}$, such that ϕ^* converges to the largest invariant subset of $V^{-1}(r) \cap (\overline{u_C^{-1}(0)} \cup (u_D^{-1}(0) \cap G(u_D^{-1}(0))))$. Let us denote such invariant subset as $\mathcal{I}_r$. Now note that, since the sensor footprint is radially unbounded from below, $V(X)$ is radially unbounded from above. However, since $\Gamma(T(t), \mathcal{P}(t))$ is nondecreasing, $V(t)$ is nonincreasing. Hence, any complete solution of the hybrid system is also bounded, and, therefore, it is precompact. We can conclude that any complete solution of the hybrid system converges to $\mathcal{I}_r$. Hence, to complete the proof it is sufficient to show that any state $X \in \mathcal{I}_r$ corresponds to a Voronoi tessellation. Suppose by contradiction that $X \in \mathcal{I}_r$ but X does not correspond to a Voronoi tessellation. Then, by the Definition 6.1 of a Voronoi tessellation, one or more of the following conditions occur:

$$\frac{\partial \Gamma}{\partial p_i}(T, \mathcal{P}) \neq 0_3 \quad \text{for some } i \in \{1, \ldots, N\}, \tag{6.31a}$$

$$\sum_{\xi_i \in \{x_i, y_i, z_i\}} \text{Skew}(\xi_i) \frac{\partial \Gamma}{\partial \xi_i}(T, \mathcal{P}) \neq 0_3 \quad \text{for some } i \in \{1, \ldots, N\}, \tag{6.31b}$$

$$f(T_j, L_h) > f(T_i, L_h) \quad \text{for some } h \in \mathcal{L}_i \text{ and some } (i, j) \in \{1, \ldots, N\}^2. \tag{6.31c}$$

Hence, the value of $\Gamma(T(t), \mathcal{P}(t))$ is bound to increase either along the system flow or in correspondence to the system jump $t_{ij}^{(\nu_{ij}+1)}$. Consequently, the value of $V(X)$ is bound to decrease, and X is bound to exit from $\mathcal{I}_r$. But this is a contradiction, because $\mathcal{I}_r$ is invariant. We thus conclude that any complete solution of the hybrid system converges to a Voronoi tessellation. □

6.8 Making the agent trajectories collision-safe

In this section, we propose a collision avoidance module to incorporate in the proposed controller so that the trajectories generated by the sensing agents are collision-safe.

Collision avoidance can be incorporated in the proposed control algorithm by modifying the definition of the coverage function so that it takes into account the possible proximity of the sensor to other bodies, such as the other sensors in the network or obstacles in the environment.

Consider the following collision avoidance function

$$\varphi(p,b) = \begin{cases} 0, & \|p-b\| > \rho, \\ 1/\rho^2 - 1/\|p-b\|^2, & \|p-b\| \leq \rho, \end{cases} \tag{6.32}$$

where $\rho > 0$ is a safety radius and $p, b \in \mathbb{R}^3$. Note that $\varphi(p,b) \leq 0$ and $\varphi(p,b) < 0$ if $\|p-b\| > \rho$, and that $\varphi(p,b) \to -\infty$ when $p - b \to 0$. The results in this section can be generalized to other collision avoidance functions with these characteristics, but for simplicity we are going to refer to (6.32).

We abuse of notation, we define the coverage attained by each sensing agent as

$$\Gamma_i(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i) = \sum_{j \in \mathcal{L}_i(t)} f(T_i(t), L_j) + \sum_{b \in \mathcal{B}_i(t)} \varphi(p_i(t), b), \tag{6.33}$$

where $\mathcal{B}_i(t)$ is a set of points with which agent i must avoid collisions. The linear velocity of each agent is chosen consequently as

$$v_i(t) = \frac{\partial \Gamma_i}{\partial p_i}(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i). \tag{6.34}$$

Similarly, we define the coverage attained by the whole sensor network while avoiding collisions as

$$\Gamma(T(t), \mathcal{L}(t), \mathcal{B}) = \sum_{i=1}^{N} \Gamma_i(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i), \tag{6.35}$$

where we have denoted $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_N)$. Collision avoidance is guaranteed because, if $p_i(t)$ is sufficiently close to a body $b \in \mathcal{B}_i$, the collision avoidance term will dominate any other contribution to the linear velocity $v_i(t)$ of the agent, so that the distance between the agent and the body cannot be reduced further.

Note that incorporating a collision avoidance term in the coverage function allows to model effects such as the proximity of an agent to another body hindering the performances of the agent. For example, the body may obstruct the field of view of the agent or influence the motion of the agent in an undesired way.

In general, avoiding collisions with moving bodies may compromise the monotonicity of the coverage function, thus negating the convergence of the system to a Voronoi tessellation. However, as long as the bodies $\mathcal{B}$ are stationary, the monotonicity of the coverage (and, consequently, the convergence to a Voronoi tessellation) is preserved. In fact, in this case, the coverage attained by each agent is

only affected by the motion of that agent, and the agent motion is still a gradient climb of the coverage function $\Gamma_i(T_i, \mathcal{L}_i, \mathcal{B}_i)$, which means that $\Gamma_i(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i)$ is nondecreasing for each individual agent between any two consecutive communication instances. The formal derivation of this result is analogous to the proof of Lemma 6.1, and it is omitted here for brevity.

Another scenario of interest is to avoid collisions between different sensors in the network. To this aim, we can use the same collision avoidance function (6.32), but we need to modify the definition of the total coverage attained by the network as follows:

$$\Gamma(T(t), \mathcal{L}(t), \mathcal{B}) = \sum_{i=1}^{N} \Gamma_i(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i) + \sum_{(i,j)\in\mathcal{E}} \varphi(p_i(t), p_j(t)), \tag{6.36}$$

where $\mathcal{E}$ denotes the set of all possible unordered pairs (i, j) with $i, j \in \{1, \ldots, N\}$ and $i \neq j$. For this scenario, the linear velocity of each agent needs to be chosen as

$$v_i(t) = \frac{\partial \Gamma_i}{\partial p_i}(T_i(t), \mathcal{L}_i(t), \mathcal{B}_i) + \sum_{j\in\{1,\ldots,N\}\setminus\{i\}} \frac{\partial \varphi}{\partial p_i}(p_i(t), p_j(t)). \tag{6.37}$$

With this choice of $v_i(t)$, one can show that the total coverage attained by the network, as defined by (6.36), is nondecreasing. The derivation is again similar to the proof of Lemma 6.1, and it is omitted here for brevity.

6.9 Numerical simulations

In this section, we are going to exemplify the use of the proposed coverage algorithm with two numerical simulations. The simulations are programmed using the middleware ROS (Quigley et al., 2009), and the controller of each sensor is implemented as a single ROS node. In particular, for each simulated agent i, we initiate a controller node that executes the proposed coverage algorithm, computing $v_i(t)$ and $\omega_i(t)$, and a simulator node that integrates the kinematics (6.3) to reproduce the motion of the sensor. The different controller nodes communicate with each other to transfer landmarks to each other by means of ROS messages and services. Similarly, each controller node communicates with the corresponding simulator node by means of ROS messages. This setup presents at least two advantages: first, the simulation reproduces faithfully the distributed nature of the control algorithm; secondly, the same ROS nodes used in the simulation can be used to execute the control algorithm on physical robots. In all simulations, the perception attained by the agents is modeled by footprint (6.10), and the communication instances $t_{ij}^{(k)}$ are triggered by the recursive rule (6.23) and (6.24).

Surveillance of a planar environment

In the first simulation, we consider a team of $N = 4$ sensing agents, which are required to monitor a square planar environment with side of length 6. The room is abstracted into a set of $M = 625$ landmarks, which are equally spaced about the room. Hence, each landmark can be thought of as a tile within the planar environment to survey. In this simulation, there is no orientation information associated with the landmarks, and therefore the orientations of the landmarks are not assigned. Moreover, no collision avoidance module is used in this simulation. Initially, all the landmarks are assigned to the same agent. The initial position of agent i is chosen as $p_i = (0, 1.25 - 0.25i, 0)$, while the initial orientation is chosen as $R_i = I_3$ for all the sensors. The footprint parameters are chosen as $D^* = 1$ and $\gamma = 0$.

The results of this simulation are shown in Figures 6.3 to 6.5. From Figures 6.3 to 6.5, we can see how the sensors are progressively deployed about the room, and how the landmark distribution assumes patterns corresponding to better coverage costs. The final configuration is verified to be a Voronoi tessellation, within a tolerance comparable to the numerical accuracy of the simulation.

Surveillance of a 3D structure

In this simulation, we consider $N = 4$ simulated sensing agents and a landmark set extracted from a point cloud of originated by laser-scanning a cylindrical structure. The proposed coverage algorithm is modified to account for collision avoidance as described in Section 6.8, with $\mathcal{B}_i = \{L_1, \ldots, L_M\}$. The initial positions of the sensors are taken as $[-1.5, -1.5, 0]^\intercal$, $[-1.5, 1.5, 0]^\intercal$, $[-1.5, -1.5, 1]^\intercal$ and $[-1.5, 1.5, 1]^\intercal$. The footprint parameters are chosen as $D^* = 1$ and $\gamma = 1$. The simulation is run for 100 seconds. The results of the simulation are illustrated in Figures 6.6 and 6.7. Figure 6.6 comprises six snapshots of the configuration assumed by the sensor network during the simulation.

Figure 6.7 illustrates the coverage attained by the sensors throughout the simulation, as well as the total coverage attained by the network. All the quantities are negative since the footprint used is negative semidefinite. From Figure 6.7, we can see that the coverage attained by each sensor presents large discontinuities across the communication instants, since landmarks are transferred at those instants. On the other hand, we can see that the total coverage attained by the network is non-decreasing, and converges to a value corresponding to the equilibrium configuration that we can see in the last snapshot of Figure 6.6. The small spikes in the plot must be attributed to the short delays in the message transmission in ROS.

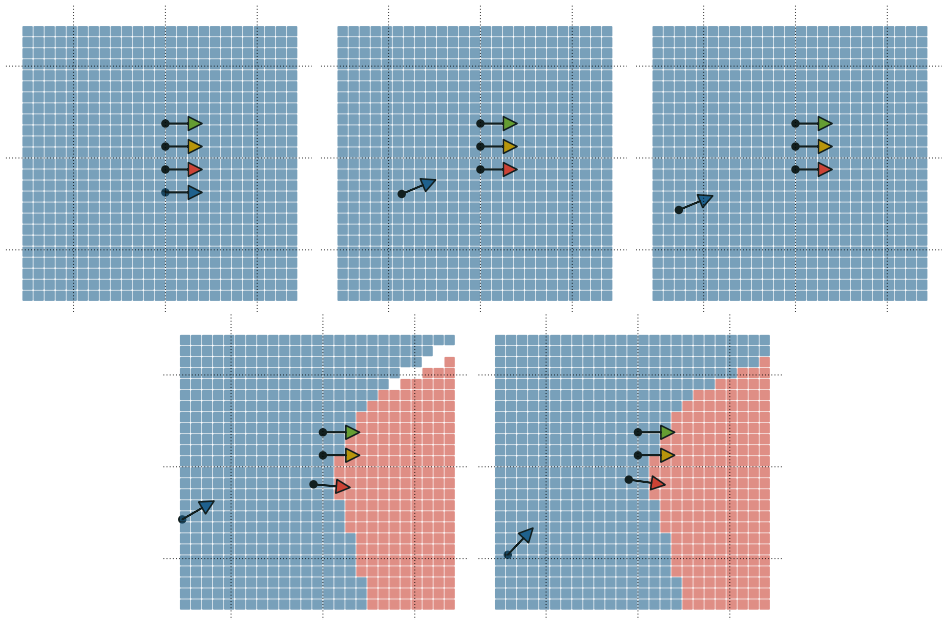

Figure 6.3: Results of the simulation described in Section 6.9. Each sensing agent is represented as a colored arrow, and each landmark is represented as a square tile. Each tile has the same color of the agent that it is currently assigned to. Blank tiles are in the process of being transferred from one agent to another while the snapshot is being taken. For each agent i, the tail of the arrow is in the point p_i, while the head of the arrow is in the point $p_i + x_i$. For each landmark, the corresponding tile is centered in the position q_j. Seconds elapsed in each snapshot, in lexicographical order: 0, 0.2, 0.5, 1.0, 2.0.

6.10 Preliminary experimental evaluation

The proposed coverage algorithm has been subject to preliminary experimental evaluations. The experiments have been conducted in the KTH *Smart Mobility Lab*, with a sensor network comprising one physical aerial robot and three simulated aerial robots. The physical robot is a 3DR IRIS+ quadcopter. Both the physical and the simulated robots are endowed with a controller running on a ROS node. A ROS node is also used to implement a simple integrator model for the simulated robots. The position of the physical robot is measured by means of a motion capture system, while the positions of the simulated robots are computed by the simulator, and sent to the controller by ROS. To reproduce the distributed nature of the algorithm, each simulator and each controller is implemented as a separate ROS node, and the different nodes communicate with each other by means of ROS messages and services. In this section, we shall present two experiments: one for the coverage of a planar environment, and one for the coverage of a 3D structure.

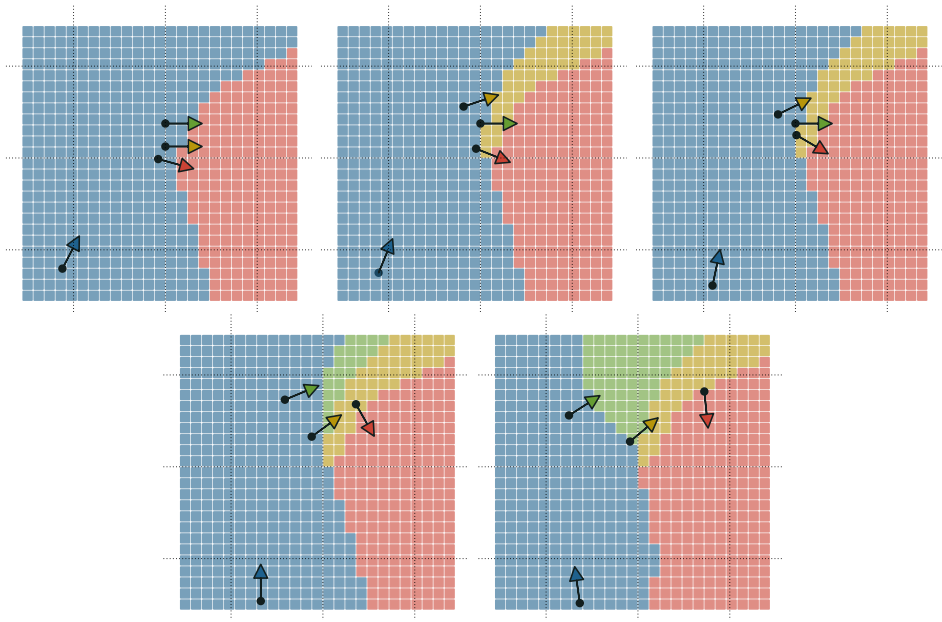

Figure 6.4: Results of the simulation described in Section 6.9. Each sensing agent is represented as a colored arrow, and each landmark is represented as a square tile. Each tile has the same color of the agent that it is currently assigned to. Blank tiles are in the process of being transferred from one agent to another while the snapshot is being taken. For each agent i, the tail of the arrow is in the point p_i, while the head of the arrow is in the point $p_i + x_i$. For each landmark, the corresponding tile is centered in the position q_j. Seconds elapsed in each snapshot, in lexicographical order: 5.0, 10.0, 15.0, 20.0, 25.0.

In these experiments, the model of the perception of the landmarks attained by the agents, the control of the motion of the agents, and the rules for scheduling the communication instances may differ slightly from those described in this chapter. Therefore, for each experiment, we give a reference where a detailed description of the algorithm can be found.

Experimental evaluation for the coverage of a planar environment

In this section we describe an experimental evaluation of the proposed algorithm for the surveillance of a planar environment. A more detailed description of the experimental setup is found in Sposato (2016). In this experiment, the landmarks are represented as colored tiles, which are projected onto the lab floor, while the agents are represented as colored arrows. The landmarks are color-coded so that each landmark is colored as the agent that it is currently assigned to. We consider 500 landmarks as a discrete abstraction of a 5.0 by 4.0 meter rectangular domain. There is no orientation information associated with the landmarks, and there is no

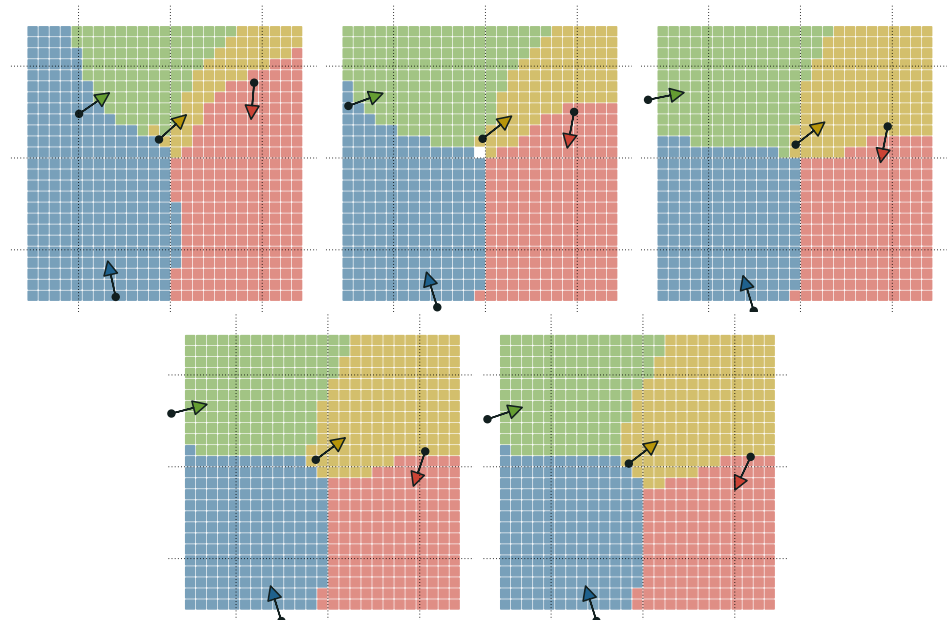

Figure 6.5: Results of the simulation described in Section 6.9. Each sensing agent is represented as a colored arrow, and each landmark is represented as a square tile. Each tile has the same color of the agent that it is currently assigned to. Blank tiles are in the process of being transferred from one agent to another while the snapshot is being taken. For each agent i, the tail of the arrow is in the point p_i, while the head of the arrow is in the point $p_i + x_i$. For each landmark, the corresponding tile is centered in the position q_j. Seconds elapsed in each snapshot, in lexicographical order: 30.0, 40.0, 50.0, 75.0, 105.0.

collision avoidance strategy applied. The results of the experiments are summarized in Figures 6.8 and 6.9, and a video can be found at Sposato et al. (2017).

Experimental evaluation of the coverage of a 3D structure

In this section we describe an experimental evaluation of the proposed algorithm for the surveillance of a 3D structure.

For this experiment, we use the Ascending Technologies NEO hexacopter. A monocular camera developed by Skybotix AG (weight of 0.088 kg, is attached on the protective case on the front side of the NEO. The camera is used to capture the surveyed structure from the perspective of the sensing agents during the execution of the algorithm. A more detailed description of the experimental setup is found in Adaldo et al. (2017d), together with a video of the experiment.

The experiment has been performed in the FROST Lab at LTU. To better suit

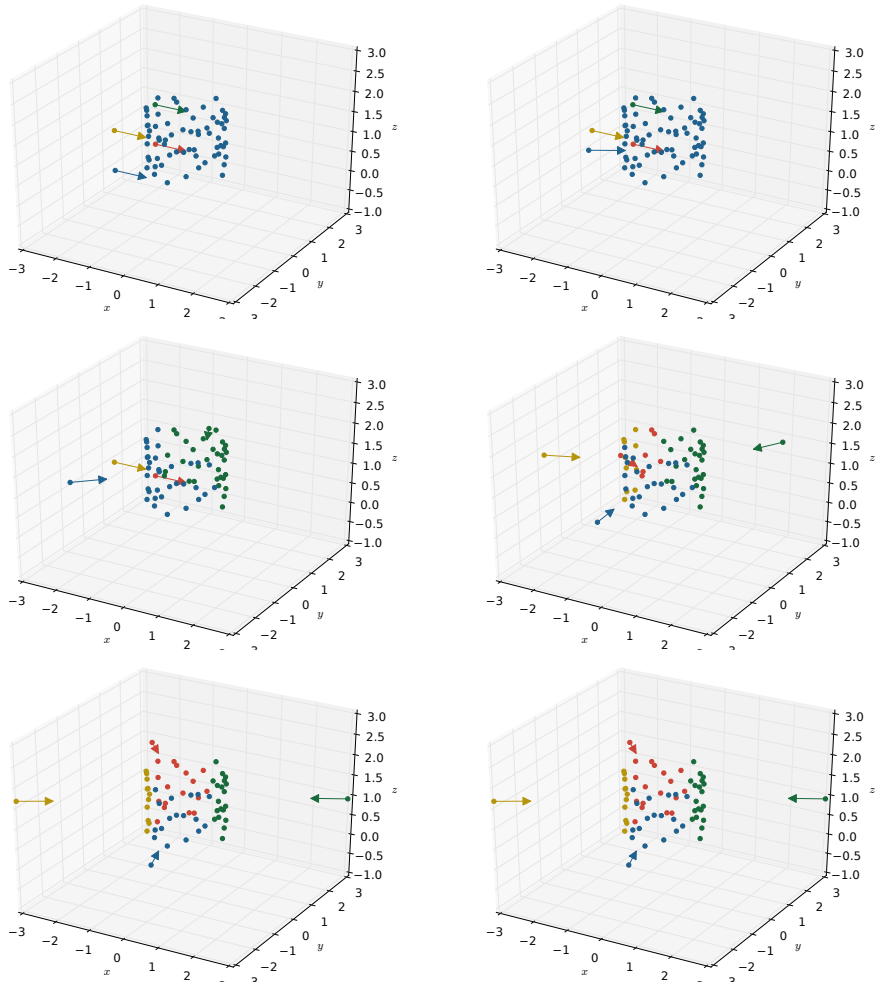

Figure 6.6: Snapshots of the configuration assumed by the agent network in the simulation described in Section 6.9. Each agent i is represented as a colored arrow, with the tail of the arrow being the position p_i of the sensor, and the direction of the arrow being x_i. Each landmark L_j is represented as a colored circle in position q_j. (The orientation of the landmarks is not represented to avoid cluttering the picture.) In lexicographical order, with $t = 0$ being the start of the simulation, the snapshots are taken at: $t = 0$, $t = 5$, $t = 25$, $t = 50$, $t = 85$, $t = 100$.

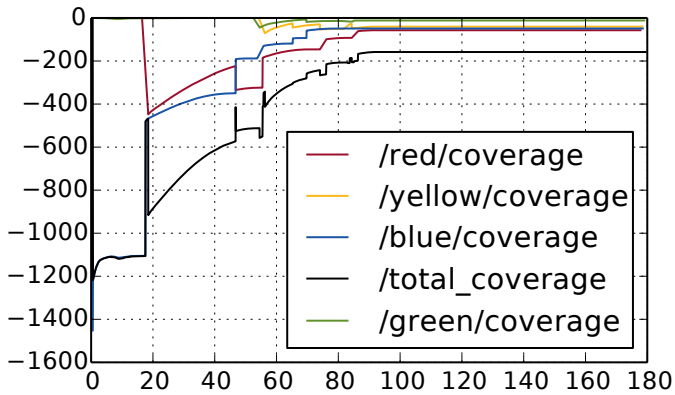

Figure 6.7: Coverage attained by each sensor and total coverage attained by the network throughout the simulation described in Section 6.9. The short spikes in the total coverage signal are due to asynchronous message transmission among ROS nodes.

the dimensions of the laboratory, the footprint parameters are set as $D^* = 1.3$ and $\gamma = 0.5$. Figure 6.10 depicts the artificial structure to inspect assembled for the experimental trial. The cylindrical structure in the pictures is provided for visual purposes, while for the landmark generation we use a cylinder object is with radius of 0.165 m and height 1 m. For the safety of the flight, and to avoid ground effects and collisions with the ceiling, the landmarks are extracted only between 0.5 m and 1.1 m height. We consider a network of two agents to survey the described structure.

In this experiment, the Vicon motion-capture system has been utilized for precise object localization, and this information is utilized by the NEO for the task execution. The waypoints generated by the algorithm are converted into position-velocity-yaw trajectories, which are inputs for the autopilot of the NEO. For safety reasons, the generated trajectories are executed on the same NEO, one at a time.

Figure 6.10 depicts the position of the two agents (left), and the image stream captured by each agent (middle and right). Additionally, the reference path, (i.e., the trajectory which is followed by agent) and the point cloud of the surveyed object are shown in Figure 6.12. The starting points of the agents are $[-1.7, -1.5, 0]^\intercal$ and $[-1.7, -1.5, 0]^\intercal$, while the final positions are $[0, 1.3, 0.68]^\intercal$ and $[0.5, -1.7, 0.8]^\intercal$. Moreover, the error between reference and trajectory is mainly caused by ground effects and errors in the attitude controller. Furthermore, Figure 6.11 also includes the point cloud that used for obtaining the landmark set. Finally, Figure 6.12 shows the coverage attained by each sensor along its path, and the total coverage attained by the network.

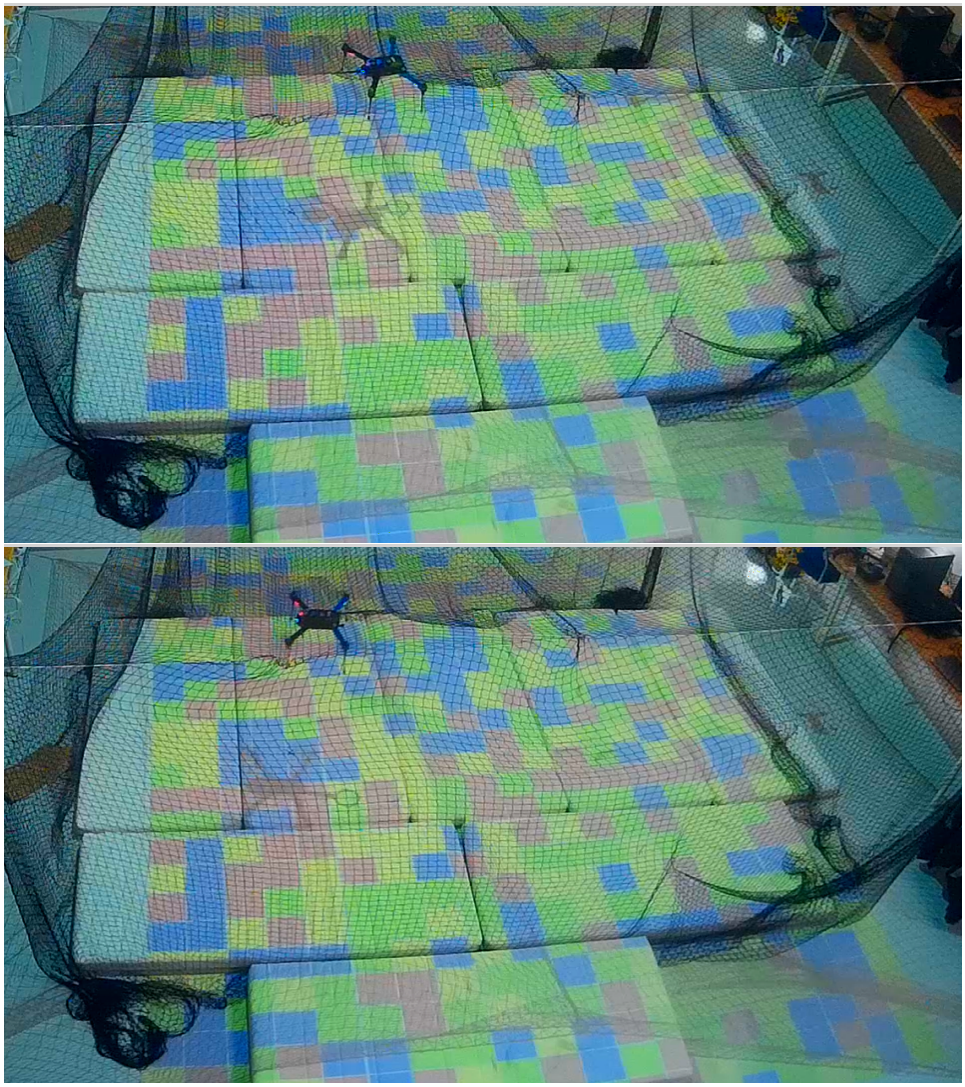

Figure 6.8: Results of the preliminary experimental evaluation described in Section 6.10. The physical robot is represented as a green arrow. The rounded arrows represent the positions and orientations generated by the optimization procedures in the coverage algorithm, and constitute reference waypoints for the robots. First part of the experiment; top: start of the experiment; bottom: 18 seconds elapsed.

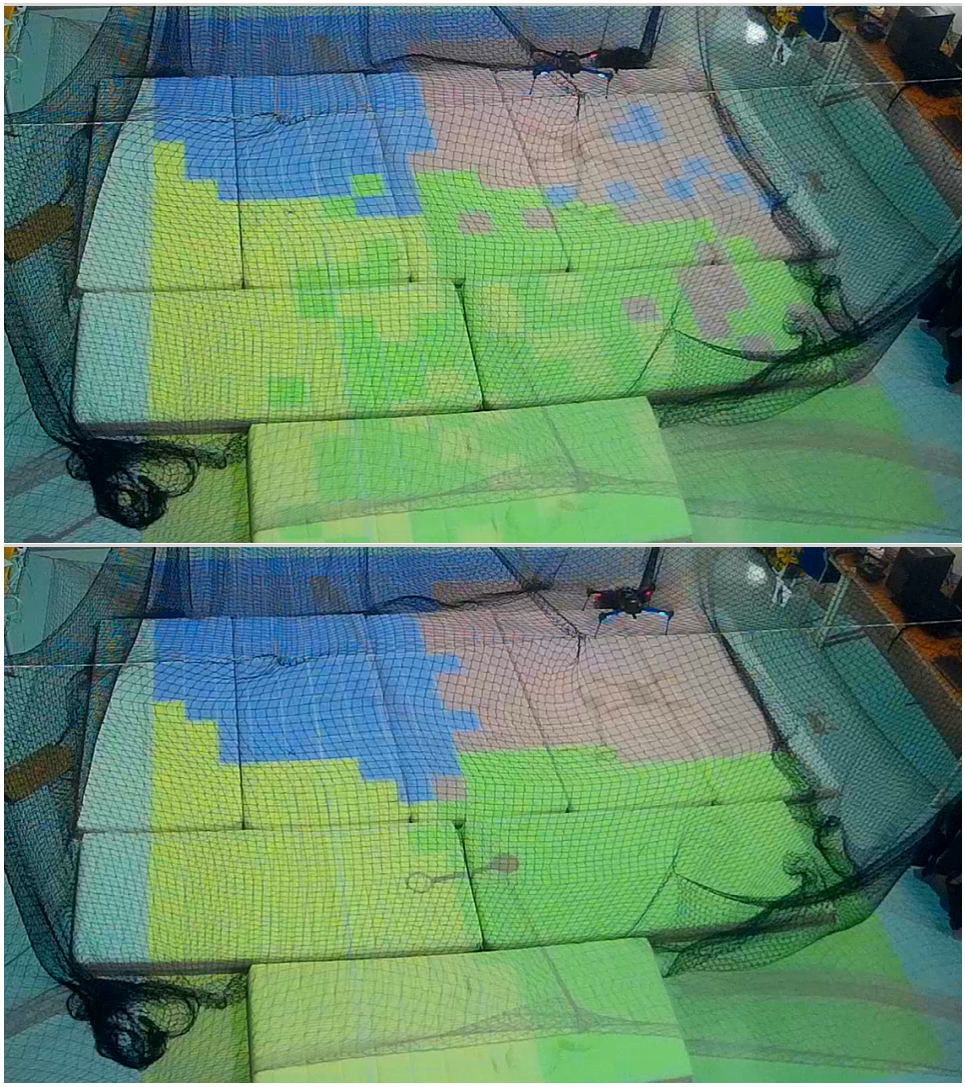

Figure 6.9: Results of the preliminary experimental evaluation described in Section 6.10. The physical robot is represented as a green arrow. The rounded arrows represent the positions and orientations generated by the optimization procedures in the coverage algorithm, and constitute reference waypoints for the robots. Second part of the experiment; top: 80 seconds elapsed; bottom: 180 seconds elapsed.

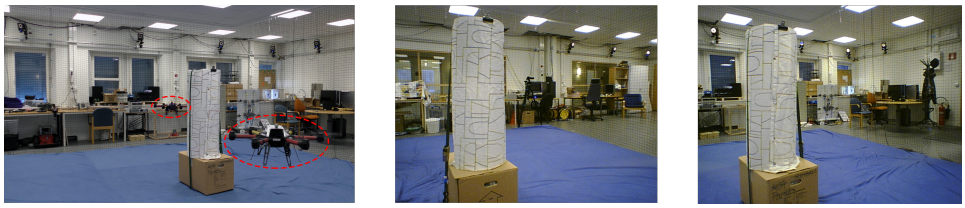

Figure 6.10: Snapshots from the experimental evaluation described in Section 6.10. Left: position of two agents during the experiment. Middle: visual feedback from the first agent. Right: visual feedback from the second agent.

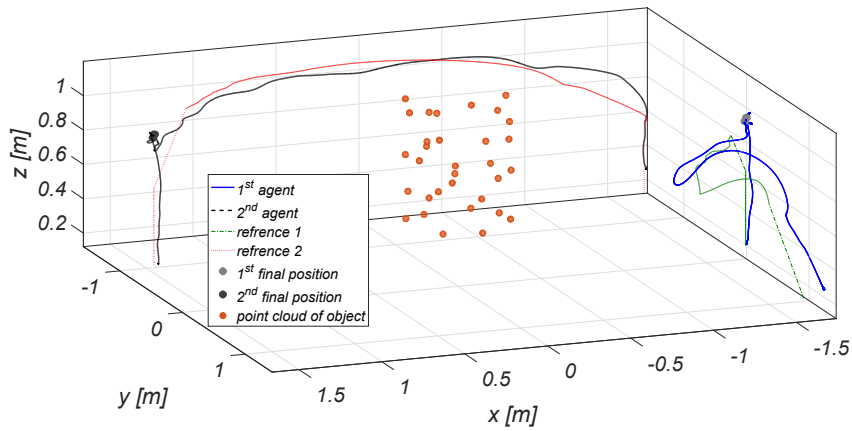

Figure 6.11: Trajectories followed by the agents during the experimental evaluation described in Section 6.10.

6.11 Summary

Motivated by applications in the field of robotic surveillance, in this chapter we have investigated a coverage problem of 3D structures with robotic sensor networks. We have introduced an abstraction of the structure to survey, and we have generalized the concept of Voronoi tessellation to fit our problem, defining a coverage score function as a measure of the quality of the surveillance attained by the sensor network. Then, we have proposed a control algorithm that drives the sensors to a Voronoi tessellation while the coverage score function is nondecreasing. We have also extended the algorithm to deal with collision avoidance problems. The algorithm has been validated with simulations and with experiments.

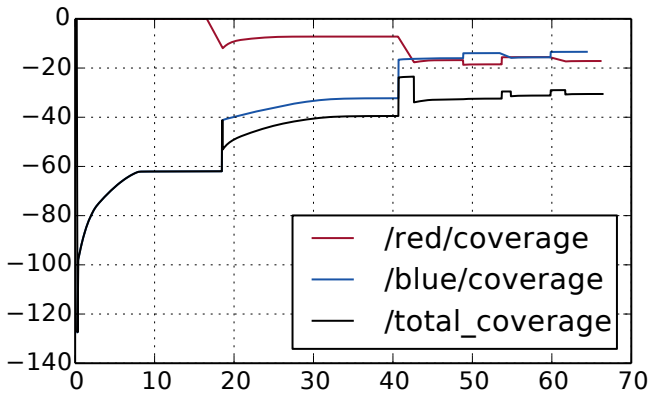

Figure 6.12: Coverage attained by each sensing agent along its path, and total coverage attained by the network during the experimental evaluation described in Section 6.10. The short spikes in the total coverage signal are due to asynchronous message transmission among ROS nodes.

Chapter 7

Cloud-supported effective coverage control

Questo di sette è il più gradito giorno,
pien di speme e di gioia:
diman tristezza e noia
recheran l'ore, ed al travaglio usato
ciascuno in suo pensier farà ritorno.

G. Leopardi,
Il sabato del villaggio*, vv. 38–42.*

In this chapter we consider our fourth and last multi-agent coordination scenario. Namely, we address the problem of inspecting a 3D structure with a team of autonomous sensing agents. Instead of communicating with each other, the agents upload information on a cloud repository to keep track of the progress of the inspection. The system modeling is similar to that proposed in Chapter 6, but the controllers operate with a different logic.

The rest of this chapter is organized as follows. In Section 7.1, we review the existing related work and highlight the novel contributions offered in the chapter. In Section 7.2, we define the mathematical model that we use as an abstract description of the effective coverage task under investigation. In Section 7.3, we define a hybrid controller that we use to steer the motion of a single sensing agent. In Section 7.4, we describe the proposed effective coverage controller for a single sensing agent. In Section 7.5, we describe the cloud repository used to store information about the progress of the task, and we define the proposed cloud-supported algorithm for multi-agent effective coverage. In Section 7.6, we describe a numerical simulation that validates our theoretical results. In Section 7.7, we describe the setup for a preliminary experimental evaluation of the proposed algorithm on an aerial robotic

platform. Section 7.8 concludes the chapter with a summary of the results.

7.1 Introduction

The problem that we study in this chapter falls within the scope of coverage problems for networks of mobile sensing agents. These problems have attracted a notable volume of research in the past few decades, because they constitute a flexible model for numerous applications, such as deployment, inspection and surveillance with networked robots (Cassandras and Li, 2005).

Coverage problems can be divided in two large categories. *Static coverage* problems (Cortes et al., 2004, Durham et al., 2012, Stergiopoulos and Tzes, 2013, Stergiopoulos et al., 2015) consist in finding a good placement for the sensing agents, and possibly controlling the agents to reach their target placements. On the other hand, *dynamic coverage* problems (Hussein and Stipanovic, 2007, Panagou et al., 2016, Wang and Hussein, 2010), also known as *effective coverage* problems, consist in controlling the motion of the agents to survey an environment continuously, until it has been searched sufficiently well. Awareness coverage control (Song et al., 2011, 2013) can be considered a form of dynamic coverage where the agents are also required to learn a density function which characterizes the importance of each point in the environment. The problem addressed in this work falls within the class of dynamic coverage problems.

In classical works on dynamic coverage problems (Hussein and Stipanovic, 2007), the sensors have circular sensing patterns, meaning that their sensing power is maximal at their own position, and decays with the distance from the sensor. This model is only appropriate to describe omnidirectional sensors, such as temperature sensors or circular laser scans. In more recent works (Panagou et al., 2016), the sensors have anisotropic sensing patters, which allows to model a larger variety of sensing devices, such as monocular cameras or cone-shaped laser scans. However, these models are still deficient in describing certain settings, such as the inspection of structure, the surveillance of a building, or the reconstruction of a 3D surface with depth sensors, because they do not capture the morphology of the structure to inspect. Conversely, in this chapter we consider a network of sensing agents with generic, anisotropic and heterogeneous sensing patterns, and we also take into account the 3D geometry of the object to inspect. Moreover, classical works assume that each sensing agent is continuously aware of the motion of all other agents, or that the agents can communicate continuously. More recently, intermittent inter-agent communication has been considered (Wang and Hussein, 2010). In this work, inter-agent communication is completely replaced by communication with a shared information repository, or *cloud*. Connections with the cloud are event-triggered and intermittent, and the amount of data exchanged upon each connection is limited. Cloud-supported solutions are becoming increasingly popular for various types of multi-agent problems (Bowman et al., 2016, Patel et al., 2016, Hale and Egerst-

edt, 2015), thanks to the recent explosion of cloud-based services and technologies. Note that the cloud is not a centralized omniscient computer. In this work, the cloud is barely a shared information repository which receives asynchronous and partial information about the progress of the inspection. All calculations need to be performed locally and online by the individual agents. Finally, most existing work on effective coverage address the problem of inspecting a planar, continuous area. Conversely, we consider the inspection of a 3D structure abstracted into a finite set of landmarks, where each landmark carries information about the local curvature of the surface. This setup is not only more general with respect to considering a planar continuous environment, but also computationally more tractable, and allows the proposed control algorithm to be implemented on small embedded processors.

The algorithm is formally shown to complete the inspection in finite time, and it is demonstrated by simulating a team of sensing agents in the ROS framework (Quigley et al., 2009), paving the way to experimental evaluation. Each agent is simulated as a different ROS node, effectively reproducing the distributed nature of the algorithm.

The rest of this chapter is organized as follows. In Section 7.2, we define the mathematical model that we use as an abstract description of the effective coverage task under investigation. In Section 7.3, we define a hybrid controller that we use to steer the motion of a single sensing agent. In Section 7.4, we describe the proposed effective coverage controller for a single sensing agent. In Section 7.5, we describe the cloud repository used to store information about the progress of the task, and we define the proposed cloud-supported algorithm for multi-agent effective coverage. In Section 7.6, we describe a numerical simulation that validates our theoretical results. In Section 7.7, we describe the setup for a preliminary experimental evaluation of the proposed algorithm on an aerial robotic platform. Section 7.8 concludes the chapter with a summary of the results.

7.2 System model and problem statement

In this chapter, we use a similar abstraction as the one described in Chapter 6. Namely, we abstract the structure to inspect into a finite set of landmarks, and each landmark is characterized by its position and orientation. Then, we consider a network of sensing agents with a simple kinematic model. However, the models of the sensors and the landmarks are slightly different than those employed in Chapter 6, since they are simplified to best suit the inspection scenario at hand. In the rest of this section, we define this system model formally.

We consider a set of N sensing agents indexed as $1, \dots, N$, and we denote $\mathcal{N} = \{1, \dots, N\}$. The ith agent is characterized by its position $p_i(t) \in \mathbb{R}^3$ and its orientation $n_i(t) \in \mathcal{S}^2$, and it is denoted $A_i(t) = (p_i(t), n_i(t))$. A sensing agent is an abstraction of a mobile sensor, such as a camera or a laser scan. The orientation

of the agent defines the direction that the agent is looking at. For example, if the sensor is a camera, the orientation of the agent corresponds to the direction that the camera is pointing to. Similarly to the model considered in Chapter 6, the kinematics of the agents is given by

$$\dot{p}_i(t) = v_i(t), \tag{7.1a}$$
$$\dot{n}_i(t) = -\operatorname{Skew}(n_i(t))\omega_i(t), \tag{7.1b}$$

for all $i \in \mathcal{N}$. Here v_i and ω_i are control inputs; v_i is called the linear velocity of the agent, while ω_i is called the angular velocity of the agent. We recall that Skew denotes the Skew operator, which corresponds to the cross product, in the sense that, for any $u, v \in \mathbb{R}^3$, we have $u \times v = \operatorname{Skew}(u)v$.

The agents are required to inspect a surface which is abstracted into a set of M landmarks indexed as $1, \dots, M$, where each landmark corresponds to a point on the surface, and we denote $\mathcal{M} = \{1, \dots, M\}$. Like a sensing agent, the jth landmark is defined by its position $q_j \in \mathbb{R}^3$ and its orientation $m_j \in \mathcal{S}^2$ (where, this time, the orientation m_j corresponds to the outward normal to the surface evaluated at q_j) and it is denoted $L_j = (q_j, m_j)$. However, the positions and orientations of the landmarks are constant.

Definition 7.1. *The perception of a generic landmark L attained by a generic agent $A = (p, n)$ is a function of the position and orientation of the landmark with respect to the agent. We denote this function as* $\operatorname{per}(A, L)$*, and we let it take values in* $[0, 1]$*, where* $\operatorname{per}(A, L) = 0$ *means that the landmark L is not visible at all by agent A, while* $\operatorname{per}(A, L) = 1$ *denotes the best possible perception. We require the following three properties for the perception function:*

P1 $\operatorname{per}(A, L)$ *is continuously differentiable with respect to both p and n;*

P2 there exists $R > 0$ such that, if $\|p - q\| > R$, then $\operatorname{per}(A, L) = 0$*,* $\partial \operatorname{per}(A, L)/\partial p = 0_3$ *and* $\operatorname{Skew}(n)(\partial \operatorname{per}(A, L)/\partial n) = 0_3$*;*

P3 for each $L \in \mathbb{R}^3 \times \mathcal{S}^2$, there exists $A_L^* \in \mathbb{R}^3 \times \mathcal{S}^2$ such that $\operatorname{per}(A_L^*, L) = 1$*,* $\partial \operatorname{per}(A_L^*, L)/\partial p = 0_3$ *and* $\operatorname{Skew}(n)(\partial \operatorname{per}(A_L^*, L)/\partial n) = 0_3$*.*

Property P1 is a technical assumption needed to prove our main result. Property P2 entails that a landmark cannot be perceived by a sensor if it is too far. Property P3 entails that for any pose of a landmark there exists a sensor pose that yields the best possible perception, and such sensor pose constitutes a local maximum of the perception function.

For each landmark L_j, we define the instantaneous coverage $\gamma_j(t)$ as the sum of the perceptions of that landmark attained by the N sensors:

$$\gamma_j(t) = \sum_{i=1}^{N} \mathrm{per}(A_i(t), L_j). \tag{7.2}$$

Moreover, we define the *cumulated coverage* $\Gamma_j(t)$ as the integral of the instantaneous coverage:

$$\Gamma_j(t) = \int_0^t \gamma_j(\tau)\,\mathrm{d}\tau\,. \tag{7.3}$$

Note that, since $\gamma_j(t) \geq 0$, $\Gamma_j(t)$ is a nondecreasing function of the time.

The control objective is that the cumulated coverage of each landmark reaches a desired value C^*. This objective can be formalized as follows:

Definition 7.2. *The inspection is completed successfully when* $\Gamma_j(t) \geq C^*$ *for all* $j \in \mathcal{M}$.

Correspondingly, we introduce the *landmark coverage errors*

$$e_j(t) = \max\{0, C^* - \Gamma_j(t)\}, \tag{7.4}$$

and the *total coverage error*

$$E(t) = \sum_{j=1}^{M} e_j(t), \tag{7.5}$$

so that the control objective corresponds to driving all the landmark coverage errors to zero, or equivalently, driving the total coverage error to zero. Note that the coverage errors $e_j(t)$ are by definition nonnegative and nonincreasing.

7.3 Hybrid control of a single agent to reach a desired position and orientation

Before we delve into the proposed control strategy, we shall show that the agent kinematics (7.1) can be controlled to reach any desired position and orientation $A^* = (p^*, n^*)$ asymptotically. To this aim, consider a generic agent $A_i(t) = (p_i(t), n_i(t))$ with kinematics (7.1) and under the hybrid controller defined in Figure 7.1, with:

$$v_i(t) = p^* - p_i(t), \tag{7.6a}$$

$$q^0 : \ \omega_i(t) = z_i, \tag{7.6b}$$

$$q^1 : \ \omega_i(t) = \mathrm{Skew}(n_i(t))n^*, \tag{7.6c}$$

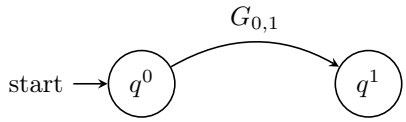

Figure 7.1: Hybrid automaton corresponding to the pose controller for a single agent. See (7.6) for the continuous-time dynamics corresponding to each discrete state and for the guard conditions.

$$G_{0,1} : n_i(t)^\intercal n^* \geq -\varsigma, \tag{7.6d}$$

and where $\varsigma \in (0, 1)$ is a control parameter, and $z_i \in \mathcal{S}^2$ is any unit vector orthogonal to $n_i(0)$. We denote this controller as Controller 7.1. Roughly speaking, Controller 7.1 drives agent i to gradually approach the pose (p^*, n^*). If the initial orientation of the sensor is almost the opposite of the desired orientation (a condition quantified as $n_i(t)^\intercal n^* < -\varsigma$), then the controller undergoes an initial phase (corresponding to state q_0) where the agents rotates around a fixed axis, until such initial alignment is amended (a condition quantified as $n_i(t)^\intercal n^* \geq -\varsigma$).

Lemma 7.1. *Consider a generic agent $A_i(t) = (p_i(t), n_i(t))$ with kinematics (7.1). Under Controller 7.1, the point (p^*, n^*) is an asymptotically stable equilibrium.*

Proof. Since the kinematics of $p_i(t)$ and $n_i(t)$ are decoupled, they can be analyzed separately. Since $v_i(t) = p^* - p_i(t)$ regardless of the discrete state q_i, p^* is an asymptotically stable equilibrium for $p_i(t)$. As for $n_i(t)$, consider the candidate Lyapunov function $V(n_i) = 1 - n_i^\intercal n^*$, and distinguish the two cases (i) $n_i(0)^\intercal n^* \geq -\varsigma$ and (ii) $n_i(0)^\intercal n^* < -\varsigma$. In Case (i), the controller switches immediately to state q^1, and we have

$$\begin{aligned} \dot{V}(n_i) &= -\dot{n}_i^\intercal n^* = -(-\operatorname{Skew}(n_i)\omega_i^\intercal n^* \\ &= -(-\operatorname{Skew}(n_i)\operatorname{Skew}(n_i)n^*)^\intercal n^* \\ &= -\|\operatorname{Skew}(n_i)n^*\|^2 \leq 0. \end{aligned} \tag{7.7}$$

Note that, from (7.7), we have that $n_i(t)^\intercal n^*$ is nondecreasing and that $\dot{V}(n_i) = 0$ if and only if $n_i = \pm n^*$. However, since $n_i(0)^\intercal n^* \geq -\varsigma$, we conclude that $\dot{V}(n_i) = 0$ if and only if $n_i = n^*$. Hence, $V(n_i)$ is a Lyapunov function and n_i converges to n^* asymptotically. In Case (ii), the controller is initially in state q^0, and n_i rotates around z_i with constant angular speed. After n_i has described a rotation of at most $2\arccos(\varsigma)$, we have $n_i^\intercal n^* \geq -\varsigma$, which causes the controller to switch to state q^1, and we can reason as in Case (i). □

An important consequence of Lemma 7.1 is that, if the initial conditions $A_i(0)$ and the target A^* are chosen out of a compact set $\mathcal{I}$, then the agent reaches any arbitrarily small neighborhood of $A^* = (p^*, n^*)$ in finite time. (Note that, since $\mathcal{S}^2$

is compact, the condition that $A_i(0)$ and A^* should lie in a compact set reduces to $p_i(0)$ and p^* being in a compact set.) In particular, if A^* is such that $\mathrm{per}(A^*, L) = 1$ for some landmark L, we have the following corollary.

Corollary 7.1. *Let a sensing agent $A_i(t)$ with kinematics (7.1) be controlled by Controller 7.1, with $A^* = A_L^*$ is such that $\mathrm{per}(A^*, L) = 1$. Then, if $A_i(0), A_L^* \in \mathcal{I}$ where $\mathcal{I}$ is a compact set, for any $\epsilon' < 1$ there exists a finite time $T_{\mathcal{I},\epsilon'}$ such that $\mathrm{per}(A_i(t), L) \geq \epsilon'$ for all $t \geq T_{\mathcal{I},\epsilon'}$.*

7.4 Effective coverage control for one agent

In this section, we describe a controller that lets a single sensing agent attain the effective coverage of a set of landmarks. This controller will serve as a stepping stone to introduce our multi-agent cloud-supported control scheme in Section 7.5.

First, note that when the coverage mission is performed by a single agent, the coverage of each landmark only depends on the motion of that agent. Therefore, the agent can keep track of the coverage error associated to each landmark with only local information. Denoting as usual the pose of the agent as $A_i(t) = (p_i(t), n_i(t))$, the derivative of the total coverage error is

$$\dot{E}(t) = - \sum_{j \in \mathcal{M}:\ e_j(t) > 0} \mathrm{per}(A_i(t), L_j) \leq 0. \tag{7.8}$$

Since the control objective is to drive $E(t)$ to zero, we can control $A_i(t)$ according to a gradient descent of $\dot{E}(t)$, so that the decay of $E(t)$ is as fast as possible.

The proposed single-agent controller is a hybrid controller that switches between a gradient descent of $\dot{E}(t)$ and Controller fig. 7.1, the latter being used as a backup when the gradient of $E(t)$ is too small. The gradient-descent controller can be written as follows:

$$v_i(t) = - \frac{\partial \dot{E}(t)}{\partial p_i}, \tag{7.9a}$$

$$\omega_i(t) = \mathrm{Skew}(n_i(t)) \frac{\partial \dot{E}(t)}{\partial n_i}. \tag{7.9b}$$

Writing $\ddot{E}(t) = (\partial \dot{E}/\partial p_i)^\intercal \dot{p}_i(t) + (\partial \dot{E}/\partial n_i)^\intercal \dot{n}_i(t)$, and using (7.1), we can see that under Controller 7.9 we have

$$\ddot{E}(t) = -\|v_i(t)\|^2 - \|\omega_i(t)\|^2 \leq 0. \tag{7.10}$$

However, $\ddot{E}(t)$ is not defined for t such that, for some $j \in \mathcal{M}$, $e_j(t) = 0$ and $e_j(\tau) > 0$ in a left neighborhood of t. In these time instants, instead, $\dot{E}(t)$ instantaneously

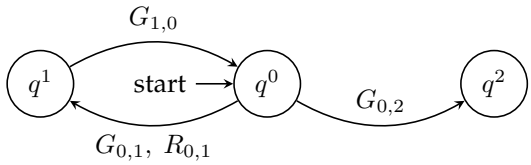

Figure 7.2: Effective-coverage controller for a single agent. States: q^0 runs (7.1) and (7.9); q^1 runs (7.1) and (7.6) with $A^* = A^*_{L_\iota}$; q^2 terminates. Guards: $G_{0,1}$ is (7.11); $G_{1,0}$ is (7.12); $G_{0,2}$ is $E(t) = 0$. Reset map: $R_{0,1}$ is $\iota :\in \{1, \dots, M : \ e_\iota(t) > 0\}$.

increases by $\mathrm{per}(A_i(t), L_j)$, because L_j has been completely inspected and stops contributing to the decay in the total coverage error.

The controller starts in the gradient-descent mode (7.9). The condition to switch to (7.6) is

$$\dot{E}(t) > -\epsilon \wedge E(t) > 0, \tag{7.11}$$

where $\epsilon \in (0, 1)$ is a constant threshold. Condition (7.11) means that the inspection is not complete, but the decay of $E(t)$ is close to zero. When switching to (7.6), one landmark L_ι is selected among those with a positive coverage error, and the reference pose is set as $A^*_{L_\iota}$. Note that, when this transition happens, at least one landmark with positive coverage error must exist, since $E(t) > 0$. The condition to switch back to (7.9) is

$$e_\iota(t) = 0 \vee \dot{E}(t) \leq -\epsilon', \tag{7.12}$$

where $\epsilon' \in (\epsilon, 1)$. Condition (7.12) means that either L_ι has been completely inspected ($e_\iota(t) = 0$) or the decay of the coverage error is faster ($\dot{E}(t) \leq -\epsilon'$). The inspection terminates when the condition $E(t) = 0$ is detected.

The proposed switching controller is given in Figure 7.2 in the form of a hybrid automaton, and it is referred to as Controller 7.2 in the rest of the chapter.

Now we can prove that a sensing agent with kinematics (7.1) controlled by Controller 7.2 completes the inspection in finite time.

Theorem 7.1. *Consider an agent $A_i(t) = (p_i(t), n_i(t))$ with kinematics (7.1), and let it be controlled by Controller 7.2. Let $\mathcal{I} \subset \mathbb{R}^3 \times \mathcal{S}^2$ be a compact set that encloses the initial pose $A_i(0)$ of the sensor and all the landmarks $L_1, \dots, L_M$. Then, there exists $T > 0$ such that $E(t) = 0$ for all $t \geq T$.*

Proof. First note that, when Controller 7.2 is in state q^0, we have $\dot{E}(t) \leq -\epsilon$. Since $E(t) \in [0, C^*]$ by definition, Controller 7.2 can be in state q^0 for a time not larger than C^*/ϵ. However, under state q^0, $\dot{E}(t)$ is nonincreasing. Therefore, the only way to trigger a transition to state q^1 is that the coverage error $e_j(t)$ of some landmark L_j

reaches zero, causing $\dot{E}(t)$ to instantaneously increase by $\mathrm{per}(A_i(t), L_j)$. However, from Corollary 7.1, we know that, once Controller 7.2 is in state q^1, after a time not larger than $T_{\mathcal{I},\epsilon'}$, we will have $\mathrm{per}(A_i(t), L_\iota) \geq \epsilon'$, which implies $\dot{E}(t) \leq -\epsilon'$. Therefore, Controller 7.2 can only remain in state q^1 for $T_{\mathcal{I},\epsilon'}$ for each landmark. Since there are M landmarks, Controller 7.2 can be in state q^1 for at most a time of $MT_{\mathcal{I},\epsilon'}$ before the inspection is complete. Hence, the theorem is proved with $T = C^*/\epsilon + MT_{\mathcal{I},\epsilon'}$. □

Remark 7.1. *A sensible choice of ϵ and ϵ' is needed to avoid a slow inspection and frequent switching between states q^0 and q^1 in Controller 7.2. For example, a larger ϵ' reduces the switching frequency, but may slow down the inspection, because the agent spends more time focusing on a single landmark rather than on the global coverage error.* ■

7.5 Cloud-supported effective coverage control for multiple agents

In this section, we consider the case that the inspection is performed by a team of N agents aided by a shared information repository (cloud). In this scenario, each individual agent does not know the coverage error $e_j(t)$ associated to each landmark, or the total coverage error $E(t)$. Therefore, for each agent i and each landmark j, we introduce the *estimated coverage error* $\hat{e}^i_j(t)$, which is initialized as $\hat{e}^i(0) = C^*$ and evolves as

$$\dot{\hat{e}}^i_j(t) = -\,\mathrm{per}(A_i(t), L_j). \tag{7.13}$$

Similarly, we let

$$\hat{E}^i(t) = \sum_{j=1}^{M} \hat{e}^i_j(t). \tag{7.14}$$

Each agent can intermittently communicate with the cloud to upload and download information about the progress of the inspection. The cloud substitutes inter-agent communication, and allows the agents to gather their contributions to the inspection. For each landmark $j \in \mathcal{M}$, the cloud maintains an estimate $\hat{e}^{\mathrm{cloud}}_j(t)$ of the coverage error associated to that landmark, which is initialized as $\hat{e}^{\mathrm{cloud}}_j(0) = C^*$. Hence, the amount of information contained in the cloud scales linearly with the number M of landmarks, and does not grow over time. We denote as $t_{i,k}$ the time when agent i connects with the cloud for the kth time. (Conventionally, $t_{i,0} = 0$ for all agents.) Cloud accesses are considered as instantaneous events. This assumption is mild because the time scale of wireless communication is arguably orders of magnitude faster than the time scale of the physical motion of the agents. Only one agent at a time can access the cloud: if it happens that $t_{i,k} = t_{j,h}$ for some agents i, j and some integers k, h, the two accesses happen one after the other in any order. Between two consecutive connections $t_{i,k}$ and $t_{i,k+1}$ to the cloud, agent i

needs to keep track only of its own contribution to the coverage. This contribution can be captured simply with a nonnegative scalar c_{ijk} defined as

$$c_{ijk} = \int_{t_{i,k-1}}^{t_{i,k}} \mathrm{per}(A_i(t), L_j)\,\mathrm{d}t\,. \tag{7.15}$$

When agent i connects to the cloud at time $t_{i,k}$, the estimated coverage errors are updated as

$$\hat{e}_j^{\mathrm{cloud}}(t_{i,k}^+) := \max\{0, \hat{e}_j^{\mathrm{cloud}}(t_{i,k}^-) - c_{ijk}\}, \tag{7.16a}$$

$$\hat{e}_j^{i}(t_{i,k}^+) := \max\{0, \hat{e}_j^{\mathrm{cloud}}(t_{i,k}^-) - c_{ijk}\}. \tag{7.16b}$$

This update means that the contribution c_{ijk} is incorporated in the coverage error estimated by the cloud and coverage error estimated by the agent is immediately updated to match the one estimated by the cloud. In this way, the contribution given to the coverage by each agent is collected in the cloud, and can be accessed later by other agents. Comparing (7.4) with (7.16), we can see that estimates of the coverage errors have the following remarkable properties:

$$\hat{e}_j^{i}(t_{i,k}^+) \le \hat{e}_j^{i}(t_{i,k}^-); \tag{7.17a}$$

$$e_j(t) \le \min\{\hat{e}_j^{\mathrm{cloud}}(t), \hat{e}_j^{i}(t)\}. \tag{7.17b}$$

Property (7.17a) means that a connection with the cloud can only cause the local estimates of the coverage errors to decrease. Property (7.17b) means that the estimates of the coverage errors are always overestimates; in particular, $\hat{E}^i(t) = 0$ implies $E(t) = 0$. This gives a natural stopping condition for the agents: when $\hat{E}^i(t) = 0$, agent i knows that the inspection is complete. Between two consecutive cloud accesses, each agent is controlled according to its estimated coverage errors. Namely, (7.9) is replaced with

$$v_i(t) = -\frac{\partial \dot{\hat{E}}^i(t)}{\partial p_i}, \tag{7.18a}$$

$$\omega_i(t) = \mathrm{Skew}(n_i(t))\frac{\partial \dot{\hat{E}}^i(t)}{\partial n_i}. \tag{7.18b}$$

The cloud accesses are triggered according to the following recursive rule:

$$t_{i,k+1} = \inf\{t > t_{i,k} :\ C_{i,k}(t) \ge \varsigma'(M_i(t)+1) \text{ or } \hat{E}^i(t) = 0\}, \tag{7.19}$$

where

$$C_{i,k}(t) = \sum_{j=1}^{M} \int_{t_{i,k}}^{t} \mathrm{per}(A_i(\tau), L_j)\mathrm{d}\tau, \tag{7.20a}$$

$$M_i(t) = \mathrm{card}\,\{j \in \mathcal{M} :\ \hat{e}_j(t) > 0\}, \tag{7.20b}$$

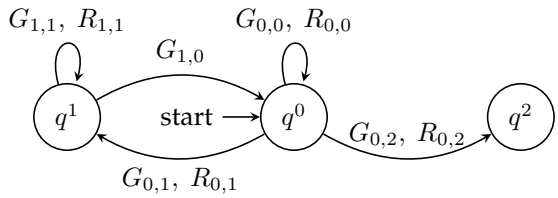

Figure 7.3: Cloud-supported effective-coverage controller for a multi-agent team. States: q^0 runs (7.1) and (7.18); q^1 runs (7.1) and (7.6); q^2 terminates. Guards: $G_{0,1}$ is (7.11); $G_{1,0}$ is (7.12); $G_{0,2}$ is $\hat{E}^i(t) = 0$; $G_{0,0}$ and $G_{1,1}$ are (7.19). Updates: $R_{0,1}$ is $\iota \in \{1, \dots, M : e_\iota(t) > 0\}$; $R_{0,0}$, $R_{1,1}$ and $R_{0,2}$ are (7.16).

with $\mathrm{card}(\cdot)$ denoting the cardinality of a set, and where ς' is a positive constant. This rule has the intuitive meaning that a cloud access is triggered when the agent has accumulated enough coverage contribution to share with the other agents ($C_{i,k}(t) \geq (M_i(t)+1)\varsigma'$) or when the inspection is complete ($\hat{E}^i(t) = 0$). The value of ς' represents a tradeoff between how often the agents access the cloud and how promptly they upload their contributions on the cloud. The proposed controller is formalized as a hybrid automaton in Figure 7.3 and is referred to as Controller 7.3. Note that the looping transitions in q^0 and q^1 represent the connections to the cloud. Upon these connections, agent i shares its contributions c_{ijk}, so that the estimated coverage errors $\hat{e}^i_j$ and $\hat{e}^{\text{cloud}}_j$ can be updated according to (7.16).

In the following Theorem 7.2 we prove formally that a team of sensing agents controlled by Controller 7.3 completes the inspection in finite time.

Theorem 7.2. *Consider a team of mobile sensing agents with poses $A_i(t) = (p_i(t), n_i(t))$, with $i \in \mathcal{N}$ and let the agents be controlled by Controller 7.3. Let $\mathcal{I} \subset \mathbb{R}^3 \times \mathcal{S}^2$ be a compact set that encloses the initial poses $A_i(0)$ of all the agents and all the landmarks $L_1, \dots, L_M$. Then, there exists $T > 0$ such that $E(t) = 0$ for all $t \geq T$.*

Proof. First note that, under the proposed controller, the estimated errors $\hat{E}^i(t)$ are nonincreasing. In fact, the motion of the agents imposes $\dot{\hat{E}}^i(t) \leq 0$, while, from (7.19) we can see that the cloud accesses impose $\hat{e}^i_j(t^+_{i,k}) \leq \hat{e}^i_j(t^-_{i,k})$, which summing over the landmarks yields $\hat{E}^i(t^+_{i,k}) \leq \hat{E}^i(t^-_{i,k})$. Similarly to the proof of Theorem 7.1, note that when Controller 7.3 is in state q^0, we have $\dot{\hat{E}}^i(t) \leq -\epsilon$. Since $\hat{E}^i(t) \in [0, C^*]$ by definition, and since $\hat{E}^i(t)$ has been shown to be nonincreasing, the controller can only remain in q^0 for a time of at most C^*/ϵ. However, since, under (7.18), $\dot{\hat{E}}^i(t)$ is nonincreasing, the only way for Controller 7.3 to transit to state q^1 is that the estimated error $\hat{e}^i_j(t)$ of a landmark reaches zero, causing $\dot{\hat{E}}^i(t)$ to drop by $\mathrm{per}(A_i(t), L_j)$. Since there are M landmarks, the controller can only

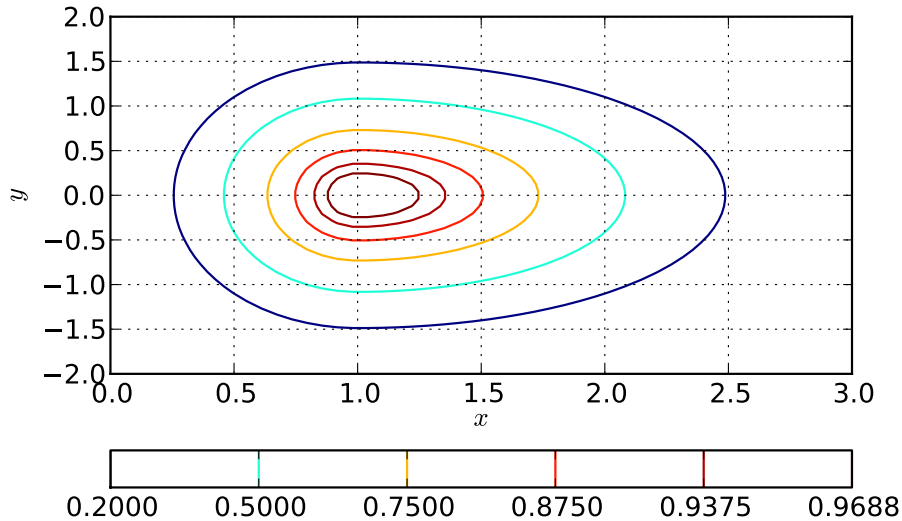

Figure 7.4: Contour plot of $f(p, n, q)$ in (7.22d) for $p = 0_3$, $n = (1, 0, 0)$ and $q \in [x, y, 0]$ with $x \in [0, 3]$, $y \in [-2, 2]$.

transit to state q^1 for at most M times. Once the controller is in state q^1, by Corollary 7.1, it will take at most a time $T_{\mathcal{I},\epsilon'}$ to have $\dot{\hat{E}}^i(t) \leq -\epsilon'$ and transit back to q^0. Hence, the controller can only remain in q^1 for at most a time of $MT_{\mathcal{I},\epsilon'}$. We must conclude that $\hat{E}^i(t)$ reaches zero in at most $MT_{\mathcal{I},\epsilon'} + C^*/\epsilon$. Since $E(t) \leq \hat{E}^i(t)$ by design, this result also implies that $E(t)$ reaches zero in at most $MT_{\mathcal{I},\epsilon'} + C^*/\epsilon$. □

7.6 Simulation

In this section, we present a simulation of the proposed cloud-supported distributed controller. We consider a network of $N = 4$ agents and a set of $M = 100$ landmarks sampled from a surface with the shape of an extruded sinusoid. The desired coverage for all landmarks is set to $C^* = 100$. The sensing pattern of the agents is chosen as follows:

$$\text{per}(A_i, L_j) = f(p_i, n_i, q_j)\lfloor n_i^\intercal m_j \rfloor, \tag{7.21}$$

where

$$\lfloor x \rfloor = \max\{x, 0\}, \tag{7.22a}$$

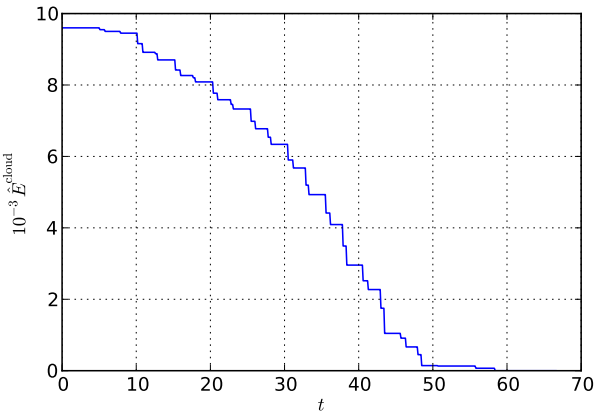

Figure 7.5: Coverage error estimated by the cloud during the simulation described in Section 7.6.

$$g(p,n,q) = \left\lfloor 1 - \frac{\|q-p-rn\|^2}{R^2} \right\rfloor, \tag{7.22b}$$

$$h(p,n,q) = \left\lfloor 1 - \frac{((q-p-rn)^\intercal n)^2}{r^2} - \frac{\|q-p-rn\|^2 - ((q-p-rn)^\intercal n)^2}{R^2} \right\rfloor, \tag{7.22c}$$

$$f(p,n,q) = \begin{cases} g(p,n,q) & (q-p-rn)^\intercal n > 0, \\ h(p,n,q) & (q-p-rn)^\intercal n < 0, \end{cases} \tag{7.22d}$$

and where $0 < r < R$. Recall that $L_j = (q_j, m_j)$ is the pose of landmark j. A contour plot of $f(p,n,q)$ is given in Figure 7.4. One can verify that (7.21) satisfies the properties of a perception function as listed in Section 7.2.

Figure 7.5 shows the total coverage error estimated by the cloud: from this picture, we can see that the inspection is completed in about 65 seconds. Figure 7.6 shows the cloud accesses for each agent over time; from this picture we can see that each agent accesses the cloud less than 15 times to complete the inspection. Figure 7.7 illustrates the trajectories followed by the agents; Figure 7.8 shows which controller is active for each agent. Finally, Figure 7.9 shows four snapshots of the configuration of the agents and the landmarks during the simulation. In these snapshots, each agent is represented as an arrow located in p_i and pointing in the direction of n_i, while each landmark is represented as a dot located in q_j. The orientations of the landmarks are not represented to avoid cluttering the pictures. The colors of the landmarks vary from red to blue to represent the value of $\hat{e}_j^{\text{cloud}}$ from C^* to zero.

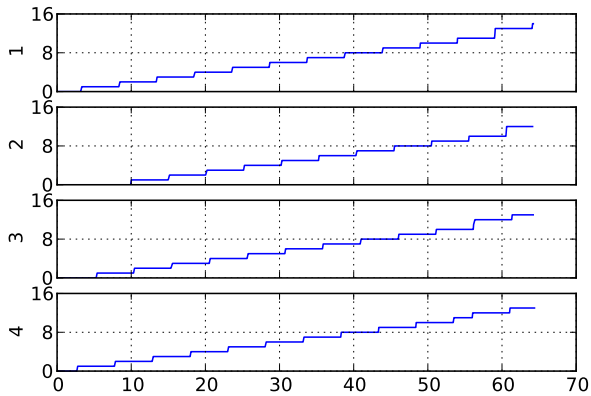

Figure 7.6: Number of cloud accesses for each sensing agent during the simulation described in Section 7.6.

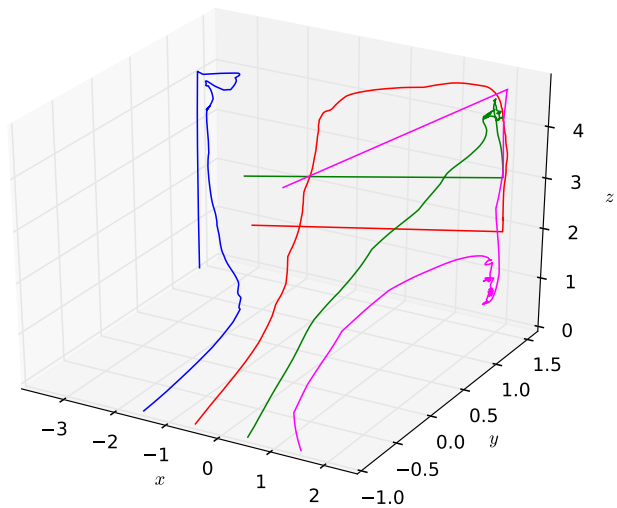

Figure 7.7: Trajectories followed by the sensing agents during the simulation described in Section 7.6.

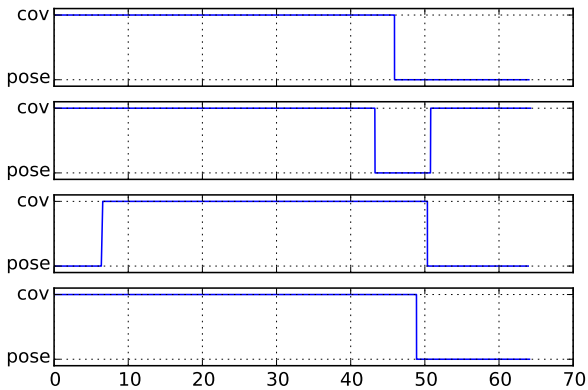

Figure 7.8: Discrete state in Controller 7.3 for each sensing agent during the simulation described in Section 7.6: pose for q^1, corresponding to (7.6); cov for q^0, corresponding to (7.18)).

From Figure 7.9, we can see how the agents adjust their orientation to follow the local curvature of surface.

7.7 Preliminary experimental evaluation

A preliminary experimental evaluation of the proposed controller has been carried out in collaboration with the FROST Lab at LTU. In this experiment, we consider the case of one UAV inspecting a cylindrical structure. As a structure to inspect, we use a cylinder-shaped fountain located on the grounds of LTU. As a sensing agent, we employ the NEO hexacopter, which has been already described in Section 6.10. The outdoor localization of the UAV is obtained by means of a system of ultrawideband transreceivers. The system measures the distance of the UAV from an array of anchors, and then uses trilateration of filtering to infer the position of the UAV within an assigned reference frame.

The experiment is performed as follows. We manually generate approximately 20 landmarks distributed around the body of the fountain in a spiraling fashion. Then, we run a preliminary version of the proposed controller to generate a reference trajectory for the UAV. Finally, we feed the reference trajectory to the UAV.

Given the preliminary nature of this evaluation, there is no associated dataset available. However, we observe that the UAV circumnavigates the body of the fountain as prescribed. A snapshot of the experiment is shown in Figure 7.10.

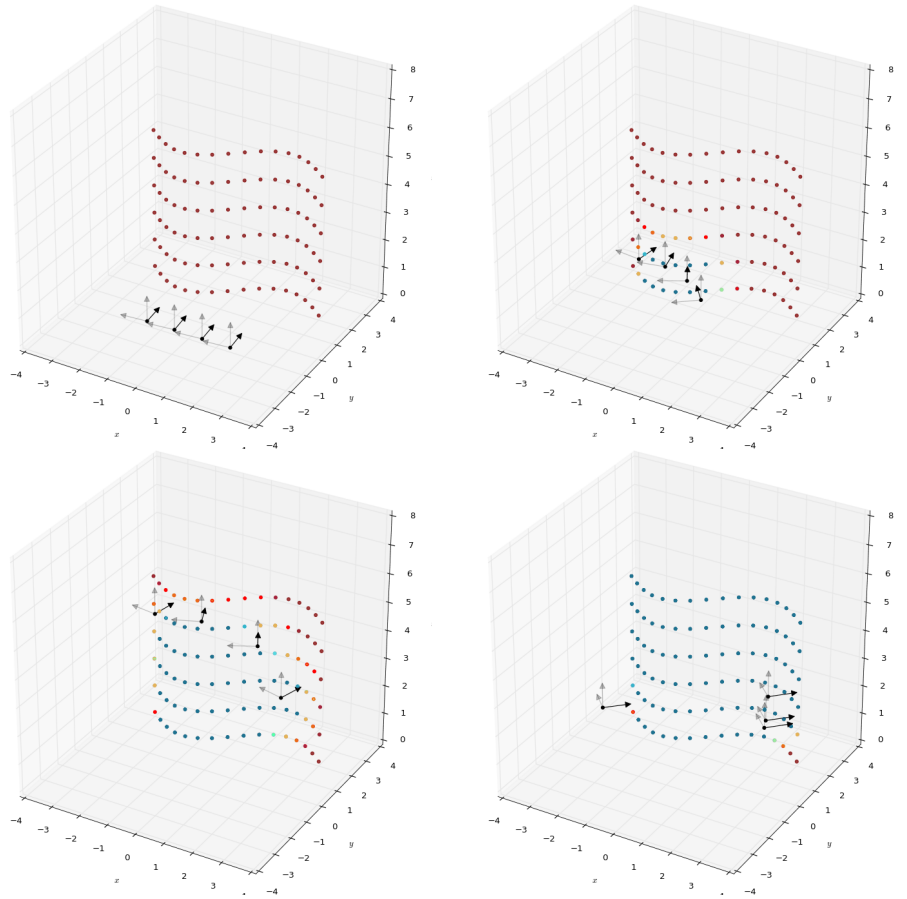

Figure 7.9: Position and orientations of the sensing agents during the simulation described in Section 7.6. Times from left to right: 0, 20, 40, 60.

7.8 Summary

In this chapter, we have proposed a distributed control algorithm for the inspection of a 3D surface with a team of mobile sensing agents with generic, heterogeneous sensing patterns. The controller is based on intermittent connections of the agents with a cloud repository, which eliminates the need for inter-agent communication. We have shown that the controller guarantees that the inspection is completed in finite time. We have validated the proposed controller in a simulation, and we have described a preliminary experimental evaluation on an aerial robotic platform.

Figure 7.10: A snapshot of the preliminary experimental evaluation described in Section 7.7.

Chapter 8

Conclusions and future work

Garzoncello scherzoso,
cotesta età fiorita
è come un giorno d'allegrezza pieno,
giorno chiaro, sereno,
che precorre alla festa di tua vita.

G. Leopardi,
Il sabato del villaggio, *vv. 43–47.*

As technology advances, networks of interconnected devices become more common and more pervasive, and, at the same time, more complex and more challenging to analyze and control. When controlling a network system, the aim is usually to make a desired collective behavior arise from the individual dynamics of each agent in the system and from local interactions among the agents. In most realistic applications, the communication between one agent and the rest of the network is intermittent, rather than continuous. Moreover, in some modern control infrastructures, a set of distributed devices with limited computational and communication power are connected to a larger shared unit, which can either be a simple information repository or have some computational capabilities.

In this thesis, we have studied several control strategies to achieve coordination in network systems using intermittent communication. In some cases, we have considered a scenario where a set of devices connect to a shared information repository, rather than communicating with each other directly.

Naturally, there remain many challenging open questions on the topic of multi-agent coordination with limited communication.

In the rest of this chapter, we summarize the results that we have obtained in Chapters 3 to 7, and we discuss possible directions for future research.

8.1 Conclusions

In Chapter 3, we have derived a decentralized event-based protocol for pinning control of networks of nonlinear systems with switching topologies. The protocol prescribes piecewise constant control signals. Each agent in the network updates its control signal when a given condition regarding the state of its neighborhood is verified. Namely, a function of the state of the neighborhood of the agent is compared to a given threshold function. When the state function overcomes the threshold function, a control update is triggered. To avoid continuous inter-agent communication, each agent can estimate the state of its neighborhood with a predictor, and broadcast its control input to its neighbors upon the control updates. We have quantified the pinning controllability of the network in terms of the local dynamics of the agents, of the variations in the network topology, and of the variations in the number and location of the pinned nodes. We have shown that, if the parameter that quantifies the pinning controllability of the network is large enough with respect to the Lipschitz constant of the dynamics of the agents, and to the convergence rate of the threshold function, then the network converges to the reference trajectory asymptotically without exhibiting Zeno behavior in the sequences of the control updates. We have shown also that the convergence rate of the threshold function is a lower bound for the convergence rate of the network to the reference trajectory. Hence, the convergence rate of the threshold function can be used as a parameter in the control design. Namely, a larger convergence rate leads to faster tracking, but tends to induce a larger number of control updates. However, as long as the convergence rate of the threshold function is chosen below a value that scales linearly with the pinning controllability of the network, the closed-loop system does not exhibit Zeno behavior.

In Chapter 4, we have derived a control algorithm for multi-agent networks where inter-agent communication is completely substituted by the exchange of data over a shared repository hosted on a cloud. In the proposed algorithm, each agent schedules its own accesses to the cloud in a recursive fashion, and independently of the other agents. When an agent accesses the cloud, it uses the information that it has downloaded to schedule the next access. The scheduling is based on the comparison of a time-varying function of the data downloaded from the repository with a time-varying threshold function. Namely, when an agent accesses the repository, it schedules the following access at the earliest time when the function of the downloaded data overcomes the threshold function. The convergence rate of the threshold function is a parameter in the control design, and acts as a lower bound on the convergence rate of the closed-loop system. The proposed setup has been applied to a formation problem of mobile agents with second-order dynamics sub-

ject to disturbances. We have shown that the proposed algorithm achieves bounded convergence if the disturbances are persistent, and asymptotic convergence if the disturbances slowly vanish. In both cases, we have shown that the sequence of the accesses to the cloud repository by each agent does not exhibit Zeno behavior, as long as the the convergence rate of the threshold function is smaller than a constant that depends on the topology of the information exchange through the cloud.

In Chapter 5, we have designed a cloud-supported control algorithm to attain localization and circumnavigation of an unknown target with a network of mobile agents. We have proposed a scenario where the agents have limited communication capabilities, and, to attain the desired circumnavigation, they may measure the bearing of the target to circumnavigate and access a shared information repository hosted on a cloud. Both the bearing measurements and the accesses to the cloud are modeled as discrete events, which are performed intermittently by each agent and asynchronously by different agents. We have designed event-triggered and self-triggered rules to schedule the bearing measurements and the cloud accesses performed by the agents. The rule to schedule the cloud accesses is similar to that designed in Chapter 4, but it is tailored to the kinematic model of the agents presented in this chapter. We have investigated how the proposed scheduling rules attain the desired circumnavigation objective, both analytically and with simulations. Finally, we have presented a preliminary setup for experimental evaluation of the proposed algorithm.

In Chapter 6, we have designed a family of distributed protocols for coverage control of anisotropic sensor networks. The environment or the structure that the sensors are required to survey is abstracted into a finite set of landmarks, where each landmark is the abstraction of a point or a small area of interest. Differently than the existing works on the coverage problem, we let each landmark be defined by a position and an orientation, where the orientation of the landmark carries information about the normal to the surface to inspect in the vicinity of the landmark itself. Each sensing agent is initially assigned a subset of the landmarks to monitor. To improve the coverage of the environment, each sensor can change its position and orientation, but also transfer the ownership of some of its landmarks to another agent that has a better perception of those landmarks. In the proposed control algorithm, the control of the motion of each agent is designed as a simple gradient climb of the coverage attained by that sensor, with respect to the landmarks that are assigned to it. Communication among different agents is used to transfer the responsibility for some landmarks from one agent to another, where the latter has a better perception of those landmarks. Such communication instances can be considered instantaneous events, and they are always pairwise. We have generalized the classical definition of Voronoi tessellation to describe locally optimal configurations of the positions and orientations of the agents and of the distribution of the landmarks among the agents. Then, we have shown that the proposed control algorithm drives the network to a configuration corresponding to a Voronoi tessel-

lation. We have discussed a possible extension of the proposed algorithm to handle collision avoidance between two different agents, as well as between one agent and one exogenous body. The proposed algorithm has been validated by means of simulations as well as experiments with aerial robotic platforms. Although we have not demonstrated these features directly, the designed control scheme allows for the use of sensing agents that have different footprints from each other, and behaves robustly to the injection of new agents in the network and the removal of some agents from the network.

In Chapter 7, we have designed a distributed, cloud-supported control algorithm to attain the inspection of a 3D structure with a network of autonomous sensing agents. The structure to inspect is abstracted into a finite set of landmarks, in a similar way as described in Chapter 6. We have described a hybrid controller to steer the pose of a sensing agent to a desired target pose, and we have shown how such a controller can be used to have the inspection completed by a single agent. Then, we have designed a mechanism to store information about the progress of the inspection in a cloud repository, by having a sensing agent access the cloud intermittently and asynchronously. We have shown how the use of the cloud repository can be used to reduce redundant inspection when the inspection mission is assigned to two or more agents. We have proven formally that, with the proposed control design, the inspection is completed in finite time. The theoretical results have been validated by a numerical simulations, and we have executed a preliminary experimental evaluation on an aerial robotic platform.

Our overall result can be summarized as follows. We have found that several coordinated behaviors of a network of autonomous agents (behaviors that are traditionally attained by means of continuous communication among the agents) can also be attained by means of intermittent, event-triggered, and asynchronous communication. Some of these behaviors can also be attained or by substituting agent-to-agent communication with the asynchronous exchange of information over a cloud repository. Remarkably, we have observed that the performances attained by a network with intermittent communication can be made as close as desired to those attained by a similar network where continuous communication is allowed. Performances that we have considered are, for example, the convergence rate of the network to a synchronized state, or the property that a network of sensing agent converges to a Voronoi tessellation. Performances are affected by a number of parameters that can be appropriately tuned. In some cases, driving the performances to the same level as attained by a network with continuous communication implies that the amount of communication instances required grows unbounded.

The coordinated behaviors that we have investigated directly are pinning control, formation control, coverage control, and effective coverage control. These behaviors have immediate applications in robotics, such as formation control for robot swarms, autonomous surveillance, and autonomous inspection.

For the coverage and effective coverage behaviors, we have also found that using intermittent communication schemes makes it easier to generalize existing algorithms so that they are applicable to a wider variety of scenarios, such as the surveillance or inspection of a 3D structure with anisotropic sensing agents.

All our results are validated by numerical simulations. The results on circumnavigation, coverage, and effective coverage have also been subjected to preliminary experimental evaluation on aerial robotic platforms. We have observed that the proposed controllers for coverage and effective coverage lend themselves organically to implementation as ROS nodes. Implementing the controllers as ROS nodes has allowed us to use them both as simulated controllers and controllers for the aerial platforms, without any modification to the scripts being necessary, if not for parameter tuning.

8.2 Future work

In Chapter 3, we have designed an event-triggered algorithm for pinning synchronization which works under the assumption that the network is connected to the reference trajectory sufficiently often. When dealing with control protocols for coordination in a multi-agent network, one always needs to make some assumptions on the connectedness of the network topology. However, we have only proved that these assumptions are sufficient conditions for asymptotic convergence of the network to the reference trajectory, but not that they are necessary. Therefore, it would be interesting to study whether the same event-triggered protocol, or a similar one, can be used to attain pinning controllability under even less conservative connectivity assumptions. For example, we may consider *jointly connected* networks. A jointly connected network may be disconnected at any individual time instant, but there exists a positive duration that, if one considers the union of the edges that are present in the graph over any time interval of that duration, then one obtains a connected graph. Other possible extensions of the results obtained in Chapter 3 involve considering networks with directed topologies and more general control protocols than diffusive coupling.

In Chapter 4, we have presented a general framework for cloud-supported multi-agent coordination, but we have derived a control law and access scheduling rule only for the case when the control objective can be formulated as a consensus of the states of second-order agents. Viable future research may address more general coordination objectives for the proposed framework. Another research challenge arises if we attribute a cost to each cloud access, and we try to account for such cost explicitly in the control design. For example, one may consider an LQR cost function and incorporate an additional impulsive term to penalize the cloud accesses of each agent.

In Chapter 5, we have presented a cloud-supported control framework that drives a

network of mobile agents to circumnavigate a target while forming a regular polygon around the target. The proposed algorithm is based on event-triggered bearing measurements of the position of the target. The same control objective can be considered in the scenario that the agents have access to event-triggered measurements of their distance from the target. Collision avoidance between different agents is an important feature in realistic settings. The circumnavigation algorithm that we have proposed in Chapter 5 does not handle collision avoidance explicitly, and future work may address the incorporation of a collision avoidance scheme that preserves the guarantee to reach a polygonal formation around the target.

Coverage control is widely open to improvements. The existing algorithms based on Voronoi tessellations, including the algorithm that we have designed in Chapter 6, converge to a local optimum of the cost function that represents the coverage achieved by the sensor network, and the research ground is open for solutions that try to avoid this type of equilibria. Depending on the footprints of the sensing agents and on the control algorithm, a single agent may also converge to a local optimum of the cost function that represents the coverage achieved by that sensor. Future research may focus on a footprint design that allows a sensor to avoid the local optima of its own coverage function. Since the footprint of a sensing agent is a function of both the position and the orientation of the agent, the design of a footprint that escapes local optima should relate to the concept of *geodesic convexity*. Geodesic convexity is a generalization of convexity that takes into account the possibility that a function is defined on the sphere, or on a subset of the sphere. For most practical applications of coverage control, it is important to incorporate a collision avoidance scheme. Collision avoidance is achieved as a side effect in Voronoi-based coverage control for sensors with omnidirectional footprints, but this benefit does not extend to networks of sensor with general footprints. A research challenge is to design a collision avoidance scheme which preserves as many properties as possible of the original control algorithm, such as the monotonic improvement of the global coverage function, and the intermittent nature of the communication between different agents.

In Chapter 7, we have introduced a distributed and cloud-supported control framework for multi-agent inspection control. This framework does not handle collision avoidance (neither between different agents nor between an agent and an external body), and collision avoidance is assumed to be cared for at a lower level of the control design. Future work may investigate a possible variant of the proposed control design which handles collision avoidance directly while still guaranteeing that the inspection is completed in finite time. Another possible future development of this work may address the optimality of the trajectories undertaken by agents during the inspection. In fact, a trajectory can be assigned a cost in terms of its length or of its duration, and the cost of the inspection can be defined as the sum of the cost of the paths undertaken by each agent involved. We may then try to design a controller that minimizes the inspection cost, or at least a controller that offers

guarantees in terms of upper bounds for the inspection cost. To make the problem more tractable, we may introduce one more degree of discretization by abstracting the space into a finite set of discrete locations for the agents, so that a trajectory can be defined as a sequence of locations over time. Then, given a set of trajectories that completes the inspection, one may use gradient-descent or learning techniques to modify the trajectories and attain a reduced cost.

Finally, we mention a few possibilities for future work that transcend the specificities of each single scenario.

Generally speaking, in this thesis we have designed algorithms for networks where the number of agents is arbitrary. The convergence properties of the proposed algorithms have been demonstrated to hold for any number of agents. However, all numerical simulations and experimental validation have been conducted with only few agents. Hence, at least from an implementation perspective, scalability of the proposed algorithms remains worth investigating. In particular, unmodeled aspects such as the time required to pass information between devices may have a more tangible impact on the collective behavior of the network when the number of agents is large.

Similar observations can be made about resilience. The coordination algorithms proposed in this thesis all exhibit some resilience properties, in the sense that, if an agent should be added to the network or removed from the network, the new system would find itself in a valid initial condition to reach the desired coordination. However, the existing agents in the network need to be informed that a new agent has been injected or that one agent has been removed. In some cases, a removed agent may be holding some data that needs to be transfered to the remaining agents. In the numerical simulations presented in this thesis, these features are handled at the simulation level, which roughly corresponds to a centralized solution. The implementation of these features in a decentralized fashion remains an open problem.

The algorithms presented in the thesis each employ a number of numerical constants, whose value often has a macroscopic influence the execution of the algorithm. We have usually chosen the values of these parameters according to intuition or trial-and-error. In some cases, the meaning of these parameters, and their influence on the collective behavior of the network has been discussed thoroughly, but such analysis has not led to a systematic tuning method. Hence, systematic tuning of the parameters of the distributed controllers presented in the thesis is left as a possible future development.

Bibliography

Godi, fanciullo mio; stato soave,
stagion lieta è cotesta.
Altro dirti non vo'; ma la tua festa
ch'anco tardi a venir non ti sia grave.

G. Leopardi,
Il sabato del villaggio*, vv. 48–51.*

J. A. Acebron, L. L. Bonilla, C. J. Vicente, F. Ritort, and R. Spigler. The Kuramoto model: a simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, 77(1):137–185, 2005.

A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Event-triggered pinning control of complex networks with switching topologies. *IEEE Conference on Decision and Control*, 2014.

A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Event-triggered pinning control of switching networks. *IEEE Transactions on Control of Network Systems*, 2(2):204–213, 2015a.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Control of multi-agent systems with event-triggered cloud access. *European Control Conference*, 2015b.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Multi-agent trajectory tracking with event-triggered cloud access. *IEEE Conference on Decision and Control*, 2016.

A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Hybrid coverage and inspection control for anisotropic mobile sensor teams. *IFAC World Congress*, 2017a.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Coordination of multi-agent systems with intermittent access to a cloud repository. T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, editors, *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*. Springer, 2017b.

A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Cloud-supported formation control of second-order multi-agent systems. *IEEE Transactions on Control of Network Systems*, Online, 2017c.

A. Adaldo, S. S. Mansouri, C. Kanellakis, D. V. Dimarogonas, K. H. Johansson, and G. Nikolakopoulos. Cooperative coverage for surveillance of 3D structures. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017d.

A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Cloud-supported effective coverage of 3D structures. *European Control Conference*, 2018.

A. Anta and P. Tabuada. To sample or not to sample: self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou. Synchronization in complex networks. *Physics Reports*, 469(3):93–153, 2008.

T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, 2009.

M. Barahona and L. M. Pecora. Synchronization in small-world systems. *Physical Review Letters*, 89(5), 2002.

I. V. Belykh, V. N. Belykh, and M. Hasler. Blinking model and synchronization in small-world networks with a time-varying coupling. *Physica D: Nonlinear Phenomena*, 195:188–206, 2004.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424:175–308, 2006.

A. Boccia, A. Adaldo, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Tracking a mobile target by multi-robot circumnavigation using bearing measurements. *IEEE Conference on Decision and Control*, 2017.

S. L. Bowman, C. Nowzari, and G. J. Pappas. Coordination of multi-agent systems via asynchronous cloud communication. *IEEE Conference on Decision and Control*, 2016.

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

F. Bullo, R. Carli, and P. Frasca. Gossip coverage control for robotic networks: dynamical systems on the space of partitions. *SIAM Journal on Control and Optimization*, 50(1):419–447, 2012.

R. Carli, F. Fagnani, P. Frasca, and S. Zampieri. Gossip consensus algorithms via quantized communication. *Automatica*, 46(1):70–80, 2010.

T. L. Carroll and L. M. Pecora. Synchronizing chaotic circuits. *IEEE transactions on circuits and systems*, 38(4):453–456, 1991.

C. G. Cassandras and W. Li. Sensor Networks and Cooperative Control. *European Journal of Control*, 11(4-5):436–463, 2005.

C. Cavaliere, D. Mariniello, A. Adaldo, F. Lo Iudice, D. V. Dimarogonas, K. H. Johansson, and M. di Bernardo. Cloud-supported self-triggered control for multi-agent circumnavigation. *Submitted to the IEEE Conference on Decision and Control*, 2018.

T. Chen, X. Liu, and W. Lu. Pinning complex networks by a single controller. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(6):1317–1326, 2007.

J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

C. De Persis and P. Frasca. Robust self-triggered coordination with ternary controllers. *IEEE Transactions on Automatic Control*, 58(12):3024–3038, 2013.

M. Deghat, I. Shames, B. D. O. Anderson, and L. Fellow. Localization and circumnavigation of a slowly moving target using bearing measurements. *IEEE Transactions on Automatic Control*, 59(8):2182–2188, 2014.

M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

D. Della Penda. *Device-to-Device Communication in Future Cellular Networks*. Doctoral thesis, KTH Royal Institute of Technology, 2018. URL `http://kth.diva-portal.org/smash/get/diva2:1187546/FULLTEXT01.pdf`.

R. Dietsel. *Graph Theory*. Springer, 2016.

D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57 (5):1291–1297, 2012.

S. Dormido, D. V. Dimarogonas, J. Sánchez, K. H. Johansson, and M. Guinaldo. Distributed event-based control strategies for interconnected linear systems. *IET Control Theory {&} Applications*, 7(6):877–886, 2013.

Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics*, 28(2):364–378, 2012.

Y. Fan, L. Liu, G. Feng, and Y. Wang. Self-triggered consensus for multi-agent systems with Zeno-free triggers. *IEEE Transactions on Automatic Control*, 60 (10):2779–2784, 2015.

E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni. Multi-AUV control and adaptive sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

E. Garcia and P. J. Antsaklis. Model-based event-triggered control for systems with quantization and time-varying network delays. *IEEE Transactions on Automatic Control*, 58(2):422–434, 2013.

E. Garcia, Y. Cao, and D. W. Casbeer. Decentralized event-triggered consensus with general linear dynamics. *Automatica*, 50(10):2633–2640, 2014.

R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability and Robustness*. Princeton Univerisity Press, 2012.

R. Gould. *Graph Theory*. Dover Publications, 2012.

R. O. Grigoriev, M. C. Cross, and H. G. Schuster. Pinning control of spatiotemporal chaos. *Physical Review Letters*, 79(15):2795–2798, 1997.

A. Gusrialdi, S. Hirche, T. Hatanaka, and M. Fujita. Voronoi based coverage control with anisotropic sensors. *American Control Conference*, 2008.

P. Haffner and S. Venkataraman. Unmanned aerial vehicle, 2012.

M. T. Hale and M. Egerstedt. Differentially private cloud-based multi-agent optimization with constraints. *American Control Conference*, 2015.

W. P. M. H. Heemels, J. H. Sandee, and P. P. Van Den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4): 571–590, 2008.

W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. *IEEE Conference on Decision and Control*, 2012.

T. Henningsson, E. Johannesson, and A. Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44:2890–2895, 2008.

M. Heymann, F. Lin, G. Meyer, and S. Resmerita. Analysis of Zeno behaviors in a class of hybrid systems. *IEEE Transactions on Automatic Control*, 50(3): 376–383, 2005.

I. I. Hussein and D. M. Stipanovic. Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance. *IEEE Transactions on Control Systems Technology*, 15(4):642–657, 2007.

K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of Zeno hybrid automata. *Systems & Control Letters*, 38:141–150, 1999.

S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse. State-of-the-art technologies for UAV inspections. *IET Radar, Sonar & Navigation*, 12(2):151–164, 2018.

Y. Kantaros, M. Thanou, and A. Tzes. Distributed coverage control for concave areas by a heterogeneous Robot-Swarm with visibility sensing constraints. *Automatica*, 53:195–207, 2015.

H. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

T.-H. Kim and T. Sugie. Cooperative control for target-capturing task based on a cyclic pursuit strategy. *Automatica*, 43:1426–1431, 2007.

A. Kwok and S. Martinez. Unicycle coverage control via hybrid modeling. *IEEE Transactions on Automatic Control*, 55(2):528–532, 2010.

J. P. LaSalle. *The Stability and Control of Discrete Processes*. Springer, 1986.

J. Le Ny and G. J. Pappas. Adaptive deployment of mobile robotic networks. *IEEE Transactions on Automatic Control*, 58(3):654–666, 2013.

X. Li, X. Wang, and G. Chen. Pinning a complex dynamical network to its equilibrium. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(10): 2074–2087, 2004.

B. Liu, T. Chu, L. Wang, and G. Xie. Controllability of a leader-follower dynamic network with switching topology. *IEEE Transactions on Automatic Control*, 53 (4):1009–1013, 2008.

Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. Controllability of complex networks. *Nature*, 473:167–173, 2011.

D. Liuzza, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Distributed model based event-triggered control for synchronization of multi-agent systems. *IFAC Symposium on Nonlinear Control Systems*, Toulose, France, 2013.

D. Liuzza, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Distributed model based event-triggered control for synchronization of multi-agent systems. *Automatica*, 73:1–7, 2016.

S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

W. Lu. Adaptive dynamical networks via neighborhood information: Synchronization and pinning control. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 17, 2007.

J. Lygeros, K. Johansson, S. Simic, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

D. Mariniello, A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Cooperative circumnavigation with event-triggered bearing measurements, 2017. URL `https://zenodo.org/record/1067181`.

J. A. Marshall, M. E. Broucke, and B. A. Francis. Formations of vehicles in cyclic pursuit. *IEEE Transactions on Automatic Control*, 49(11):1963–1974, 2004.

J. A. Marshall, M. E. Broucke, and B. A. F. Macdonald. Pursuit formations of unicycles. *Automatica*, 42:3–12, 2006.

S. Martinez, J. Cortes, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007.

T. Matsumoto. A Chaotic Attractor from Chua's Circuit. *IEEE Transactions on Circuits and Systems*, 31(12):1055–1058, 1984.

M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton Univerisity Press, 2010.

G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38 (8), 1965.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

C. Nowzari and J. Cortes. Self-triggered coordination of robotic networks for optimal deployment. *Automatica*, 48:1077–1087, 2012.

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

R. Olfati-saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

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

D. Panagou, D. M. Stipanovic, and P. G. Voulgaris. Distributed dynamic coverage and avoidance control under anisotropic sensing. *IEEE Transactions on Control of Network Systems*, 2016.

R. Patel, P. Frasca, J. W. Durham, R. Carli, and F. Bullo. Dynamic partitioning and coverage control with asynchronous one-to-base-station communication. *IEEE Transactions on Control of Network Systems*, 3(1):5589–5594, 2016.

L. Paull, S. Saeedi, M. Seto, and H. Li. AUV navigation and localization: A review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, 2014.

M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Distributed algorithms for environment partitioning in mobile robotic networks. *IEEE Transactions on Automatic Control*, 56(8):1834–1848, 2011.

L. M. Pecora and T. L. Carroll. Synchronization in chaotic systems. *Physical Review Letters*, 64(8):821–824, 1990.

L. M. Pecora and T. L. Carroll. Master stability functions for synchronized coupled systems. *Physical Review Letters*, 80(10):2109–2112, 1998.

L. M. Pecora, T. L. Carroll, G. A. Johnson, D. J. Mar, and J. F. Heagy. Fundamentals of synchronization in chaotic systems, concepts, and applications. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 7(4):520–543, 1997.

M. Porfiri and M. di Bernardo. Criteria for global pinning-controllability of complex networks. *Automatica*, 44(12):3100–3106, 2008.

M. Porfiri and F. Fiorilli. Node-to-node pinning control of complex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19, 2009.

R. Postoyan, A. Anta, W. P. M. H. Heemels, P. Tabuada, and D. Nesic. Periodic event-triggered control for nonlinear systems. *IEEE Conference on Decision and Control*, 2013.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg. ROS: an open-source robot operating system. *IEEE International Conference on Robotics and Automation*, 2009.

W. Ren. Consensus based formation control strategies for multi-vehicle systems. *2006 American Control Conference*, 2006.

W. Ren. Information consensus in multi-vehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

W. Ren and R. W. Beard. *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008.

Scania AB. The Scania report 2017. Technical report, 2017.

R. Sepulchre, D. A. Paley, and N. E. Leonard. Stabilization of planar collective motion: All-to-all communication. *IEEE Transactions on Automatic Control*, 52 (5):811–824, 2007.

G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, 2013.

I. Shames, B. Fidan, and B. D. O. Anderson. Close target reconnaissance with guaranteed collision avoidance. *International Journal of Robust and Nonlinear Control*, 21:1823–1840, 2011.

I. Shames, S. Dasgupta, B. Fidan, and B. D. O. Anderson. Circumnavigation using distance measurements under slow drift. *IEEE Transactions on Automatic Control*, 57(4):889–903, 2012.

C. Song, G. Feng, Y. Fan, Y. Wang, C. Song, G. Feng, Y. Fan, and Y. Wang. Decentralized adaptive awareness coverage control for multi-agent networks. *Automatica*, 47:2749–2756, 2011.

C. Song, L. Liu, G. Feng, Y. Wang, and Q. Gao. Persistent awareness coverage control for mobile sensor networks. *Automatica*, 49:1867–1873, 2013.

Q. Song, J. Cao, and W. Yu. Second-order leader-following consensus of nonlinear multi-agent systems via adaptive pinning control. *Systems & Control Letters*, 59: 553–562, 2010.

M. Sposato. *Multiagent Cooperative Coverage Control*. Master thesis, KTH Royal Institute of Technology, 2016. URL `http://kth.diva-portal.org/smash/get/diva2:931263/FULLTEXT01.pdf`.

M. Sposato, A. Adaldo, D. V. Dimarogonas, and K. H. Johansson. Anisotropic coverage control, 2017. URL `https://zenodo.org/record/1067172`.

Y. Stergiopoulos and A. Tzes. Spatially distributed area coverage optimisation in mobile robotic networks with arbitrary convex anisotropic patterns. *Automatica*, 49(1):232–237, 2013.

Y. Stergiopoulos, M. Thanou, and A. Tzes. Distributed collaborative coverage-control schemes for non-convex domains. *IEEE Transactions on Automatic Control*, 60(9):2422–2427, 2015.

S. H. Strogatz. From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143:1–20, 2000.

S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001.

J. O. Swartling, I. Shames, K. H. Johansson, and D. V. Dimarogonas. Collective circumnavigation. *Unmanned Systems*, 2(3):219–229, 2014.

P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

P. V. Teixeira, D. V. Dimarogonas, and K. H. Johansson. Multi-agent coordination with event-based communication. *American Control Conference*, 2011.

V. Turri. *Look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning*. Doctoral thesis, KTH Royal Institute of Technology, 2018. URL `https://www.diva-portal.org/smash/get/diva2:1197579/FULLTEXT01.pdf`.

A. van der Shaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, 2000.

X. F. Wang and G. Chen. Pinning control of scale-free dynamical networks. *Physica A*, 310:521–531, 2002.

Y. Wang and I. I. Hussein. Awareness coverage control over large-scale domains with intermittent communications. *IEEE Transactions on Automatic Control*, 55(8):1850–1859, 2010.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

J. Wei, A. R. Everts, M. K. Camlibel, and A. J. van der Schaft. Consensus dynamics with arbitrary sign-preserving nonlinearities. *Automatica*, 83:226–233, 2017a.

J. Wei, S. Zhang, A. Adaldo, X. Hu, and K. H. Johansson. Finite-time attitude synchronization with a discontinuous protocol. *IEEE International Conference on Control & Automation (ICCA)*, 2017b.

J. Wei, S. Zhang, A. Adaldo, J. Thunberg, X. Hu, and K. H. Johansson. Finite-time attitude synchronization with a distributed discontinuous protocol. *IEEE Transactions on Automatic Control*, Online, 2018.

S. I. Williams. Wind turbine inspection, 2010. URL `https://patentimages.storage.googleapis.com/23/71/23/71cc6f43d0da3a/US20100103260A1.pdf`.

H. S. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, 1966.

W. Wu, W. Zhou, and T. Chen. Cluster synchronization of linearly coupled complex networks under pinning control. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(4):829–839, 2009.

W. Xia and J. Cao. Pinning synchronization of delayed dynamical networks via periodically intermittent control. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(1), 2009.

D. Zelazo and M. Mesbahi. Edge agreement: Graph-theoretic performance bounds and passivity analysis. *IEEE Transactions on Automatic Control*, 56(3):544–555, 2011.

Z. Zeng, X. Wang, and Z. Zheng. Convergence analysis using the edge Laplacian: Robust consensus of nonlinear multi-agent systems via ISS method. *International Journal of Robust and Nonlinear Control*, 26:1051–1072, 2015.

M. Zhong and C. G. Cassandras. Distributed coverage control and data collection with mobile sensor networks. *IEEE Transactions on Automatic Control*, 56(10): 2445–2455, 2011.

J. Zhou, J.-A. Lu, and J. Lü. Pinning adaptive synchronization of a general complex dynamical network. *Automatica*, 44:996–1003, 2008.