Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
AUTOMATICA E RICERCA OPERATIVA

Ciclo XXIV

Settore Concorsuale di Afferenza: 09/G1
Settore Scientifico Disciplinare: ING–INF/04

NONLINEAR CONTROL STRATEGIES FOR COOPERATIVE CONTROL OF MULTI–ROBOT SYSTEMS

Presentata da: **Lorenzo Sabattini**

Coordinatore:
Prof. Alberto Caprara

Relatore:
Prof. Claudio Melchiorri

Esame finale anno 2012

Abstract

This thesis deals with distributed control strategies for cooperative control of multi–robot systems. Specifically, distributed coordination strategies are presented for groups of mobile robots.

The formation control problem is initially solved exploiting artificial potential fields. The purpose of the presented formation control algorithm is to drive a group of mobile robots to create a completely arbitrarily shaped formation. Robots are initially controlled to create a regular polygon formation. A bijective coordinate transformation is then exploited to extend the scope of this strategy, to obtain arbitrarily shaped formations. For this purpose, artificial potential fields are specifically designed, and robots are driven to follow their negative gradient.

Artificial potential fields are then subsequently exploited to solve the coordinated path tracking problem, thus making the robots autonomously spread along predefined paths, and move along them in a coordinated way.

Formation control problem is then solved exploiting a consensus based approach. Specifically, weighted graphs are used both to define the desired formation, and to implement collision avoidance. As expected for consensus based algorithms, this control strategy is experimentally shown to be robust to the presence of communication delays.

The global connectivity maintenance issue is then considered. Specifically, an estimation procedure is introduced to allow each agent to compute its own estimate of the algebraic connectivity of the communication graph, in a distributed manner. This estimate is then exploited to develop a gradient based control strategy that ensures that the communication graph remains connected, as the system evolves. The proposed control strategy is developed initially for single–integrator kinematic agents, and is then extended to Lagrangian dynamical systems.

Acknowledgements

This thesis is the result of several years of work, and I need to thank who made it possible.

I would like to thank my advisor, Prof. Claudio Melchiorri, for giving me this opportunity.

Many thanks go to Dr. Cristian Secchi and Prof. Cesare Fantuzzi, for their teachings, their advices and their help.

I'd like to thank Dr. Nikhil Chopra for his support during my six months at the University of Maryland.

I would have never been able to start studying without the support of my parents: I really need to thank them for their support.

And, of course, my greatest thanks go Francesca, my wife. Thanks for being there.

Contents

Chapter 1

Introduction

Cooperative multi–robot systems have several advantages, over single–robot systems [1, Chapters 40–41]:

- multiple robots can accomplish tasks that are too complex for a single robot,
- different robots can gather together complementary abilities,
- some tasks are inherently distributed, in space or time,
- multiple robots can work simultaneously, thus solving parallelizable problems in less time,
- redundancy provided by multiple robots may increase the overall robustness of the system.

These are among the main reasons why multi–robot systems have been intensively studied, in the last few decades.

A specific class of robots that are often exploited in cooperative system is that of mobile robots. Groups of mobile robots can be exploited in several applications. Main examples are search–and–rescue operations [2–4], cooperative transportation [5, 6], exploration of unknown terrains [7–9], service operation in domestic [10, 11] or industrial environment [12–14], military tasks [15, 16].

Generally speaking, control strategies for multi–robot systems may be divided into two categories: centralized and distributed:

When adopting a *centralized* control architecture (Fig. 1.1a), a central computation unit gathers information from all the members of the team. Subsequently, the central unit sends the desired control input to all the team members.

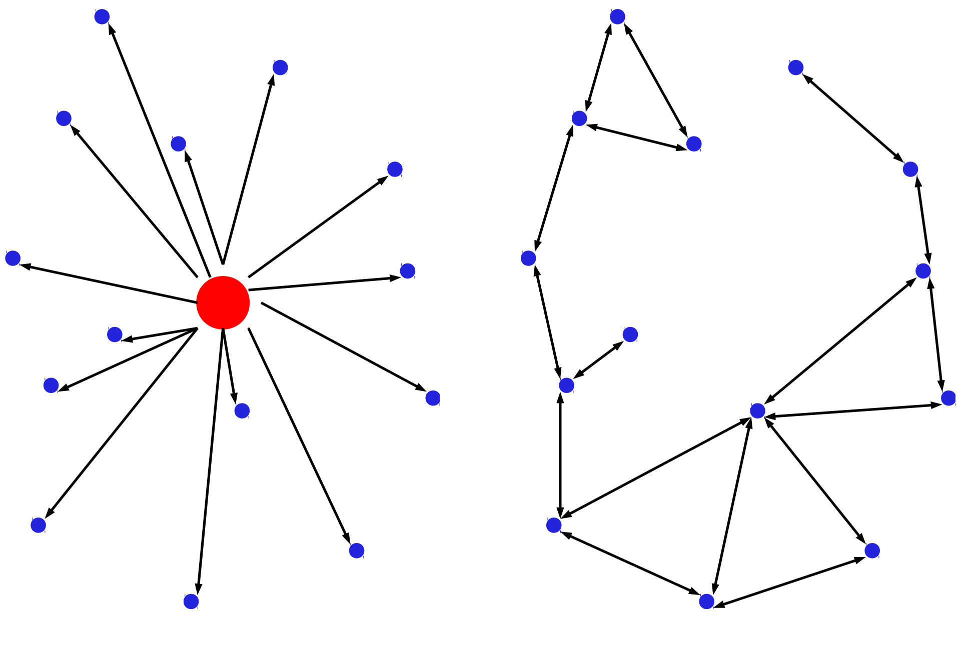

(a) Centralized architecture (b) Distributed architecture

Figure 1.1: Centralized and distributed control architectures

Conversely, in a *distributed* control architecture (Fig. 1.1b), each entity of the group computes its own control input, based on information acquired locally, from its neighbors.

While a centralized architecture generally yields to an easier design of the control algorithm, it clearly lacks of robustness [17]. In fact, in the case of failure of the central entity, the group is no longer able to perform the task. Conversely, in a distributed architecture, the failure of a single entity doesn't necessarily prevent the completion of the task: generally speaking, the rest of the group may be still able to perform the desired operation, possibly with lower performances. This is the main reason why this work focuses on distributed control strategies.

The idea of having a group of mobile entities cooperating in a distributed manner comes from several examples that can be easily found in the nature. The so called *social animals* are a remarkable example of cooperating entities (Fig. 1.2). Exploiting cooperation, animals are able to fulfill incredibly challenging tasks. Consider insects, like ants or bees: simple entities, with very limited cognitive capabilities, are able to organize themselves in a very complex way, and to complete incredible challenges: ant colonies (Fig. 1.2a) and beehives (Fig: 1.2b) are remarkable examples of complex behaviors, real-

(a) An ant colony

(b) A swarm of bees

(c) A school of fish, hunted by a shark

(d) A pride of lions hunting

Figure 1.2: Examples of social animals

ized by means of the cooperation of simple entities. Another example is represented by school of fish (Fig. 1.2c): coordinating their movements, the animals are able to increase exponentially each one's perception capabilities, without explicit communication. This kind of cooperation helps them in finding food, and avoiding predators. Furthermore, also predators, in some cases, exploit cooperation for hunting, as in the case of lions (Fig. 1.2d).

Considering these fascinating examples, the idea of imitating natural behaviors in robotics appears quite attractive.

Artificial potential fields are one of the main techniques that allow groups of robots to imitate social animals' behaviors. Specifically, control strategies based on artificial potential fields drive robots to move along the negative gradient of the composition of some specifically designed artificial potential fields. The shape of these potential fields encodes the desired behavior. In fact, potential fields can be used to:

- aggregate robots that are too far away from each other,
- avoid collisions among robots,
- avoid collisions with obstacles,
- move robots to the desired position.

Hence, they can be used to imitate the so called *social forces*, the forces that occur among social animals (see [17] and references therein). As observed by the biologists, in fact, the behavior of social animals can be modeled as the composition of some simple behaviors, each of which can be defined by means of an appropriate potential function.

In order to implement any kind of control strategy in a distributed manner, it is necessary for the robots to exchange information with their neighbors. There are basically two different ways of exchanging information: communication and sensing. Generally speaking, *communication* entails a bidirectional information exchange: if the i–th robot can communicate with the j–th one, it is reasonable to assume that the j–th robot can communicate with the i–th one as well. Conversely, an information exchange based on pure *sensing* is generally unidirectional: the fact that the i–th robot can acquire some data about the i–th one doesn't necessarily mean the converse.

Clearly, bidirectional explicit communication is more effective than unidirectional sensing. This is due to the fact that information flows faster through the team of robots. Furthermore, *structured* data (e.g. results of local computations) may be exchanged as well. The main drawback is in the fact that a communication infrastructure is needed, that is computationally demanding. Moreover, typically mobile robots are powered by means of batteries: avoiding the use of communication modules increases the battery life. Another problem is related to the environmental conditions: communication is not always possible, for instance in the presence of radio disturbances, or in the case of long distances among the robots.

The communication architecture among the robots is often modeled as a *graph* [18,19], that is usually referred to as the *communication graph*. Generally speaking, a graph $\mathcal{G}$ represents the interconnection among a set of *nodes*: if two nodes are interconnected, and *edge* exists among them. The *neighborhood* of a node is defined as the set of its *neighbors*, that is the set of nodes to whom it is connected through an edge.

Hence, in multi–robot systems, each robot is represented as a node of the graph, and the link between two robots is represented as an edge of the graph. In order to represent the communication architecture in multi–robot systems, two different classes of graphs may be adopted: directed graphs and undirected graphs.

- In an *undirected graph* the information exchange is bidirectional: for every couple of nodes i and j, if the $i \to j$ edge exists, then the $j \to i$ edge exists as well. Undirected graphs are thus usually exploited to model explicit bidirectional communication among the robots.

- In a *directed graph* the information exchange is unidirectional: for every couple of nodes i and j, the fact that the $i \to j$ edge exists doesn't automatically imply the existence of the $j \to i$ edge. Directed graphs are thus usually exploited to model unidirectional communication among the robots, that may be based on pure sensing.

1.1 Contribution and thesis outline

This thesis focuses on distributed control strategies for cooperative control of multi–robot systems. Each chapter focuses on a specific topic, and starts with an introduction section, that includes a detailed analysis of the state of the art, based on the literature review.

Chapter 2 describes the MORE–pucks *experimental framework*, that is exploited for validating the control strategies described in the subsequent chapters. Specifically, an hardware and software platform is described, that has been designed to implement multi–robot control strategies on a group of e–puck robots. Even though the control software is implemented on a central computer, that communicates via bluetooth with the e–puck robots, distributed implementation of control strategies is emulated, letting each robot exploit only local information. Another experimental framework, based on iRobot Roomba robots, that was first introduced in [20], is described as well.

Chapter 3 introduces distributed control strategies for the coordination of multi–robot systems, first introduced in [21–24]. Specifically, *artificial potential fields* are exploited for formation control purpose. In fact, artificial potential fields are designed to obtain a regular polygon formation, that is subsequently deformed by means of an appropriately designed coordinate transformation, thus obtaining a completely *arbitrarily shaped formation*. This control strategy is proven to be asymptotically stable, and to be local minima free, unlike traditional strategies available in the literature.

Subsequently, the previously designed artificial potential fields are modified, in order to solve a slightly different problem: *coordinated path tracking*. Specifically, as shown also in [25, 26], robots are controlled to move along a predefined closed curve path: in a completely distributed way, without any global synchronization, the composition of appropriately defined artificial potential fields make them spread along the curve and move along it, while keeping the desired speed.

The formation control problem is then tackled exploiting *graph theory* based strategies. In fact, while artificial potential fields are an effective way of solving formation control problem, they have some criticalities: one of the main drawbacks is in the fact that delays in the communication channels drive the system to instability. For this reason, a formation control strategy based on *edge–weighted graphs*, first introduced in [27, 28], is described: specifically, edge–weights are exploited both for formation control and for collision avoidance. In fact, traditional approaches use graph based strategies for formation control only, while artificial potential fields are introduced for collision avoidance. Avoiding the use of artificial potential fields, the proposed control strategy is robust to the presence of delays, as shown in simulations.

To perform a common task in a distributed manner, it is crucial to ensure that information exchange may take place, as the system evolves. For this reason, **Chapter 4** deals with *connectivity maintenance*. Specifically, a distributed control strategy is described to ensure the connectivity of the communication graph. This control strategy was first introduced in [29–32]. In the literature, several approaches to connectivity maintenance have been proposed. These approaches can be divided into two categories: approaches to maintain the local connectivity, and approaches to maintain the global connectivity. Maintaining the local connectivity entails designing a controller that ensures that, if a communication link is active at time $t = 0$, it will be active $\forall t \geq 0$. However, imposing the maintenance of each single communication link is often too restrictive. In fact, to ensure that information exchange among all the robots is possible, it is necessary to guarantee only the *global connectivity* of the communication graph. Loosely speaking, it is acceptable that a few links are broken, as long as the overall graph is still connected: if necessary, redundant links can be removed, and new ones can be introduced. As a measure of the global connectivity of a graph is the second smallest eigenvalue of the Laplacian matrix, the proposed strategy implements a gradient descent of an appropriately designed function of the eigenvalue itself.

In order to implement this strategy, a *distributed estimation procedure* is introduced, to let each robot compute its own estimate of the eigenvalue and of its gradient. The boundedness of the estimation error is shown to be a sufficient condition to ensure connectivity maintenance. The proposed strategy is implemented initially for *single–integrator kinematic agents*, and is then extended to *Lagrangian dynamical systems*, as shown in [33]. The presence of additional external control laws is considered as well.

Appendix A summarizes some of the main results on *graph theory* used throughout the thesis.

Chapter 2

MORE–pucks: a multi–robot experimental framework

This chapter describes an experimental test–bed for multi–robot experiments. More specifically, a software platform is described that allows the user to test control algorithms on a multi–robot experimental setup based on e–puck robots. An overhead camera is exploited for the identification and the localization of the robots, that are moving inside a bounded arena. The software platform has been developed based on open source and cross–platform libraries. Validation experiments for the control strategies described in the following Chapters will be developed within this experimental framework.

2.1 Introduction

This chapter describes a hardware and software platform designed to implement control strategies on a multi–robot experimental setup based on e–puck robots [34]. This experimental framework has been designed from scratch in cooperation with the ARSControl research group at the University of Modena and Reggio Emilia, Italy.

Since research on groups of autonomous mobile robots is a quite popular topic, several different experimental setups have been developed in the recent years. General purpose software platforms, such as [35] and [36], provide support for the development of experimental tests of control algorithms independently of the the particular hardware used. Thus, the same algorithm can be tested on different kinds of robots, also on simulated ones.

On these lines, a preliminary experimental setup has been realized inspired by [37], creating a group of three mobile robots. Each one of them is based on an iRobot Roomba

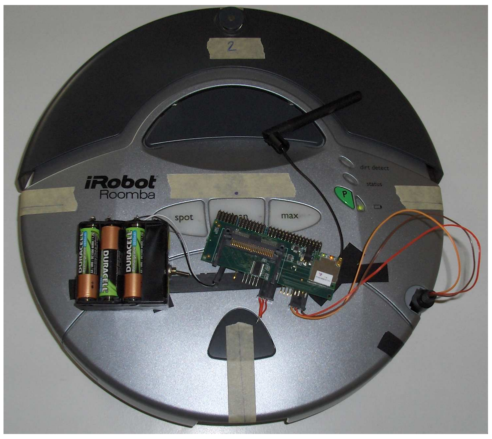


Figure 2.1: iRobot Roomba robot vacuum cleaner connected to Gumstix computer

robot vacuum cleaner[1], connected to a Gumstix Connex board[2], as shown in Fig. 2.1. The interaction among the main components of this experimental setup is described in Fig. 2.2. Gumstix Connex is a very small sized single–board Linux computer, which can be connected to the Roomba via serial port, controlling the wheels' motors and reading data from the sensors. Furthermore, it provides WiFi connectivity. The control strategy is implemented by means of the Player Robot Device Interface: the Gumstix board runs the Player server, while the control strategy is implemented on a remote computer, which controls the group of robots exploiting a WiFi network. It is worth noting that, due to the limited computational resources of the Gumstix board, it is not possible to execute complex control strategies directly on the board itself. Hence, even to emulate the implementation of decentralized strategies, the controller is executed on a remote computer.

Since the Roomba robots are not equipped with any exteroceptive sensor, the localization is performed by means of odometric measurements. To obtain a good accuracy, measurement and correction of the systematic odometric errors have been performed, exploiting the methodology described in [38]. WiFi network is used to broadcast each robot's position to the other robots, while the presence of proximity sensors may be simulated: for instance, each robot may be allowed to use only the positions of its neighbors (i.e. robots that are closer than the sensing range).

Even though this experimental setup ensures flexibility and reusability of the software on different hardware platforms, it presents serious lacks in terms of performances, as shown for instance in [20].

To obtain better performances, in the literature several software platforms have been

[1]http://www.irobot.com
[2]http://www.gumstix.com

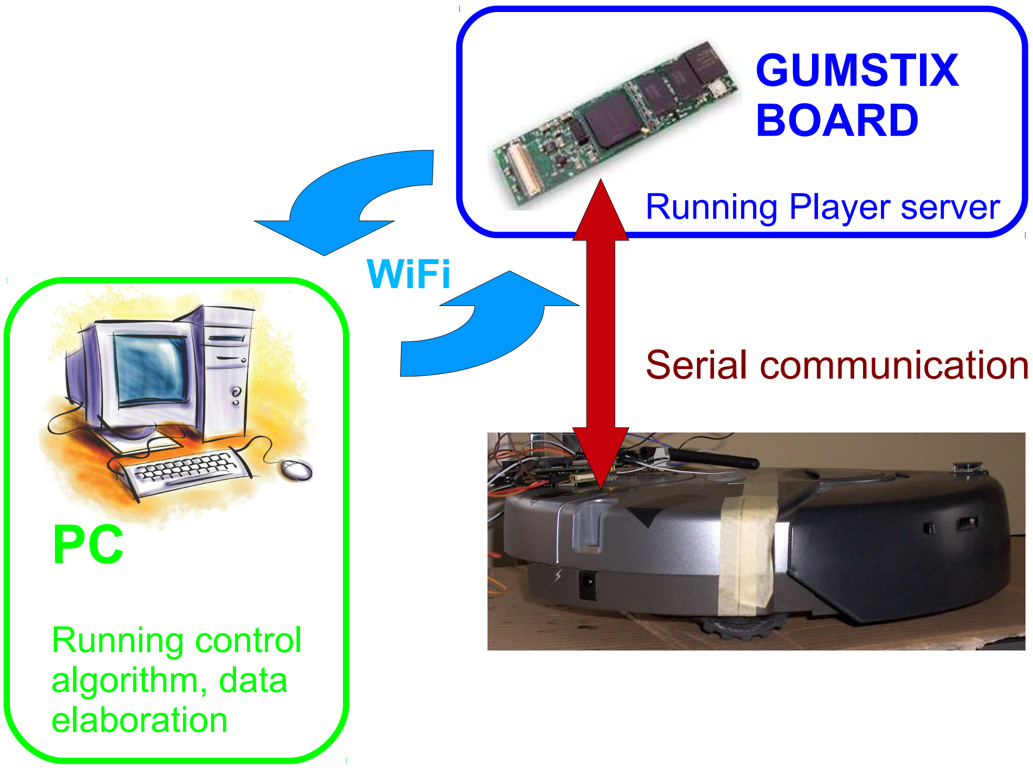


Figure 2.2: Scheme representing the main components of the iRobot Roomba based experimental setup

developed ad hoc for some particular robots. E–puck robots [34], for instance, are equipped with a simple and effective software platform that allows the users to control each sensor and actuator of the robots. In [39] an experimental platform for the control of groups of multiple e–puck robots has been introduced. The robots are moving in a bounded arena, and their positions are tracked by means of an overhead camera. The acquisition and elaboration of the image, and the control algorithm, are implemented as Matlab functions and scripts on a central PC. As stated by the authors in [39], Matlab has been used because it is easy to use, but the efficiency of the code (in term of speed of execution) can be improved using other (lower level) programming tools.

The software platform described in this chapter has been developed in C/C++ language. The main goal was to obtain an efficient and easy–to–use platform for the implementation of experimental tests on groups of mobile robots. The movement of the robots have been constrained inside an arena, equipped with an overhead camera. Exploiting colored markers, the overhead camera allows the tracking of the position of each robot. The elaboration of the images acquired by the overhead camera is developed exploiting some of the functions provided within the OpenCV library [40].

A Graphical User Interface (GUI) has been provided for the supervision of the experimental setup. The GUI, developed by means of the Nokia Qt library [41], allows the user to increase and reduce the number of robots included in the experiment, to monitor the position of each robot, and to implement the desired controllers. Analysis and plot tools are available as well.

The project has been developed on a Personal Computer equipped with Microsoft Windows XP Operative System. However, all the libraries that have been used are cross–platform, thus the software can be compiled and executed under different Operative Systems, such as Linux or Mac OS.

2.1.1 Outline

The outline of the Chapter is as follows. Section 2.2 describes the hardware test–bed used within the MORE–pucks project, based on the use of e–puck robots moving in a bounded arena, equipped with an overhead camera. Section 2.3.1 describes the image processing operations that allow the system to exploit the overhead camera for identification and localization purposes. Section 2.3.2 provides technical details about the modular structure of the software. Experimental tests developed with this platform will be described in the following Chapters.

2.2 E–puck robots and arena design

E–puck robots [34] are cheap and small sized mobile robots developed by the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Each robot (Fig. 2.3) has a diameter of $75mm$, and is equipped with a dsPIC30 microcontroller. The robots have a differential drive kinematic structure, with two wheels actuated by means of two stepper motors. A bluetooth interface is provided for the communication between each e–puck robot and a computer. The robots are equipped with several sensors as well, such as a small CMOS front camera, eight infrared proximity sensors on the perimeter, microphones and 3D accelerometers.

Inspired by [39], an arena was built to constraint the movement of the robots, equipped with an overhead camera, used for localization purposes. The arena (Fig. 2.4) has a rectangular shape, with dimension $2.0m \times 1.5m$. A metal structure holds a USB web–

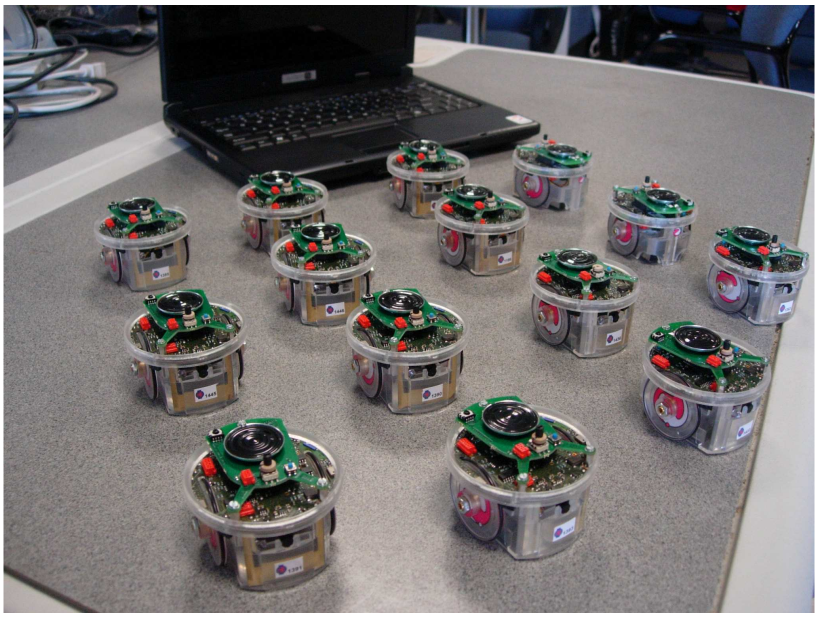


Figure 2.3: E–puck robots

cam on the top of the middle of the arena, at a height of $1.7m$ from the floor of the arena. The arena has been developed with a $4:3$ ratio between the sides, in order to exploit the entire size of the image acquired by the camera, that has a $4:3$ ratio. The correct height of the camera from the floor of the arena has been determined empirically.

Each robot has been equipped with a colored marker (Fig. 2.4). Different colors are used as unique identifiers during the visual localization process. Further details regarding the visual localization process will be provided in the following section.

2.3 Software architecture

2.3.1 Localization and visual odometry

One of the most commonly used localization systems is odometry. Starting from a known initial position, the current position of the robot is computed exploiting the readings of the encoders on the wheels. As is well known (see e.g. [38]), odometry is affected by error, that accumulates as the robot moves. Since, during the experiments, robots are constrained to move in a limited–size arena, their position can be computed by means of the visual feedback provided by an overhead camera.

This section will explain how the position is computed in our software platform: this part of the system will be referred to as *visual odometry*.

Each robot is equipped with a colored marker, represented in Fig. 2.4. The background color of each marker is used as a unique identifier for each robot (as shown in Fig. 2.4,

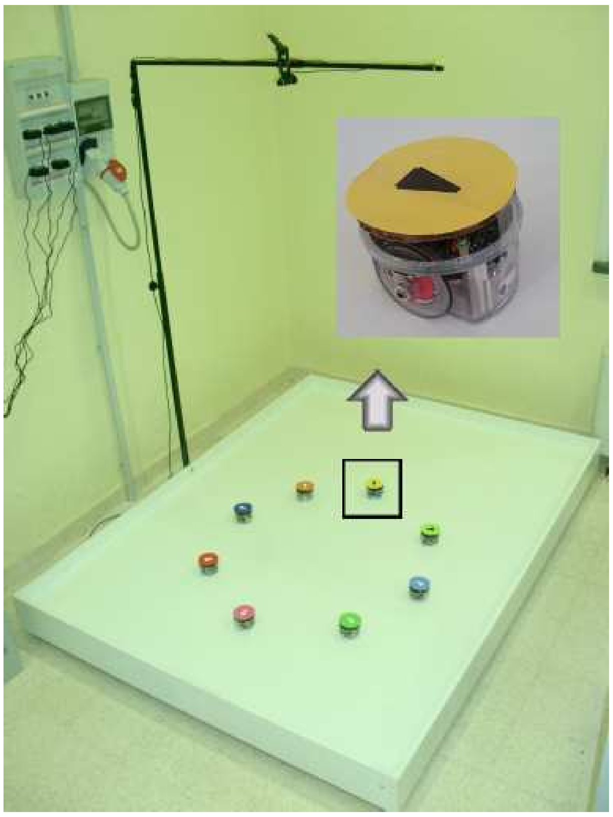


Figure 2.4: The arena, equipped with the overhead camera, with eight e–puck robots. On each e–puck robot, a colored marker is used for identification and localization

each robot's marker has a different background color), while the triangle inside the marker is used to compute its position.

Colored markers have two purposes: they provide a unique identifier for each robot (identification), and they are used to compute the current position of each robot (localization).

Identification

The arena has a uniform white background. Each robot is identified by the color of the background of its marker. More specifically, the image acquired by the camera is analyzed using the RGB color model [42]: the color of each pixel is identified with a specific value for each channel, red (R), green (G), and blue (B). A filter is applied to the image, in order to identify a region of interest for each robot. The mean and the standard deviation values for the three RGB channels are specified for each robot. This allows the system to identify whether a pixel corresponds to a robot (and, in this case, to find which robot it is) or to the white background.

The built–in OpenCV filtering function [40] does the image filtering. The main drawback of this function is that it analyzes the whole image for each color to be found: thus,

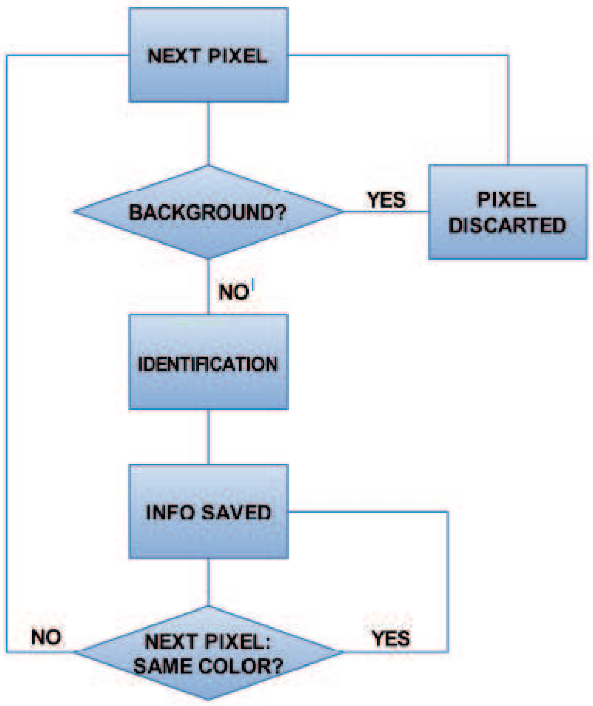


Figure 2.5: Filtering function used for the identification of the markers' colors

to identify N robots, each pixel of the image must be processed N times. This makes the system not scalable with the number of robots, and causes an unacceptably high computation time even for a relatively small number of robots. This is the main reason why a custom filtering function has been developed.

Each pixel of the image can either be part of the background, or part of the marker of one of the robots. The markers are compact sets of pixels: this means that all the pixels with the same color are in a limited area. Furthermore, if a pixel with a particular color is found, it's quite likely that also its neighboring pixels have the same color. Hence, the filtering function algorithm is explained in the flow–chart in Fig. 2.5:

- the image is analyzed per rows;
- if a pixel's color corresponds to the background, the pixel is ignored;
- if a pixel's color corresponds to one of the robots' identifiers, the next pixel will be tested first to understand if it has the same color.

This strategy makes the filtering operation considerably faster than using the built–in OpenCV function, because the heaviest computation (testing a pixel for all the possible ranges of colors) is done (ideally) only for the pixels that correspond to the borders of the robots. To make the computation even faster, this identification procedure is done

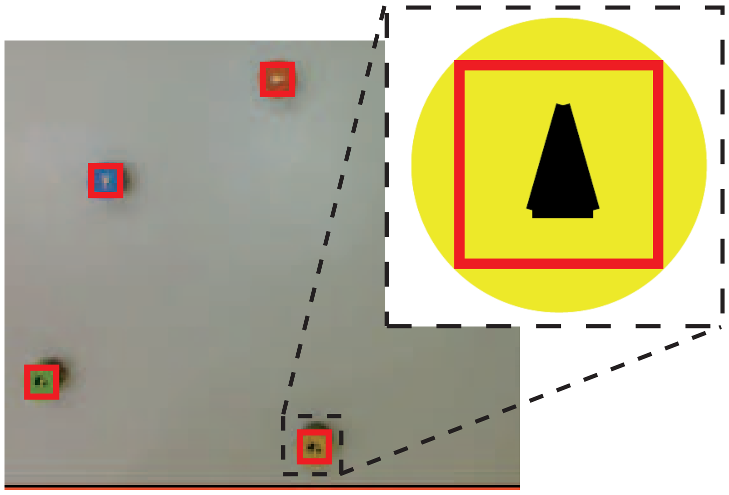


Figure 2.6: Region Of Interests (ROI) created on each robot

not on all the pixels, but only on the ones corresponding to an even row and an even column. It is worth noting that the filtering function has been implemented in a scalable way. More specifically, since each pixel of the image is analyzed only once, regardless of the number of robots involved, the computation time does not increase heavily as the number of robots increases.

The output of this filtering procedure is to create a region of interest for each robot. As shown in Fig. 2.6, it is a square set of pixels that, for each robot, contains the triangle used for the following localization procedure.

Localization

In the center of each marker, an isosceles triangle is drawn, with an high–contrast color: white (if the marker's background color is dark) or black (is the marker's background color is light).

After the identification phase, a region of interest is defined for each robots. Each region of interest contains the triangle and part of the marker's background color, as shown in Fig. 2.6.

The purpose of the localization phase is to compute the pose of each robot. The term *pose* indicates both the position and the orientation of the robot.

The position of the robot is defined as the barycenter of the isosceles triangle. Moreover, the orientation of the robot is defined as the angle between the height of the isosceles triangle and the x–axis of the absolute reference frame.

The positions of the corners of the isosceles triangle are computed exploiting the strategy described in [43]: corners are identified as points of discontinuity in the image.

Thus, corners are identified as point in which the second–order derivative of the intensity is high, both in x and y direction. Further details can be found in [43].

Once the corners have been found, the position of the robot is defined as the barycenter of the triangle, i.e. the average of the positions of the corners. Once the top corner of the triangle (i.e. the corner between the two identical sides of the isosceles triangle) has been identified, the orientation of the robot is computed as the angle between the height of the triangle and the x–axis of the absolute reference frame.

2.3.2 The Core Software and the Graphical User Interface (GUI)

The software platform has been developed in a modular way. As show in the class diagram in Fig. 2.7, several C/C++ classes has been defined to implement the different modules and sub–modules of the software architecture. The main modules are the Core Software and the Graphical User Interface (GUI).

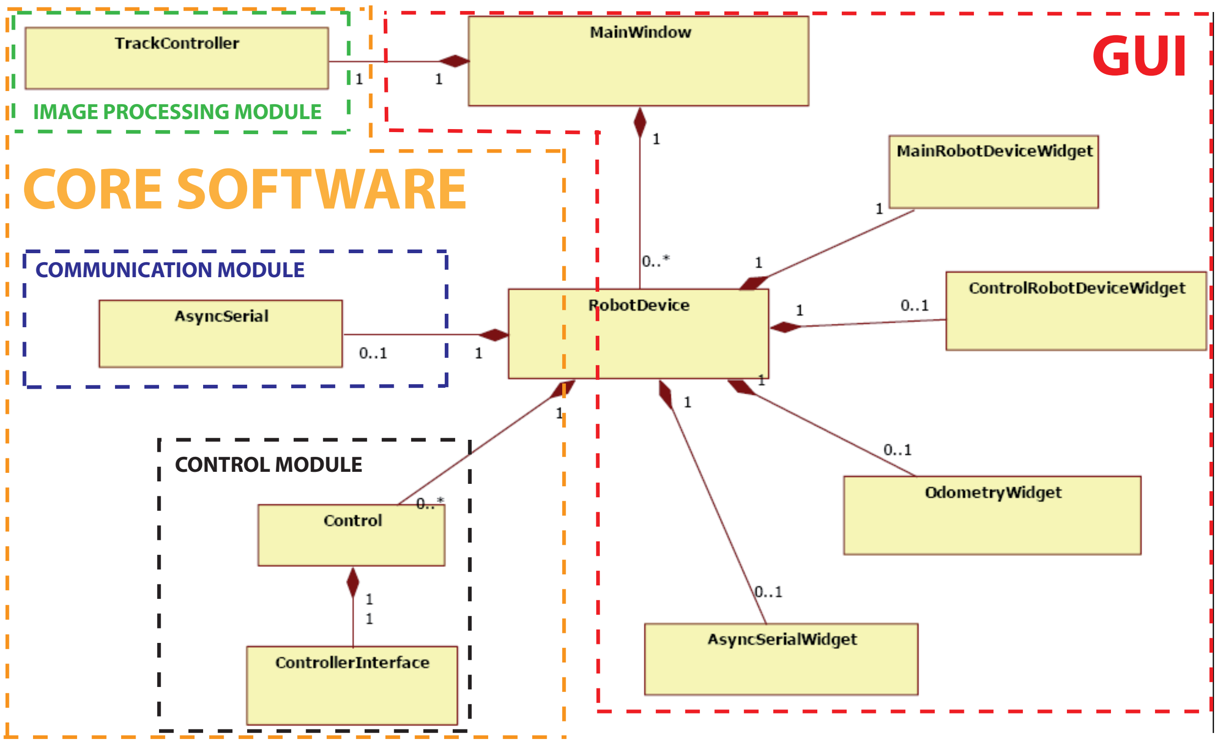


Figure 2.7: Class diagram of the software

The Core Software

The Core Software is the composition of three sub–modules: the image processing module, the communication module, and the control module.

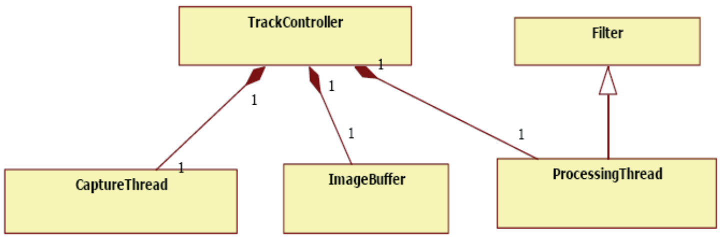


Figure 2.8: Class diagram of the image processing module

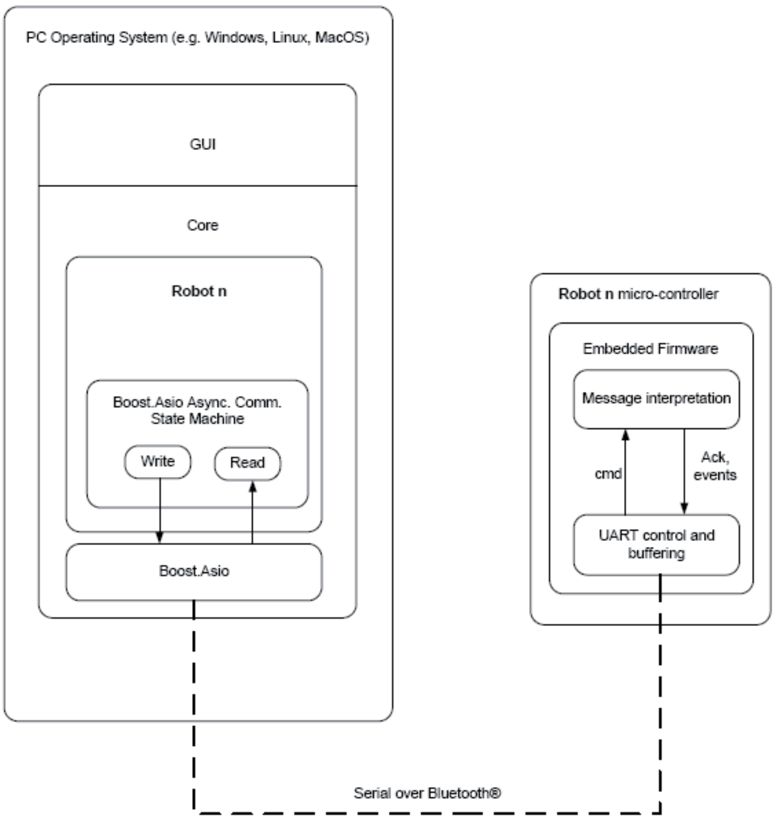


Figure 2.9: Scheme of the communication architecture between the e–puck robot and the PC

Image processing module The image processing module implements the visual odometry procedure described in the previous section. As shown in Fig. 2.7, the image processing is managed by the class *TrackController*, whose decomposition is shown in Fig. 2.8. Specifically, the class *CaptureThread* implements an infinite loop that acquires images from the web–cam and saves it into a buffer (*ImageBuffer*). The class *ProcessingThread* extracts the image from the buffer and computes the elaboration described in the previous section. A separate class has been created for the heaviest part of the image processing, namely the filtering process (class *Filter*).

Communication module The scheme of the communication architecture between the computer and each robot is described in Fig. 2.9. The communication is implemented exploiting the bluetooth interface of the e–puck robots. More specifically, a serial communication over bluetooth channel is implemented, by means of the *Boost.Asio* library. *Boost.Asio* is provided within the *Boost* C++ library, that is a cross–platform C++ library for network and low level I/O programming with an asynchronous model support. Further details about the *Boost* C++ library can be found in [44].

The main purpose of the communication module is to send commands to the robots, and to acquire data from the robots' sensors.

A modified version of the e–puck BTCom protocol has been implemented (details can be found in [45]). The main improvement to this protocol is the support for asynchronous communication.

The communication protocol is based on the exchange of fixed length text messages. To read some sensor data:

- the PC sends to the robot a request message;
- once received the message, the robot saves the request, and sends a confirmation to the PC;
- the robot sends the data asynchronously, every time the values change;
- the PC acquires the incoming data in a buffer, and makes them available for the application programs.

Sending commands for the actuators is managed in the same way.

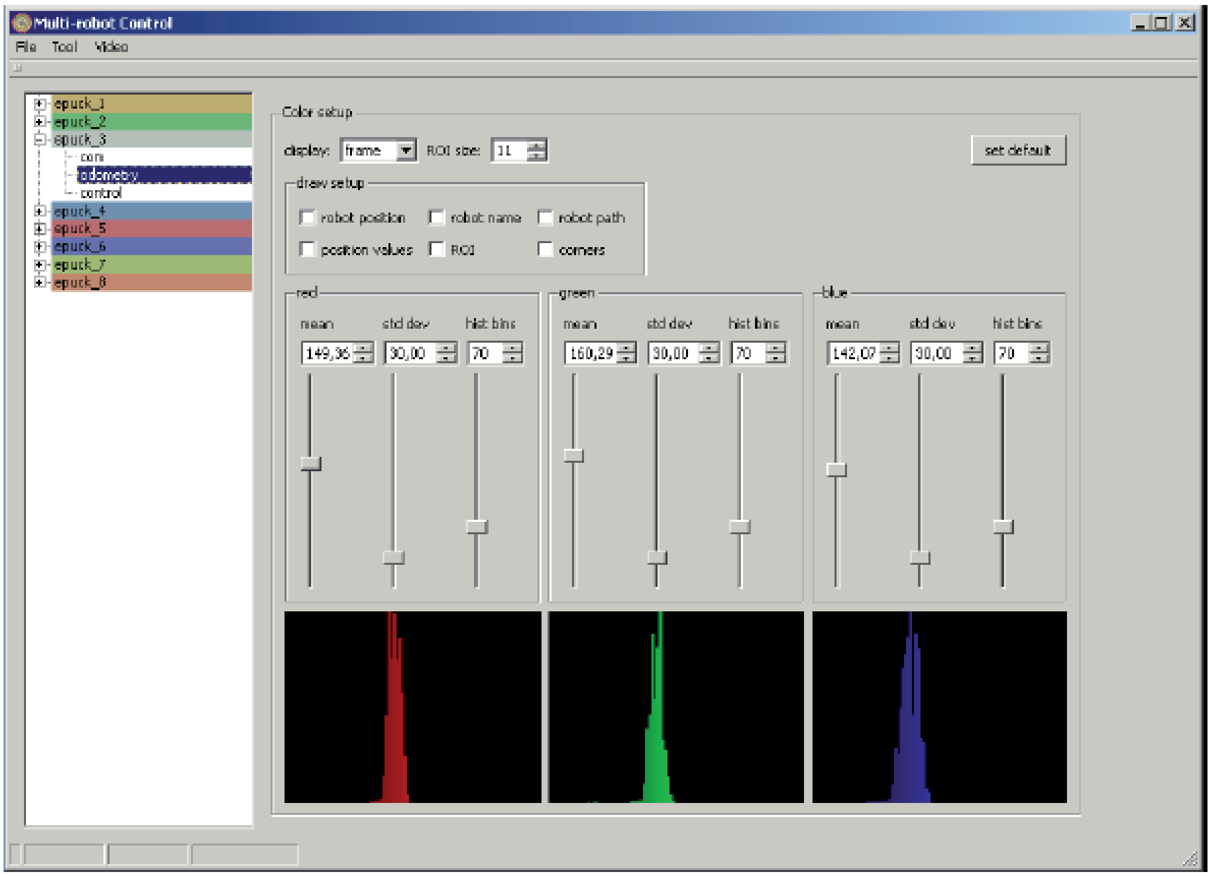
The asynchronous approach helps the execution of the program as the number of robots increases, because there is no need to insert blocking points to wait for the other robots to end their current operation.

Control module The control module is in charge of associating one of the available control functions with the desired robot. Control functions are C/C++ functions, written by the user to obtain some desired behavior.

The Graphical User Interface (GUI)

The Graphical User Interface (GUI) has been developed exploiting the Nokia Qt library [41]. The class decomposition scheme of the GUI is shown in Fig. 2.7.

As show in Fig. 2.10, the GUI is represented by the *main window*, where Qt Widgets (i.e. modules) can be added to control the different parts of the software platform, as shown in Fig. 2.11.



display: frame
ROI size: 11

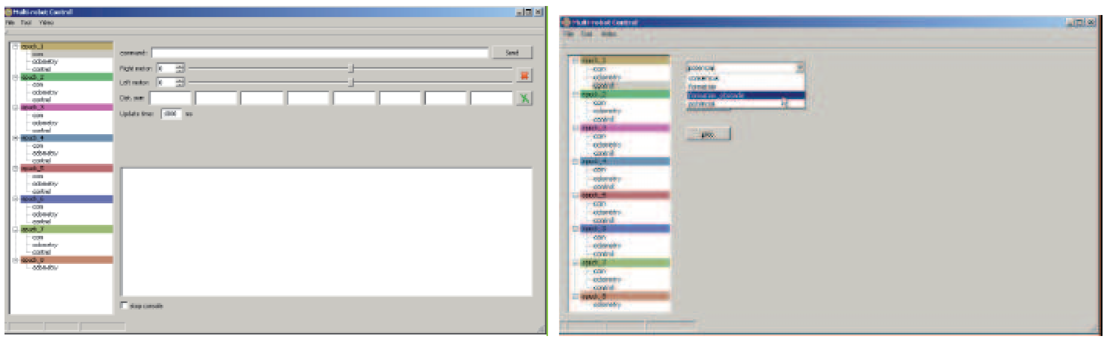
	mean	std dev	hist bins
red	149,36	30,00	70
green	160,29	30,00	70
blue	142,07	30,00	70

Figure 2.10: The Graphical User Interface (GUI)

More specifically, the user can add *virtual robots*, that appear in a tree list. The user can then add a *communication widget* to each virtual robot, thus linking the virtual robot to a real, physical, robot. Then, adding an *odometry widget* to a virtual robot allows the user to link each robot with the corresponding marker's background color, so that the system can perform the visual odometry. Finally, adding a *control widget* to a virtual robot, the user selects the desired control function to be executed on each robot.

Additional features of the GUI include the possibility to record videos of the arena during the experiments, and storing log files of the positions of the robots, to be used for further off–line elaborations (e.g. to draw plots or compute statistical analysis, with external programs such as Matlab).

The modular architecture allows the user to add widget at runtime. More specifically, this means that it is possible to add or remove robots while performing an experiment. This allows, for example, to test the scalability of control algorithms, since it is possible to analyze the behavior of the system as robots are added or removed.



(a) (b)

Figure 2.11: GUI Widgets: communication widget and control widget

2.4 Validation experimental tests

The MORE–pucks software platform has been exploited for the experimental validation of the control strategies described in the following chapters.

On the project's web–page, http://www.arscontrol.unimore.it/morepucks, several videos can be found. This videos show the system implementing different control algorithms. Tutorials and extended documentation are available as well.

Generally speaking, as described in the previous section, the control module is in charge of selecting one of the available control functions, and to assign it to the desired robots.

Control functions are written in C/C++ code. Generally speaking, a function is assigned to a specific robot, but has access to the data acquired by the sensors of the entire group. In other words, each robot has access to the sensor data (e.g. the positions) of all the other robots. This is useful to emulate the presence of an explicit communication channel between the robots, or to emulate the presence of some proximity sensors.

Once written and saved as a source file, each control function can be assigned to a robot by exploiting the control widget of the GUI, that allows the user to select the function from a list.

2.5 Discussion

In this chapter a software platform has been described for the development of experimental tests in the field of multi–robot systems. The purpose of this software platform is the control of a group of e–puck robots moving in a bounded arena, which is equipped with an overhead camera used for identification and localization purposes.

The software platform has been developed from scratch, based on the use of open–source cross–platform libraries, such as OpenCV, Boost and Nokia Qt. The software platform is freely available for download, under GPLv3 license.

As described in Section 2.3.2, the software has been developed in a modular way. This characteristic makes the addition of new features easy to implement.

Furthermore, the modularity of the software is exploitable also from the user's point of view. More specifically, the GUI is defined as a collection of widgets: this allows the addition and subtraction of robots at runtime.

The use of a central computer and of an overhead camera allow the user to directly test centralized algorithms, but also to emulate the execution of decentralized algorithms, simulating the presence of proximity sensors and of direct communication channels among the robots.

The image elaboration, that is the most computationally demanding part of the software platform, has been developed with custom functions, to guarantee a faster execution with respect to the built–in OpenCV functions. This increased the scalability of the software, because the execution time does not heavily increase with the number of robots. Nevertheless, the image filtering procedure needs further improvements, in order to make the system able to identify a higher number of colors.

The software platform has been developed based on open source and cross–platform libraries. The name of the project, MORE–pucks, comes from the acronym of the University of MOdena and REggio Emilia. The software is is freely available for download, under GPLv3 license [46], at http://www.arscontrol.unimore.it/morepucks.

Chapter 3

Formation control and coordinated curve tracking

In this chapter some coordination control strategies for multi–robot systems are described to solve formation control and coordinated curve tracking problems. Artificial potential fields and consensus based controllers will be exploited to design the control strategies. Specifically, the first section describes how to design artificial potential fields to obtain a formation with the shape of a regular polygon. The proof asymptotic stability of the system is based on the definition of a proper Lyapunov function. The absence of local minima will be ensured as well. Then, exploiting a bijective coordinate transformation to deform the polygonal formation, completely arbitrarily shaped formations will be obtained. Subsequently, the previously described artificial potential fields will be modified, in order to achieve coordinated tracking of closed curve paths. This will lead to the definition of a completely decentralized algorithm, that doesn't require any global synchronization. The formation control problem will then be addressed by means of a consensus based control strategy. Weighted graphs will be used to obtain the desired formation–shape while avoiding collisions among the robots. Since mobile robots usually move in unknown and unstructured environments, the control strategy will be extended to make the robots avoid collision with obstacles as well.

3.1 Introduction

Formation control has been widely studied in the last few years, due to the increasing interest in autonomous vehicles. Groups of mobile robots can be used to perform tasks that a single robot cannot be able to complete. Examples are the movement of large or heavy objects [47], or the exploration of wide areas [48].

In the literature, many different approaches to formation control can be found. The main existing approaches can be divided into two categories: centralized and distributed. Because of the intrinsic unreliability of centralized methods [17], the focus of this Chapter is on distributed ones: all the agents are equal, and if one of them stops working, the other ones can still complete their task.

Many distributed strategies have been proposed to make groups of mobile robots move in a cohesive way [49–51], imitating the behavior of large groups of animals (e.g. school of fish). However, the aim of the control strategy introduced in Section 3.2 is quite different: the group of mobile robots is required to create a formation with an exact desired geometric shape.

This kind of control strategy can be applied into several different fields. For example, in the industrial field, this formation control strategy can be applied to a group of Automated Guided Vehicles (AGVs) moving in a warehouse for goods delivery. The main idea is to make a group of AGVs cooperatively deliver a certain amount of goods, moving in a formation. The creation of a formation with the desired shape is useful to precisely constrain the action zone of the AGVs, thus reducing the chance of collisions with other entities (e.g. human guided vehicles).

Another possible application is in the exploration of unknown environments. A group of mobile robots moves inside an unknown environment, while acquiring data from some exteroceptive sensors. With respect to a single robot exploring an unknown environment, a group of robots can acquire much more information. However, data acquired by each robot need to be merged in a coherent way. If the mobile robots keep a known formation while moving in the environment, their relative positions are known, that makes the process of merging sensor data more effective.

Artificial potential fields Artificial potential based control strategies make robots move along the negative gradient of the composition of some artificial potential fields. Correctly shaping these potential fields allows one to impose a desired behavior to a group of robots. Artificial potential fields are a very powerful control strategy. Different potential fields can be designed to obtain different objectives, for example

- to make a robot move to a desired goal position,
- to make a robot avoid collisions with obstacles,
- to avoid collisions among different robots,
- to make a group of robots move in a cohesive way.

From the composition of these artificial potential fields, the desired behavior for the group emerges.

While most of the artificial potential based formation control strategies have the aim of controlling only the overall swarm geometry (examples can be found in [52, 53], and references therein), recently some strategies have appeared to control the exact shape of the formation. One possible approach is to deploy a group of robots over a desired curve [54–56].

Conversely, the control strategy introduced in Section 3.2 exploits some specifically designed artificial potential fields to obtain a completely arbitrarily shaped formation. In particular, the artificial potential fields are designed to provably make the robots create a regular polygon formation. Subsequently, a bijective coordinates transformation will be exploited to obtain a arbitrarily shaped formation. Formal proof of the asymptotic stability of the system, based on the definition of a proper Lyapunov function, will be provided. Previous potential based strategies to obtain formations with an exact geometric shape [17] have the drawback that, as the number of agents increases, many local minima appear. Local minima are asymptotically stable undesired equilibrium points. Thus, they are one of the main problem in potential based strategies [57], because they make the agents stop in undesired positions. Conversely, the control strategy introduced in Section 3.2 is formally guaranteed to be unaffected by the problem of local minima: thus, the desired formation is always created.

Consider the regular polygon formation. Introducing a time varying coordinate transformation, it is possible to make the coordinate system rotate, in order to make the robots move along the circumcircle of the polygon. Hence, exploiting this strategy, it is possible to solve the coordinated path tracking problem, as shown in Section 3.3.

Among the main applications of this control strategy, a remarkable example is represented by Automated Guided Vehicles (AGVs) moving in an industrial environment. As represented, for instance, in Fig. 3.1, AGVs may be employed for end–of–line operations, for example delivery of goods from the production machines to the warehouse.

Since movement of goods is a crucial activity in industrial applications, this kind of problem has already been tackled, in the literature. A remarkable example is described in [12]: a decentralized control strategy has been developed to make AGVs autonomously move on a roadmap, for goods delivery in a warehouse. However, considering typical industrial plants, the environment is generally more cluttered: therefore, the admissible paths for the AGVs often assume very strange shapes. Thus, the control strategy introduced in Section 3.3 is suitable for paths with completely arbitrary shapes.

In the literature, many control strategies have been proposed for tracking of paths. Traditional approaches (see e.g. [58] and references therein) generally make the mobile

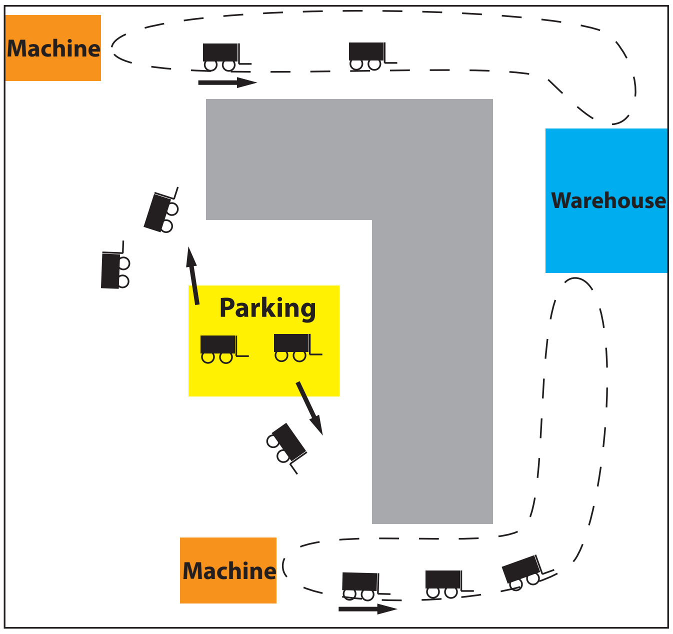


Figure 3.1: End–of–line industrial scenario

robot follow a reference point that moves along the trajectory, by means of error feedback. Even though these strategies are very effective for a single vehicle to track a trajectory, it's not straightforward to extend them to the multi–vehicle case.

In the multi–vehicle case, each robot has to track the path without colliding with the other ones, and maintaining a desired distance from them. For this purpose, traditional collision avoidance strategies, for example potential based ones [59], are not suitable. An adapted potential based control strategy is presented in [60] for automatic driving on highways. The composition of the artificial potentials makes the vehicles change the lane to overtake other vehicles, thus avoiding collisions. Conversely, due to safety issues, the single lane scenario (i.e. the vehicles never leave the path, and synchronize their motion along it) is often more interesting, for industrial applications.

A well studied application for groups of mobile robots coordinated over a closed curve path is boundary tracking [61, 62], i.e. the deployment of a group of robots along the boundary of a certain zone, for instance a nuclear plant, as shown in Fig. 3.2. The same concepts can be used for environmental monitoring, i.e. the deployment of a group of active sensors around a forest fire, a poisonous oil spill or an ocean contamination. However, generally boundaries are approximated by convex (or star–convex) curves [63, 64], since a higher precision in the definition of the boundary is not needed for environmental monitoring. Furthermore, the aim of these algorithms is to spread the robots over the boundary and then to stop them [63], or to make them patrol a small segment of the boundary moving alternatively forward and backward [65].

Figure 3.2: Mobile robots can be used for patrolling the boundary of a nuclear plant

Hence, Section 3.3 describes how to modify the control strategy introduced in Section 3.2, making the coordinate system rotate, in order to make the robots move along a circumference. However, it is not always possible to find a suitable coordinates transformation to relate a circumference with a completely arbitrary shaped curve. Thus, the artificial potential fields will be appropriately redefined, in order to avoid the use of coordinates transformations.

Weighted graph consensus algorithms Artificial potential fields are a very effective way to implement formation control and collision avoidance strategies. However, one of the main drawbacks in using potential fields is the fact that delays in the communication channels drive the system to instability [66].

Hence, in Section 3.4 edge–weighted graphs [67, 68] will be exploited to drive a group of robots to create the desired formation while avoiding collisions.

A remarkable example of consensus based formation control strategy has been introduced in [69]: in this work, the authors describe how to exploit consensus algorithms to obtain a formation of autonomous vehicles whose interconnection is described by means of a graph. One of the main advantages of consensus algorithms is the fact that the agreement is reached even in the presence of delays in the communication [70]. Furthermore, the multi–agent system keeps a stable behavior even in the presence of a varying communication topology [69]. However, to include collision avoidance among the agents into consensus based formation control strategies, typically repulsive potential fields are added [70]. As previously stated, one of the main drawbacks in using potential fields is the fact that delays in the communication channels drive the system to instability [66], as shown in the simulations described in Section 3.4.4.

The control strategy described in Section 3.4 follows a different approach. Specifically, a control strategy is introduced that exploits edge–weighted graphs [67] both for the creation of a desired formation and for collision avoidance. In particular, non–constant edge–weight functions are exploited to obtain the desired formation while avoiding collisions among the robots. Moreover, supposing that the robots are moving in unknown environments, this approach can be extended to avoid collisions with detected obstacles, by introducing virtual agents projected on their surface [71]. The repulsive action caused by the introduction of virtual agents could drive the system to configurations where the desired shape is not maintained [72]. Therefore, the intensity of the inter–robot influence [73] may be regulated, in order to modify the rigidity of the formation, thus ensuring the shape maintenance.

Avoiding the use of artificial potentials, this approach is robust to the presence of communication delay, being fully consensus based. Simulations are provided for validation purpose.

3.1.1 Outline

The outline of the Chapter is as follows. Section 3.2 describes an artificial potential field based control strategy that aims at obtaining completely arbitrarily shaped formations of mobile robots. Modified artificial potential fields are then exploited in Section 3.3 to perform cooperative path tracking, in a completely decentralized way. Section 3.4 described a strategy based on graph theory for formation control and collision avoidance.

3.2 Arbitrarily Shaped Formations of Mobile Robots: Artificial Potential Fields and Coordinate Transformation

3.2.1 Regular polygon control law

Consider a group of n point mass holonomic agents characterized by the following dynamics:

$$\ddot{x}_i = u_i \quad i = 1, ..., n \tag{3.1}$$

where $x_i \in \mathbb{R}^2$ is the position of the i–th agent. The dynamic behavior considered here is quite simple, but all the results obtained hereafter can be extended to nonholonomic vehicles. In fact, many strategies can be found (e.g. [74] and [75]) to feedback linearize several classes of nonholonomic vehicles. Furthermore, the agents are supposed to be able to localize themselves exactly. For applications in indoor environment (e.g. AGVs moving in a warehouse), localization can be obtained, for instance, by means of laser triangulation. On the other hand, in case of outdoor applications (e.g. mobile robots for exploration of unknown environment), localization can be obtained exploiting a GPS receiver.

Let S_R be the sensing range of each agent. Each agent knows only the positions of its neighbors, which are the agents that are closer than S_R.

The objective is to make the agents create a formation with the shape of a regular polygon with n sides. More specifically, the length of each side (i.e. the distance between two neighboring agents) is required to be equal to $L \leq S_R$, and the circumcenter of the polygon to be in a desired position $x_c \in \mathbb{R}^2$. Let R be the radius of the circumcircle of the polygon (i.e. the distance between each agent and the circumcenter): from basic geometrical considerations, it follows that

$$R = \frac{L}{2\sin\left(\frac{\pi}{n}\right)} \tag{3.2}$$

In order to implement the control law, each agent is supposed to know the position of the center of the circumcircle, x_c, the number of agents, n, and the desired distance between two neighboring agents, L. It is worth noting that knowing the total number of agents is necessary to create a formation with an exact geometric shape.

Hence, to obtain the desired behavior, the following control law is implemented:

$$u_i = f_{ci} + \sum_{j=1;j\neq i}^{n} f_{aij} - b\dot{x}_i \tag{3.3}$$

where b is a positive constant which implements a damping action.

The first term of Eq. (3.3) is defined as follows:

$$f_{ci} = -\nabla_{x_i} V_{ci}\left(x_i\right) \tag{3.4}$$

and

$$V_{ci}\left(x_i\right) = \frac{1}{2}K_c(d_{ci} - R)^2 \tag{3.5}$$

where $d_{ci}(t) = \|x_i(t) - x_c\|$ is the current distance between the i–th agent and the desired position for center (x_c), and K_c is a positive constant. The role of this term is to take each agent at distance R from the desired position for the center of the formation. In other words, if no other potential fields were present, this term would make every agent move to a circumference with center x_c and radius R.

The second term of Eq. (3.3) is defined by the following components:

$$f_{aij} = -\nabla_{x_i} V_{aij}(x_i, x_j) \tag{3.6}$$

and

$$V_{aij}(x_i, x_j) = \begin{cases} \frac{1}{2} K_a (d_{ij} - L)^2 & \text{if } d_{ij} \le L \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where $d_{ij}(t) = \|x_i(t) - x_j(t)\|$ is the distance between the i–th agent and the j–th agent, and K_a is a positive constant. It's easy to see that function V_{aij} is continuously differentiable. This term is used to regulate the distances among the agents. This interagent potential produces a repulsive force if two agents are too close, namely if $d_{ij} < L$, and produces a null force if the distance is greater than or equal to the desired one, namely if $d_{ij} \ge L$.

Thus, the composition of these potential fields produces the following behavior:

1. All the agents move toward a circumference with center x_c and radius R. No collisions among the agents can happen, because of the presence of the control action in Eq. (3.6).

2. When all the agents lie on the circumference, the control action in Eq. (3.4) is null. The control action in Eq. (3.6) regulates the relative distances among the agents, until they are in the desired configuration.

3. In the desired configuration, the composition of the potentials gives a null control action, because the agents are on the circumference ($f_{ci} = 0\ \forall i = 1, \ldots, n$), and the distance between each couple of agents is equal to L ($f_{aij} = 0\ \forall i, j = 1, \ldots, n$).

Proposition 3.1. *The regular polygon formation is an asymptotically stable configuration.*

Proof. Let $\tilde{x} = \left[x_1^T \ldots x_n^T\ \dot{x}_1^T \ldots \dot{x}_n^T\right]^T \in \mathcal{X}$ be the state vector of the system. Consider the following Lyapunov candidate function $V : \mathcal{X} \to \mathbb{R}$, given by the total energy of the system:

$$V(\tilde{x}) = \sum_{i=1}^{n} \left[V_{ci}(x_i) + \sum_{j=1; j \ne i}^{n} V_{aij}(x_i, x_j) + \frac{1}{2} \|\dot{x}_i\|^2 \right] \tag{3.8}$$

From Eqs. (3.5), (3.7) one can trivially see that $V \geq 0$. Since V is the sum of three terms which are always positive or null, for V to be equal to zero all of them are required to be equal to zero as well. More specifically, $V = 0$ if and only if, simultaneously:

1. $\dot{x}_i = 0 \; \forall i = 1, ..., n$; i.e. all the agents are at some steady state position;
2. $V_{ci} = 0 \; \forall i = 1, ..., n$; i.e. all the agents are on the circumference with center x_c and radius R;
3. $V_{aij} = 0 \; \forall i, j = 1, ..., n$; i.e. all the agents are at a distance greater than or equal to L with respect to their neighbors ($d_{ij} \geq L \; \forall i, j = 1, ..., n$).

From basic geometrical considerations it follows that conditions 2 and 3 can hold simultaneously if and only if $d_{ij} = L \; \forall i, j = 1, ..., n$. In other words, $V \geq 0$ always, an $V = 0$ only in the regular polygon formation (no local minima).

Consider the time derivative of this function:

$$\dot{V}(\tilde{x}) = \sum_{i=1}^{n} \dot{x}_i^T \left[\nabla_{x_i} V_{ci}(x_i) + \sum_{j=1; j \neq i}^{n} \nabla_{x_i} V_{aij}(x_i, x_j) + \ddot{x}_i \right] \tag{3.9}$$

From Eqs. (3.1), (3.3), (3.4), (3.6) it is possible to obtain the following equation:

$$\ddot{x}_i = -\nabla_{x_i} V_{ci}(x_i) - \sum_{j=1; j \neq i}^{n} \nabla_{x_i} V_{aij}(x_i, x_j) - b\dot{x}_i \tag{3.10}$$

Thus, from Eq. (3.9) and Eq. (3.10):

$$\dot{V}(\tilde{x}) = -\sum_{i=1}^{n} b \|\dot{x}_i\|^2 \tag{3.11}$$

which is always less than or equal to zero.

To prove the asymptotic stability of the desired configuration, LaSalle's principle may be invoked. Function V as already been proved to be always greater than or equal to zero, and $V = 0$ only in the desired configuration. Furthermore it has been proved that $\dot{V} \leq 0$ always, and $\dot{V} = 0$ if and only if $\dot{x}_i = 0 \; \forall i = 1, ..., n$.

Thus, in a neighborhood of the desired configuration, V is positive definite, and $\dot{V}$ is negative semidefinite. $\dot{V} = 0$ if the velocities of all the agents are zero. This situation happens only in the desired configuration. In fact, if the agents are in different configurations, thanks to the control law described so far, a force different from zero makes them accelerate, thus modifying their velocities. Hence, the set

$$\mathcal{V} = \left\{ \tilde{x} \in \mathcal{X} \text{ s.t. } \dot{V}(\tilde{x}) = 0 \right\} \tag{3.12}$$

contains no trajectory of the system except the trivial trajectory $\tilde{x}(t) = \tilde{x}_D$, where $\tilde{x}_D$ is the state vector of the system where all the agents are in the desired configuration with velocity equal to zero.

Therefore, the desired configuration is asymptotically stable. □

The desired configuration is not globally asymptotically stable because undesired equilibrium configurations appear when two or more vehicles are aligned with x_c. In this case the potentials never generate a force perpendicular to the alignment direction and, therefore, the aligned agents would never play their role in the creation of the desired polygonal formation. Nevertheless, these equilibrium points are not local minima, since they are clearly unstable. In fact, an infinitesimal perturbation of the position of the aligned agents is sufficient for the potentials to create a force that leads the agents to the desired configuration. Thus, in order to avoid some agents to get stuck in this undesired configuration, when an agents detects that it's aligned with x_c and with another agent, it applies a random infinitesimal force that modifies its position in order to destroy the alignment condition and to converge to the desired polygonal configuration. The possibility that all the aligned agents apply a force in the same direction and that, therefore, the alignment condition is preserved after the perturbation, is practically zero.

Hence, the regular polygon configuration is the only asymptotically stable configuration of the system. Thus, unlike other potential–based methods [57], this control strategy is local minumum free.

3.2.2 Orientation of the polygon

The control strategy presented in the previous section admits a symmetry. In fact, given n agents, there are infinite regular polygons with n sides lying on the same circumcircle, and this control strategy just takes the agents to one admissible configuration. However, in many applications it is very useful to select exactly *one* of these infinite admissible configurations. To solve this problem, the orientation of the formation needs to be fixed. To this aim, the control law presented in the previous section will now be modified.

In Fig. 3.3 one can see three admissible configurations, obtained by rotating the polygon around its circumcenter. The system has one degree of freedom: to select one precise polygon, one condition is needed to eliminate this degree of freedom. One way to do this is to select the position of one of the vertices of the polygon. Thus, define x^* as the position to be occupied by one of the vertices of the polygon. Fixing the position of one of the vertices, the orientation of the polygon may be selected. Since all the agents are

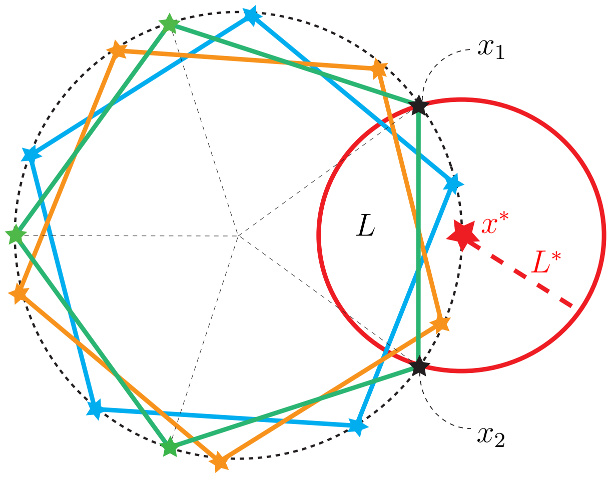


Figure 3.3: The action zone of the orientation component of the control law must be such that it influences one and only one agent at the steady state

required to be indistinguishable, selecting a priori which agent will be in the position x^* is not admissible. Thus, a new potential V_{oi} is introduced which attracts to x^* every agent that is inside a proper region of attraction. It is now necessary to define this region of attraction.

Let $\mathcal{C} = \{x \text{ s.t. } \|x - x^*\| \leq L^*\}$ be a circle whose border intersects the circumcircle of the polygon in two points x_1 and x_2 such that $\|x_1 - x_2\| = L$ (Fig. 3.3). To calculate L^*,

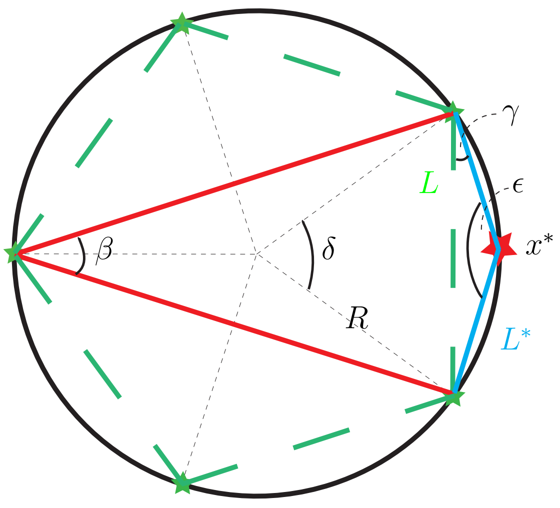


Figure 3.4: Geometric properties to calculate the radius L^* of the action zone of the orientation component

refer to Fig. 3.4, where some geometric properties among the angles are shown. Namely, given angle β, then $\delta = 2\beta$, and $\epsilon = \pi - \beta$. The following property among L, R and β holds:

$$L/2 = R \sin(\delta/2) = R \sin(\beta) \tag{3.13}$$

Thus, β can be easily calculated:

$$\beta = \arcsin(L/2R) \tag{3.14}$$

Angle γ is given by the following relation:

$$\gamma = [\pi - (\pi - \beta)]/2 = \beta/2 \tag{3.15}$$

Since

$$L/2 = L^* \cos(\gamma) \tag{3.16}$$

L^* can be calculated as follows:

$$L^* = L/\{2\cos[(\arcsin(L/2R))/2]\} \tag{3.17}$$

Assume that $\mathcal{C}$ is the region of attraction. If one agent is inside $\mathcal{C}$, the action of V_{oi} would attract this agent to x^* taking the polygon at the desired orientation. Nevertheless, if two agents are in x_1 and x_2, they are both attracted to x^* and the interaction between V_{oi} and the interagent potential creates a local minimum which deforms the final shape of the formation. On the other hand, if the border of $\mathcal{C}$ is excluded from the region of attraction, another pathological case appears. In fact, in this case, if two agents are in x_1 and x_2, none of them is attracted to x^* and the orientation of the polygon is not changed as desired. In order to avoid these undesired behaviors, the region of attraction is defined as follows:

$$\mathcal{S}^* = \{x \text{ s.t. } \|x - x^*\| < L^*\} \cup \{x_1\} \tag{3.18}$$

Note that x_1 can be substituted by x_2 as well.

Thus, the following control law is implemented:

$$u_i = f_{ci} + \sum_{j=1;j\neq i}^{n} f_{aij} + f_{oi} - b\dot{x}_i \tag{3.19}$$

This control law can be obtained from Eq. (3.3) by adding the term f_{oi}, which is defined as follows:

$$f_{oi} = -\nabla_{x_i} V_{oi}(x_i) \tag{3.20}$$

and

$$V_{oi}(x_i) = \begin{cases} \frac{1}{2} K_o (d_{oi})^2 & \text{if } x_i \in \mathcal{S}^* \\ K^* & \text{otherwise} \end{cases} \tag{3.21}$$

where $d_{oi}(t) = \|x_i(t) - x^*\|$ is the distance between the i–th agent and the point x^*, and K_o and K^* are constants, with $K_o > 0$.

As already stated, after the polygon has been created, one and only one agent would be influenced by the orientation action. But during the transient (i.e. before the polygon has been created) it can happen that two or more agents are inside $\mathcal{S}^*$. For the polygon to be correctly created, the distance between two neighboring agents is required to be equal to L, even in the presence of this orientation component. Thus, if two or more agents are inside $\mathcal{S}^*$, they must move away from each other, until they reach the correct relative positions. In other words, the gain of the orientation component (K_o) must be much smaller than the gain of the interagent component (K_a). Namely, these gains must be chosen such that $K_a \gg K_o$. This ensures that, in the presence of both the components, the orientation one becomes negligible, and the polygonal formation is correctly created. Once the agents are in the polygonal formation, only one of them is inside $\mathcal{S}^*$, and the formation is taken to the desired orientation.

3.2.3 Deformation of the polygon: bijective coordinates transformation

For many applications it is very useful to obtain formations with shapes different from regular polygons. The main idea is to obtain a formation with an arbitrary shape by deforming the regular polygon, as shown in Fig. 3.5. In this picture, the reference frame

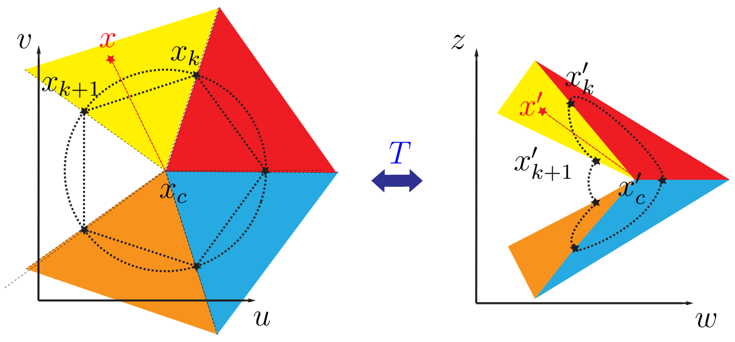


Figure 3.5: To obtain an arbitrary shape, the regular polygon is deformed by means of a bijective coordinates transformation

(w, z) represents the *real* reference frame; the *real* positions of the agents are measured with respect to the coordinate set (w, z). The reference frame (u, v) is an auxiliary reference frame. A bijective coordinates transformation T is introduced to relate the desired positions for the agents in (w, z) to the positions of the vertices of a regular polygon in (u, v).

Thus, the following control strategy is proposed:

1. Each agent measures its own position, and the positions of its neighbors, with respect to the real reference frame (w, z).

2. Each agent transforms these positions using the transformation T, and obtains the values of these positions with respect to the auxiliary reference frame (u, v).

3. Then, it calculates the control action as described in the previous sections, with respect to the auxiliary reference frame (u, v).

4. Finally, applying the inverse transformation, it finds the value of the control action with respect to the real reference frame (w, z). The control action can then be applied.

Thus, the obtained formation has the shape of a regular polygon with respect to the auxiliary reference frame (u, v), but has the desired shape with respect to the real reference frame (w, z).

Define now a bijective transformation of coordinates T which maps n arbitrary positions into the positions of the vertices of a regular polygon. It is only necessary to ensure that the distance between each couple of neighboring positions is less than the sensing range S_R.

Refer to the left–hand picture in Fig. 3.5. The (u, v) reference frame is partitioned, creating n triangular zones (where n is the number of agents in the formation). The partition is created drawing n rays: each ray starts at the circumcenter of the polygon x_c and passes through a vertex. Thus, the environment is partitioned into n zones, whose borders are these n rays.

Referring to the right–hand picture in Fig. 3.5, the (w, z) reference frame can be partitioned in a similar way. The partition is created drawing n rays: each ray starts at x'_c and passes through the desired position of an agent in the desired formation. x'_c is the image of x_c under the transformation T. The requirements on its position will be shown subsequently.

Once defined the partitions in the two coordinates sets, they need to be correlated by means of a bijective relation. This relation maps each vertex of the polygon in (u, v) into the desired position of an agent in the formation in (w, z). The circumcenter of the polygon $x_c = (u_c, v_c)^T$ is mapped into the point $x'_c = (w_c, z_c)^T$. Then, each triangular zone in the (u, v) reference frame is mapped into one triangular zone in the (w, z) reference frame. Referring to Fig. 3.5, for example, the triangular zone defined by the points (x_k, x_{k+1}) has to be mapped into the triangular zone defined by the points $\left(x'_k, x'_{k+1}\right)$, and vice versa. Thus, this mapping is defined as follows: $x \in (u, v)$ is inside the k–th zone (yellow zone in the left–hand picture in Fig. 3.5) if the argument of the vector $(x - x_c)$ is between the arguments of the vectors $(x_k - x_c)$ and $(x_{k+1} - x_c)$:

$$x \in \ k\text{–}th\ zone \text{ iff}$$

$$\angle\left(x - x_c\right) \in \left[\angle\left(x_k - x_c\right), \angle\left(x_{k+1} - x_c\right)\right[\tag{3.22}$$

and $x' \in (w,z)$ is inside the k–th zone (yellow zone in the right–hand picture in Fig. 3.5) if the argument of the vector $(x' - x'_c)$ is between the arguments of the vectors $(x'_k - x'_c)$ and $\left(x'_{k+1} - x'_c\right)$:

$$x' \in \ k\text{–}th\ zone \text{ iff}$$

$$\angle\left(x' - x'_c\right) \in \left[\angle\left(x'_k - x'_c\right), \angle\left(x'_{k+1} - x_c\right)\right[\tag{3.23}$$

Let $\bar{x} = \left(x^T, 1\right)^T \in \mathbb{R}^3$ and $\bar{x}' = \left(x'^T, 1\right)^T \in \mathbb{R}^3$. For each couple of corresponding triangular zones, a projective transformation [76] is exploited, that maps $\bar{x}$ into $\bar{x}'$. For the k–th couple of triangular zones:

$$\bar{x}' = M_k \cdot \bar{x} \tag{3.24}$$

The matrix M_k has the following structure:

$$M_k = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

where $a, b, c, d, e, f \in \mathbb{R}$. Each triangular zone is defined by three points (Fig. 3.5): (x_c, x_k, x_{k+1}) in the (u,v) coordinates set, and $\left(x'_c, x'_k, x'_{k+1}\right)$ in the (w,z) coordinates set. To find the matrix M_k, the following conditions are imposed:

$$\begin{cases} \bar{x}'_c & = M_k \cdot \bar{x}_c \\ \bar{x}'_k & = M_k \cdot \bar{x}_k \\ \bar{x}'_{k+1} & = M_k \cdot \bar{x}_{k+1} \end{cases} \tag{3.26}$$

Since $x_c, x'_c, x_k, x'_k, x_{k+1}, x'_{k+1} \in \mathbb{R}^2$, Eq. (3.26) represents a linear system of six equations, to find the six components of the matrix M_k.

It's easy to show that, if x_c, x_k and x_{k+1} are different and non–collinear, the six equations are linearly independent. Since x_c, x_k and x_{k+1} are respectively the circumcenter and two adjacent vertices of a regular polygon, they are never coincident or collinear.

A projective transformation maps a straight line into a straight line [76]. Thus the line connecting x_c and x_k is transformed into the line connecting x'_c and x'_k (Fig. 3.5). In other words, the borders of the k–th triangular zone in the (u,v) coordinates set are mapped into the borders of the k–th triangular zone in the (w,z) coordinates set, $\forall k = 1, \ldots, n$.

Since any linear transformation of a convex set yields to a convex set [77], each triangular zone is mapped into a convex set by M_k. Since the borders of each triangular zone in the (u,v) coordinates set are mapped into the borders of the corresponding triangular zone in the (w, z) coordinates set, then the matrix M_k maps every point of the k–th triangular zone in the (u, v) coordinates set into points of the k–th triangular zone in the (w, z) coordinates set, $\forall k = 1, \dots, n$.

Proposition 3.2. *The matrix M_k is invertible.*

Proof. Let

$$\begin{array}{lll} x_c & = (u_c, v_c)^T & \in \mathbb{R}^2 \\ x_k & = (u_k, v_k)^T & \in \mathbb{R}^2 \\ x_{k+1} & = (u_{k+1}, v_{k+1})^T & \in \mathbb{R}^2 \end{array} \tag{3.27}$$

be the positions of the center of the polygon, and of two adjacent vertices. Let

$$\begin{array}{lll} x'_c & = (w_c, z_c)^T & \in \mathbb{R}^2 \\ x'_k & = (w_k, z_k)^T & \in \mathbb{R}^2 \\ x'_{k+1} & = (w_{k+1}, z_{k+1})^T & \in \mathbb{R}^2 \end{array} \tag{3.28}$$

be their corresponding transformed points. The matrix M_k will be shown to be singular if and only if x'_c, x'_k, x'_{k+1} are coincident or collinear.

The matrix M_k is singular if $\det M_k = 0$:

$$\det M_k = \det \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} = a \cdot e - b \cdot d = 0 \tag{3.29}$$

Solving the linear system of six equations in Eq. (3.26), the condition in Eq. (3.29) can be rewritten as follows:

$$-\frac{-w_k z_{k+1} + w_c z_{k+1} - w_c z_k + w_k z_c + w_{k+1} z_k - w_{k+1} z_c}{u_{k+1} v_c - u_{k+1} v_k - v_c u_k + v_k u_c + u_k v_{k+1} - u_c v_{k+1}} = 0 \tag{3.30}$$

The denominator must be different from zero. By means of a simple translation of the coordinates set, it's always possible to consider

$$x_c = (u_c, v_c)^T = (0, 0)^T \tag{3.31}$$

Thus, the denominator is equal to zero if

$$-u_{k+1} v_k + u_k v_{k+1} = 0 \tag{3.32}$$

The condition in Eq. (3.32) is verified if and only if

- $x_k = x_{k+1}$, or
- the arguments of vectors $(x_k - x_c)$ and $(x_{k+1} - x_c)$ are equal.

These conditions are never verified, because x_k and x_{k+1} are two different vertices of the polygon, and x_c is the center of the circumcircle of the polygon.

Thus, matrix M_k is singular if the numerator in Eq. (3.30) is equal to zero. By means of a simple translation of the coordinates set, it's always possible to consider

$$x'_c = (w_c, z_c)^T = (0, 0)^T \tag{3.33}$$

Thus the numerator is equal equal to zero if

$$-w_k z_{k+1} + w_{k+1} z_k = 0 \tag{3.34}$$

The condition in Eq. (3.34) is verified if and only if

- $x'_k = x'_{k+1}$, or
- the arguments of vectors $(x'_k - x'_c)$ and $\left(x'_{k+1} - x'_c\right)$ are equal.

The first condition means that the desired position of two different agents must be different. This appears to be a very natural condition: it doesn't have any physical meaning to obtain a formation in which two or more agents occupy the same position at the same time.

To satisfy the second condition, x'_c must be non–collinear to any couple of desired position for the agents in the formation. This is the only condition that has to be satisfied during the choice of x'_c. Since the number of agents in the formation is finite, it is always possible to find a suitable position for x'_c. □

It is worth noting that the coordinates transformation defined so far can be calculated by each agent without any centralized controller. Each agent must only know the desired positions that define the shape of the formation.

The triangular zones have been assumed to be convex sets, so far. While this is always true in the (u, v) reference frame, because the triangular zones are defined by means of the vertices of a regular polygon, this condition can be violated in the (w, z) reference frame in many cases of interest (e.g. bottom left–hand picture in Fig. 3.6).

The borders of the triangular zones are rays starting at x'_c and passing through the desired position of an agent in the formation. If the angle between a couple of adjacent rays is greater than π, the corresponding zone is non–convex. To apply the strategy described so far, the partition needs to be modified, in order to obtain only convex zones. More specifically, the non–convex zone needs to be split, thus obtaining two convex triangular zones. To do this, an auxiliary point is introduced, which defines and additional ray.

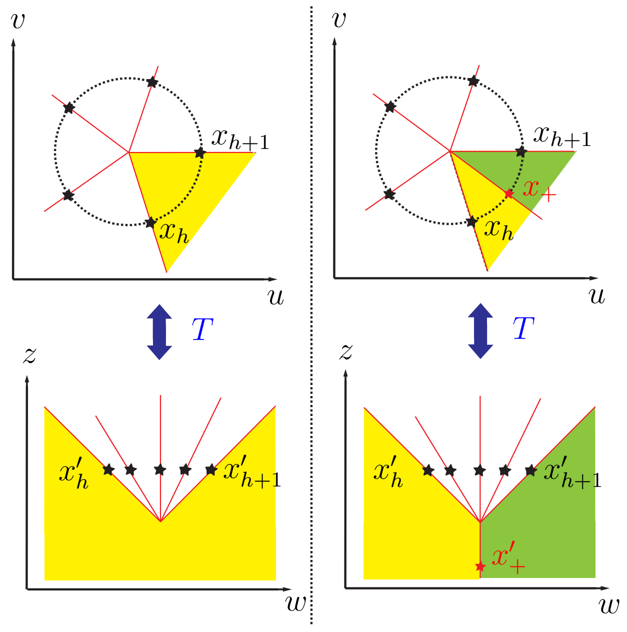


Figure 3.6: Adding an auxiliary point, all the zones of the partitions are convex

More specifically, let α'_h and α'_{h+1} be the arguments of vectors $(x'_h - x'_c)$ and $\left(x'_{h+1} - x'_c\right)$ respectively. Furthermore, let $\Delta\alpha'_h = \left|\alpha'_h - \alpha'_{h+1}\right|$. If $\Delta\alpha'_h > \pi$, a point x'_+ is introduced, such that

$$\angle\left(x'_+ - x'_c\right) = \alpha'_+ = \alpha'_h + \Delta\alpha'_h/2 \tag{3.35}$$

Then, the partition of the environment (right–hand picture in Fig. 3.6) is done considering $n+1$ points: the desired positions of the n agents, and the auxiliary point x'_+.

To make the transformation bijective, a corresponding auxiliary point, named x_+, must be added in the (u, v) reference frame as well. Let α_h and α_{h+1} be the arguments of vectors $(x_h - x_c)$ and $(x_{h+1} - x_c)$ respectively. The argument of vector $(x_+ - x_c)$ will be the following:

$$\angle\left(x_+ - x_c\right) = \alpha_+ = \alpha_h + \left|\alpha_{h+1} - \alpha_h\right|/2 = \alpha_h + \pi/n \tag{3.36}$$

It is worth noting that x_+ is used only for the definition of the bijective mapping: it does not directly influence the control action (it is not an attraction point for the agents).

Thus, as initially stated, this control strategy allows the creation of completely arbitrarily shaped formations. Some examples are provided in Fig. 3.7.

The bijective coordinates transformation T defined so far can be described as a variable matrix:

$$x'_i = T_i\left(x_i\right) \cdot x_i \tag{3.37}$$

where x_i and x'_i represent the position of the i–th agent, in the (u, v) and in the (w, z)

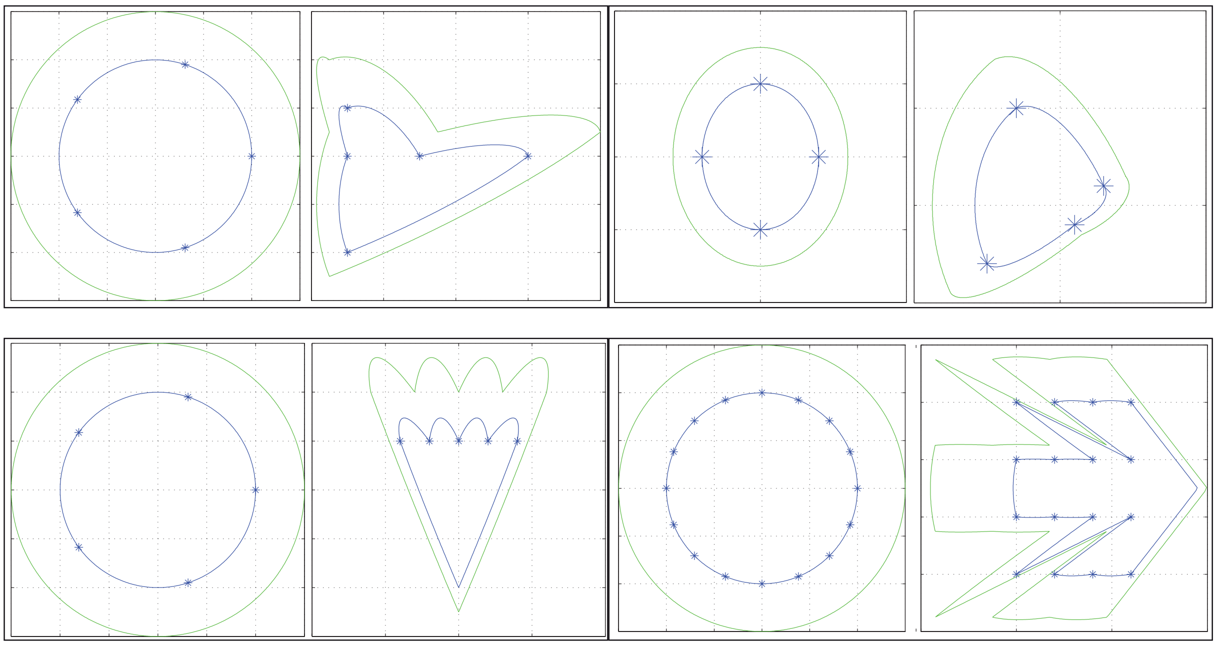


Figure 3.7: Some examples of different shapes that can be obtained starting from a polygonal formation, exploiting the bijective coordinates transformation described in the section

coordinates set, respectively. $T_i\left(x_i\right) = M_k$ if x_i is inside the k–th triangular zone. Let

$$\begin{array}{ll} \mathbf{x} & = \left[\bar{x}_1^T \dots \bar{x}_n^T\right]^T \quad \in \mathbb{R}^{3n} \\ \mathbf{x}' & = \left[\bar{x}_1'^T \dots \bar{x}_n'^T\right]^T \quad \in \mathbb{R}^{3n} \end{array} \tag{3.38}$$

Let the *total* transformation matrix $\mathbf{T}$ be defined such that

$$\mathbf{x}' = \mathbf{T}\left(\mathbf{x}\right) \cdot \mathbf{x} \tag{3.39}$$

The matrix $\mathbf{T}$ is a block diagonal matrix with the following structure:

$$\mathbf{T}\left(\mathbf{x}\right) = \begin{bmatrix} T_1\left(x_1\right) & 0 & \dots & \dots & 0 \\ 0 & T_2\left(x_2\right) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & T_{n-1}\left(x_{n-1}\right) & 0 \\ 0 & \dots & \dots & 0 & T_n\left(x_n\right) \end{bmatrix} \tag{3.40}$$

The matrix $\mathbf{T}$ is clearly invertible, since it is the block diagonal composition of invertible matrices.

Let $\mathbf{x_D}$ be the desired configuration of the agents in the (u, v) reference frame, i.e. if $\mathbf{x} = \mathbf{x_D}$ the agents create a formation with the shape of a regular polygon in the (u, v)

reference frame. Let $\mathbf{x}'_\mathbf{D}$ be the desired configuration of the agents in the (w, z) reference frame, i.e. if $\mathbf{x}' = \mathbf{x}'_\mathbf{D}$ the agents create a formation with the desired shape in the (w, z) reference frame. The coordinates transformation is defined such that

$$\mathbf{x}'_\mathbf{D} = \mathbf{T}\left(\mathbf{x}_\mathbf{D}\right) \cdot \mathbf{x}_\mathbf{D} \tag{3.41}$$

In Section 3.2.1, the control strategy has been proven to be asymptotically stable and local minimum free. In other words, applying this control strategy, the regular polygon formation is always created, namely

$$\lim_{t\to\infty} \mathbf{x}\left(t\right) = \mathbf{x}_\mathbf{D} \tag{3.42}$$

Hence, applying the coordinates transformation to Eq. (3.42):

$$\lim_{t\to\infty} \mathbf{x}'\left(t\right) = \lim_{t\to\infty} \mathbf{T}\left(\mathbf{x}\left(t\right)\right) \cdot \mathbf{x}\left(t\right) = \mathbf{T}\left(\mathbf{x}_\mathbf{D}\right) \cdot \mathbf{x}_\mathbf{D} = \mathbf{x}'_\mathbf{D} \tag{3.43}$$

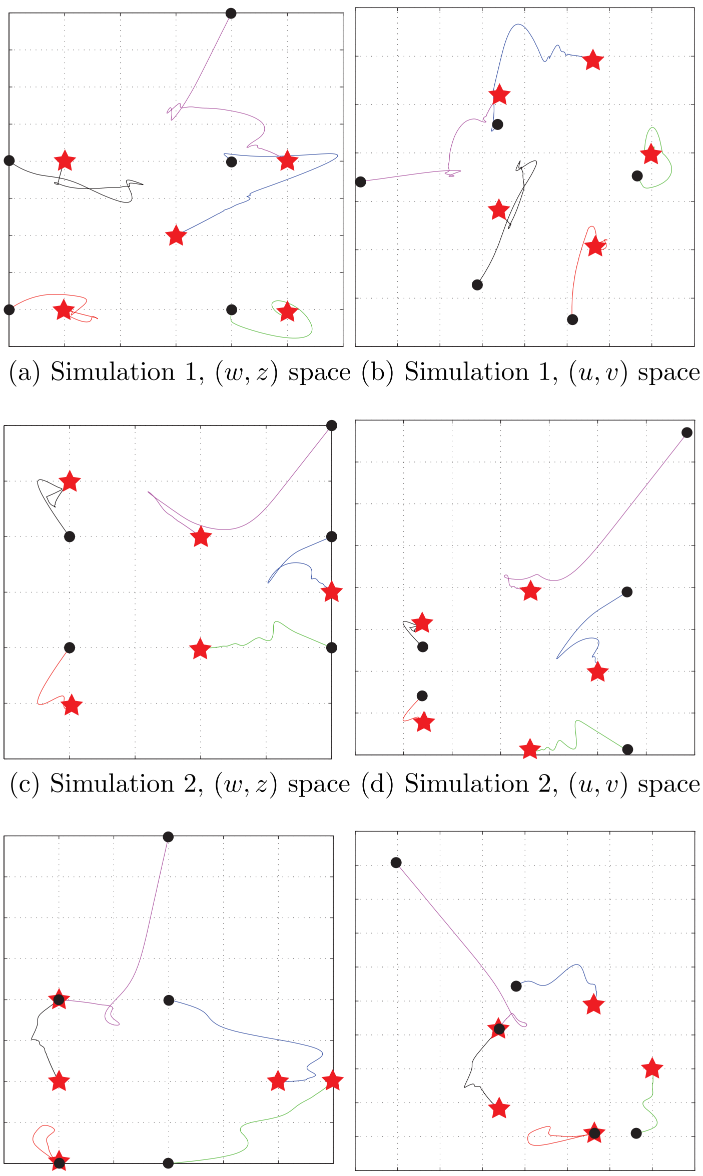
In other words, with this control strategy the desired formation is always created.

3.2.4 Simulations and experiments

Matlab simulations

Several Matlab simulations have been performed, for validation purpose. Point mass agents have been considered, with unitary mass. The presence of proximity sensors have been simulated: each agent only knew the positions of its neighbors (i.e. agents that are closer than the sensing range S_R). During the simulations, the number of the agents involved has been varied, as well as their desired positions. As expected, the agents always converge to the desired positions. The trajectories covered by five point mass agents realizing different formations are represented in Fig. 3.8. In the simulations, the following parameters have been used: $K_c = 80$, $K_a = 100$, $K_o = 30$. With these parameters, the time taken by the group to create the formation is always less then 20 seconds.

Fig. 3.9 shows the trajectories covered by five agents moving in the environment while keeping an arrow shaped formation. The movement of the formation is obtained by translating the point x'_c. The desired positions for the agents are represented as relative positions with respect to x'_c. Thus, as x'_c translates, even the minima of the composition of the potential fields translate. Therefore, the agents move preserving the shape of the formation, as shown in Fig. 3.9.


(a) Simulation 1, (w, z) space (b) Simulation 1, (u, v) space

(c) Simulation 2, (w, z) space (d) Simulation 2, (u, v) space

(e) Simulation 3, (w, z) space (f) Simulation 3, (u, v) space

Figure 3.8: Trajectories simulated with Matlab: black dots are the starting positions, red stars are the final positions. Trajectories are plotted with respect to the real reference frame (w, z) and the auxiliary one (u, v), respectively

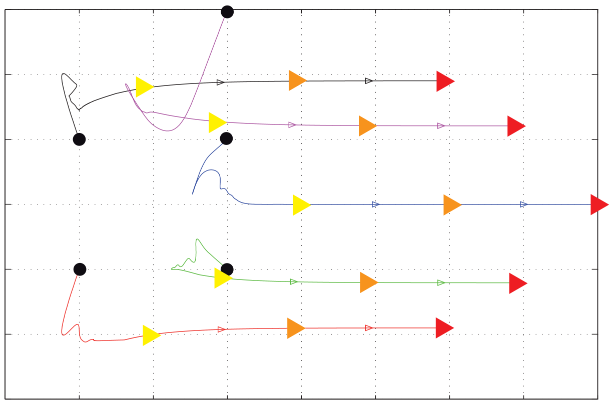


Figure 3.9: Agents moving while maintaining a formation: different colors represent different instant of time

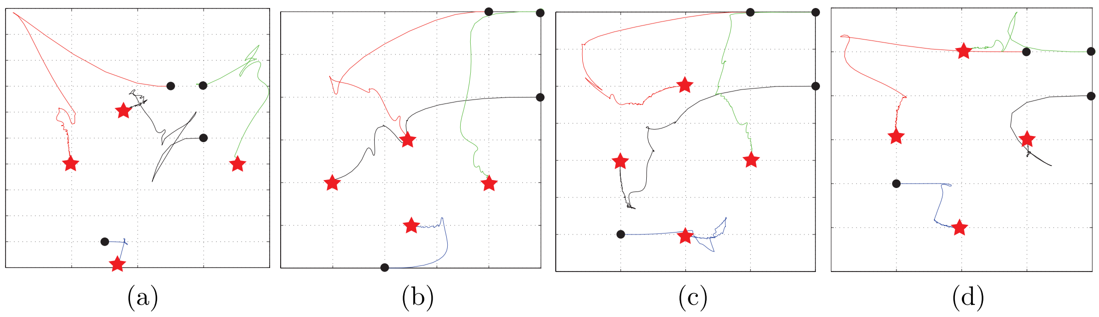


Figure 3.10: Trajectories simulated with Palyer/Stage: black dots are the starting positions, red stars are the final positions

Player/Stage simulations

To validate the control strategy with realistic simulations, several tests have been developed within the Player/Stage environment. More specifically, the control strategy has been implemented by means of the Player Robot Device Interface[1]: a useful feature of Player Robot Device Interface is the Stage Multiple Robot Simulator, which enables the simulation of algorithms with a realistic mobile robot model. More specifically, a differentially driven mobile robot model has been adopted. Although the control strategy described so far has been developed for holonomic point mass agents, it has been applied to nonholonomic robots exploiting the dynamic feedback linearization strategy described in [75].

In these simulations, each robot is supposed to have the capability to localize itself within the environment, while the presence of proximity sensors has been simulated: each robot only knew the positions of its neighbors (i.e. robots that are closer than the sensing range S_R).

During the simulations, the number of the involved agents has been varied, as well as their desired positions. As expected, the agents always converge to the desired positions. The trajectories covered by four simulated mobile robots realizing different formations are represented in Fig. 3.10. In the simulations, the following parameters have been used: $K_c = 8$, $K_a = 10$, $K_o = 3$. With these parameters, the time taken by the group to create the formation is always less then 50 seconds.

[1]http://playerstage.sourceforge.net/

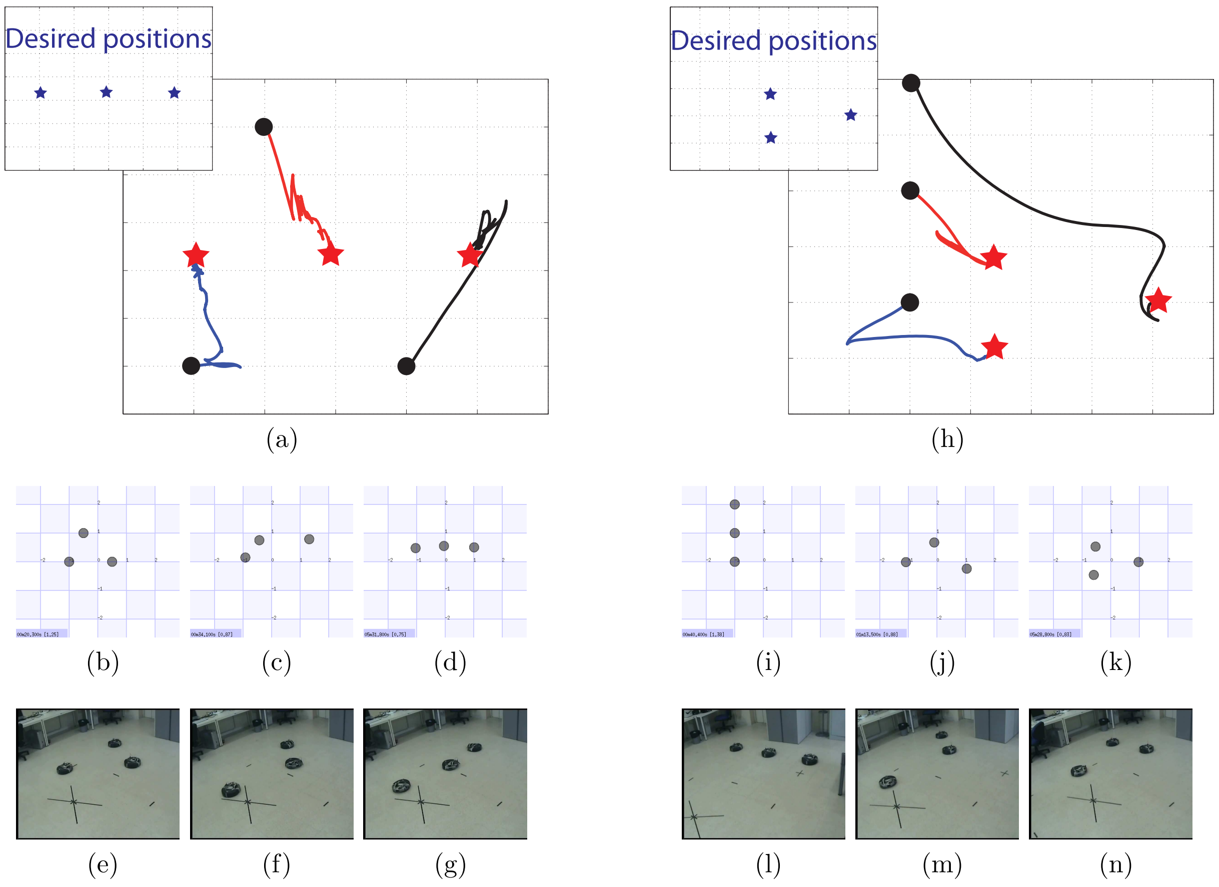


Figure 3.11: Trajectories traveled by the robots: black dots are the starting positions, red stars are the final positions; snapshots of simulated and real robots

Experimental results

Several experimental tests has been performed to validate the control strategy presented so far, exploiting the iRobot Roomba based experimental setup described in Chapter 2.

During the experimental tests, the initial positions of the robots have been varied, as well as the desired shape of the formation.

Two experiments are shown in Fig. 3.11. Figs. 3.11a, 3.11h show the trajectories traveled by the robots: data are extracted from the log of the odometric measurements of real robots. Figs. 3.11b, 3.11c, 3.11d and Figs. 3.11i, 3.11j, 3.11k show snapshots from the Stage simulation of the control algorithm. Figs. 3.11e, 3.11f, 3.11g and Figs. 3.11l, 3.11m, 3.11n show snapshots from the same experiment on real robots.

Due to the limited performances of the Gumstix board, the control program is executed with a quite slow frequency. Furthermore, the communication over WiFi network introduces some non–negligible delays. Hence, the presence of a large sampling period makes the use very small gains necessary to obtain a stable behavior of the system. In fact, during the experiments, the following parameters have been used: $K_c = 0.08$, $K_a = 0.08$, $K_o = 0.03$, $b = 0.2$.

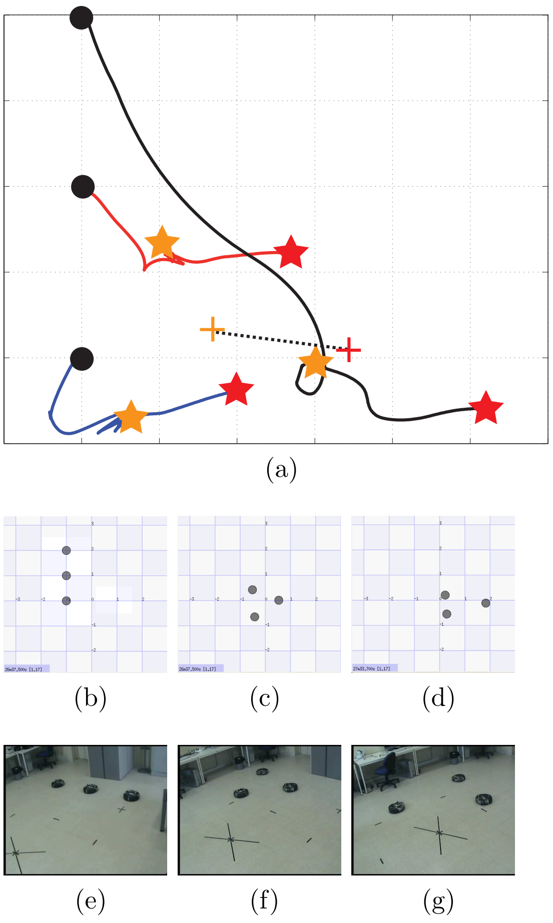


Figure 3.12: Agents moving while maintaining a formation: different colors represent different instants of time. The cross represents the current position of x'_c

Fig. 3.12 shows three robots that, after creating a formation, move in the environment while keeping the formation. Fig. 3.12a shows the trajectories traveled by the robots: data are extracted from the log of the odometric measurements of real robots. Figs. 3.12b, 3.12c, 3.12d show snapshots from the Stage simulation, while snapshots from the same experiment on the real group of robots are shown in Figs. 3.12e, 3.12f, 3.12g. Figs. 3.12b, 3.12e represent the initial condition of the robots (black dots in Fig. 3.12a). Initially the agents create the formation, as in the previous experiments, until they are in the configuration represented in Figs. 3.12c, 3.12f (yellow stars in Fig. 3.12a). Once the formation has been created, the movement of the formation is obtained by translating the point x'_c. Point x'_c is fixed for the first 60 seconds of the experiment, and then translates in the environment. Thus, as x'_c translates, even the minima of the composition of the potential fields translate. Therefore, the agents move preserving the shape of the formation, as shown in Figs. 3.12d, 3.12g (red stars in Fig. 3.12a).

3.3 Coordinated Closed–Curve Path Tracking for Multi–Robot Systems

3.3.1 Deformation of a circumference

With some modifications, the control strategy described in Section 3.2 can be exploited for trajectory tracking as well. Let $\mathcal{C}$ be the closed curve that defines the desired trajectory, and let T be the bijective mapping that relates $\mathcal{C}$ with a circumference. Then, a further coordinates transformation is introduced, defined by the following matrix:

$$M(t) = \begin{bmatrix} \cos\omega t & -\sin\omega t \\ \sin\omega t & \cos\omega t \end{bmatrix} \tag{3.44}$$

where t is the time, and $\omega \in \mathbb{R}$ is the angular speed. The matrix M defines the movement around the circumference. Thus, from the composition of M and T, the previously described artificial potential fields make a point move along the desired curve, with speed proportional to ω.

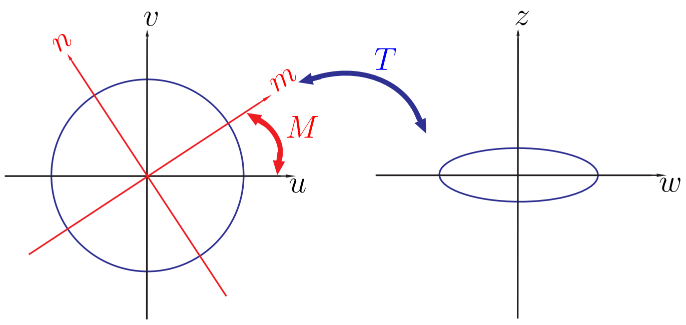


Figure 3.13: Deformation of a circumference to obtain an arbitrary curve

More specifically, refer to Fig. 3.13. The reference frame (w, z) represents the *real* reference frame; the *real* positions of the robots are measured with respect to the coordinate set (w, z). The reference frames (u, v) and (m, n) are auxiliary reference frames. The bijective coordinates transformation T relates the points of the curve $\mathcal{C}$ in (w, z) to the points of a circumference in (u, v). The matrix M relates the points of the circumference in (m, n) to the points of a circumference in (u, v). The two circumferences are equal, and their centers coincide with the origin of the reference frames (m, n) and (u, v) respectively. The reference frame (m, n) is obtained as a rotation at speed ω around the origin of the reference frame (u, v).

Thus, with the control law described in Eq. 3.19, the robots create a regular polygon formation with respect to the rotating reference frame (m, n). By means of the matrix

M, the robots create a polygon that rotates at speed ω with respect to the reference frame (u, v). In other words, the robots move at speed ω along the circumference, and the desired distance between each couple of neighbors is kept. Finally, by means of the transformation T, the robots move at constants speed along the curve $\mathcal{C}$.

The main drawback in this methodology is that it is not always possible to find a suitable transformation T once defined the desired shape of the curve $\mathcal{C}$. In fact, it is worth noting that the bijective transformation presented Section 3.2, as shown e.g. in Fig. 3.14, is not suitable to define any desired curve $\mathcal{C}$. In fact, when a group of mobile robots is controlled to create a desired formation, robots move to the desired positions and stop. Thus, only the final positions of the robots are of interest. As shown in Fig. 3.14, the vertices of the polygon are mapped into the desired positions, while the rest of the circumference assumes a strange and uncontrolled shape. This is clearly unacceptable in the path tracking application. In this case, in fact, the whole shape of the curve is clearly of interest.

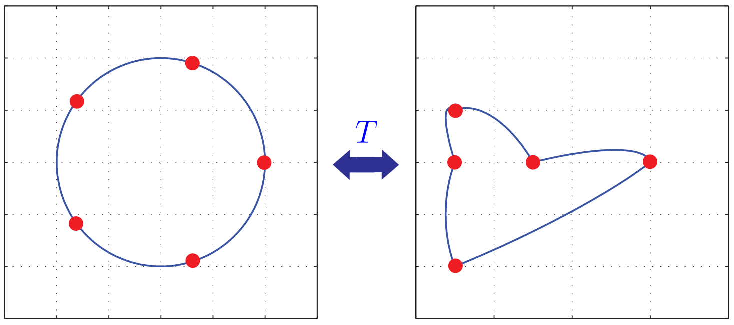


Figure 3.14: Bijective coordinates transformation is not suitable to define a closed–curve path

3.3.2 Paths described with implicit functions

In this section a modified control strategy is presented, to implement path tracking considering a wider class of curves. To overcome the difficulties in finding an appropriate transformation T to deal with completely arbitrarily shaped curves, the control law is modified. Specifically, a control law is introduced that makes the robots move along an arbitrarily shaped curve that can be described by means of an implicit function $f(x) = 0$, $x \in \mathbb{R}^2$.

To make a group of robots converge to the desired curve, they can be controlled to perform a gradient descent of f^2 [56]. Thus the following control law is introduced:

$$v_i = -K\nabla f^2 + f_{ti} + f_{di} - b\dot{x}_i \tag{3.45}$$

where K and b are positive constants, and b implements a damping action.

The term $-K\nabla f^2$ is orthogonal to the curve $\mathcal{C}$ in every point of the space. The role of this term is to make the robots converge to the desired curve $\mathcal{C}$.

The role of the term f_{ti} is to make the i–th robot move along the curve $\mathcal{C}$ at the desired speed. To this aim, the force f_{ti} is tangent to the curve at every time. More specifically, f_{ti} is described as follows:

$$f_{ti} = \omega \cdot R_\theta \cdot \frac{-\nabla f^2}{\|\nabla f^2\|} \tag{3.46}$$

where $\omega \in \mathbb{R}$ is a constant, proportional to the desired speed for the robot along the curve, and R_θ is a rotation matrix. The rotation matrix R_θ is defined as follows:

$$R_\theta(x_i) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.47}$$

where:

$$\theta(x_i) = \begin{cases} -\pi/2 & \text{if } f(x_i) > 0 \\ \pi/2 & \text{otherwise} \end{cases} \tag{3.48}$$

In other words, $\theta(x_i) = -\pi/2$ if the i–th robot is outside the curve $\mathcal{C}$, and $\theta(x_i) = \pi/2$ if it is inside the curve. As shown for example in Fig. 3.15, this definition of the rotation matrix R_θ leads to a movement along the curve in counterclockwise direction if $\omega > 0$, and in clockwise direction if $\omega < 0$.

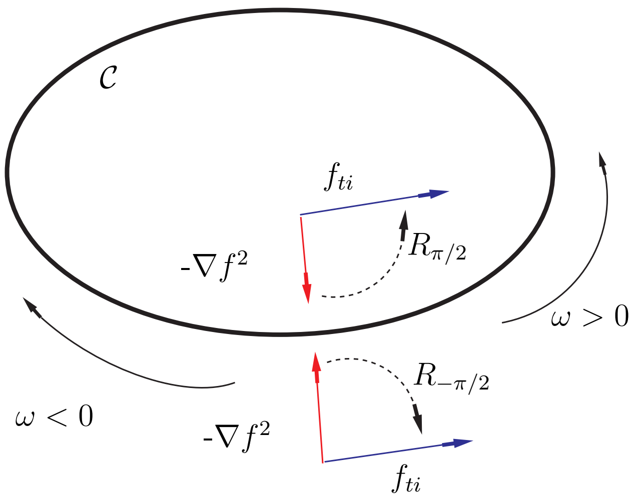


Figure 3.15: The force f_{ti} is perpendicular to the negative gradient of f^2

Clearly Eq. (3.46) is not defined when $\|\nabla f^2\| = 0$. This condition is verified only when the robot is on the curve $\mathcal{C}$. In this case, the control action is required to drive the robot along the curve. In other words, the force f_{ti} needs to be still tangent to the curve

$\mathcal{C}$. To obtain this, Eq. (3.46) may be slightly modified as follows:

$$f_{ti}\left(x_i\right) = \begin{cases} \omega \cdot R_\theta\left(x_i\right) \cdot \dfrac{-\nabla f^2\left(x_i\right)}{\left\|\nabla f^2\left(x_i\right)\right\|} & \text{if } f\left(x_i\right) \neq 0 \\ \omega \cdot R_\theta\left(x_p\right) \cdot \dfrac{-\nabla f^2\left(x_p\right)}{\left\|\nabla f^2\left(x_p\right)\right\|} & \text{otherwise} \end{cases} \tag{3.49}$$

where x_p is an arbitrary point on the line perpendicular to the curve $\mathcal{C}$ passing through x_i. Hence, the direction of the vector defined by Eq. (3.49) is the same of the desired one, defined in Eq. (3.46).

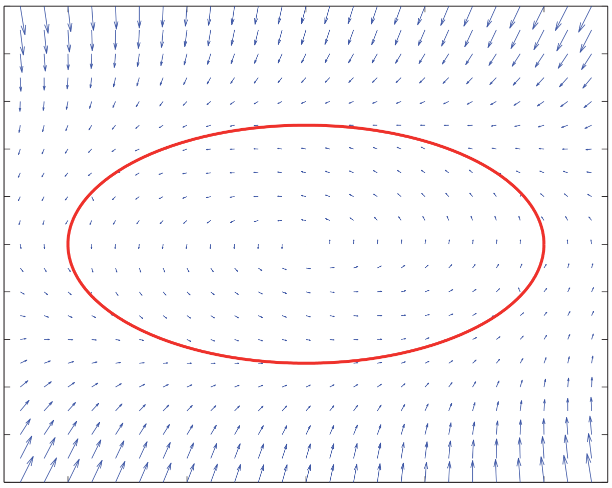


Figure 3.16: The composition of the negative gradient of f^2 and of the force f_{ti} makes the robots converge to the curve and move along it

Fig. 3.15 shows the composition of the negative gradient of f^2 and of the force f_{ti}: the composition of these two actions drives the robots to converge to the curve $\mathcal{C}$, and then move along it.

The role of the term f_{di} is to take the robot i at the desired distance from the other robots. This force is given by the composition of two terms:

$$f_{di} = \sum_{j=1;j\neq i}^{n} f_{rij} + \sum_{j=1;j\neq i}^{n} f_{qij} \tag{3.50}$$

The term f_{rij} implements a repulsive action if robot i and robot j are closer than the safety distance d_s. The value of d_s is the minimum distance that ensures that collisions between two robots never happen. More specifically:

$$f_{rij} = -\nabla_{x_i} V_{rij}\left(x_i, x_j\right) \tag{3.51}$$

and

$$V_{rij}\left(x_i, x_j\right) = \begin{cases} \dfrac{1}{2} K_r (d_{ij} - d_s)^2 & \text{if } d_{ij} \leq d_s \\ 0 & \text{otherwise} \end{cases} \tag{3.52}$$

where $d_{ij}(t) = \|x_i(t) - x_j(t)\|$, and K_r is a positive constant. The definition this function introduces a non–smooth control action, that can be avoided introducing a smooth bump function, as in [78].

To regulate the relative positions of the agents along the curve, the term f_{qij} is introduced. In fact, the term f_{rij} regulates only the euclidean distances among the agents. The fact that the euclidean distances between each couple of agents are equal to the desired one doesn't imply at all that the agents are deployed along the curve as desired.

Let u be a curvilinear abscissa, defined on the curve $\mathcal{C}$. The term f_{qij} is active only when robot i and robot j are on the curve $\mathcal{C}$. This term implements a repulsive action based on the value of the curvilinear abscissa that corresponds to the positions of robot i and robot j on the curve $\mathcal{C}$. Given the position of the robot $x_i \in \mathbb{R}^2$, the corresponding curvilinear abscissa u_i is defined as the value of the curvilinear abscissa that corresponds to the point of the curve that is closest to x_i.

The force f_{qij} is tangent to the curve $\mathcal{C}$ in the position of robot i. This force implements a repulsive action if the distance between robot i and robot j is less than the desired minimum distance u_d. Let u_i and u_j be the value of the curvilinear abscissa corresponding to the positions of robot i and robot j respectively, and let $u_{ij} = |u_i - u_j|$. Thus, f_{qij} is defined as follows:

$$f_{qij} = \begin{cases} K_u \cdot R_\theta(x_i) \cdot \dfrac{-\nabla f^2(x_i)}{\|\nabla f^2(x_i)\|}(u_i - u_j) & \text{if } u_{ij} \leq u_d \\ 0 & \text{otherwise} \end{cases} \tag{3.53}$$

where K_u is a positive constant.

To ensure collision avoidance, forces f_{rij} and f_{qij} are assumed to be much stronger than $-K\nabla f^2$ and f_{ti}. This is obtained by means of an appropriate choice of the parameters K_r and K_u.

It is worth noting that the force f_{rij} should be active only for collision avoidance. This means that, to avoid interference between f_{rij} and f_{qij}, the parameter d_s must be chosen in order to define a region much smaller than the one defined by u_d. Furthermore, the curve must be defined such that its curvature do not cause any collision among the agents.

Proposition 3.3. *Under the control law in Eq.* (3.45), *the robots asymptotically converge to the curve $\mathcal{C}$ and, after the transient, never leave it*

Proof. The motion of the robots can be considered as the composition of two components of motion:

- the motion in direction parallel to the curve $\mathcal{C}$,

- the motion in direction perpendicular to the curve $\mathcal{C}$.

Namely:

$$\dot{x}_i = \dot{x}_{i\perp} + \dot{x}_{i\|} \tag{3.54}$$

To prove the convergence of the motion to the curve $\mathcal{C}$, only the perpendicular component, namely $\dot{x}_{i\perp}$, is of interest.

As defined so far, the forces f_{ti} and f_{qij} don't have any component in the direction perpendicular to the curve $\mathcal{C}$. Thus, these forces do not influence the dynamics of $\dot{x}_{i\perp}$.

The force f_{rij} is active only for collision avoidance: this means that it can be different from zero only during the initial transient, when the robots start moving from their initial positions, and it can happen that two or more robots are closer than the safety distance. Therefore, f_{rij} can be considered zero after the initial transient.

Thus, from Eqs. (3.1), (3.45) the following dynamics may be obtained:

$$\ddot{x}_{i\perp} = -K\nabla f^2 - b\dot{x}_{i\perp} \tag{3.55}$$

To prove that the robot converges to the curve, it is necessary to prove the asymptotic stability of the following set:

$$\begin{cases} x_i \in \mathcal{C} \\ \dot{x}_{i\perp} = 0 \end{cases} \tag{3.56}$$

Consider the following Lyapunov candidate function:

$$V(x_i) = Kf^2(x_i) + \frac{1}{2}\|\dot{x}_{i\perp}\|^2 \tag{3.57}$$

which is trivially non–negative, and equal to zero only when the conditions in Eq. (3.56) are verified. The time derivative of this function is the following:

$$\dot{V}(x_i) = \left(K\nabla f^2 + \ddot{x}_{i\perp}\right)^T \dot{x}_{i\perp} \tag{3.58}$$

From Eq. (3.55):

$$\dot{V}(x_i) = -b\|\dot{x}_{i\perp}\|^2 \tag{3.59}$$

which is always less than or equal to zero. The asymptotic stability can be proved by invoking LaSalle's principle. □

Proposition 3.4. *Under the control law in Eq.* (3.45)*, after the transient, once on the curve the robots move along the curve at a constant speed*

Proof. With respect to the decomposition of the motion of the robot described in Eq. (3.54), in this case, only the component of the motion which is parallel to the curve $\mathcal{C}$, namely $\dot{x}_{i\|}$, is of interest.

By definition, the gradient of f^2 doesn't have any component in the direction parallel to $\mathcal{C}$. As stated before, the force f_{rij} can be considered zero after the initial transient.

Thus, from Eqs. (3.1), (3.45), the dynamics may be rewritten as follows:

$$\ddot{x}_{i\|} = f_{ti} + \sum_j f_{qij} - b\dot{x}_{i\|} \tag{3.60}$$

As stated before, the forces f_{qij} are much stronger than f_{ti}. Therefore, if the robots are on the curve and the distance between two neighbors is less than the desired one, the forces f_{qij} make them deploy along the curve as desired. Once the robots have deployed along the curve, the forces f_{qij} are no longer active, and Eq. (3.60) can be rewritten as follows:

$$\ddot{x}_{i\|}(t) = f_{ti}(t) - b\dot{x}_{i\|}(t) \tag{3.61}$$

Since only the dynamics in direction parallel to the curve are under consideration, it follows from Eq. (3.49) that, along this direction, $f_{ti}(t) \equiv \omega$. Thus, Eq. (3.61) can be rewritten as follows:

$$\ddot{x}_{i\|}(t) = \omega - b\dot{x}_{i\|}(t) \tag{3.62}$$

The differential equation in Eq. (3.62) can be easily integrated, thus obtaining

$$\dot{x}_{i\|}(t) = (\omega/b) + ce^{-bt} \tag{3.63}$$

where c is an arbitrary constant. Then, as time goes to infinity:

$$\lim_{t\to\infty} \dot{x}_{i\|}(t) = (\omega/b) = \text{constant} \tag{3.64}$$

This proves that, asymptotically, the robots move along the curve $\mathcal{C}$ at a constant speed proportional to ω. $\square$

It is worth noting that, since all the terms of the control strategy are independent of the total number of robots, sudden addition or subtraction of robots is managed automatically, as shown in the experiments described in Section 3.3.4.

Simulations and discussion

Several Matlab simulations have been performed for validation purposes. For example, Fig. 3.17 shows the path covered by three point mass agents that, starting from random

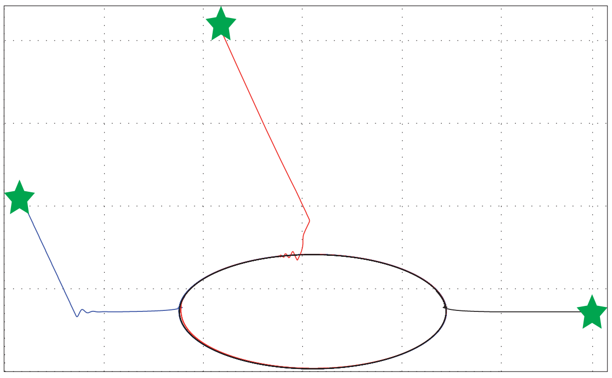


Figure 3.17: Three agents moving along an elliptic path

initial positions, initially move toward an elliptical curve, and then move along this curve at a constant speed. The control strategy works as expected: the agents are attracted to the curve until they reach it. Then they move along the curve, and are never forced to leave it.

The main drawback of the control strategy presented so far is that, even though many curves can be represented as implicit functions, with this formulation it is not possible to represent completely arbitrarily shaped curves. The next Section will show how to extend this control strategy, in order to deal with completely arbitrarily shaped curves.

3.3.3 Paths described with parametric functions

Generally speaking, a closed curve in $\mathbb{R}^2$ can be described by means of a parametric function $x = g(u)$, with $x \in \mathbb{R}^2$ and $u \in \mathbb{R}$. In the literature, many methods can be found to define these parametric functions. For example, arbitrarily shaped closed curves can be defined by means of Bezier curves, B–splines or NURBS [79].

Since, in general, it is not always possible to obtain an implicit formulation of the curve $\mathcal{C}$ from its parametric formulation, the algorithm will now be adapted, in order to avoid the use of the implicit formulation.

Let L be the length of the curve $\mathcal{C}$, i.e. the curvilinear abscissa $u \in [0, L]$. Since the curve $\mathcal{C}$ is closed, $g(0) = g(L)$.

The control law in Eq. (3.45) will then be modified, in order to allow the computation of the forces without the expression of $f(x)$.

In the control strategy presented in the previous section, the gradient descent of f^2 is performed in order to drive each robot toward the curve $\mathcal{C}$. The same result can be obtained replacing the term $(-K\nabla f^2)$ in Eq. (3.45) with the following term:

$$-K\nabla f^2(x_i) \rightarrow -K\nabla d^2(x_i) \tag{3.65}$$

where $d(x_i)$ is the distance between the i–th robot and the curve $\mathcal{C}$, i.e.:

$$d(x_i) = \min_{u \in [0,L]} \|x_i - g(u)\| \tag{3.66}$$

The other terms of the control law in Eq. (3.45) are modified accordingly. Specifically, the force f_{ti} is replaced by the force f'_{ti}, defined as follows:

$$f'_{ti}(x_i) = \omega \cdot R_\theta \cdot \frac{-\nabla d^2(x_i)}{\|\nabla d^2(x_i)\|} \tag{3.67}$$

and the force f_{qij} is replaced by the force f'_{qij}, defined as follows:

$$\begin{aligned} &\|f'_{qij}\| = \|f_{qij}\| \\ &f'_{qij}(x_i) = \|f'_{qij}\| \cdot R_\theta(x_i) \cdot \frac{-\nabla d^2(x_i)}{\|\nabla d^2(x_i)\|} \frac{(u_i - u_j)}{|u_i - u_j|} \end{aligned} \tag{3.68}$$

It is worth noting that, in case $\|\nabla d^2(x_i)\| = 0$, the control laws defined in Eqs. (3.67), (3.68) can not be computed. However, this issue may be solved as for the control law defined in Eq. (3.46): instead of x_i, it is possible to use any point x_p which is on the line perpendicular to $\mathcal{C}$ passing through x_i.

Hence, in the case of closed curves described with parametric function, the following control law is applied:

$$v_i = -K\nabla d^2 + f'_{ti} + f'_{di} - b\dot{x}_i \tag{3.69}$$

where

$$f'_{di} = \sum_{j=1;j\neq i}^{n} f_{rij} + \sum_{j=1;j\neq i}^{n} f'_{qij} \tag{3.70}$$

The following Propositions demonstrate the effectiveness of the proposed control law.

Proposition 3.5. *Under the control law in Eq.* (3.69)*, the robots asymptotically converge to the curve $\mathcal{C}$ and, after the transient, never leave it*

Proof. Consider the following Lyapunov function:

$$W(x_i) = Kd^2(x_i) + \frac{1}{2}\|\dot{x}_{i\perp}\|^2 \tag{3.71}$$

Proposition 3.3 can then be applied to prove the asymptotic stability of the set described in Eq. (3.56). □

Proposition 3.6. *Under the control law in Eq.* (3.69)*, after the transient, once on the curve the robots move along the curve at a constant speed*

Proof. As in Proposition 3.4, only the dynamics in direction parallel to the curve $\mathcal{C}$ may be considered. From Eq. (3.67), it follows that, along this direction

$$f_{ti}(t) \equiv f'_{ti}(t) \equiv \omega \tag{3.72}$$

Proposition 3.4 can then be applied to prove that, asymptotically, the robots move along the curve $\mathcal{C}$ at a constant speed proportional to ω. □

3.3.4 Implementation issues

In order to apply the previously described control law on simulated or real systems, it is necessary to approximate the control law itself. In fact, the parametric definition of the closed curve $\mathcal{C}$ will be approximated with a finite number of points, in a realistic scenario. The implementation of this approximation procedure will be described in the next subsection, and will be followed by the description of some Matlab simulations, and experiments or real robots.

Approximation of the control law

This section will show how to implement the previously described control strategy in case the curve $\mathcal{C}$ is defined by means of a finite number of points.

Let x_i be the position of the i–th robot. At each time, the robot can compute the closest point of the curve, i.e. the value u^* of the curvilinear abscissa such that

$$u^* = \arg\min_{u\in[0,L]} \left\| x_i - g(u) \right\| \tag{3.73}$$

It can happen that u^* is not uniquely defined, i.e. more than one point of the curve have the same minimum distance from the i–th robot. In particular, this can happen when the i–th robot is approaching the (non–convex) curve, and u^* defines the point where the robot enters the curve. In this case, u^* can be chosen randomly among the minimum distance points. Once the robot is on the curve, this ambiguity will not happen anymore.

Once defined u^*, the control law in Eq. (3.65) may be approximated as follows:

$$-K\nabla d^2 \approx -\nabla U_{att} \tag{3.74}$$

where

$$U_{att} = \frac{1}{2} K \left\| x_i - g(u^*) \right\|^2 \tag{3.75}$$

It is easy to show that the approximation is well posed. In fact, as the number of points used to describe the curve $\mathcal{C}$ goes to infinity,

$$\left\| x_i - g(u^*) \right\| \longrightarrow d(x_i) \tag{3.76}$$

The force f'_{ti} needs to be approximated as well. In order to do that, the composition of the negative gradient of $-K\nabla d^2$ and f'_{ti} is approximated as follows:

$$-K\nabla d^2 + f'_{ti} \approx -\nabla U^{\omega}_{att} \tag{3.77}$$

with

$$U^{\omega}_{att} = \frac{1}{2}K \left\| x_i - g\left((u^* + \omega) \mod L\right)\right\|^2 \tag{3.78}$$

where $(u) \mod L$ is the reminder of the division of u by L.

The choice of this kind of approximation can be justified as follows: under this control law, the robot is not attracted to $g(u^*)$, but it is attracted to $g\left((u^* + \omega) \mod L\right)$, where $\omega \in \mathbb{R}$ is proportional to the desired speed along the curve.

- When the robot is not on the curve, it is attracted to the curve $\mathcal{C}$ as desired, since $g\left((u^* + \omega) \mod L\right)$ is clearly a point of $\mathcal{C}$.
- When the robot is on the curve, the point $g(u^*)$ is the robot's own position. Thus, being attracted to $g\left((u^* + \omega) \mod L\right)$ it is forced to move along the curve, at a speed proportional to ω.

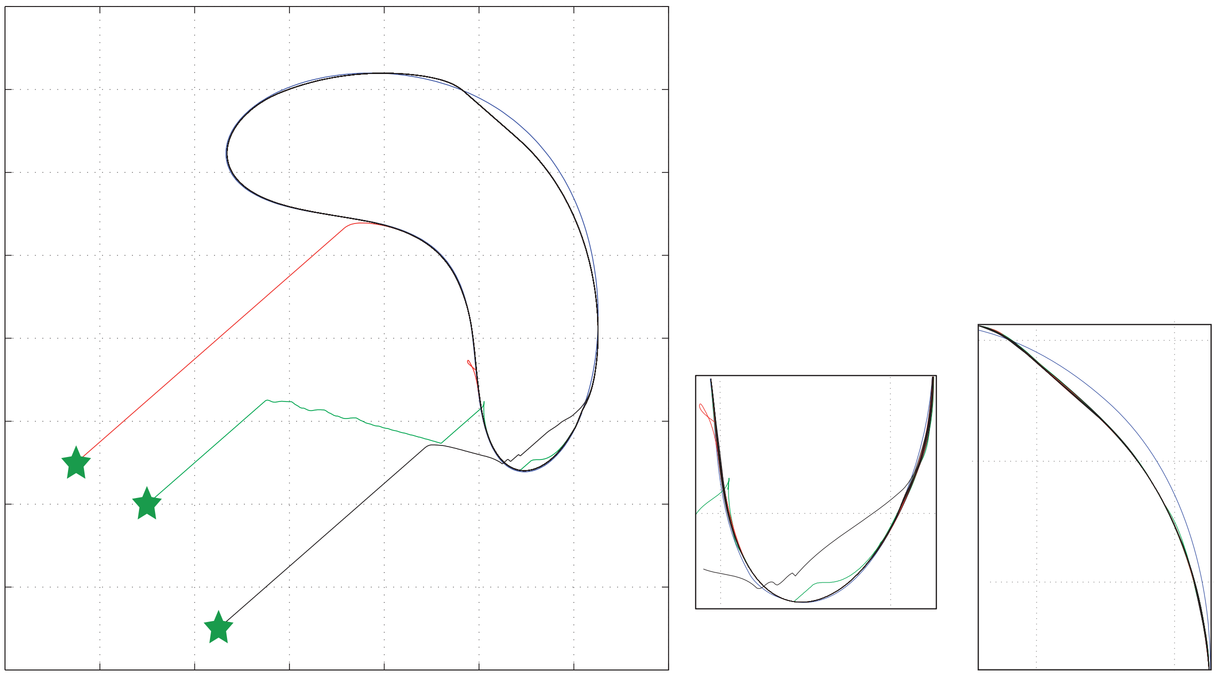
The other terms of the control law are defined as described in the previous section.

Thus, the approximated control law introduced in this section implements both the actions perpendicular and parallel to the curve $\mathcal{C}$, making the robots converge to the curve and move along it.

It is worth noting that the choice of the value of ω must be related to the shape of the curve, to guarantee a good tracking performance. In fact, if the curve, for instance, presents a sharp bend, a high value of ω will make the robots cross the bend according to a straight line, instead of following the curve as desired.

Matlab simulations

Several Matlab simulations have been performed for validation purpose. In these simulations, point mass agents have been considered, tracking the desired curve $\mathcal{C}$, defined by means of the B–spline formulation. As shown in Fig. 3.18, the agents, starting from random initial positions, reach the curve $\mathcal{C}$ and move along it. The speed of the agent is not uniform along the curve: this is obtained by means of a non–uniform discretization of the curve. This is useful to make the robots move faster in some zones and slower in some other zones. For example, this strategy can be exploited to slow down the robots while loading or unloading goods on them.



(a) (b) (c)

Figure 3.18: Three agents moving along the desired curve

Fig. 3.18a shows the trajectories covered by three agents that, starting from random initial positions, reach the desired curve and move along it. It can be seen that the tracking of the path is quite good, except for two zones, described in Fig. 3.18b and Fig. 3.18c.

Fig. 3.18b shows the transient behavior: when the agents approach the curve for the first time, they need a certain amount of time to obtain the desired distances among each others. In fact, this undesired behavior is not repeated anymore: once the agents have reached the correct distances, they track the curve as desired.

The wrong tracking of the path shown in Fig. 3.18c is due to the discretization of the curve. In fact, in this zone the discretization of the curve is coarser than the rest of the curve. A non–uniform discretization is useful if in some zones a precise tracking is not needed (e.g. because in some zones of the environment there are no obstacles), because it reduces the number of points to be stored to describe the curve.

Experiments

Several experiments have been performed exploiting the MORE–pucks experimental setup described in Chapter 2.

To deal with the fact that these robots are nonholonomic systems, the feedback linearization technique presented in [75] has been exploited.

During the experiments, the following values have been used for the parameters: $K = 100$, $K_u = K_r = 500$, $b = 15$, $\omega = 5$, with curves defined by 400 points.

Similarly to the simulations, experiments show the effectiveness of the control strategy described so far: in fact, after the transient, the robots correctly deploy along the curve and track it.

In Fig. 3.19, some snapshots of an experimental test involving seven robots are shown. Specifically, Fig. 3.19a shows the initial positions of the robots. As shown in Fig. 3.19b, initially only four robots are activated, and start tracking the curve. The sequential addition of the other three robots is shown in Figs. 3.19c, 3.19d 3.19e.

The positions of the robots have been recorded, and the mean value of the tracking error (i.e. the distance between the each robot and the curve) has been computed after 15 runs of the experiments. As shown in Fig. 3.20, the curve tracking is quite accurate since, after the transient, the mean error becomes less than $1cm$.

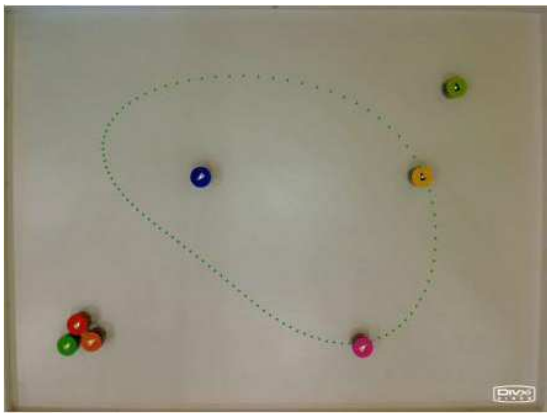
3.3.5 Presence of multiple tasks

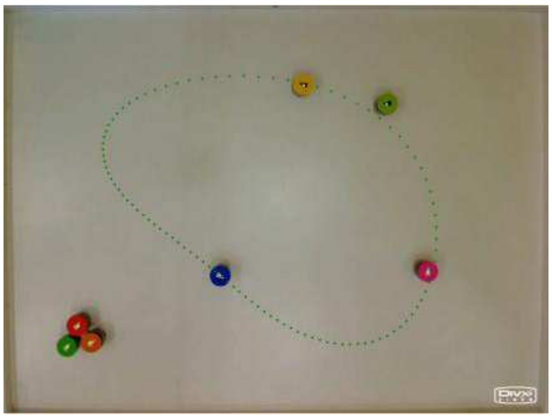
Discussion

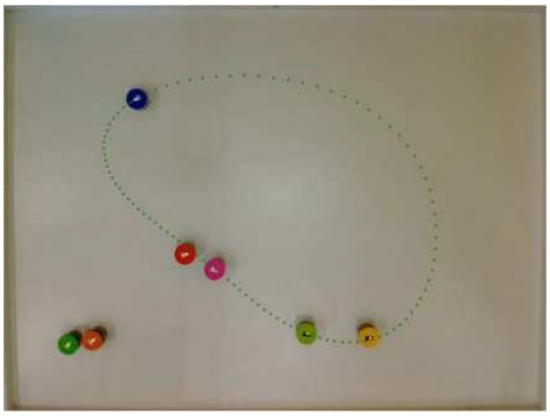
In several applications, multiple tasks to be completed (i.e. multiple curves to be tracked) may be simultaneously available. This scenario can arise, for instance, in industrial end–of–line applications, where different kinds of goods are to be delivered to different locations in the warehouse.

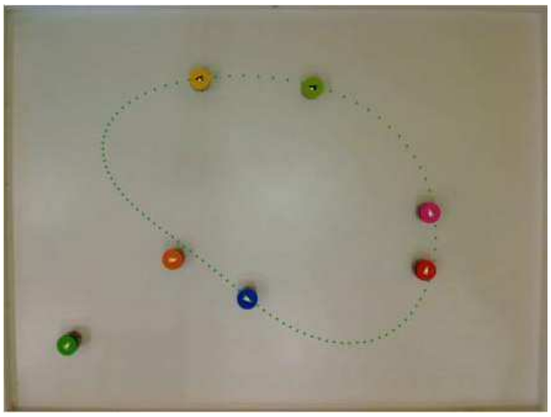
Typically, in industrial applications, a centralized system manages the different tasks to be completed. More specifically, the centralized management system allocates each robot to a predefined mission. The scenario under consideration is different: each robot can access a shared *task list*, and the robots are supposed to autonomously spread among the different tasks to be completed.

Similar problems have been widely studied in the last few years, under the class of *distributed task assignment* problems. Task assignment is the problem of spreading a set of agents to solve a finite number of tasks. Task assignment can be optimally solved in a centralized implementation: see e.g. [80–83] and references therein. Since, generally speaking, centralized implementations are less robust than distributed ones, several distributed task assignment algorithms have been introduced. One way to solve the task assignment problem in a distributed way is to let each agent compute a local estimate of the global situation: consensus algorithms [69] can be exploited for this purpose [84]. This kind of algorithms lead to a consistent estimate of the global situation among the


(a)


(b)


(c)


(d)

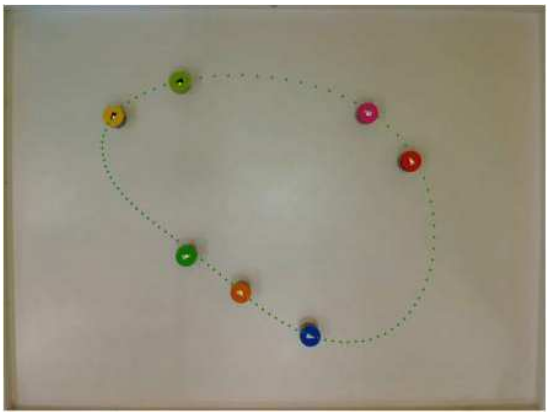

(e)

Figure 3.19: Snapshots of the experimental validation of the multi–robot curve tracking algorithm

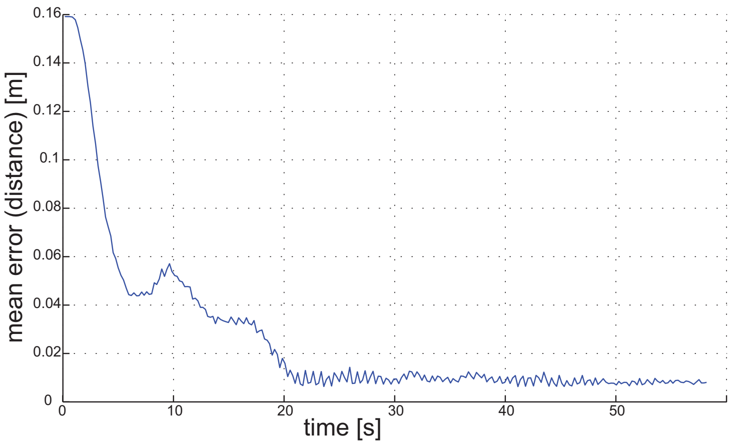


Figure 3.20: Mean error of the curve tracking

agents, but require a significant time to compute a solution [85].

Auction based algorithms (see e.g. [86,87] and references therein) have been shown to efficiently produce suboptimal solutions. Generally speaking, each agent places a bid for a task, and the highest bidder is assigned to the task. As shown in [85] and references therein, several decentralized algorithms have been introduced to compute the auction winner in a decentralized implementation.

As shown in [87], although auction based algorithms are computationally efficient, they are usually not robust to dynamically changing network topologies. However, dynamic changes in the network topology are likely to appear in the problem under consideration. More specifically, consider an industrial environment, where different robots become available at different time (e.g. because they have completed some previously assigned task). Furthermore, a quite simple framework in under consideration: the robots are indistinguishable, i.e. anyone can execute any task. For this purpose, a simple message passing algorithm in now introduced, that may be used for each robot to select the path to be tracked.

Message passing for path assignment

This section describes a simple message passing algorithm, which represents a *high level* control layer that enables each robot to select the right task.

When a new robot approaches a path, message passing starts among the robots that are already tracking the same path. The answer to the new robot is sent based on the number of robots that are already moving along the path (that can be simply computed in a decentralized manner, as explained later on). Loosely speaking, the new robot is not allowed to join the path if there already *too many* robots moving along it.

More specifically, the algorithm is based on the following assumptions and definitions:

1. The robots can communicate, by means of message passing, when their distance is less then a certain communication radius.

2. Each robot has a unique identifier (UID).

3. Let L be the length of the curve to be tracked, and let u_d be the desired distance between two neighboring agents on the curve. Then, $(L/u_d) = N \in \mathbb{N}$. In other words, u_d is defined so that so that an exact number N of robots is allowed to track the curve.

4. If the distance between two robots is less than or equal to u_d, they can communicate.

5. The k–th robot is attracted to the curve $\mathcal{C}$ by means of the control strategy described in the previous Section, while n robot are already moving along the curve $\mathcal{C}$. Thus, if it finds that $n \geq (L/u_d)$, the k–th robot will move to a different task.

6. Let $\Delta_{j+1} = |u_{j+1} - u_j|$ and $\Delta_{j-1} = |u_{j-1} - u_j|$ be the distances, in terms of curvilinear abscissa, between the j–th robot and its neighbors along the curve.

Thus, the proposed algorithm is the following:

- The k–th robot sends to the j–th robot a message with its own UID, UID_k.

- The j–th robot starts Algorithm 1.

Algorithm 1 Reply of the j–th robot to the k–th robot request

```
1:  if (Robot $j$ is not on the curve) then
2:      Robot $j$ → Robot $k$: $msg = [0, 0]$
3:  else
4:      if (($\Delta_{j+1} > u_d$) AND ($\Delta_{j-1} > u_d$)) then
5:          Robot $j$ → Robot $k$: $msg = [1, 1]$
6:      else
7:          Robot $j$ → Robot $j+1$: $msg_{out} = [UID_k, UID_j, 0]$
8:          Algorithm 2
9:      end if
10: end if
```

Algorithm 1 describes how the j–robots computes the reply message to the request of the k–th robot. The reply message is a two–bit message, and its meaning is the following:

Algorithm 2 Message passing among the robots on the curve, to understand if one more robot is allowed to track the curve as well

```
$msg_{in}$ = incoming message
$msg_{out}$ = outgoing message
The $i$–th robot receives $msg_{in}$:
$msg_{in} = [UID_k, UID_j, m]$, $m = 0, 1$
 1: if $msg_{in}(2) == UID_i$ then
 2:    Robot $i \to$ Robot $k$: $msg_{out} = [1, msg_{in}(3)]$
 3: else
 4:    if $msg_{in}(3) == 1$ then
 5:       Robot $i \to$ Robot $i-1$: $msg_{out} = msg_{in}$
 6:    else
 7:       if $(\Delta_{i+1} > u_d)$ then
 8:          Robot $i \to$ Robot $i-1$:
             $msg_{out} = [UID_k, UID_j, 1]$
 9:       else
10:          Robot $i \to$ Robot $i+1$:
             $msg_{out} = [UID_k, UID_j, 0]$
11:       end if
12:    end if
13: end if
```

- $msg = [0, 0]$: the j–th robot is not on the curve $\mathcal{C}$.
- $msg = [1, 0]$: the j–th robot is on the curve $\mathcal{C}$, and $n \geq (L/u_d)$. The k–th robot must move to a different task.
- $msg = [1, 1]$: the j–th robot is on the curve $\mathcal{C}$, and $n < (L/u_d)$. The k–th robot can move along the curve $\mathcal{C}$.

If the condition in line 4 of Algorithm 1 is true, then the distance between the j–th robot and its neighbors is strictly greater than u_d. Thus, under Assumption 3, the number of robots on the curve $\mathcal{C}$ is less than the maximum allowed, and the k–th robot is allowed to move along the curve $\mathcal{C}$ as well (Fig. 3.21a).

Otherwise, if this condition is not verified for the j–th robot, it is necessary to check whether it is verified for another robot on the curve $\mathcal{C}$ (Fig. 3.21b). This is the purpose of Algorithm 2. In this case, the message is a vector with three components:

1. The first component is the UID of the k–th robot.
2. The second component is the UID of the j–th robot. This is the robot that started the message passing along the curve.

3. The last component can be 0 or 1:
    - it is set to 0 if the k–th robot is not allowed to move along the curve $\mathcal{C}$.
    - it is set to 1 if the k–th robot is allowed to move along the curve $\mathcal{C}$.

If the condition in line 7 of Algorithm 2 is true, then the distance between the i–th robot and its following neighbor is strictly greater than u_d. Thus, under Assumption 3, the number of robots on the curve $\mathcal{C}$ is less than the maximum allowed, and the k–th robot is allowed to move along the curve $\mathcal{C}$ as well (Fig. 3.21c). Thus, the third component of the outgoing message is set to one, and this message is sent back to the previous robot. The message is delivered to the j–th robot, that allows the k–th robot to move along the curve $\mathcal{C}$.

Conversely, if the condition in line 7 of Algorithm 2 is not verified for any robot on the curve $\mathcal{C}$, then the number of robots currently on the curve is greater than or equal to the maximum allowed (Fig. 3.21d). In this situation, the message passes through all the robots, and no one of them sets to 1 the third component of the message.

When the message comes back to the to the j–th robot, the condition in line 1 is true. The message passing among the robots on the curve ends, and the j–th robot sends to the k–th one the correct answer.

To quantify the complexity [88] of this algorithm, the worst case may be considered: starting from the j–th robot, the message passes through all the robots until it reaches the $(j-1)$–th, and then goes back until it reaches the j–th one again. Let n be the number of robots currently on the curve $\mathcal{C}$. In the worst case, the message passing along the curve involves $2\,(n-1)$ messages. Furthermore, one message is sent from the k–th to the j–th robot, and the answer is sent back. Thus, the total number of messages exchanged in the worst case is $2n$. Hence, the communication complexity of this algorithm is linear with the number of robots involved.

3.4 A Graph–Based Collision–Free Distributed Formation Control Strategy

3.4.1 Weighted Graph-Based Formation Achieving

This section will describe a formation control strategy, obtained exploiting a consensus–based algorithm. For further details, see **APPENDIX**. Hence, consider a group of single

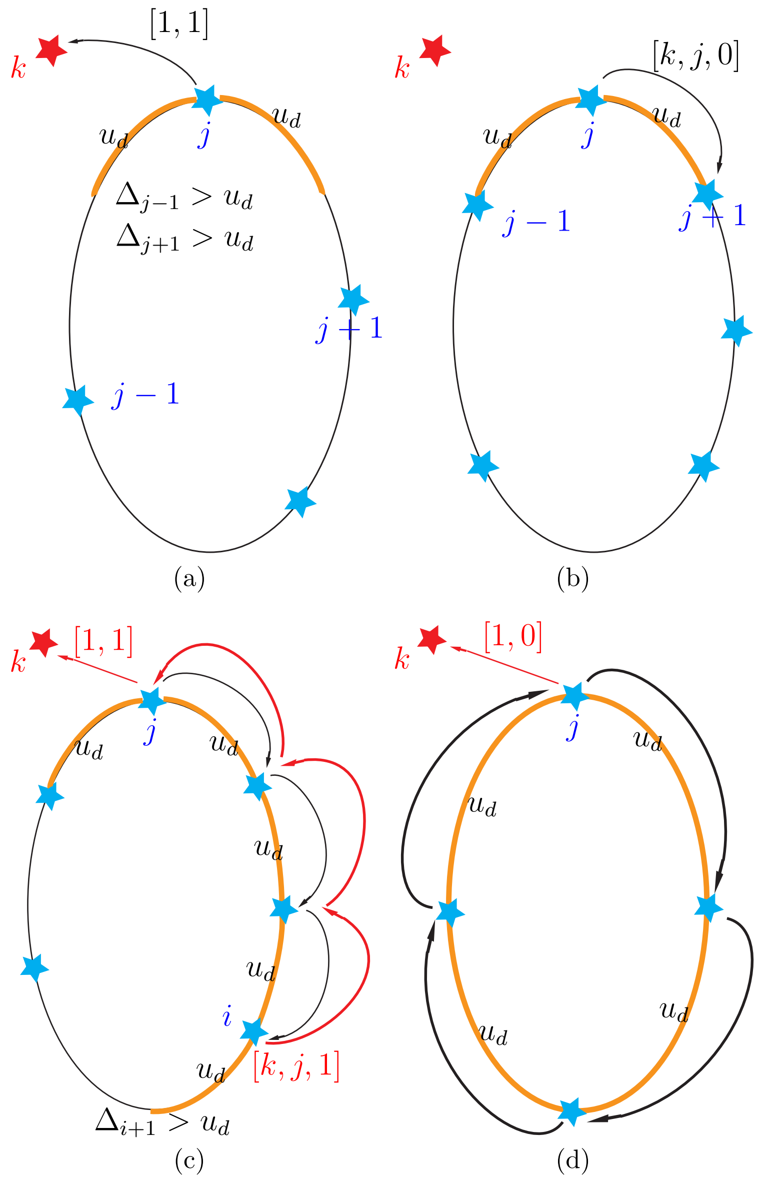


Figure 3.21: Message passing algorithm

integrator agents, whose dynamics are described as follows:

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} w_{ij}(x)(x_i - x_j) \tag{3.79}$$

Given an appropriate choice of the edge–weights $w_{ij}(x)$, $\forall\ (v_i, v_j) \in E$, it is possible to obtain any desired formation: the convergence of each edge–lengths to some desired values will be formally proven: this implies the creation of the desired formation. Furthermore, avoidance of collisions among the agents will be addressed as well within the same control law. More specifically, it will proven that, given a safety distance δ, if the initial configuration of the system is such that all the inter–agents distances are strictly greater than δ, then they will never go below this value.

Let l_{ij} be the edge vector between agents i and j, i.e. $l_{ij} = x_i - x_j$. Furthermore, let a collision–free realization of a graph G be defined as

$$\mathcal{D}^{\epsilon}_{G,\delta} = \left\{ x \in \mathbb{R}^{nN} :\ (\delta + \epsilon) \leq \|l_{ij}\| \leq D_M, \forall (v_i, v_j) \in E \right\} \tag{3.80}$$

for some positive ϵ. D_M is the maximum allowed distance that guarantees connectivity between the agents.

Then, define an edge–tension function V_{ij} as follows (Fig. 3.22a):

$$V_{ij}(\delta, x) = \begin{cases} \alpha_{ij}\left(\coth\left(\dfrac{\|l_{ij}\| - \delta}{K_{ij}}\right) + \dfrac{\|l_{ij}\|}{K_{ij}} - V_{ij}^{min}\right) \\ \hfill \text{if } (v_p, v_v) \in E \\ \\ 0 \hfill \text{otherwise} \end{cases} \tag{3.81}$$

where $K_{ij} > 0$ is a constant, $\alpha_{ij} > 0$ is a value used to define the *intensity* of the inter–robot influence [73], and $V_{ij}^{min} > 0$ is defined such that

$$\min_{\|l_{ij}\| > \delta} V_{ij}(\delta, x) = 0 \tag{3.82}$$

This function is non–negative, and has a strict minimum in $\|l_{ij}\| = D_{ij}$, with

$$D_{ij} = \delta + \frac{1}{2} K_{ij} \log\left(3 + 2\sqrt{2}\right) \tag{3.83}$$

The choice of the value of the constant $K_{ij} > 0$ is related to the position of the minimum of the edge–tension function, i.e. the desired distance for each couple of agents. From Eq. (3.81), it follows that

$$\frac{\partial V_{ij}(\delta, x)}{\partial x_i} = \begin{cases} \alpha_{ij}\left(-\text{csch}^2\left(\dfrac{\|l_{ij}\| - \delta}{K_{ij}}\right) + 1\right) \cdot \dfrac{(x_i - x_j)}{K_{ij}\,\|l_{ij}\|} \\ \hfill \text{if } (v_i, v_j) \in E \\ \\ 0 \hfill \text{otherwise} \end{cases} \tag{3.84}$$

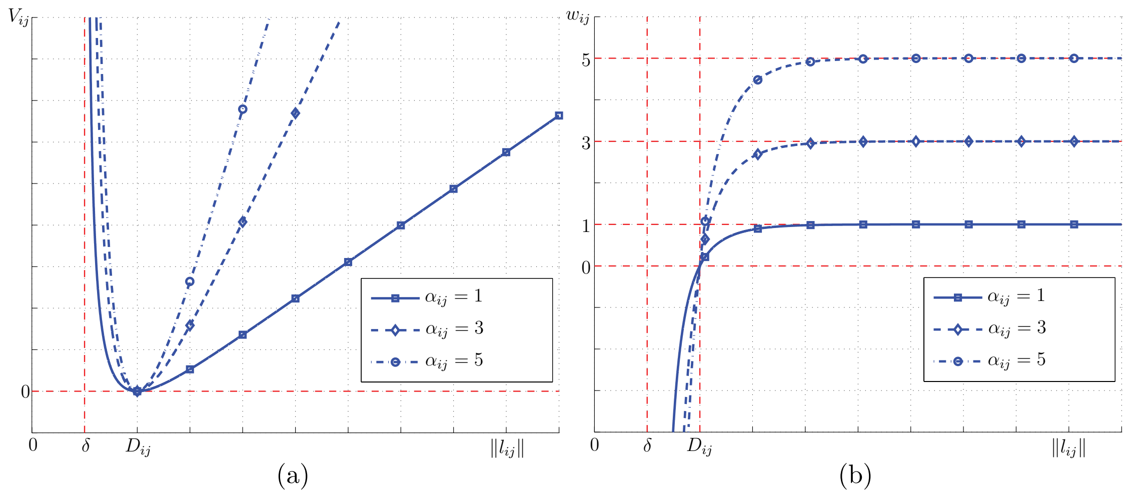


Figure 3.22: Edge–tension function $V_{ij}(\delta, x)$ with respect to $\|l_{ij}\|$ (Fig. 3.22a) and edge–weight function with respect to $\|l_{ij}\|$ (Fig. 3.22b)

The total tension energy of the graph G can then be defined as follows:

$$V(\delta, x) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} V_{ij}(\delta, x) \tag{3.85}$$

The edge–weights are defined as follows (Fig. 3.22b):

$$w_{ij} = \alpha_{ij} \left(-\text{csch}^2 \left(\frac{\|l_{ij}\| - \delta}{K_{ij}} \right) + 1 \right) \tag{3.86}$$

For ease of notation, hereafter α_{ij} will be assumed to be equal to 1, $\forall i, j$. Nevertheless, all the following proofs still hold for arbitrary values of $\alpha_{ij} > 0$.

Proposition 3.7. *Given an initial position $x_0 \in \mathcal{D}^{\epsilon}_{G,\delta}$, for some $\epsilon > 0$, then, if the system is driven by the control law in Eq.* (3.79)*, with the edge–weights defined in Eq.* (3.86)*, the total tension energy of the graph G defined in Eq.* (3.85) *does not increase.*

Proof. From Eqs. (3.79), (3.84), (3.86), the control law of the system can be rewritten as follows:

$$\dot{x}_i = -\nabla_{x_i} V(\delta, x(\tau)) \sum_{j \in \mathcal{N}_i} K_{ij} \|l_{ij}\| \tag{3.87}$$

Assume that $x(\tau) \in \mathcal{D}^{\epsilon'}_{G,\delta}$, for some $\epsilon' > 0$, at time τ. The total tension energy of the graph $V(\delta, x)$, defined in Eq. (3.85), is a positive function, and is zero only in the desired configuration, i.e. $\|l_{ij}\| = D_{ij}\ \forall l_{ij} \in E$.

The time derivative of the total tension energy function is defined as follows:

$$\dot{V}\left(\delta, x\left(\tau\right)\right) \quad = \nabla_x V\left(\delta, x\left(\tau\right)\right)^T \cdot \dot{x}\left(\tau\right) = -\sum_{i=1}^{N} \frac{\dot{x}_i^T \dot{x}_i}{\sum\limits_{j\in\mathcal{N}_i} K_{ij} \left\|l_{ij}\right\|} \tag{3.88}$$

Thus, for any $x\left(\tau\right) \in \mathcal{D}_{G,\delta}^{\epsilon'}$, $\dot{V}\left(\delta, x\left(\tau\right)\right)$ is non–positive, which proves the statement. □

The total tension energy function $V\left(\delta, x\right)$ has been defined in Eq. (3.85) as the sum of positive definite functions $V_{ij}\left(\delta, x\right)$, that are described in Eq. (3.81). Since these functions are clearly equal to zero only when the agents are in the desired configuration, it is possible to conclude that the desired formation is a global minimum for the total tension energy $V\left(\delta, x\right)$. However, it is possible for the system to evolve to some local minima of the total tension energy. In order to avoid local minima, the Virtual Relabeling algorithm, that will be described in Section 3.4.3, may be exploited.

In order to ensure that the presented control algorithm avoids collisions between agents achieving formation, the following proposition is provided.

Proposition 3.8. *Given an initial position $x_0 \in \mathcal{D}_{G,\delta}^{\epsilon}$, for some $\epsilon > 0$, under the control law in Eq.* (3.79)*, with the edge–weights defined in Eq.* (3.86)*, collisions among the agents are always avoided.*

Proof. Let δ be the safety distance for the agents, i.e. if the distance between each couple of agents is greater than δ, collisions are avoided. The proof of the statement is based on the fact that, as $V\left(\delta, x\left(\tau\right)\right)$ decreases (or at least does not increase), no edge–length will approach δ.

In order to prove that, let

$$\hat{V}_\epsilon = \max_{x\in\mathcal{D}_{G,\delta}^{\epsilon}} V\left(\delta, x\right) \tag{3.89}$$

Since, inside the set $\mathcal{D}_{G,\delta}^{\epsilon}$, the function $V\left(\delta, x\right)$ is bounded, this maximum exists. Let M_1 be the number of edges whose length is less than D_{ij}, and let M_2 be the number of edges whose length is greater than or equal to D_{ij}. Thus, the maximum $\hat{V}_\epsilon$ is obtained when M_1 edge–lengths are equal to the minimum allowed length, i.e. $\left\|l_{ij}\right\| = (\delta + \epsilon)$, while M_2 edge–lengths are equal to the maximum allowed length i.e. $\left\|l_{ij}\right\| = D_M$. Furthermore, M_1 or M_2 are also allowed to be equal to zero. Thus:

$$\hat{V}_\epsilon = M_1 \left(\coth\left(\frac{(\delta+\epsilon)-\delta}{K^*}\right) + \frac{\delta+\epsilon}{K^*}\right) - V_*^{min} + M_2 \left(\coth\left(\frac{D_M-\delta}{K^*}\right) + \frac{D_M}{K^*} - V_*^{min}\right) \tag{3.90}$$

where K^* and V_*^{min} are the mean values of K_{ij} and V_{ij}^{min}.

A bound for the minimal edge–length will now be found, that can generate this value for the total tension energy. Consider this total tension energy as if it were generated from one single edge, whose edge–length is $\hat{l}_\epsilon \leq (\delta + \epsilon)$, while all the other edge–lengths are equal to D_{ij}, thus their contribution to the total tension energy is zero. The edge–length $\hat{l}_\epsilon$ is defined such that

$$\hat{V}_\epsilon = \coth\left(\frac{\hat{l}_\epsilon - \delta}{K}\right) - V_\epsilon^{min} \tag{3.91}$$

where $V_\epsilon^{min} > 0$. To prove the statement, it is necessary to prove that $\hat{l}_\epsilon > \delta$. Substituting Eq. (3.91) into Eq. (3.89):

$$\begin{aligned}&\coth\left(\frac{\hat{l}_\epsilon - \delta}{K}\right) = \\ &V_\epsilon^{min} + M_1\left(\coth\left(\frac{\epsilon}{K^*}\right) + \frac{\delta+\epsilon}{K^*} - V_*^{min}\right) + M_2\left(\coth\left(\frac{D_M - \delta}{K^*}\right) + \frac{D_M}{K^*} - V_*^{min}\right)\end{aligned} \tag{3.92}$$

From the definition of the edge–tension function in Eq. (3.81):

$$\left(\coth\left(\frac{D_M - \delta}{K^*}\right) + \frac{D_M}{K^*} - V_*^{min}\right) \geq 0 \tag{3.93}$$

Since $M_1 \geq 0$, $M_2 \geq 0$ and $V_\epsilon^{min} > 0$, the following inequality holds:

$$\coth\left(\frac{\hat{l}_\epsilon - \delta}{K}\right) \geq 0 \tag{3.94}$$

Then, it is possible to conclude that $\left(\hat{l}_\epsilon - \delta\right) > 0$, which implies $\hat{l}_\epsilon > \delta$. Thus, since $\hat{l}_\epsilon$ is bounded from δ, then, as V decreases (or at least does not increase), no edge–length will tend to δ. □

3.4.2 Obstacle avoidance

One of the main issues that arises when robots have to be coordinated in an unstructured or unknown environment is that they have to take into account the presence of obstacles.

Consider, without loss of generality, the case where a robot detects an obstacle at a distance $d_o \leq d_{sens}$, where d_{sens} represents the effective range of the on board sensors able to detect obstacles all around the robot. In that case, as described in [71], a *virtual agent* is projected on the obstacle by the robot that detects it. While in [71] artificial potential fields are used for collision avoidance purposes, the control strategy described in this section automatically deals with collision avoidance, by including the virtual agents (i.e.

the obstacles) into the previously described graph–based algorithm. More specifically, this control algorithm can be extended simply by introducing a new *virtual edge* to the connectivity graph to represent the link between the real robot and the virtual agent.

It is worth noting that the number of virtual agents can be different with respect to the number of detected obstacles. In fact, as long as robots are moving in an unknown environment, they cannot distinguish an obstacle from another one. This means that different robots detecting the same object will project on it different virtual agents. As an example, consider the system represented in Fig. 3.23: three robots are moving close to an obstacle and, when robots R_1 and R_2 detect it, they project on the estimated surface of the obstacle two different virtual agents V_1 and V_2.

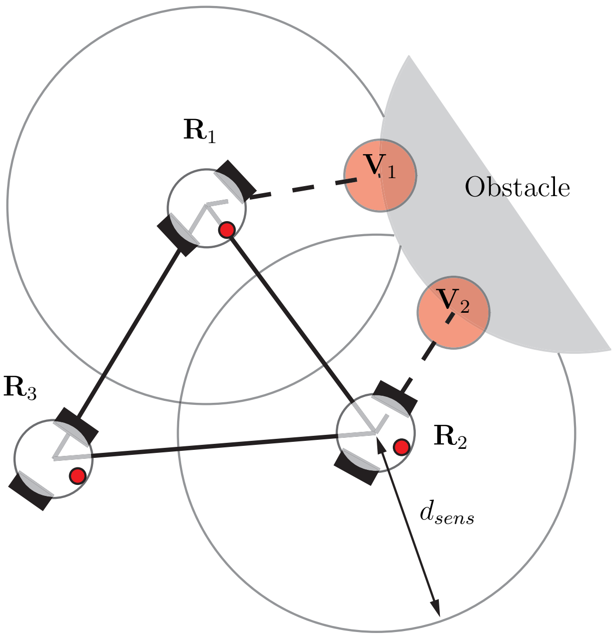


Figure 3.23: Three robots moving in formation: two of them (R_1, R_2) detect an obstacle and project on it the corresponding virtual agents (V_1, V_2). Virtual edges are represented as dashed lines.

Assumption 3.1. *The distance between each couple of obstacles is supposed to be greater then the size of a robot.*

Without loss of generality, consider the case where, while N agents are moving in the environment, the p–th one senses an obstacle. In this case, the dynamics of the agents that don't sense the obstacle are not directly influenced by its presence. On the contrary, the p–th agent defines a virtual agent, whose position corresponds to the position of the obstacle. The dynamics of the virtual agent are described as

$$\dot{x}_v = f\left(x_v, t\right) \tag{3.95}$$

This function is completely unknown: even if the obstacle is static, the position of the virtual agent depends on the motion of the p–th agent, and on the curvature of the surface of the obstacle, that is supposed to be unknown in advance. Furthermore, if the obstacle is not static (e.g. the obstacle is a non–cooperative vehicle), its law of motion is supposed to be unknown.

Proposition 3.9. *The edge–weight function introduced in Eq.* (3.86) *ensures the avoidance of collisions with obstacles.*

Proof. To include the virtual agent defined once an obstacle is sensed, the total tension energy of the graph G defined in Eq. (3.85) may be modified as follows:

$$V(\delta, \delta_o, x) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V_{ij}(\delta, x) + V_{pv}(\delta_o, x) \tag{3.96}$$

where $V_{pv}(\delta_o, x)$ is the edge–tension function related to the edge between the p–th real agent and the v–th virtual agent, and δ_o is the safety distance required between robots and obstacles. Namely:

$$V_{pv}(\delta_o, x) = \begin{cases} \alpha_{pv}\left(\coth\left(\dfrac{\|l_{pv}\| - \delta_o}{K_v}\right) + \dfrac{\|l_{pv}\|}{K_v} - V_{pv}^{min}\right) & \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (v_p, v_v) \in E & \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

where l_{pv} is the edge vector between the p–th real agent and the v–th virtual agent, i.e. $l_{pv} = x_p - x_v$, and $\alpha_{pv} > 0$ is a constant value that can be used to modulate the intensity of the interactions between real and virtual agents.

Thus, to prove the avoidance of collisions, Proposition 3.8 can be applied with this modified total tension energy function. □

Using the same approach, obstacle avoidance can be ensured even in the presence of more than one obstacle sensed by more than one agent (i.e. many virtual agents are added to the graph).

The value of the constant $K_v > 0$ can be chosen such that, when an agent senses an obstacle and defines a virtual agent, the corresponding edge–weight is always negative, thus always introducing a repulsive action.

It is worth noting that, in the presence of virtual agents, since their movement is not influenced by the position of the real robots, the graph becomes directed. This causes that Proposition 3.7 does not hold when virtual agents are added to the graph. This implies that the shape of the formation is not preserved in the presence of obstacles. However, the multi–robot system is not supposed to embed the virtual agents inside the formation: virtual agents are introduced only for collision avoidance purposes. Hence, robots will overcome the obstacles without maintaining the shape of the formation, i.e. by performing split–and–rejoin maneuvers, or by reducing the inter–robot distances. Examples of these maneuvers will be shown in the simulations described in the next section.

3.4.3 Local minima avoidance

The following section describes an algorithm to let the robots autonomously escape from local minima of the total tension energy function $V(\delta, x)$, introduced in Eq. (3.85).

Virtual Relabeling

As demonstrated in Proposition 3.7, the desired formation configuration is the global minimum of the of the total tension energy function $V(\delta, x)$ introduced in Eq. (4.131). However, it is possible for the system to evolve to some local minumum configuration. To make the robots escape from local minima, the *virtual relabeling* algorithm may be exploited.

As long as the communication graph is connected, all the robots can calculate the position of the centroid of the group. Then, by computing their own position with respect to the centroid of the group, they can find an agreement on the position they should occupy in the final formation. In order to do this, each robot needs to acquire information from all the other robots of the group. For this purpose, the *data broadcasting* algorithm, that

```
ind := [1 ... N];
while 1 do
    new_ind := ind;
    [Xc, Yc] := CalculateCoM();
    v1 := GetClockwise([Xc, Yc]);
    v2 := CreateSorted(v1);
    if v2(i) != ind(i)
        new_ind(i) := v2(i);
    end if
    ind := new_ind;
    AchieveFormation(ind);
end while
```

Table 3.1: Pseudo code for *virtual relabeling.*

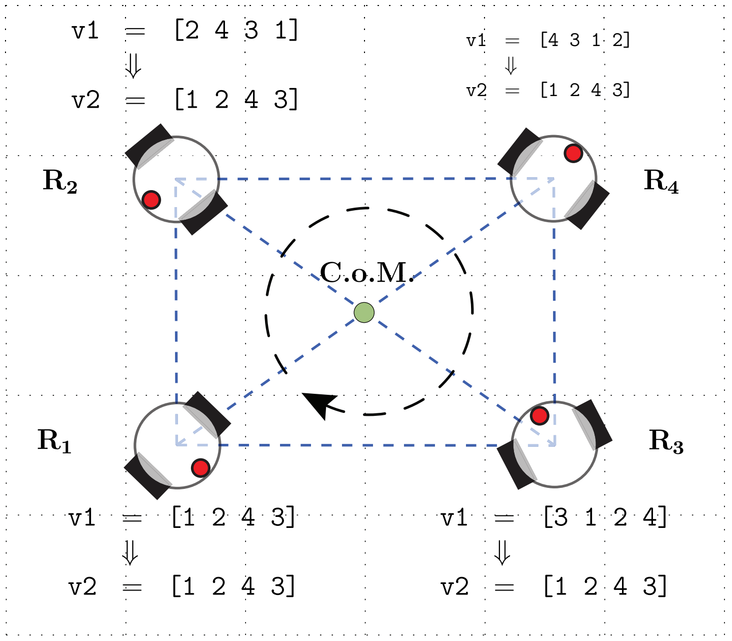


Figure 3.24: Example of four robots stuck in a local minimum configuration while achieving a square formation. The indices vectors `v1` and `v2` are defined by the relabeling algorithm.

will be introduced in Section 3.4.3, may be exploited.

Therefore, the *virtual relabeling* algorithm, defined as in Table 3.1, may implemented on each robot. More specifically:

- `[Xc, Yc] := CalculateCoM();` calculates the centroid of the group exploiting directly acquired and broadcast data;
- `v1 := GetClockwise([Xc, Yc]);` creates a vector where the indices of all the detected robots are saved according to clockwise direction with respect to the centroid;
- `v2 := CreateSorted(v1);` creates a new index vector where all the indices are stored from 1 up to N starting from the previously defined vector `v1`.

To better clarify the relabeling algorithm, in Fig. 3.24 a typical example of local minima for a system involving four robots achieving a square formation is depicted. In particular, the *virtual relabeling* algorithm is applied by each robot in order to calculate the correct neighbors configuration.

As reported, each robot calculate its own `v1` vector depending on its position with respect to the centroid and the teammates. Then, starting from `v1`, the vector `v2` is calculated in order to redefine the actual label that each robot should use in order to achieve the right configuration. In the depicted example, the robots of the group reach an agreement on `v2` and each of them is able to detect that robots $\mathbf{R_3}$ and $\mathbf{R_4}$ are in the wrong position with respect to their teammates.

Data Broadcasting

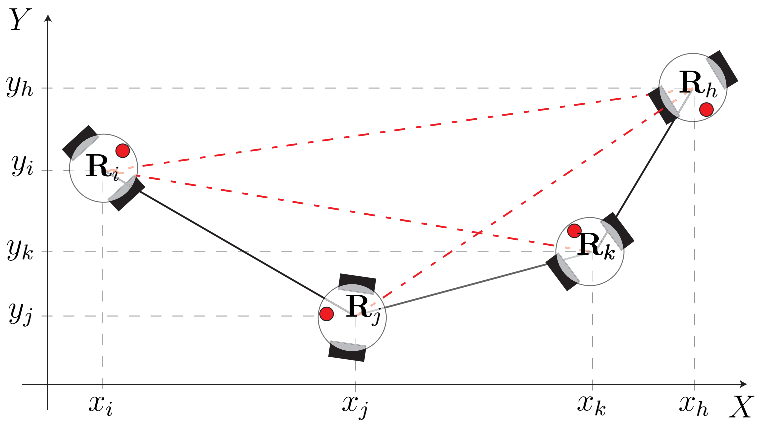


Figure 3.25: Example of data broadcasting. Solid lines represent the existing communication links while dash–dotted lines represents the broadcast ones. As an example, robot $\mathbf{R_j}$ broadcasts the values $[(x_k - x_j),\ (y_k - y_j)]^T$ and $[(x_h - x_j),\ (y_h - y_j)]^T$ to robot $\mathbf{R_i}$, thus allowing it the to localize teammates that are not directly seen.

One of the main problem related with robots with a limited communication range is that they can not always acquire information about the whole swarm, that is some robots can not see each other. Hence, the virtual relabeling algorithm introduced in Section 3.4.3 can not be implemented. However, as long as the communication graph is connected, this problem may be overcome by introducing data broadcasting between teammates, i.e. by allowing each robot to transmit information about the relative position of connected teammates with respect to itself, thus transforming *de facto* a connected communication graph into a complete graph. As an example, consider the three robots depicted in Fig. 3.25. The corresponding *Neighbor* sets are defined as

$$\mathcal{N}_i = \{\mathbf{R}_j\} \qquad \mathcal{N}_j = \{\mathbf{R}_i,\ \mathbf{R}_k\} \qquad \mathcal{N}_k = \{\mathbf{R}_j,\ \mathbf{R}_h\} \quad \mathcal{N}_h = \{\mathbf{R}_k\}$$

Generally speaking, given a couple of communicating robot, it is possible to define $\Delta_{ij}^x = (x_i - x_j)$ and $\Delta_{ij}^y = (y_i - y_j)$ as the relative distance between robot i and robot j on the x-axis and on the y-axis respectively. Thus, the i-th robot can estimate the relative position of the h-th one by exploiting the following equations:

$$(x_h - x_i) = (x_j - x_i) + \Delta_{hj}^x, \quad (y_h - y_i) = (y_j - y_i) + \Delta_{hj}^y$$

In order to reduce the measurement errors, the data broadcast by each robot are grouped into a data packet containing the sender ID and the relative position of their neighbors with respect to itself. Each of the transmitted data is associated with a *hop count* c_{ij} ($\forall i = 1 \ldots N, j \in \mathcal{N}_i$) that is incremented each time a robot broadcasts position data that are not directly measured, thus allowing the receiver to chose the data with the lowest *hop count*. The general scheme of the string transmitted by the generic j-th robot of a swarm is the $4\|\mathcal{N}_j\|+2$ elements vector defined as:

Vector element:	**j**	$\|\mathcal{N}_j\|$	i	$x_i - x_j$	$y_i - y_j$	c_{ij}
Vector index:	0	1	3	4	5	6

where $\|\mathcal{N}_j\|$ is the cardinality of the neighbors subset and the terms 3–6 are repeated $\forall i \in \mathcal{N}_j$.

As an example, by considering the system depicted in Fig. 3.25, the data transmitted by $\mathbf{R}_j$ are stored in the following vector:

j	3	i	Δ^x_{ij}	Δ^y_{ij}	1	k	Δ^x_{kj}	Δ^y_{kj}	1	h	Δ^x_{hj}	Δ^y_{hj}	2

It is worth noting that the data broadcasting algorithm may be exploited to improve the obstacle avoidance ability of the system as well. In fact, each robot may also broadcast the position of the virtual agents defined when an obstacle is detected, as described in Section 3.4.2.

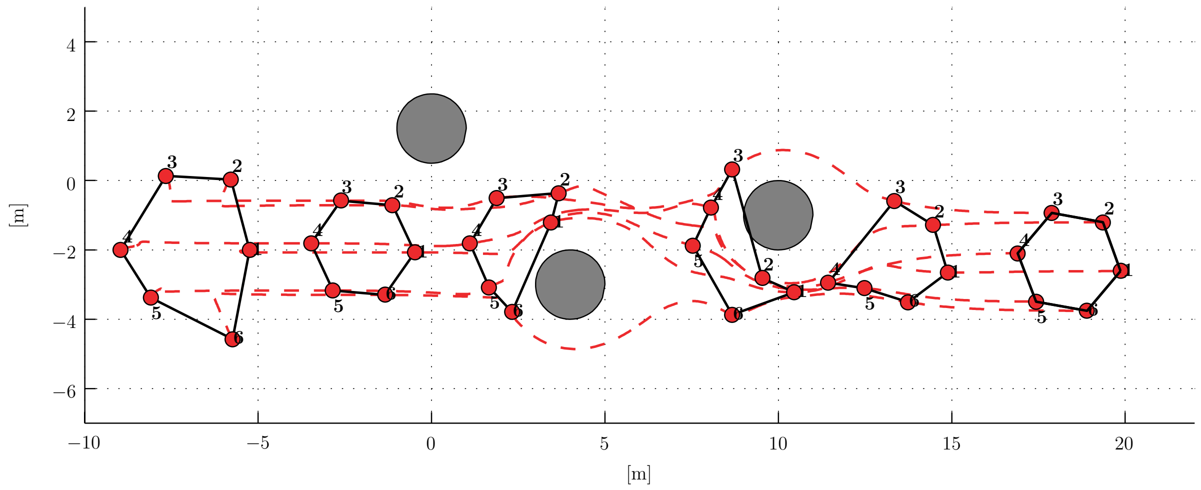
The data broadcasting algorithm requires the communication graph to be connected. In order to ensure connectivity of the communication graph, several strategies may be exploited (see e.g [29, 67] and references therein).

3.4.4 Simulations and Experiments

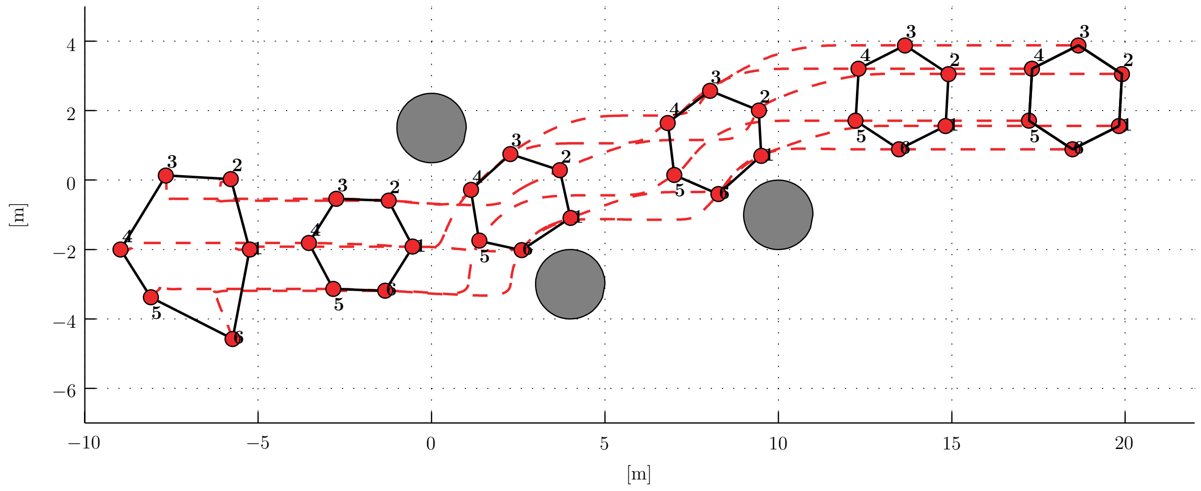
To validate the control strategy presented so far, several simulations and experiments. have been implemented. Differential–drive robots have been considered: to deal with the fact that these model represents a nonholonomic system, the feedback linearization technique presented in [89] has been applied. To make the formation move in a desired direction, a common offset has been added to the control law in Eq. (3.79), that describes the desired speed of the barycenter of the formation.

Matlab Simulations

Several simulations have been performed using Matlab/Simulink. Specifically, six robots have been simulated, that were supposed to move in an environment where three round obstacles were placed on their trajectory while achieving a formation with $1.5m$ radius. To emphasize the different behaviors that come out by changing the value of α_{ij}, two different simulations are reported in Figs. 3.26a 3.26b: in the first one, where the inter–robot action is modulated by $\alpha_{ij} = 1$, robots exhibit a flexible behavior when obstacles are encountered. In the second one, where $\alpha_{ij} = 10$, the formation is too rigid to be able to split in order to overcome the obstacles, thus the formation preserves its shape while *sliding* over them.



(a) Formation moving with $\alpha_{ij} = 1$, $\alpha_{pv} = 1$.



(b) Formation moving with $\alpha_{ij} = 10$, $\alpha_{pv} = 1$.

Figure 3.26: Simulation with six robots engaged in a formation task while moving in an unknown environment with three obstacles (in gray) with different values of α_{ij}.

Communication delay The behavior of the system in the presence of communication delay has been tested by means of simulations.

More specifically, three robots have been simulated, moving in an obstacle–free environment. Initially, the control law introduced in [69] was implemented, adding an artificial potential field for collision avoidance, as described in [70]. Simulations show that,

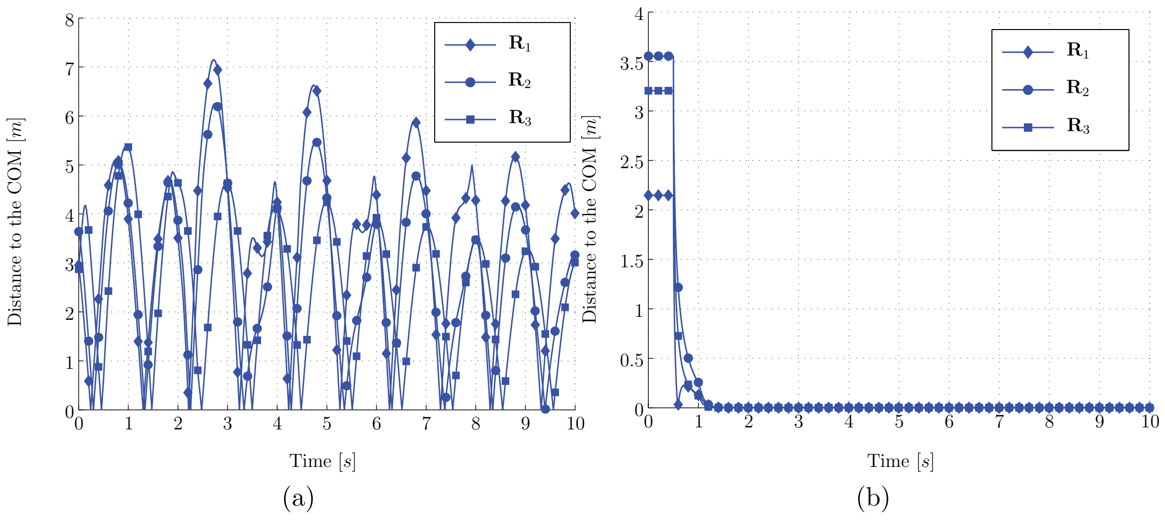


Figure 3.27: Error between the actual and the desired distance between each robot and the centroid of the formation with a communication delay of 0.5 s, by exploiting the formation control algorithm introduced in [69] (Fig. 3.27a) and exploiting the algorithm introduced in this section (Fig. 3.27b).

as proved in [66], artificial potential fields are not robust with respect to communication delays. In fact, Fig. 3.27a shows the error between the actual and the desired distance between each robot and the centroid of the formation, with a communication delay of $0.5s$. As expected, the system does not converge to the desired formation.

Conversely, simulations show that the control law introduced in this section is robust with respect to communication delays, as expected for consensus based control laws [70]. Fig. 3.27b shows the error between the actual and the desired distance between each robot and the centroid of the formation, with a communication delay of $0.5s$. As expected, the error quickly converges to zero.

Experiments

Several experiments have been performed exploiting the MORE–pucks experimental setup described in Chapter 2.

Two different experimental setups have been used to test the algorithm. In the first setup, four robots starting from random positions converge to the desired square formation while avoiding collisions and, after $20s$ they start moving along the x–axis with a constant speed.

In the second setup, an obstacle is placed in the middle of the arena in a position unknown by the system, thus robots have to overcome it while moving in formation along the x–axis.

For each setup, the experiments were run 10 times, and data were collected. The average mean square error (MSE) between the actual and desired distances for each pair of robots are represented in Fig. 3.28a and 3.28b respectively. In particular, in the second setup, it can be seen that at time $\approx 35s$ the formation detects the obstacle, thus the variance increases.

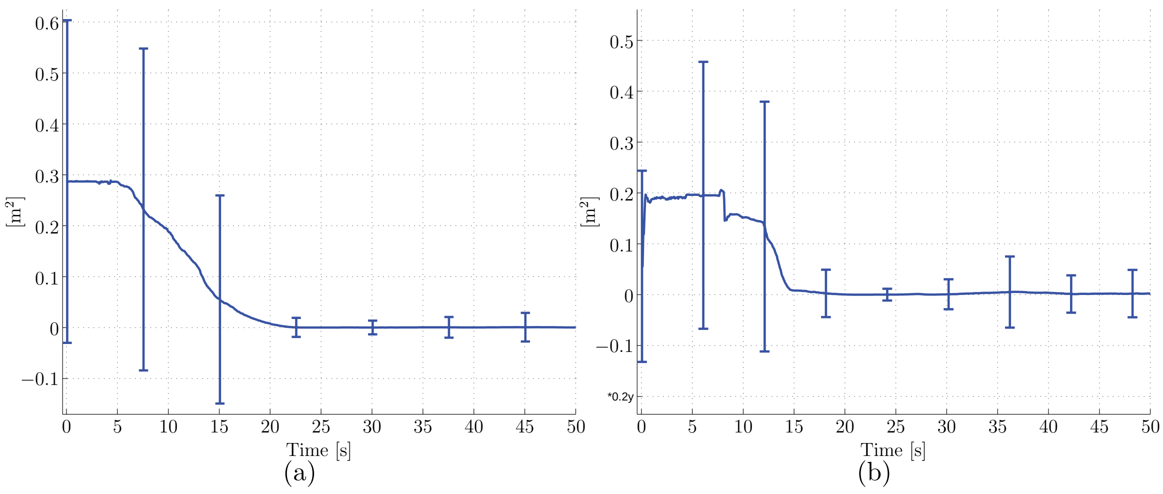


Figure 3.28: Average and standard deviation of the mean square error between the actual and desired distances for each pair of robots, depending on time, in a real arena *without* obstacles (Fig. 3.28a) and *with* obstacles (Fig. 3.28b).

3.5 Discussion

In this chapter, three different decentralized control strategy for the coordination of multi–robot systems have been presented.

Artificial potential fields and consensus based controllers have been be used to design the control strategies. Specifically, Section 3.2 describes a formation control strategy, first introduced in [21–24], that exploits artificial potential fields to design regular polygon formations. Formal proof of the asymptotic stability of the system, based on the definition of a proper Lyapunov function. The absence of local minima is ensured as well. The control strategy is then extended to arbitrarily shaped formation, exploiting a bijective coordinate transformation to deform the polygonal formation.

Simulations and experiments show the effectiveness of the proposed control strategy. The desired formations are always created, regardless the original positions of the robots, and the shape of the formation itself. Moreover, as expected collisions among the robots are always avoided.

The main advantages of this formation control strategy are the following:

- **Asymptotic stability** is guaranteed.
- Unlike other artificial potential based control strategies, **local minima** do not appear.
- It provides a high degree of **flexibility**, since formations can be obtained with completely arbitrary shapes.

One of the main drawbacks of the formation control strategy presented in this section is the sub–optimality of the paths traveled by the robots. As can be inferred by the results of the simulations shown in Fig. 3.8, the shape of the paths is mainly due to the coordinate transformation.

In order to improve the performances of this control strategy, different coordinate transformation may be developed, in order to generate more optimal paths.

In Section 3.3, inspired by the previously described artificial potential fields, a coordinated path tracking algorithm is described. First introduced in [25, 26], this control strategy makes a group of mobile robots track a path given by an arbitrarily shaped desired curve. This control strategy is a completely decentralized algorithm, since there is no need for any centralized controller or global synchronization.

By means of this control strategy, a group of mobile robots, starting from random initial positions, is able to reach the desired curve and then move along it. Once reached the curve, the robots never leave it. Furthermore, collision avoidance and desired spacing between neighboring robots is ensured.

With respect to previous works on tracking, the main advantage of this control strategy is the fact that it combines tracking of paths with the coordination of multiple mobile robots. Furthermore, this result is obtained without any global controller, and without the need of knowing information about the whole group of robots. This ensures the correct behavior even in the presence of sudden addition or subtraction of one or more robots.

Section 3.4 addresses the formation control problem by means of a consensus based control strategy. Specifically, weighted graphs are exploited to drive a group of robots to a predefined configuration while avoiding mutual collisions, exploiting a consensus based algorithm, first introduced in [27, 28]. An appropriate edge–weight function has been defined that provably guarantees the convergence to the desired formation, as well as the avoidance of collisions among the robots. The framework is extended for accomplishing the avoidance of collisions among robots and obstacles as well.

Analytical proof of the convergence of the system has been provided, based on tools from the graph theory. Furthermore, the approach has been validated by means of simulations and real experiments in noisy environment.

Chapter 4

Global connectivity maintenance

To accomplish cooperative tasks, robotic systems are often required to communicate with each other. Thus, maintaining connectivity of the interagent communication graph is a fundamental issue in the field of multi–robot systems. This chapter describes a completely decentralized control strategy for global connectivity maintenance of the interagent communication graph, for single integrator kinematic agents. A gradient–based control strategy is introduced that exploits decentralized estimation of the algebraic connectivity. The proposed control algorithm guarantees the global connectivity of the communication graph without requiring maintenance of the local connectivity between robotic systems. The control strategy is demonstrated to be effective in the presence of external control actions as well. The control strategy is extended also to consider groups of dynamic agents, described as Lagrangian systems.

4.1 Introduction

Connectivity maintenance is a crucial issue in the field of multi–robot systems. As mobile robots have limited communication capabilities, for the completion of a desired cooperative task, it is important to ensure information exchange can occur among the robotic systems. The communication architecture among the robots is often modeled as a graph (see e.g. [19]), and is usually referred to as the communication graph. Thus, the problem of ensuring that information exchange can occur is mathematically translated into guaranteeing that the communication graph is connected.

Decentralized control of groups of mobile robots has several applications, such as surveillance, exploration of unknown environments, search–and–rescue, automatic warehouses. In this kind of problems, robots are supposed to coordinate their motion, in order to achieve the global objective. Generally speaking, one of the main challenges is due to the fact that robots move into cluttered environments: unknown terrains to be explored, industrial environments for goods transportation, damaged buildings in search–and–rescue applications. This implies that collisions with obstacles must be avoided, while robots are performing their task. Thus, obstacle avoidance can interfere with the primary task of the robots: problems can arise, for instance, if some robots are trapped. One of the main issues, in this case, is to ensure the connectivity maintenance: due to the limited communication capabilities, in fact, the trapped robots can lose the connectivity with the rest of the group, which implies that the global objective can not be, in general, achieved.

In the literature, several approaches to connectivity maintenance have been proposed. These approaches can be divided into two categories: approaches to maintain the local connectivity, and approaches to maintain the global connectivity.

Maintaining the local connectivity entails designing a controller that ensures that, if a communication link is active at time $t = 0$, it will be active $\forall t \geq 0$. Examples of decentralized algorithms for local connectivity maintenance can be found in [67, 90–95]. The main advantage of these control algorithms is that the maintenance of the connectivity is formally proven. Nevertheless, imposing the maintenance of each single communication link is often too restrictive. In fact, to ensure that information exchange among all the robots is possible, it is necessary to guarantee only the *global* connectivity of the communication graph. Loosely speaking, it is acceptable that a few links are broken, as long as the overall graph is still connected: if necessary, redundant links can be removed, and new ones can be introduced. For instance, in [96] a path planning strategy is introduced that allows temporary loss of connectivity, provided that the connectivity will then be again ensured in some predefined future time.

As shown in [97], a measure of the connectivity of a graph is the value of the second–smallest eigenvalue of the Laplacian matrix of the graph. On these lines, a connectivity maintenance control strategy was introduced in [98], where each agent built a local estimate of the communication graph, and computed the value of the second–smallest eigenvalue of the Laplacian matrix. A distributed market–based algorithm was then implemented, in order to cooperatively decide whether a link could be safely removed or not.

To avoid the direct analysis of the influence of each single link on the connectivity of the communication graph, gradient based strategies can be exploited. In [99] a gradient based control strategy was proposed to guarantee that the second–smallest eigenvalue of the Laplacian matrix is greater than zero. The main drawback of this control strategy is the fact that the eigenvalue was computed in a centralized way. Decentralized estimation procedures for the computation of the second–smallest eigenvalue of the Laplacian matrix were introduced in [100] and [101]. This estimate is then used in [100] to implement an optimization algorithm, that aims at increasing the value of the algebraic connectivity of the graph. Optimization algorithms have been also implemented to increase the algebraic connectivity of the graph without estimation of the value of the eigenvalue itself, as shown in [102–104].

Increasing the algebraic connectivity is pursued in [101] exploiting a gradient based control strategy, that is implementable because an estimate of the gradient of the second–smallest eigenvalue of the Laplacian matrix is computed as well. In cooperative control tasks, connectivity maintenance is usually necessary for completing a desired task via an external control. As will be discussed in Section 4.5, it can be demonstrated via simulations that, in the presence of certain (bounded) external control laws, the control strategy described in [101] may not guarantee the connectivity of the communication graph.

Motivated by the above discussion, this chapter describes a decentralized control strategy to guarantee maintenance of the global connectivity. Inspired by [101], this control strategy relies on a decentralized estimation procedure of the second–smallest eigenvalue of the Laplacian matrix. Specifically, the main contribution described in this chapter is the following:

1. An estimation procedure is introduced, that provides bounded estimation errors. The previous work in [101] demonstrates the convergence of the estimation system, without considering the presence of estimation errors. Conversely, in Proposition 4.1, the dynamics of the estimation system are explicitly considered, and the boundedness of the state of the estimation system is proven. This result leads to the demonstration of the boundedness of the estimation error, in Proposition 4.3 and Proposition 4.4.

2. The boundedness of the estimation errors will be shown to be a necessary element to guarantee the connectivity maintenance. Simulations will show that, in the presence of certain (bounded) external control laws, the control strategy described in [101] may not guarantee the connectivity of the communication graph. Conversely, Theorem 4.1 demonstrates that the control strategy presented in this chapter ensures the connectivity maintenance in any case.

3. Connectivity will be formally proven to be guaranteed even in the presence of any bounded external control action.

4. The control strategy will be extended to guarantee connectivity for groups of Lagrangian systems.

4.1.1 Outline

The outline of the Chapter is as follows. Section 4.2 introduces a distributed estimation strategy, to allow each agent to compute its own estimate of λ_2 and of its gradient. This estimates are then exploited in Section 4.3 to develop a connectivity maintenance control strategy for single integrator kinematic agents, that takes into account the presence of both estimation errors and external control laws. This control strategy is then extended in Section 4.4 to implement connectivity maintenance for group of Lagrangian dynamical systems. Simulations and experiments are described in Section 4.5.

4.2 Estimation of the algebraic connectivity of the graph

Consider a group on N cooperating agents. Let $\mathcal{G}$ be the communication graph, that is the graph that model the communication architecture among them. Let L be the Laplacian matrix of $\mathcal{G}$. As shown in [97], a measure of the connectivity of a graph is the value of the second–smallest eigenvalue of the Laplacian matrix of the graph, that will be hereafter referred to as λ_2.

This section will describe a decentralized estimation procedure, that allows each agent to compute its own estimate of λ_2.

For the sake of clarity, a brief overview of the estimation procedure introduced in [101] will now be provided. Specifically, the estimation of λ_2 is computed by exploiting the estimation of the corresponding eigenvector v_2. The power iteration procedure described in [105] is utilized to design the following update law:

$$\dot{\tilde{v}}_2 = -k_1 \text{Ave}\left(\{\tilde{v}_2^i\}\right)\mathbf{1} - k_2 L\tilde{v}_2 - k_3\left(\text{Ave}\left(\left\{(\tilde{v}_2^i)^2\right\}\right) - 1\right)\tilde{v}_2 \tag{4.1}$$

where $k_1, k_2, k_3 > 0$ are the control gains, and $\text{Ave}(\cdot)$ is the averaging operation. Furthermore, $\tilde{v}_2^i$ is defined as the i–th agent's estimate of v_2^i, the i–th component of the eigenvector v_2, and $\tilde{v}_2 = \left[\tilde{v}_2^1, \ldots, \tilde{v}_2^N\right]^T$. Additional details can be found in [101].

To implement the update law in Eq. (4.1) in a decentralized way, the averaging operation is implemented by means of the PI average consensus estimator described in [106]:

$$\begin{aligned} \dot{z}^i &= \gamma \left(\alpha^i - z^i\right) - K_p \sum_{j \in \mathcal{N}_i} \left(z^i - z^j\right) + K_i \sum_{j \in \mathcal{N}_i} \left(w^i - w^j\right) \\ \dot{w}^i &= -K_i \sum_{j \in \mathcal{N}_i} \left(z^i - z^j\right) \end{aligned} \tag{4.2}$$

Further details can be found in [106].

Since there are two averaging operations in the update law in Eq. (4.1), two PI consensus estimators must be run:

- the first one, with input $\alpha^{i,1} = \tilde{v}_2^i$, provides z_1^i as the i–th agent's estimate of $\text{Ave}\left(\{\tilde{v}_2^i\}\right)$;
- the second one, with input $\alpha^{i,2} = \left(\tilde{v}_2^i\right)^2$, provides z_2^i as the i–th agent's estimate of $\text{Ave}\left(\left\{\left(\tilde{v}_2^i\right)^2\right\}\right)$.

Thus, each agent can run the decentralized version of the update law in Eq. (4.1):

$$\dot{\tilde{v}}_2^i = -k_1 z_1^i - k_2 \sum_{j \in \mathcal{N}_i} a_{ij} \left(\tilde{v}_2^i - \tilde{v}_2^j\right) - k_3 \left(z_2^i - 1\right) \tilde{v}_2^i \tag{4.3}$$

As demonstrated in [101], the i–th agent can compute its estimate of λ_2, namely λ_2^i, as follows:

$$\lambda_2^i = \frac{k_3}{k_2} \left(1 - z_2^i\right) \tag{4.4}$$

The convergence of the estimation system to the real value of λ_2 was formally proven in [101] assuming that each agent could compute the current values of $\text{Ave}\left(\{\tilde{v}_2^i\}\right)$ and $\text{Ave}\left(\left\{\left(\tilde{v}_2^i\right)^2\right\}\right)$. In order to explicitly take into account the estimation errors provided by the PI average consensus estimators, a modified estimator will now be introduced. This estimator will be shown to provide bounded errors in the estimate of λ_2, even in the presence of estimation errors from the PI average consensus estimators.

In order to accomplish this goal, the following decentralized update law is introduced:

$$\dot{\tilde{v}}_2^i = \; -k_1 z_1^i - k_2 \sum_{j \in \mathcal{N}_i} a_{ij} \left(\tilde{v}_2^i - \tilde{v}_2^j\right) - k_3 \left(z_2^i - 1\right) \tilde{v}_2^i - k_4 \left|\tilde{v}_2^i\right| \tilde{v}_2^i \tag{4.5}$$

for some constant $k_4 > 0$. The update law in Eq. (4.5) is obtained by slightly modifying Eq. (4.3), by introducing an additional term. As will be shown in the simulations provided in Section 4.5, the introduction of this additional term deteriorates the estimation, with respect to the original update law introduced in [101]. However, the presence of this

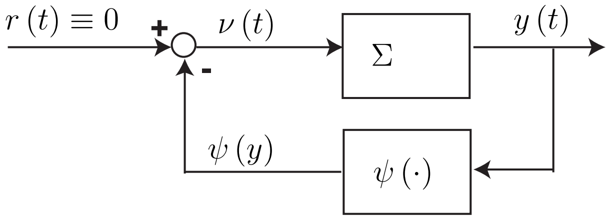


Figure 4.1: Representation of the estimation system

additional term will be subsequently shown to be necessary to guarantee the boundedness of the estimation error of λ_2. This is a fundamental element to guarantee the connectivity maintenance, which is the goal of the control strategy presented in this chapter.

In order to prove the boundedness of the estimation error of λ_2, the boundedness of the state of the estimation system will now be proven.

Let $\chi = \left[\tilde{v}_2^T {z_1}^T {w_1}^T {z_2}^T {w_2}^T\right]^T$ be the state vector of the estimation system, that embeds the decentralized power iteration update law described in Eq. (4.5), whose state vector is represented by $\tilde{v}_2$, and the two PI average consensus estimators described in Eq. (4.2), whose state is represented by $\left[{z_1}^T {w_1}^T\right]^T$ and $\left[{z_2}^T {w_2}^T\right]^T$ respectively. Thus, the estimation dynamics can be represented as the feedback interconnection of a linear dynamic system Σ with a memoryless nonlinearity $\psi(\cdot)$, as described in Fig. 4.1. More specifically, the linear dynamic system Σ is defined as follows:

$$\Sigma : \begin{cases} \dot{\chi}(t) & = \Lambda\chi(t) + B\nu(t) \\ y(t) & = C\chi(t) \end{cases} \tag{4.6}$$

where

$$\Lambda = \begin{bmatrix} -k_2 L & -k_1 I_N & 0_N & 0_N & 0_N \\ \gamma I_N & -\gamma I_N - K_p L_* & K_i L_* & 0_N & 0_N \\ 0_N & -KiL_* & 0_N & 0_N & 0_N \\ 0_N & 0_N & 0_N & -\gamma I_N - K_p L_* & K_i L_* \\ 0_N & 0_N & 0_N & -KiL_* & 0_N \end{bmatrix}$$
$$B = \begin{bmatrix} I_N & 0_N \\ 0_N & 0_N \\ 0_N & 0_N \\ 0_N & I_N \\ 0_N & 0_N \end{bmatrix} \quad C = \begin{bmatrix} I_N & 0_N & 0_N & 0_N & 0_N \\ 0_N & 0_N & 0_N & I_N & 0_N \end{bmatrix} \tag{4.7}$$

where I_N is the identity matrix of size N, and 0_N is the zero matrix of size N. The matrices L and L_* are the weighted and unweighted Laplacian matrix, respectively. The parameters K_i and K_p have been introduced in Eq. (4.2). As shown in Fig. 4.1, the input ν is defined as $\nu(t) = -\psi(y(t))$, where $\psi(\cdot)$ will be defined later on.

From the definition of the matrix C in Eq. (4.7), it follows that

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \tilde{v}_2 \\ z_2 \end{bmatrix} \tag{4.8}$$

Given a vector $\xi \in \mathbb{R}^N$, let $\operatorname{diag}(\xi)$ be the diagonal matrix whose diagonal elements are the entries of the vector ξ. Let $\xi_s \in \mathbb{R}^N$ be a vector whose entries are the square of the corresponding entries of ξ, namely $\xi_s = \left\{(\xi_i)^2\right\}$. It is easy to prove that

$$\xi_s = \operatorname{diag}(\xi)\,\xi = \xi^T \operatorname{diag}(\xi) \tag{4.9}$$

Hence, the memoryless nonlinearity $\psi(\cdot)$ is then defined as follows:

$$\psi(y) = \begin{bmatrix} k_3\left(\operatorname{diag}(y_2) - I_N\right) y_1 + k_4 \operatorname{diag}\left(\{|y_1^i|\}\right) y_1 \\ -\gamma \operatorname{diag}(y_1)\, y_1 \end{bmatrix} \tag{4.10}$$

The following proposition proves the boundedness of the estimation system's state.

Proposition 4.1. *Consider the dynamics of the estimation system, described by Eqs.* (4.6), (4.10). *Given any initial condition* $\chi(0)$, *the norm of the state vector of the estimation system,* $\|\chi(t)\|$, *is bounded.*

Proof. First, the existence of a value $S > 0$ will be demonstrated such that, if $\|\tilde{v}_2\| \geq S$, then $\|\chi\|$ does not increase over time.

Let

$$W(\chi) = \frac{1}{2}\chi^T \chi \geq 0 \tag{4.11}$$

where, for the sake of simplicity, the dependence on time has been dropped. The time derivative of this function may be computed as follows:

$$\dot{W}(\chi) = \chi^T \dot{\chi} = \chi^T \left[\Lambda \chi + B\nu\right] \tag{4.12}$$

The matrix Λ can be decomposed as the sum of the matrices Λ_{diag} and Λ_{skew}, defined as follows:

$$\begin{aligned} \Lambda_{diag} &= \\ &\begin{bmatrix} -k_2 L & 0_N & 0_N & 0_N & 0_N \\ 0_N & -\gamma I_N - K_p L_* & 0_N & 0_N & 0_N \\ 0_N & 0_N & 0_N & 0_N & 0_N \\ 0_N & 0_N & 0_N & -\gamma I_N - K_p L_* & 0_N \\ 0_N & 0_N & 0_N & 0_N & 0_N \end{bmatrix} \\ \Lambda_{skew} &= \begin{bmatrix} 0_N & k_1 I_N & 0_N & 0_N & 0_N \\ -\gamma I_N & 0_N & K_i L_* & 0_N & 0_N \\ 0_N & -K_i L_* & 0_N & 0_N & 0_N \\ 0_N & 0_N & 0_N & 0_N & K_i L_* \\ 0_N & 0_N & 0_N & -K_i L_* & 0_N \end{bmatrix} \end{aligned} \tag{4.13}$$

Since L and L_*, being Laplacian matrices, are symmetric and positive semidefinite, Λ_{diag} is negative semidefinite. Imposing $k_1 = \gamma$, Λ_{skew} is skew–symmetric. Thus, Eq. (4.12) may be rewritten as follows:

$$\dot{W}\left(\chi\right) = \chi^T \Lambda \chi + \chi^T B \nu = \chi^T \Lambda_{diag} \chi + \chi^T B \nu \tag{4.14}$$

Substituting Eqs. (4.7), (4.10) into Eq. (4.14):

$$\begin{aligned} &\dot{W}\left(\chi\right) = \\ &= \chi^T \Lambda_{diag} \chi - k_3 \tilde{v}_2^T \left[\mathrm{diag}\left(z_2\right) - I_N\right] \tilde{v}_2 + {z_2}^T \left[\gamma \mathrm{diag}\left(\tilde{v}_2\right)\right] \tilde{v}_2 - k_4 \tilde{v}_2^T \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) \tilde{v}_2 \\ &= \chi^T \Lambda_{diag} \chi + \left(-k_3 \tilde{v}_2^T \mathrm{diag}\left(z_2\right) + \gamma z_2^T \mathrm{diag}\left(\tilde{v}_2\right)\right) \tilde{v}_2 + k_3 \tilde{v}_2^T I_N \tilde{v}_2 - k_4 \tilde{v}_2^T \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) \tilde{v}_2 \end{aligned} \tag{4.15}$$

Given two vectors $\xi, \phi \in \mathbb{R}^N$, the vector

$$\zeta = \xi^T \mathrm{diag}\left(\phi\right) = \phi^T \mathrm{diag}\left(\xi\right) \tag{4.16}$$

is the vector whose components are the products of the corresponding components of ξ and ϕ, namely $\zeta = \{\xi_i \phi_i\}$.

Hence, Eq. (4.15) can then be rewritten as follows:

$$\begin{aligned} &\dot{W}\left(\chi\right) = \\ &= \chi^T \Lambda_{diag} \chi + \left(-k_3 \tilde{v}_2^T \mathrm{diag}\left(z_2\right) + \gamma \tilde{v}_2^T \mathrm{diag}\left(z_2\right)\right) \tilde{v}_2 + k_3 \tilde{v}_2^T I_N \tilde{v}_2 - k_4 \tilde{v}_2^T \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) \tilde{v}_2 \end{aligned} \tag{4.17}$$

Imposing $k_3 = \gamma$, Eq. (4.17) can be rewritten as follows:

$$\dot{W}\left(\chi\right) = \chi^T \Lambda_{diag} \chi + \gamma \tilde{v}_2^T I_N \tilde{v}_2 - k_4 \tilde{v}_2^T \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) \tilde{v}_2 \tag{4.18}$$

From the definition of Λ_{diag} in Eq. (4.13), Eq. (4.18) can be rewritten as follows:

$$\begin{aligned} \dot{W}\left(\chi\right) = \ & -\tilde{v}_2^T k_2 L \tilde{v}_2 - {z_1}^T \gamma I_N z_1 - {z_1}^T K_p L_* z_1 - {z_2}^T \gamma I_N z_2 - {z_2}^T K_p L_* z_2 + \gamma \tilde{v}_2^T I_N \tilde{v}_2 \\ & - k_4 \tilde{v}_2^T \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) \tilde{v}_2 \end{aligned} \tag{4.19}$$

From Eq. (4.19), the following inequality may be derived:

$$\dot{W}\left(\chi\right) \ \leq -\tilde{v}_2^T \left(k_4 \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) - \gamma I_N\right) \tilde{v}_2 \tag{4.20}$$

Let

$$\Omega_i\left(\chi\right) = -k_4 \left|\tilde{v}_2^i\right|^3 + \gamma \left|\tilde{v}_2^i\right|^2 \ \ \forall i = 1, \ldots, N \tag{4.21}$$

and let $\Omega\left(\chi\right) = \sum_{i=1}^{N} \Omega_i\left(\chi\right)$, namely:

$$\begin{aligned} \Omega\left(\chi\right) \ &= -k_4 \sum_{i=1}^{N} \left|\tilde{v}_2^i\right|^3 + \gamma \sum_{i=1}^{N} \left|\tilde{v}_2^i\right|^2 \\ &= -\tilde{v}_2^T \left(k_4 \mathrm{diag}\left(\left\{\left|\tilde{v}_2^i\right|\right\}\right) - \gamma I_N\right) \tilde{v}_2 \end{aligned} \tag{4.22}$$

Thus, from Eqs. (4.19), (4.22) it follows that $\dot{W}(\chi) \leq \Omega(\chi)$. The function $\Omega(\chi)$ has a strict maximum Ω^M when $|\tilde{v}_2^i| = \frac{2\gamma}{3k_4} < \frac{\gamma}{k_4}$ $\forall i = 1, \ldots, N$.

Namely, $\Omega^M = N \cdot \bar{\Omega}$, where:

$$\bar{\Omega} = \left[-k_4 \left(\frac{2\gamma}{3k_4} \right)^3 + \gamma \left(\frac{2\gamma}{3k_4} \right)^2 \right] \tag{4.23}$$

In order to compute an upper–bound on $|\tilde{v}_2^i|$ $\forall i = 1, \ldots, N$, the worst case may be considered. More specifically, each entry of the vector $\tilde{v}_2$ will be shown to be bounded. To do this, suppose that all the entries of the vector $\tilde{v}_2$ are bounded, such that $|\tilde{v}_2^i| < \frac{\gamma}{k_4}$, except the j–th one.

In this case, the following inequality holds:

$$\Omega(\chi) \leq (N-1)\bar{\Omega} + \Omega_j(\chi) = (N-1)\bar{\Omega} - k_4 \left|\tilde{v}_2^j\right|^3 + \gamma \left|\tilde{v}_2^j\right|^2 \tag{4.24}$$

Worst case means that letting more than one component of $\tilde{v}_2$ be greater than $\frac{\gamma}{k_4}$ would decrease the value on the right–hand side of Eq. (4.24). The existence of a value α will now be shown, such that, if $\left|\tilde{v}_2^j\right| > \alpha$, then $\Omega_j(\chi) > (N-1)\bar{\Omega}$, and then $\Omega(\chi) < 0$. More specifically, $\Omega(\chi) < 0$ if $\left|\tilde{v}_2^j\right| > \alpha > 0$ such that:

$$\alpha^3 > \frac{\gamma}{k_4}\alpha^2 + \frac{(N-1)\bar{\Omega}}{k_4} \tag{4.25}$$

Hence, $\dot{W}(\chi) \leq \Omega(\chi) < 0$ if $|\tilde{v}_2^i| > \alpha$ for at least one value of $i = 1, \ldots, N$. Thus, $\exists S > 0$ such that, if $\|\tilde{v}_2\| \geq S$, then $W(\chi)$ does not increase over time, which implies that $\|\chi\|$ does not increase over time as well.

The boundedness of $\|\chi\|$ will now be demonstrated even when $\|\tilde{v}_2\| < S$.
Let $\zeta_1 = \left[{z_1}^T {w_1}^T\right]^T$ and $\zeta_2 = \left[{z_2}^T {w_2}^T\right]^T$ be the state vectors of the PI average consensus estimators. Thus, $\chi = \left[\tilde{v}_2^T \zeta_1^T \zeta_2^T\right]^T$. As proved in [106], the PI average consensus estimators are input–to–state stable (ISS) systems. The boundedness of $\|\tilde{v}_2\|$ implies the boundedness of the inputs of the PI average consensus estimators. In fact, as stated in Section 4.3, these inputs are v_2^i and $(v_2^i)^2$, respectively. Thus, both $\|\zeta_1\|$ and $\|\zeta_2\|$ are bounded, given $\|\tilde{v}_2\| < S$. $\square$

From Proposition 4.1, it follows that $\exists M > 0$ such that $\|\chi(t)\| \leq M$, $\forall t \geq 0$.
Since $\|\tilde{v}_2(t)\| \leq \|\chi(t)\|$ and $\|z_2(t)\| \leq \|\chi(t)\|$, it is possible to conclude that $\|\tilde{v}_2(t)\| \leq M$ and $\|z_2(t)\| \leq M$, $\forall t \geq 0$.

4.3 Connectivity maintenance for single integrator agents

This section will describe an algorithm for global connectivity maintenance, for groups of single integrator kinematic agents.

For the sake of clarity, this algorithm will be first introduced in a centralized framework: suppose that each agent can compute the actual value of the algebraic connectivity of the communication graph. This assumption will be removed subsequently, exploiting the decentralized estimation procedure introduced in Section 4.2.

Hence, consider a group of N single–integrator agents, i.e.:

$$\dot{p}_i = u_i^c \tag{4.26}$$

where $p_i \in \mathbb{R}^m$ is the position of the i–th agent, and u_i^c is the control input. Let $p = \left[p_1^T \dots p_N^T\right]^T \in \mathbb{R}^{Nm}$ be the state vector of the multi–agent system. Furthermore, let R be the maximum communication range for each agent, i.e. the j–th agent is inside $\mathcal{N}_i$ if $\|p_i - p_j\| \leq R$.

Let L be the Laplacian matrix of the communication graph, then the connectivity is guaranteed if the second smallest eigenvalue of L (that, hereafter, will be referred to as λ_2) is strictly greater than zero. Hence, let $\epsilon > 0$ be the desired lower–bound for the value of λ_2. The control strategy will then be designed to ensure that the value λ_2 never goes below ϵ. To this end, an energy function will be used for generating the decentralized connectivity maintenance control strategies.

Definition 4.1. *An energy function*

$$V\left(\lambda_2\left(p\right) - \epsilon\right) : \mathbb{R}^{Nm} \mapsto \mathbb{R}$$

exhibits the following properties:

(P1) It is continuously differentiable.

(P2) it is non–negative.

(P3) it is non–increasing with respect to λ_2.

(P4) it approaches a constant value, as λ_2 increases.

The control design essentially drives the robots to perform a gradient descent of $V(\cdot)$, in order to ensure connectivity maintenance. Since λ_2 and its gradient are global quantities, the following centralized connectivity maintenance control law may be defined:

$$u_i^c = -\frac{\partial V(\lambda_2(p) - \epsilon)}{\partial p_i} = -\frac{\partial V(\lambda_2(p) - \epsilon)}{\partial \lambda_2}\frac{\partial \lambda_2}{\partial p_i} \tag{4.27}$$

Without loss of generality, the following energy function will be adopter, hereafter:

$$V(p) = \coth(\lambda_2 - \epsilon) \tag{4.28}$$

The energy function (Fig. 4.2) is non–increasing (with respect to λ_2) and non–negative, for any $\lambda_2 > \epsilon$.

According to the energy function defined in Eq. eq:totaltension, the control law is defined as follows:

$$u_i^c = -\frac{\partial V(p)}{\partial p_i} \tag{4.29}$$

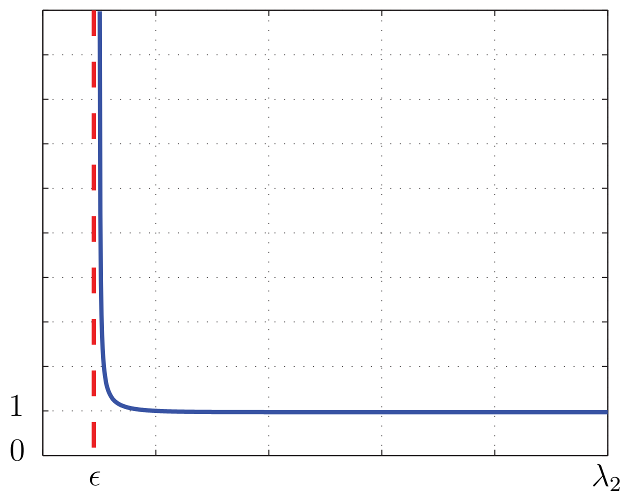


Figure 4.2: Energy function $V(p) = \coth(\lambda_2 - \epsilon)$

From Eq. (4.28):

$$\frac{\partial V(p)}{\partial \lambda_2} = \operatorname{csch}^2(\lambda_2 - \epsilon) \tag{4.30}$$

As shown in Fig. 4.3, the magnitude of this multiplicative coefficient increases suddenly as λ_2 decreases. Subsequently, this property will be shown to be fundamental to guarantee the connectivity maintenance in the presence of external control laws. It is important to note that, as will be subsequently shown, an appropriate choice of the lower–bound ϵ is crucial for guaranteeing connectivity maintenance when dealing with estimation errors, and external control laws as well.

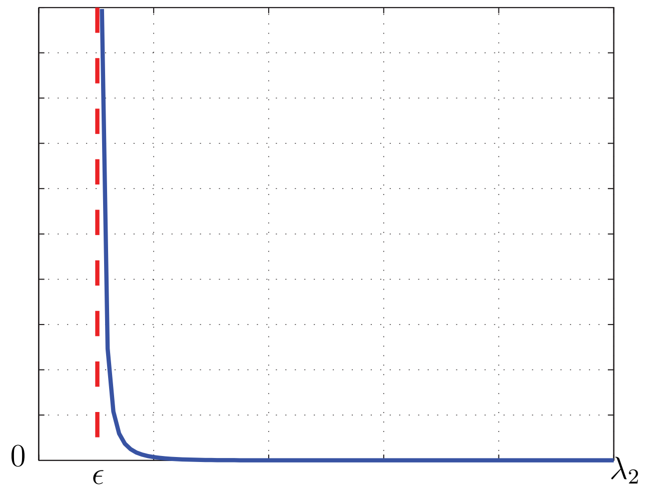


Figure 4.3: $\frac{\partial V(p)}{\partial \lambda_2} = \operatorname{csch}^2(\lambda_2 - \epsilon)$

From Eqs. (4.29), (4.28), the control law can be rewritten as follows:

$$u_i^c = \operatorname{csch}^2(\lambda_2 - \epsilon)\frac{\partial \lambda_2}{\partial p_i} \tag{4.31}$$

Inspired by [101], the edge–weights for the inter–agent communication graph may be defined as follows:

$$a_{ij} = \begin{cases} e^{-\left(\|p_i - p_j\|^2\right)/\left(2\sigma^2\right)} & \text{if } \|p_i - p_j\| \leq R \\ 0 & \textit{otherwise} \end{cases} \tag{4.32}$$

The scalar parameter σ is chosen to satisfy the threshold condition $e^{-\left(R^2\right)/\left(2\sigma^2\right)} = \Delta$, where Δ is a small predefined threshold. The presence of a non–zero threshold Δ ensures that that the edge–weight is different from zero if the distance between two agents is exactly equal to R. This definition of the edge–weights introduces a discontinuity in the control action, that can be avoided introducing a smooth bump function, as in [78].

Let v_2 be the eigenvector corresponding to the eigenvalue λ_2. As shown in [101], $\frac{\partial \lambda_2}{\partial p_i}$ can be computed as follows:

$$\frac{\partial \lambda_2}{\partial p_i} = v_2^T \frac{\partial L}{\partial p_i} v_2 = \sum_{j \in \mathcal{N}_i} \frac{\partial a_{ij}}{\partial p_i}\left(v_2^i - v_2^j\right)^2 \tag{4.33}$$

where v_2^i and v_2^j are the i–th and the j–th components of v_2, respectively. Then, from the definition of the edge–weights a_{ij} given in Eq. (4.32):

$$\frac{\partial \lambda_2}{\partial p_i} = \sum_{j \in \mathcal{N}_i} -a_{ij}\left(v_2^i - v_2^j\right)^2 \frac{p_i - p_j}{\sigma^2} \tag{4.34}$$

Thus, the control law in Eq. (4.31) can be rewritten as follows:

$$u_i^c = -\operatorname{csch}^2 \left(\lambda_2 - \epsilon\right) \sum_{j \in \mathcal{N}_i} a_{ij} \left(v_2^i - v_2^j\right)^2 \frac{p_i - p_j}{\sigma^2} \tag{4.35}$$

Let $\mathcal{D}_\epsilon$ be a set where the communication graph is connected, above a desired connectivity threshold ϵ, i.e.:

$$\mathcal{D}_\epsilon = \left\{p \in \mathbb{R}^{Nm} \text{ s. t. } \lambda_2 > \epsilon\right\} \tag{4.36}$$

Proposition 4.2. *Consider the system described by Eq.* (4.26)*. Given an initial configuration $p_0 \in \mathcal{D}_\epsilon$, for some $\epsilon > 0$, then, if the system is driven by the control law in Eq.* (4.35)*, the energy function defined in Eq.* (4.28) *does not increase.*

Proof. To prove the statement, the time derivative of the energy function may be computed as follows:

$$\dot{V}\left(p\right) = \nabla_p V\left(p\right)^T \cdot \dot{p} = \sum_{i=1}^{N} \frac{\partial V}{\partial p_i}^T \cdot \dot{p}_i \tag{4.37}$$

Thus, from Eq. (4.29):

$$\dot{V}\left(p\right) = -\sum_{i=1}^{N} \dot{p}_i^T \dot{p}_i \leq 0 \tag{4.38}$$

Thus, the energy function does not increase over time. □

Hence, Proposition 4.2 guarantees that $V\left(p\right)$ does not increase over time. Thus, if the initial condition is such that $\lambda_2 > \epsilon$, the value of λ_2 will never decrease. Hence, the connectivity of the graph is always maintained.

4.3.1 Decentralized implementation of the connectivity maintenance algorithm

Since the current value of λ_2 is not available to each agent, this section we will show how to exploit the decentralized estimation procedure introduced in Section 4.3.1 to implement the control strategy introduced in Eq. (4.29) in a decentralized manner.

Summary of the different estimates of λ_2 used by the connectivity maintenance algorithm

This section will summarize the different estimates of λ_2 that will be exploited for the connectivity maintenance control algorithm.

Exploiting the estimation procedure introduced in Section 4.2, each agent computes an estimate of a component of the eigenvector v_2, namely $\tilde{v}_2^i$. Let $\tilde{v}_2 = \left[\tilde{v}_2^1 \ldots \tilde{v}_2^N\right]^T$, and let $\tilde{\lambda}_2$ be the value that the second smallest eigenvalue of the Laplacian matrix would take if $\tilde{v}_2$ were the corresponding eigenvector.

Similarly to λ_2^i (see Eq. (4.4)), $\tilde{\lambda}_2$ can be computed as follows:

$$\tilde{\lambda}_2 = \frac{k_3}{k_2}\left[1 - \mathrm{Ave}\left(\left\{\left(\tilde{v}_2^i\right)^2\right\}\right)\right] \tag{4.39}$$

As shown in [101], $\dfrac{\partial\tilde{\lambda}_2}{\partial p_i}$ can be computed as follows

$$\frac{\partial\tilde{\lambda}_2}{\partial p_i} = \tilde{v}_2^T\frac{\partial L}{\partial p_i}\tilde{v}_2 = \sum_{j\in\mathcal{N}_i}\frac{\partial a_{ij}}{\partial p_i}\left(\tilde{v}_2^i - \tilde{v}_2^j\right)^2 \tag{4.40}$$

Then, from the definition of the edge–weights a_{ij} given in Eq. (4.32):

$$\frac{\partial\tilde{\lambda}_2}{\partial p_i} = \sum_{j\in\mathcal{N}_i} -a_{ij}\left(\tilde{v}_2^i - \tilde{v}_2^j\right)^2\frac{p_i - p_j}{\sigma^2} \tag{4.41}$$

The actual value of $\tilde{\lambda}_2$ can not be computed by each agent. In fact, the real value of $\mathrm{Ave}\left(\{(\tilde{v}_2^i)\}\right)$ is not available to any agent. Nevertheless, an estimate of this average, namely z_2^i, is available to each agent. According to Eq. (4.4), each agent can compute λ_2^i, that is indeed different from both λ_2 and $\tilde{\lambda}_2$. However, as demonstrated in the Proposition 4.3 and Proposition 4.4, λ_2^i is a good estimate of both λ_2 and $\tilde{\lambda}_2$, since $\exists\Xi,\Xi' > 0$ such that

$$\begin{aligned} |\lambda_2 - \lambda_2^i| &\le \Xi \quad \forall i = 1,\dots,N \\ \left|\tilde{\lambda}_2 - \lambda_2^i\right| &\le \Xi' \quad \forall i = 1,\dots,N \end{aligned} \tag{4.42}$$

From Eq. (4.42), It is possible to conclude that

$$\left|\lambda_2 - \tilde{\lambda}_2\right| \le \Xi + \Xi' \tag{4.43}$$

We remark that, even though the actual value of $\tilde{\lambda}_2$ is not available to each agent, the partial derivatives of $\tilde{\lambda}_2$ can be computed by each agent. In fact, Eq. (4.41) can be implemented in a decentralized manner.

The following proposition proves the boundedness of the estimation error of λ_2.

Proposition 4.3. *Consider the estimation system described by Eqs.* (4.4), (4.5). *Then, the error on the estimation of* λ_2 *is bounded.*

Proof. Let $\hat{\lambda}_2 = \left[\lambda_2^1,\dots,\lambda_2^N\right]^T \in \mathbb{R}^N$ be the vector containing the estimates of λ_2 performed by each agent.

Since each agent computes its estimate of λ_2, namely λ_2^i, according to Eq. (4.4), the vector $\hat{\lambda}_2$ is defined as follows:

$$\hat{\lambda}_2 = \frac{k_3}{k_2} (\mathbf{1} - z_2) \tag{4.44}$$

Since, from Proposition 4.1, $\|z_2\|$ is bounded, then $\left\|\hat{\lambda}_2\right\|$ is bounded as well. Once defined the number of agents in the graph, the real value of λ_2 is bounded, namely $\lambda_2 \in \left[0, \lambda_2^M\right]$. More specifically:

- $\lambda_2 = 0$ if the graph is disconnected;
- $\lambda_2 = \lambda_2^M$ if the graph is complete (i.e. an edge exist between each couple of agents), and the distance between each couple of agents is such that the edge–weights a_{ij} assume their maximum value. Namely, the distance between each couple of agents is zero, and $a_{ij} = 1$ $\forall i = 1, \ldots, N$. Then, for any value of the number of agents N, λ_2^M is well defined.

Let $\delta \in \mathbb{R}^N$ be the estimation error vector, i.e. $\delta = \hat{\lambda}_2 - \lambda_2\mathbf{1}$.

Since both $\left\|\hat{\lambda}_2\right\|$ and $\|\lambda_2\mathbf{1}\| = \lambda_2$ are bounded, we can conclude that $\exists \Xi > 0$ such that $\|\delta\| \leq \Xi$. Hence, $|\lambda_2^i - \lambda| \leq \Xi$, $\forall i = 1, \ldots, N$. □

The following proposition proves the boundedness of the estimation error $\left|\lambda_2^i - \tilde{\lambda}_2\right|$, $\forall i = 1, \ldots, N$.

Proposition 4.4. *Consider the estimation system described by Eqs.* (4.4)*,* (4.5)*,* (4.39)*. Then, the error on the estimation* $\left|\lambda_2^i - \tilde{\lambda}_2\right|$ *is bounded,* $\forall i = 1, \ldots, N$.

Proof. Let $\hat{\lambda}_2 = \left[\lambda_2^1, \ldots, \lambda_2^N\right]^T \in \mathbb{R}^N$ be the vector containing the estimates of λ_2 performed by each agent.

Since each agent computes its estimate of λ_2, namely λ_2^i, according to Eq. (4.4), the vector $\hat{\lambda}_2$ is defined as follows:

$$\hat{\lambda}_2 = \frac{k_3}{k_2} (\mathbf{1} - z_2) \tag{4.45}$$

Since, from Proposition 4.1, $\|z_2\|$ is bounded, then $\left\|\hat{\lambda}_2\right\|$ is bounded as well.

As shown in the proof of Proposition 4.3, $\left\|\hat{\lambda}_2\right\|$ is bounded. Furthermore Proposition 4.1 proves that also $\tilde{v}_2$ is bounded. Hence, it is possible to conclude that $\mathrm{Ave}\left(\{(\tilde{v}_2^i)\}\right)$, i.e. the average of the components of the vector $\tilde{v}_2$, is bounded as well.

Let $\tilde{\delta} \in \mathbb{R}^N$ be the estimation error vector, i.e. $\tilde{\delta} = \hat{\lambda}_2 - \tilde{\lambda}_2\mathbf{1}$.

Since both $\left\|\hat{\lambda}_2\right\|$ and $\left\|\tilde{\lambda}_2\mathbf{1}\right\| = \tilde{\lambda}_2$ are bounded, it is possible to conclude that $\exists \Xi' > 0$ such that $\left\|\tilde{\delta}\right\| \leq \Xi'$. Hence, $\left|\lambda_2^i - \tilde{\lambda}\right| \leq \Xi'$, $\forall i = 1, \ldots, N$. □

Decentralized connectivity maintenance algorithm

Consider the control law introduced in Eq. (4.29). Since the real values of λ_2 and $\frac{\partial \lambda_2}{\partial p_i}$ are not available, the agents will actually implement the following control law:

$$u_i^c = \operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} \tag{4.46}$$

The parameter $\tilde{\epsilon}$ is defined as follows

$$\tilde{\epsilon} = \epsilon + \Xi + \Xi' \tag{4.47}$$

where ϵ is the desired lower–bound for λ_2, and Ξ, Ξ' are defined according to Eq. (4.42).

The following energy function may now be introduced:

$$\tilde{V}\left(p\right) = \coth\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \tag{4.48}$$

The following theorem shows that, assuming the initial value of λ_2 is sufficiently large (i.e. the graph is connected above a certain threshold), then the control law introduced in this chapter ensures the connectivity maintenance.

Theorem 4.1. *Consider the dynamical system described by Eqs.* (4.26), (4.46). *Then,* $\exists \epsilon, \tilde{\epsilon}$ *defined according to Eq.* (4.47) *such that, if the initial value of* $\lambda_2 > \tilde{\epsilon} + \Xi + \Xi'$, *then the value of* λ_2 *never goes below* ϵ.

Proof. To prove the statement, the time derivative of the energy function introduced in Eq. (4.48) may be computed.

From Eq. (4.48) it follows that:

$$\frac{\partial \tilde{V}}{\partial p_i} = \frac{\partial \tilde{V}}{\partial \tilde{\lambda}_2} \frac{\partial \tilde{\lambda}_2}{\partial p_i} = -\operatorname{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} \tag{4.49}$$

From Eqs. (4.26), (4.46), (4.49), the time derivative of $\tilde{V}\left(p\right)$ can be computed as follows:

$$\begin{aligned} \dot{\tilde{V}}\left(p\right) &= \nabla_p \tilde{V}\left(p\right)^T \dot{p} = \sum_{i=1}^{N} \frac{\partial \tilde{V}}{\partial p_i}^T \dot{p}_i = \\ &= \sum_{i=1}^{N} \left[-\operatorname{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \frac{\partial \tilde{\lambda}_2}{\partial p_i}\right]^T \left[\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \frac{\partial \tilde{\lambda}_2}{\partial p_i}\right] = \\ &= -\sum_{i=1}^{N} \operatorname{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \leq 0 \end{aligned} \tag{4.50}$$

Thus, the energy function does not increase over time. According to Eq. (4.43), the fact that the initial value of λ_2 is greater than $\tilde{\epsilon}+\Xi+\Xi'$ ensures that the initial value of $\tilde{\lambda}_2$ is greater than $\tilde{\epsilon}$. Hence, it is possible to conclude that the value of $\tilde{\lambda}_2$ does not decrease over time, which ensures $\tilde{\lambda}_2 \geq \tilde{\epsilon}$.

Hence, according to Eq. (4.43), it is possible to conclude that λ_2 is guaranteed to remain lower–bounded as the system evolves, specifically

$$\lambda_2 \geq \epsilon = \tilde{\epsilon} - \Xi - \Xi' \tag{4.51}$$

□

4.3.2 Connectivity maintenance in the presence of an external controller

This section extends the control law described in the previous section, considering the following control law:

$$\dot{p}_i = u_i^c + u_i^d \tag{4.52}$$

where u_i^c is the control term introduced in Eq. (4.35), while u_i^d is a control term used to obtain some desired behavior. Namely, the control term u_i^d is an unknown bounded function, i.e. $\left\| u_i^d \right\| \leq u_M$.

Proposition 4.5. *Consider the dynamical system described by Eq.* (4.52)*. Let* Ξ, Ξ' *be defined according to Eq.* (4.42)*.* $\exists \epsilon, \tilde{\epsilon}$ *such that, if the initial value of* $\lambda_2 > \tilde{\epsilon}+\Xi+\Xi'$*, then the control law in Eq.* (4.35) *ensures that the value of* λ_2 *never goes below* ϵ*.*

Proof. To prove the statement, consider the following energy function:

$$V(p) = \coth\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \tag{4.53}$$

The time derivative of this energy function may be computed as follows:

$$\frac{\partial V}{\partial p_i} = \frac{\partial V}{\partial \tilde{\lambda}_2}\frac{\partial \tilde{\lambda}_2}{\partial p_i} = -\text{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right)\frac{\partial \tilde{\lambda}_2}{\partial p_i} \tag{4.54}$$

From Eqs. (4.35), (4.52), (4.54), the time derivative of $V(p)$ can be computed as follows:

$$\dot{V}(p) = \nabla_p V(p)^T \dot{p} = \sum_{i=1}^{N} \frac{\partial V}{\partial p_i}^T \dot{p}_i = \sum_{i=1}^{N} \left[-\text{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right)\frac{\partial \tilde{\lambda}_2}{\partial p_i}\right]^T \left[\text{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right)\frac{\partial \tilde{\lambda}_2}{\partial p_i} + u_i^d\right] \tag{4.55}$$

Since u_i^d is bounded:

$$\dot{V}(p) \leq \operatorname{csch}^2\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \sum_{i=1}^{N} \left[-\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 + + \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| u_M \right] \tag{4.56}$$

From Eq. (4.56) it follows that $\dot{V}(p) \leq 0$ if

$$\sum_{i=1}^{N} \left[-\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 + \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| u_M \right] \leq 0 \tag{4.57}$$

The condition in Eq. (4.57) is verified if

$$\sum_{i=1}^{N} \left[\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \right] \geq u_M \sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| \tag{4.58}$$

According to Eq. (4.42), the fact that the initial value of λ_2 is greater than $\tilde{\epsilon} + \Xi + \Xi'$ ensures that the initial value of λ_2^i is greater than $\tilde{\epsilon}$, $\forall i = 1, \ldots, N$. Then, $\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right)$ is monotonically decreasing. Hence,

$$\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \geq \operatorname{csch}^2\left(\lambda_2^{MAX} - \tilde{\epsilon}\right) \tag{4.59}$$

where, according to Eq. (4.42)

$$\lambda_2^{MAX} = \max_{i=1,\ldots,N} \left\{\lambda_2^i\right\} \leq \tilde{\lambda}_2 + \Xi' \tag{4.60}$$

Hence, according to Eqs. (4.59), (4.60) the condition in Eq. (4.58) is verified if

$$\operatorname{csch}^2\left(\tilde{\lambda}_2 + \Xi' - \tilde{\epsilon}\right) \sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \geq u_M \sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| \tag{4.61}$$

If the following condition holds

$$\sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \neq 0 \tag{4.62}$$

then the inequality in Eq. (4.61) can be rewritten as follows:

$$\operatorname{csch}^2\left(\tilde{\lambda}_2 + \Xi' - \tilde{\epsilon}\right) \geq u_M \frac{\sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|}{\sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2} = H(p) > 0 \tag{4.63}$$

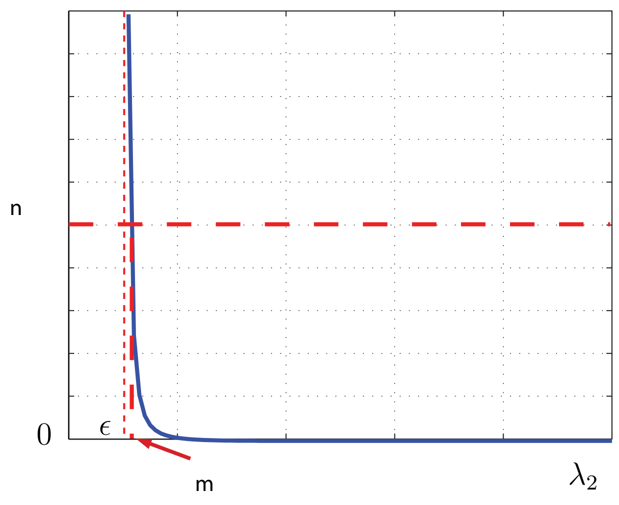


Figure 4.4: The shape of $\operatorname{csch}^2(\lambda_2 - \epsilon)$ ensures the possibility to guarantee connectivity maintenance even in the presence of (bounded) external control laws

which implies

$$\tilde{\lambda}_2 \leq \bar{\lambda}_2 = \operatorname{settcsch}\left(\sqrt{H(p)}\right) + \epsilon' \tag{4.64}$$

where $\epsilon' = \tilde{\epsilon} - \Xi'$. Therefore, as shown in Fig. 4.4, $\bar{\lambda}_2 > \epsilon'$ always exists that satisfies the condition in Eq. (4.64). Thus, $\forall \tilde{\lambda}_2 \leq \bar{\lambda}_2$ the function $\dot{V}(p) \leq 0$. Then, $\forall \tilde{\lambda}_2 \leq \bar{\lambda}_2$, the energy function $V(p)$ does not increase over time.

Let $\tilde{\lambda}_2(0) > \tilde{\epsilon}$ be the initial value of $\tilde{\lambda}_2$. If $\tilde{\lambda}_2(0) > \bar{\lambda}_2$, then the value of $\tilde{\lambda}_2$ will always be lower–bounded by $\bar{\lambda}_2$. Conversely, if $\tilde{\lambda}_2(0) \leq \bar{\lambda}_2$, then the value of $\tilde{\lambda}_2$ will increase, until $\tilde{\lambda}_2 \geq \bar{\lambda}_2$, and then it will never go below $\bar{\lambda}_2$. Namely, let

$$\hat{\lambda}_2 = \min\left(\tilde{\lambda}_2(0), \bar{\lambda}_2\right) \tag{4.65}$$

Then, the value of $\tilde{\lambda}_2$ will never go below the value of $\hat{\lambda}_2 > \epsilon'$.

In case $\left\| \dfrac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| = 0$, the condition in Eq. (4.62) is not verified. However, in this case

$$\dot{\tilde{\lambda}}_2 = \frac{\partial \tilde{\lambda}_2}{\partial p_i} \dot{p}_i = 0 \tag{4.66}$$

This implies that the value of $\tilde{\lambda}_2$ is constant over time, thus it is lower–bounded by its initial value. In both cases, $\tilde{\lambda}_2 > \epsilon'$.

Then, according to Eq. (4.43), the control law in Eq. (4.35) ensures that the value of λ_2 never goes below $\epsilon = \epsilon' - \Xi = \tilde{\epsilon} - \Xi' - \Xi$. □

Hence, given the boundedness of the estimation errors, ensuring an appropriate lower-bound on the estimate of λ_2 guarantees that the actual value of λ_2 remains strictly greater than zero.

4.3.3 Enhanced connectivity maintenance: selective control action

In this section, a selective control action is described to enhance the previously described connectivity maintenance control strategy. The objective is twofold:

1. to reduce the overall control effort introduced by the connectivity maintenance control action.
2. to reduce the interference between the connectivity maintenance control action and the main task of the system.

In order to achieve these goals, the control law given in Eq. (4.35) is modified as follows:

$$u_i^c = \gamma_i \operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} \tag{4.67}$$

where the coefficient $\gamma_i \in \mathbb{R}$ is used to modulate the control action as will be explained hereafter.

Consider now the graph $\mathcal{G}$ encoding the communication architecture of a multi–robot system. According to [18], an *edge cutset* is defined as a set of edges whose deletion would increase the number of connected components of the graph $\mathcal{G}$. If an edge cutset is constituted by a single edge, then this edge is defined as a *bridge*. In other words, if a graph is connected, deleting a bridge would cause the disconnection of the graph.

The relationship between the disconnection of a graph $\mathcal{G}$ and these concept will now be investigated.

For this purpose, define $\mathcal{N}_i$ as the neighborhood of the i–th agent, and let

$$\mathcal{N}_i = \mathcal{N}_i^c + \mathcal{N}_i^f \tag{4.68}$$

where:

- $\mathcal{N}_i^c$ is the set of the *close* neighbors of the i–th agent,
- $\mathcal{N}_i^f$ is the set of the *far* neighbors of the i–th agent.

These two sets are defined as follows:

$$\begin{aligned} \mathcal{N}_i^c &= \{j \in \mathcal{N}_i \text{ such that } \|p_i - p_j\| \leq \delta \cdot R\} \\ \mathcal{N}_i^f &= \{j \in \mathcal{N}_i \text{ such that } \|p_i - p_j\| > \delta \cdot R\} \end{aligned} \tag{4.69}$$

where $\delta \in (0,1)$ is a predefined threshold. Note that according to this definition $\mathcal{N}_i^s \cap \mathcal{N}_i^c = \emptyset$.

Moreover, the definition of *isolated agent* is introduced.

Definition 4.2. Isolated agent. *An agent j is considered* isolated *from the agent i's perspective, if it belongs to $\mathcal{N}_i^f$ and it does not belong to the $\mathcal{N}_k^c$ for any of the $k \in \mathcal{N}_i^c$, that is:*

$$j \in \mathcal{N}_i^f, \text{ and } \nexists k \in \mathcal{N}_i^c \text{ such that } j \in \mathcal{N}_k^c \tag{4.70}$$

Hence, the following definition of *critical agent* is introduced.

Definition 4.3. Critical agent. *The i–th agent identifies itself as* critical *if at least one of its neighbors is* isolated.

The definition of critical agent exhibits a symmetry property, that is:
If the i–th agent considers itself as critical by identifying the j one as isolated, then the j–th agent considers itself as critical by identifying the i one as isolated, as well.
This is a simple consequence of some geometrical facts, under the assumption of common communication range R.

Fig. 4.5 clarifies the concept of critical agent. In the figure, the grey area represents $\mathcal{N}_k^c$, the hatched area represents $\mathcal{N}_i^f$. In the left–hand picture of Fig. 4.5, no critical agents are identified: in fact, even though $j \in \mathcal{N}_i^f$, the k–th agent is a close neighbor of both the i–th and the j–th ones. Conversely, in the right–hand picture of Fig. 4.5, the i–th agent considers the i–th one as critical, and vice–versa.

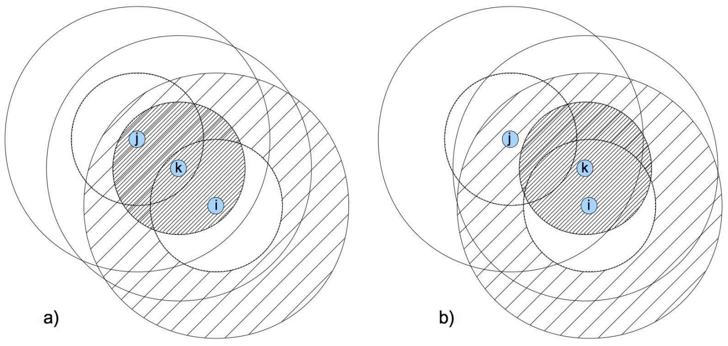


Figure 4.5: Identification of critical agents. The grey area represents $\mathcal{N}_k^c$, the hatched area represents $\mathcal{N}_i^f$. In the left–hand picture, no critical agents are identified. In the right–hand picture, the i–th agent considers the i–th one as critical, and vice–versa.

As a result, the connectivity maintenance control action is limited to those agents whose disconnection may lead to the loss of connectivity. Thus, the coefficient γ_i in Eq. (4.67) can be defined as follows:

$$\gamma_i = \begin{cases} 1 & \text{if the } i\text{–th agent is critical} \\ \rho & \text{otherwise} \end{cases} \tag{4.71}$$

with $\rho \in (0, 1)$ arbitrarily small. As will be shown in the next Section, the fact $\rho \neq 0$ is a mathematical technicality required for the correctness of the proof of Proposition 4.6.

However, since ρ can be chosen arbitrarily small, its effect can be made negligible from an implementation standpoint.

The next proposition shows that the proposed selective control action u_i^c given in Eq. (4.67) can ensure the connectivity of the communication graph for the multi–robot system under the assumption of boundedness for any additional control term u_i^d introduced to obtain some desired behavior.

Proposition 4.6. *Consider the dynamical system described by Eq.* (4.52)*. Let* Ξ, Ξ' *be defined according to Eq.* (4.42)*. Then,* $\exists\, \epsilon, \tilde{\epsilon}$ *such that, if the initial value of* $\lambda_2 > \tilde{\epsilon} + \Xi + \Xi'$*, the control law given in Eq.* (4.67) *can ensure that the value of* λ_2 *never goes below* ϵ*.*

Proof. In order to prove the statement, the partial derivative of the energy function introduced in Eq. (4.28) may be computed, with respect to an agent i, as follows:

$$\frac{\partial V}{\partial p_i} = \frac{\partial V}{\partial \tilde{\lambda}_2} \frac{\partial \tilde{\lambda}_2}{\partial p_i} = -\mathrm{csch}^2 \left(\tilde{\lambda}_2 - \tilde{\epsilon} \right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} \tag{4.72}$$

From Eqs. (4.67), (4.52), (4.72), it follows that the the time derivative of $V(p)$ can be computed as:

$$\dot{V}(p) = \\ \nabla_p V(p)^T \dot{p} = \sum_{i=1}^{N} \frac{\partial V}{\partial p_i}^T \dot{p}_i = \sum_{i=1}^{N} \left[-\mathrm{csch}^2 \left(\tilde{\lambda}_2 - \tilde{\epsilon} \right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right]^T \left[\gamma_i \mathrm{csch}^2 \left(\lambda_2^i - \tilde{\epsilon} \right) \frac{\partial \tilde{\lambda}_2}{\partial p_i} + u_i^d \right] \tag{4.73}$$

Given the boundedness of the additional control term u_i^d:

$$u_i^d \le u_M, \qquad \forall\, i = 1, \ldots, N \tag{4.74}$$

the time derivate $\dot{V}(p)$ can be restated as:

$$\dot{V}(p) \le \mathrm{csch}^2 \left(\tilde{\lambda}_2 - \tilde{\epsilon} \right) \sum_{i=1}^{N} \left[-\gamma_i \mathrm{csch}^2 \left(\lambda_2^i - \tilde{\epsilon} \right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 + \right. \\ \left. + \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| u_M \right] \tag{4.75}$$

As a result, the time derivative $\dot{V}(p) \le 0$ if the following condition holds:

$$\sum_{i=1}^{N} \left[\gamma_i \mathrm{csch}^2 \left(\lambda_2^i - \tilde{\epsilon} \right) \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \right] \ge u_M \sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| \tag{4.76}$$

According to Eq. (4.42), if the initial value of λ_2 is greater than $\tilde{\epsilon} + \Xi + \Xi'$ then the initial value of λ_2^i is greater than $\tilde{\epsilon}$, $\forall i = 1, \ldots, N$ as well. At this point, since the function $\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right)$ is monotonically decreasing with respect to λ_2^i, the following condition holds:

$$\operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \geq \operatorname{csch}^2\left(\lambda_2^{MAX} - \tilde{\epsilon}\right) \tag{4.77}$$

with λ_2^{MAX} defined as:

$$\lambda_2^{MAX} = \max_{i=1,\ldots,N}\left\{\lambda_2^i\right\} \leq \tilde{\lambda}_2 + \Xi' \tag{4.78}$$

As a result, according to Eqs. (4.77), (4.78), the inequality given in Eq. (4.76) is verified if the following holds:

$$\operatorname{csch}^2\left(\tilde{\lambda}_2 + \Xi' - \tilde{\epsilon}\right) \sum_{i=1}^{N} \gamma_i \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \geq u_M \sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\| \tag{4.79}$$

Assume now that the following condition holds:

$$\sum_{i=1}^{N} \gamma_i \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2 \neq 0 \tag{4.80}$$

As a matter of fact, this implies that the inequality in Eq. (4.79) can be rewritten as follows:

$$\operatorname{csch}^2\left(\tilde{\lambda}_2 + \Xi' - \tilde{\epsilon}\right) \geq u_M \frac{\sum_{i=1}^{N} \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|}{\sum_{i=1}^{N} \gamma_i \left\| \frac{\partial \tilde{\lambda}_2}{\partial p_i} \right\|^2} = H(p) > 0 \tag{4.81}$$

which implies

$$\tilde{\lambda}_2 \leq \bar{\lambda}_2(p) = \operatorname{settcsch}\left(\sqrt{H(p)}\right) + \epsilon' \tag{4.82}$$

where $\epsilon' = \tilde{\epsilon} - \Xi'$, and $\operatorname{settcsch}(\cdot)$ is the inverse function of $\operatorname{csch}(\cdot)$. At this point, it should be noticed $\bar{\lambda}_2(p) > \epsilon'$ always exist such that the condition given in Eq. (4.82) is satisfied, as shown in Fig. 4.4. This implies that:

$$\dot{V}(p) \leq 0, \qquad \forall \tilde{\lambda}_2 \leq \bar{\lambda}_2(p) \tag{4.83}$$

Therefore, $\forall \tilde{\lambda}_2 \leq \bar{\lambda}_2(p)$, the energy function $V(p)$ does not increase over time.

With a slight abuse of notation, let $\tilde{\lambda}_2(t)$ and $\bar{\lambda}_2(t)$ be the values of $\tilde{\lambda}_2(\cdot)$ and $\bar{\lambda}_2(\cdot)$ at time t, respectively.

Suppose $\tilde{\lambda}_2(0) > \tilde{\epsilon} > \epsilon'$ to be the initial value of $\tilde{\lambda}_2$. If $\tilde{\lambda}_2(0) > \bar{\lambda}_2(0) > \epsilon'$, then the value of $\tilde{\lambda}_2$ will always be lower–bounded by ϵ'.

Conversely, if $\epsilon' < \tilde{\lambda}_2(0) \leq \bar{\lambda}_2(0)$, then the value of $\tilde{\lambda}_2$ will increase, until $\tilde{\lambda}_2(t) \geq \bar{\lambda}_2(t)$. Then, the value of $\tilde{\lambda}_2$ will never go below ϵ'.

Note that, in the case of $\left\|\frac{\partial \tilde{\lambda}_2}{\partial p_i}\right\| = 0$, the condition given in Eq. (4.80) is not verified. Nevertheless, it should be noticed that:

$$\dot{\tilde{\lambda}}_2 = \frac{\partial \tilde{\lambda}_2}{\partial p_i}\dot{p}_i = 0 \tag{4.84}$$

which implies that the value of $\tilde{\lambda}_2$ is constant over time, thus it is lower–bounded by its initial value. In both cases, $\tilde{\lambda}_2 > \epsilon'$.

Then, according to Eq. (4.43), the control law in Eq. (4.67) ensures that the value of λ_2 never goes below $\epsilon = \epsilon' - \Xi = \tilde{\epsilon} - \Xi' - 2\Xi$. □

As a result, the boundedness of the estimation errors is a sufficient condition to prove that, by ensuring an appropriate lowerbound on the estimate of λ_2, the actual value of λ_2 can be ensured to be strictly greater than zero over time.

Definition of the critical agents

4.3.4 Identification of the critical agents

In this section, a local policy is described for the identification of the critical agents, according to Definition 4.3.

The proposed strategy exhibits the following properties:

- it does not require the agents to have a unique identifier,
- it only relies on local sensing information available to each agent,
- does not require explicit communication among the agents.

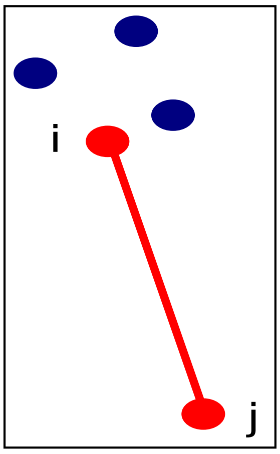
Referring to Definition 4.3, the local policy may be described with the following algorithm.

The five configurations shown in Fig. 4.6 for the communication graph are representative of all the possible scenarios. More specifically:

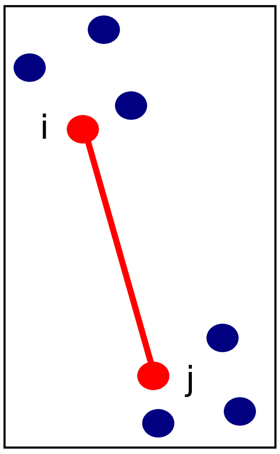
- in Fig. 4.6a, the j–th agent is isolated: only the i–th one is in its neighborhood, and they are far neighbors. Disconnecting the red link would cause the disconnection of the graph. Furthermore, none of the i–th agent's close neighbors (blue dots) is close to the j–th agent: hence they are both considered critical, and $\gamma_i = \gamma_j = 1$.

Algorithm 3 Local policy to identify the critical robots

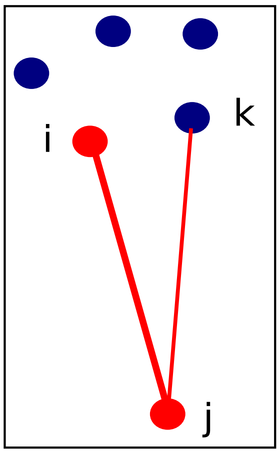
```
1: $\gamma_i = 1$
2: **if** $\left\{\mathcal{N}_i^f = \emptyset\right\}$ **then**
3:     $\gamma_i = \rho$
4: **end if**
5: **if** $\left\{\forall j \in \mathcal{N}_i^f \ \ \exists k \in \mathcal{N}_i^c \text{ s.t. } j \in \mathcal{N}_k^c\right\}$ **then**
6:     $\gamma_i = \rho$
7: **end if**
```



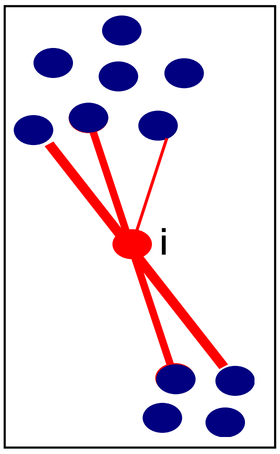
(a) i and j are identified as critical agents



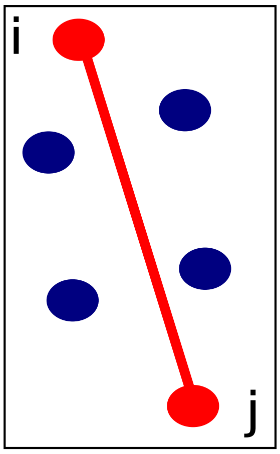
(b) i and j are identified as critical agents



(c) i and j (and k) are identified as critical agents



(d) i and all its neighbors are identified as critical agent



(e) i and j are clearly identified as non–critical agents

Figure 4.6: Decision algorithm to define the critical agents: some examples

- in Fig. 4.6b, the link between the i–th and the j–th agents links two different components of the graph. As in the previous example, both agents are considered critical, and $\gamma_i = \gamma_j = 1$.

- in Fig. 4.6c, the j–th agent is isolated, and is likely to lose connectivity from all its neighbors. Hence, it is identified as a critical agent, i.e. $\gamma_j = 1$. Analogously, the i–th agent is considered critical, as well as the k–th one.

- in Fig. 4.6d, the i–th agent is identified as isolated by all its neighbors, and is then a critical agent. Due to the symmetry property of Definition 4.3, the i–th agent's neighbors are critical agents as well.

- Fig. 4.6e represents a situation where the connectivity maintenance action is not needed: in fact, even though the j–th agent is a far neighbor of the i–th one, they have some close neighbors in common (blue dots). In this case, they are both identified as non–critical neighbors, thus $\gamma_i = \gamma_j = \rho$.

4.3.5 Rendezvous and formation control

This section exploits the control strategy described so far to address the connectivity maintenance issue within rendezvous and formation control problems.

For this purpose, the connectivity maintenance control action may be rewritten in vector form. To this aim, define $\bar{a}_{ij}$ as follows:

$$\bar{a}_{ij}\left(\lambda_2^i\right) = \gamma_i \operatorname{csch}^2\left(\lambda_2^i - \tilde{\epsilon}\right) \frac{1}{\sigma^2} \left(\tilde{v}_2^i - \tilde{v}_2^j\right)^2 a_{ij} \tag{4.85}$$

Furthermore, a modified degree matrix of the graph may be defined as $\bar{D} = \operatorname{diag}\left(\left\{\bar{d}_i\right\}\right)$, where $\bar{d}_i$ is the degree of the i–th node of the graph, i.e. $\bar{d}_i = \sum_{j=1}^{N} \bar{a}_{ij}$. The modified Laplacian matrix can be then defined accordingly, as $\bar{L} = \bar{D} - \bar{A}$.

Hence, the control law in Eqs. (4.52)–(4.67) can be rewritten in vector form:

$$\dot{p} = -\bar{L}p + u^d \tag{4.86}$$

where u^d is the vector containing the desired control laws.

In the following subsections, for the sake of clarity, the dynamics of the system will be considered in one dimension only, without loss of generality. All the results presented hereafter, in fact, may be easily extended to the m–dimensional case.

Consensus–based rendezvous

In this section the connectivity maintenance issue in the rendezvous problem, i.e. making a group of robots converge to the same position, is address. To this end, the following consensus–based control law may be introduced:

$$u^d = -L_* p \tag{4.87}$$

As shown in [107], this control law guarantees the convergence of the system to the rendezvous configuration, provided that the communication graph is connected.

Proposition 4.7. *Under the control law described by Eqs.* (4.86), (4.87)*, the connectivity of the communication graph is always ensured.*

Proof. To prove the statement, the results given in Proposition 4.6 is exploited. More specifically, the value u_M will be now derived, to be used in Eqs. (4.81), (4.82) to define the lowerbound on λ_2. In particular, since

$$u_i^d = \sum_{j \in \mathcal{N}_i} (p_i - p_j) \tag{4.88}$$

it follows that

$$\left\| u_i^d \right\| \leq u_M = NR \tag{4.89}$$

being R the greatest possible distance between two neighboring robots. □

Proposition 4.8. *Under the control strategy introduced in Eqs.* (4.86), (4.87)*, the system asymptotically converges to the rendezvous configuration.*

Proof. The proof is analogous to that of [107], considering $\tilde{L} = \bar{L} + L_*$ as the Laplacian matrix of the communication graph. □

Consensus–based formation control

In this section the connectivity maintenance issue is addressed in the formation control problem. Consensus–based control laws can be exploited for formation control as well. As in the rendezvous case, the convergence of the system to the desired configuration is guaranteed, provided that the communication graph is connected. In [69] the following control strategy has been introduced:

$$u^d = -L_* p + b \tag{4.90}$$

where

$$b = L_* \bar{p} \tag{4.91}$$

and $\bar{p}$ is a vector containing the desired positions for the robots in the formation. More specifically, $\bar{p}$ represents the desired relative position of each robot with respect to the center of the formation. Hence, the desired configuration can be described as follows:

$$p = \alpha \mathbf{1} + \bar{p} \tag{4.92}$$

for some $\alpha \in \mathbb{R}$. In other words,

$$(p - \bar{p}) \in \operatorname{span}(\mathbf{1}) \tag{4.93}$$

The bias term defined in Eq. (4.91) is now slightly modified as follows:

$$b = \{b_i(p)\} \tag{4.94}$$

where

$$b_i(p) = \begin{cases} \sum\limits_{j \in \mathcal{N}_i} \left(1 + \bar{a}_{ij}\left(\lambda_2^i\right)\right) \cdot \left(\bar{p}_i - \bar{p}_j\right) & \text{if } \lambda_2^i > k\tilde{\epsilon} \\ \sum\limits_{j \in \mathcal{N}_i} \left(1 + \bar{a}_{ij}\left(k\tilde{\epsilon}\right)\right) \cdot \left(\bar{p}_i - \bar{p}_j\right) & \text{otherwise} \end{cases} \tag{4.95}$$

for $k > 1$. Roughly speaking, when the estimate of the algebraic connectivity is sufficiently greater than $\tilde{\epsilon}$ (i.e. $\lambda_2^i > k\tilde{\epsilon}$), then the bias term is computed exploiting the following Laplacian matrix:

$$\tilde{L} = \bar{L} + L_* \tag{4.96}$$

Conversely, when the value of the estimate of the algebraic connectivity approaches $\tilde{\epsilon}$, this design of the control law ensures boundedness of u_i^d that, as shown in Proposition 4.7, is necessary to ensure connectivity.

Proposition 4.9. *Under the control law described by Eqs.* (4.86), (4.90)*, the connectivity of the communication graph is always ensured.*

Proof. The proof is analogous to that of Proposition 4.7. □

In [69] the proof of the convergence of the system to the desired formation is provided. The next proposition shows how this proof can be extended in order to guaranteed the convergence even in the presence of the connectivity maintenance control action.

Proposition 4.10. *Assume that, if $p = \bar{p}$, then $\lambda_2^i > k\tilde{\epsilon}$, $\forall i = 1, \ldots, N$. Then, under the control strategy introduced in Eqs.* (4.86), (4.90)*, the system asymptotically converges to the formation defined as $p = \bar{p}$.*

Proof. When $\lambda_2^i > k\tilde{\epsilon}$, then the control law introduced in Eqs. (4.86), (4.90) may be rewritten as follows:

$$\dot{p}(t) = -\tilde{L}(t)(p(t) - \bar{p}) \tag{4.97}$$

where the dependence on time has been added for clarity purposes.

As shown in [107], under the control law in Eq. (4.97), the system evolves until

$$(p(t) - \bar{p}) \in \ker\left(\tilde{L}\right) \tag{4.98}$$

Since $\ker\left(\tilde{L}\right) = \operatorname{span}(\mathbf{1})$, these condition is verified if

$$p(t) - \bar{p} = \alpha\mathbf{1} \tag{4.99}$$

for some $\alpha \in \mathbb{R}$. Hence,

$$p(t) = \bar{p} + \alpha\mathbf{1} \tag{4.100}$$

□

4.4 Connectivity maintenance for networked Lagrangian dynamical systems

This section extends the scope of the control strategy introduced in Section 4.3, in order to address the connectivity maintenance issue for groups of Lagrangian dynamical systems.

To this aim, a generalized communication model will now be defined. Let $p_{ij} = p_i - p_j$, and let H be some properly defined constant matrix. Suppose $H \geq 0$.

The matrix H is defined such that the j–th agent is inside $\mathcal{N}_i$ if $p_{ij}^T H p_{ij} \leq R^2$, for some parameter $R > 0$.

The edge–weights a_{ij}, first introduced in Eq. (4.32), are then re–defined as follows:

$$a_{ij} = \begin{cases} e^{-\dfrac{p_{ij}^T H p_{ij}}{2\sigma^2}} & \text{if } p_{ij}^T H p_{ij} \leq R^2 \\ 0 & \textit{otherwise} \end{cases} \tag{4.101}$$

As for the single integrator case, the scalar parameter σ is chosen to satisfy the threshold condition $e^{-\left(R^2\right)/\left(2\sigma^2\right)} = \Delta$, where Δ is a small predefined threshold.

Given the definition of the edge–weights in Eq. (4.101), the value of $\dfrac{\partial\lambda_2}{\partial p_i}$ can be computed as follows (see also Eq. (4.34)):

$$\frac{\partial\lambda_2}{\partial p_i} = \sum_{j\in\mathcal{N}_i} -a_{ij}\left(v_2^i - v_2^j\right)^2 \frac{H(p_i - p_j)}{\sigma^2} \tag{4.102}$$

4.4.1 Connectivity maintenance control strategy

Consider a group of N Lagrangian systems: Let $p_i \in \mathbb{R}^m$ be the state vector of the i–th Lagrangian agent, and let $p = \left[p_1^T \dots p_N^T\right]^T \in \mathbb{R}^{Nm}$ be the state vector of the multi–agent system.

Hence, the following dynamic model may be considered:

$$M\left(p_i\right)\ddot{p}_i + C\left(p_i, \dot{p}_i\right)\dot{p}_i + D\dot{p}_i + g\left(p_i\right) = u_i \tag{4.103}$$

where u_i is the control input. The matrix $M\left(p_i\right)$ is the symmetric positive definite mass matrix, the matrix $C\left(p_i, \dot{p}_i\right)$ represents the Coriolis effects, the matrix D represents a dissipative term due to friction, and $g\left(p_i\right)$ is the gravity term. Further details can be in [108].

As in the the previous section, let the energy function be defined as follows:

$$V\left(p\right) = \coth\left(\tilde{\lambda}_2 - \tilde{\epsilon}\right) \tag{4.104}$$

The following control law is now introduced:

$$u_i = g\left(p_i\right) + u_i^c \tag{4.105}$$

We remark that, as we are exploiting passivity based analysis, adaptive gravity compensation is possible.

As in the previous case, the control term u_i^c is defined as follows:

$$u_i^c = -\frac{\partial V}{\partial p_i} \tag{4.106}$$

Proposition 4.11. *Consider the dynamic described by Eq.* (4.103)*. Let Ξ, Ξ' be defined according to Eq.* (4.42)*, and let $\tilde{\epsilon} = \epsilon + \Xi + \Xi'$. If the initial value of $\lambda_2 > \tilde{\epsilon} + \Xi + \Xi'$, then the control law defined in Eqs.* (4.105)–(4.106) *ensures that the value of λ_2 never goes below ϵ.*

Proof. Let

$$W\left(p, \dot{p}\right) = \frac{1}{2}\sum_{i=1}^{N} \dot{p}_i^T M\left(p_i\right)\dot{p}_i + V\left(p\right) \tag{4.107}$$

The time derivative of W may be computed as follows:

$$\dot{W}\left(p, \dot{p}\right) = \dot{p}^T \nabla_p W\left(p\right) = \sum_{i=1}^{N} \dot{p}_i^T \frac{\partial W}{\partial p_i} \tag{4.108}$$

Hence:

$$\dot{W}(p,\dot{p}) = \sum_{i=1}^{N} \left(\dot{p}_i^T M(p_i)\ddot{p}_i + \frac{1}{2}\dot{p}_i^T \dot{M}(p_i)\dot{p}_i + \dot{p}_i^T \frac{\partial V}{\partial p_i} \right) \tag{4.109}$$

From Eqs. (4.103), (4.105), (4.106):

$$M(p_i)\ddot{p}_i + C(p_i,\dot{p}_i)\dot{p}_i + D\dot{p}_i = -\frac{\partial V}{\partial p_i} \tag{4.110}$$

Hence:

$$M(p_i)\ddot{p}_i = -\frac{\partial V}{\partial p_i} - C(p_i,\dot{p}_i)\dot{p}_i - D\dot{p}_i \tag{4.111}$$

Hence, from Eqs. (4.109), (4.111):

$$\dot{W}(p,\dot{p}) = \sum_{i=1}^{N} \left(-\dot{p}_i^T \frac{\partial V}{\partial p_i} - \dot{p}_i^T C(p_i,\dot{p}_i)\dot{p}_i - \dot{p}_i^T D\dot{p}_i + \frac{1}{2}\dot{p}_i^T \dot{M}(p_i)\dot{p}_i + \dot{p}_i^T \frac{\partial V}{\partial p_i} \right) \tag{4.112}$$

Hence:

$$\dot{W}(p,\dot{p}) = \sum_{i=1}^{N} \left(\frac{1}{2}\dot{p}_i^T \left(\dot{M}(p_i) - 2C(p_i,\dot{p}_i) \right) \dot{p}_i - \dot{p}_i^T D\dot{p}_i \right) = \sum_{i=1}^{N} \left(-\dot{p}_i^T D\dot{p}_i \right) \leq 0 \tag{4.113}$$

With a slight abuse of notation, hereafter functions $W(\cdot)$ and $V(\cdot)$ will be referred to as $W(t)$ and $V(t)$, even though they are not explicit functions of time.

Hence, $\forall t \geq 0$, $W(t) \leq W(0)$,$\forall t \geq 0$. From Eq. (4.107), it follows that $V(t) \leq W(t)$, $\forall t \geq 0$. Thus, it is possible to conclude that $V(t) \leq W(0)$, $\forall t \geq 0$.

Given the definition of $V(t)$ provided in Eq. (4.104), it is possible to state that V is monotonically decreasing with respect to $\tilde{\lambda}_2$, $\forall \tilde{\lambda}_2 > \tilde{\epsilon}$. According to Eq. (4.43), the fact that the initial value of λ_2 is greater than $\tilde{\epsilon} + \Xi + \Xi'$ ensures that the initial value of $\tilde{\lambda}_2$ is greater than $\tilde{\epsilon}$.

Thus, it is possible to conclude that $\exists \bar{\lambda}_2 > \tilde{\epsilon}$ such that $\tilde{\lambda}_2(t) \geq \bar{\lambda}_2$, $\forall t \geq 0$. Hence, according to Eq. (4.43), this implies that $\lambda_2 \geq \epsilon = \tilde{\epsilon} - 2\Xi - \Xi'$. □

4.4.2 Connectivity in the presence of external control laws

In this section, the presence of external control laws is explicitly considered:

$$u_i = g(p_i) + u_i^c + u_i^d \tag{4.114}$$

Specifically, the following case will be considered: u_i^d is supposed to be the gradient of an appropriately designed potential function, that is:

$$u_i^d = -\frac{\partial U(p)}{\partial p_i} \tag{4.115}$$

where $U(p)$ is supposed to be a positive definite potential function.

Proposition 4.12. *Consider the dynamic described by Eq.* (4.103)*. Let Ξ, Ξ' be defined according to Eq.* (4.42)*, and let $\tilde{\epsilon} = \epsilon + \Xi + \Xi'$. If the initial value of $\lambda_2 > \tilde{\epsilon} + \Xi + \Xi'$, then the control law defined in Eqs.* (4.114)*–*(4.106)*–*(4.115) *ensures that the value of λ_2 never goes below ϵ.*

Proof. Let

$$T(p, \dot{p}) = \frac{1}{2} \sum_{i=1}^{N} \dot{p}_i^T M(p_i) \dot{p}_i + V(p) + U(p) \tag{4.116}$$

The time derivative of T may be computed as follows:

$$\dot{T}(p, \dot{p}) = \dot{p}^T \nabla_p T(p) = \sum_{i=1}^{N} \dot{p}_i^T \frac{\partial T}{\partial p_i} \tag{4.117}$$

Hence:

$$\dot{T}(p, \dot{p}) = \sum_{i=1}^{N} \left(\dot{p}_i^T M(p_i) \ddot{p}_i + \frac{1}{2} \dot{p}_i^T \dot{M}(p_i) \dot{p}_i + \dot{p}_i^T \frac{\partial V}{\partial p_i} \dot{p}_i^T \frac{\partial U}{\partial p_i} \right) \tag{4.118}$$

From Eqs. (4.103), (4.114), (4.106), (4.115):

$$M(p_i) \ddot{p}_i + C(p_i, \dot{p}_i) \dot{p}_i + D\dot{p}_i = -\frac{\partial V}{\partial p_i} - \frac{\partial U}{\partial p_i} \tag{4.119}$$

Hence:

$$M(p_i) \ddot{p}_i = -\frac{\partial V}{\partial p_i} - \frac{\partial U}{\partial p_i} - C(p_i, \dot{p}_i) \dot{p}_i - D\dot{p}_i \tag{4.120}$$

Hence, from Eqs. (4.118), (4.120):

$$\begin{aligned} &\dot{T}(p, \dot{p}) = \\ &= \sum_{i=1}^{N} \left(-\dot{p}_i^T \frac{\partial V}{\partial p_i} - \dot{p}_i^T \frac{\partial U}{\partial p_i} - \dot{p}_i^T C(p_i, \dot{p}_i) \dot{p}_i - \dot{p}_i^T D\dot{p}_i + + \frac{1}{2} \dot{p}_i^T \dot{M}(p_i) \dot{p}_i + \dot{p}_i^T \frac{\partial V}{\partial p_i} + \dot{p}_i^T \frac{\partial U}{\partial p_i} \right) \end{aligned} \tag{4.121}$$

Hence:

$$\dot{T}(p, \dot{p}) = \sum_{i=1}^{N} \left(\frac{1}{2} \dot{p}_i^T \left(\dot{M}(p_i) - 2C(p_i, \dot{p}_i) \right) \dot{p}_i - \dot{p}_i^T D\dot{p}_i \right) = \sum_{i=1}^{N} \left(-\dot{p}_i^T D\dot{p}_i \right) \leq 0 \tag{4.122}$$

With a slight abuse of notation, hereafter $T(\cdot)$, $V(\cdot)$ and $U(\cdot)$ will be referred to as $T(t)$, $V(t)$ and $U(t)$, even though they are not explicit functions of time.

Hence, $\forall t \geq 0$, $T(t) \leq T(0)$, $\forall t \geq 0$. From Eq. (4.116), it follows that $V(t) \leq T(t)$, $\forall t \geq 0$. Thus, it is possible to conclude that $V(t) \leq T(0)$, $\forall t \geq 0$.

Given the definition of $V(t)$ provided in Eq. (4.104), it is possible to state that V is monotonically decreasing with respect to $\tilde{\lambda}_2$, $\forall \tilde{\lambda}_2 > \tilde{\epsilon}$. According to Eq. (4.43), the fact that the initial value of λ_2 is greater than $\tilde{\epsilon} + \Xi + \Xi'$ ensures that the initial value of $\tilde{\lambda}_2$ is greater than $\tilde{\epsilon}$.

Thus, it is possible to conclude that $\exists \bar{\lambda}_2 > \tilde{\epsilon}$ such that $\tilde{\lambda}_2(t) \geq \bar{\lambda}_2$, $\forall t \geq 0$. Hence, according to Eq. (4.43), it follows that $\lambda_2 \geq \epsilon = \tilde{\epsilon} - 2\Xi - \Xi'$. □

4.4.3 Application: rendezvous for fully actuated Lagrangian systems

This section will show how to apply the connectivity maintenance control algorithm to a group of fully actuated Lagrangian systems performing a rendezvous task.

Dynamics and control law

Consider a group of 6–degree–of–freedom spacecraft vehicles, whose dynamics are described in [109].

Specifically, the configuration of these vehicles is described by the following state vectors:

$$p_i = \left[x_i^T \ \theta_i^T\right]^T \quad \dot{p}_i = \left[v_i^T \ \omega_i^T\right] \tag{4.123}$$

where $x_i \in \mathbb{R}^3$ represents the position of the i–th robot, and θ_i represents the rotation of the i–th robot, expressed in terms of Euler parameters [110]. $v_i \in \mathbb{R}^3$ and $\omega_i \in \mathbb{R}^3$ are the linear and angular velocity of the i–th robot, respectively.

The following relationship holds:

$$\begin{aligned} \dot{x}_i &= v_i \\ \dot{\theta}_i &= T(p_i)\,\omega_i \end{aligned} \tag{4.124}$$

where $T(p_i)$ is a properly defined transformation matrix.

Referring to Eq. 4.103, the matrices that describe the dynamics of each spacecraft vehicle are defined as follows:

$$\begin{aligned} M(p_i) &= \begin{bmatrix} m_s I_3 & 0_{3\times3} \\ 0_{3\times3} & J_s(p_i) \end{bmatrix} \\ C(p_i, \dot{p}_i) &= \begin{bmatrix} C_t(x_i, \dot{x}_i) & 0_{3\times3} \\ 0_{3\times3} & C_r(\theta_i, \omega_i) \end{bmatrix} \\ g(p_i) &= \begin{bmatrix} g_t(x_i) \\ 0_{3\times1} \end{bmatrix} \\ D &= 0_{3\times3} \end{aligned} \tag{4.125}$$

where $0_{\zeta\times\xi}$ is a zero matrix with ζ rows and ξ columns, and I_ξ is the identity matrix of size ξ. The value m_s represents the mass of the spacecraft, while $J_s(p_i)$ is the matrix representing the moments of inertia.

From Eq. (4.125) it's easy to see that translations and rotations are decoupled, and can be independently controlled. Hence, hereafter only the translational dynamics of the system will be considered. The matrix $C_t\left(x_i, \dot{x}_i\right)$ is a Coriolis–like skew–symmetric matrix, and is defined as follows:

$$C_t\left(x_i, \dot{x}_i\right) = 2m_s\dot{v}_i \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.126}$$

The gravity term $g_t\left(x_i\right)$ is defined as follows:

$$g_t\left(x_i\right) = m_f \begin{bmatrix} \dfrac{\mu}{r_s^3} - \dot{v}_i^2 & -\ddot{v}_i & 0 \\ \ddot{v}_i & \dfrac{\mu}{r_s^3} - \dot{v}_i^2 & 0 \\ 0 & 0 & \dfrac{\mu}{r_s^3} \end{bmatrix} x_i \tag{4.127}$$

where r_s is the average radius of the orbit of the spacecraft. Let G be the universal constant of gravity, and let M_e be the mass of the Earth: then, $\mu \approx GM_e$.

Consider the following connectivity model: two robots can communicate if their Euclidean distance is less than or equal to R. More specifically, the matrix H in Eq. (4.101) is defined as follows:

$$H = \begin{bmatrix} I_3 & 0_{3\times4} \\ 0_{3\times4} & 0_{4\times4} \end{bmatrix} \tag{4.128}$$

With this definition of H, the term $p_{ij}^T H p_{ij}$ is exactly the Euclidean distance between the i–th and the j–th robot. According to the definition of the edge–weights introduced in Eq. (4.101), the i–th and the j–th agents are neighbors if their Euclidean distance is less than or equal to R. Furthermore, given the definition of the matrix H provided in Eq. (4.128), it follows that only the first three components of the control action u_i^c will be different from zero.

Define $\bar{u}_i^c \in \mathbb{R}^3$ as the vector containing the first three components of u_i^c. Hence, $u_i^c = \left[\bar{u}_i^c\ 0_{3\times1}^T\right]^T$.

In order to make the robots perform a rendezvous task, an additional control law u_i^d is added, defined as in Eq. (4.115), where the potential field $U\left(p\right)$ is defined as follows:

$$U_i = \sum_{j\in\mathcal{N}_i} \frac{1}{2} K_r \left(x_i - x_j\right) \tag{4.129}$$

where $K_r > 0$ is a properly defined constant. It's easy to prove that, as long as the communication graph is connected, this control law yields to the rendezvous of the multi–robot system.

4.5 Simulations and experiments

4.5.1 Matlab simulations

Several Matlab simulations have been implemented, for validation purposes. Both single integrator and Lagrangian agents have been simulated: the number of agents have been varied, from $N = 3$ to $N = 25$, and their initial positions have been randomly chosen.

Estimation process

Preliminary simulations have been carried out with the objective of evaluating the performance of the estimation algorithm. Specifically, five agents have been simulated while only running the estimation procedure.

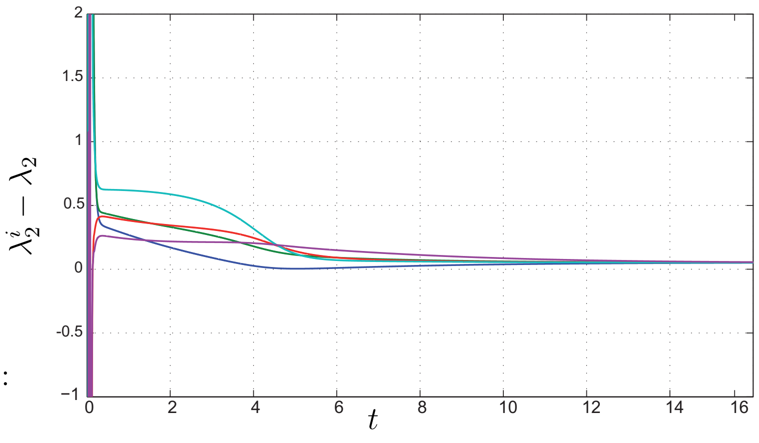


Figure 4.7: Estimation error of λ_2: $(\lambda_2^i - \lambda_2)$

In Fig. 4.7 the estimation error of λ_2 is shown: each line represents the difference between one of the estimates λ_2^i and the actual value λ_2. As expected, the estimation error is bounded.

Connectivity maintenance in the presence of an external controller, for single integrator kinematic agents

The effectiveness of the proposed connectivity maintenance control algorithm has been tested in the presence of different external control laws.

Comparison with the algorithm proposed in [101] To compare the control strategy introduced in this chapter with the one previously proposed in [101], the following external control law has been defined:

$$u_i^d = \begin{bmatrix} k\cos\left(\frac{2\pi}{N+1}i\right) \\ k\sin\left(\frac{2\pi}{N+1}i\right) \end{bmatrix} \tag{4.130}$$

for different values of $k > 0$. The results presented hereafter were obtainet with $k = 5$.

Without the connectivity maintenance controller (i.e. $u_i^c = 0$), the external control law makes the agents move away from each other. As shown in Fig. 4.8 (red dashed line), the value of λ_2 decreases, until the connectivity of the communication graph is lost. Simulations give a similar result implementing the connectivity maintenance controller described in [101], as shown in Fig. 4.8 (blue dotted line).

As expected, using the connectivity maintenance controller described in this chapter, the connectivity of the communication graph is never lost. In this setup, the value of $\tilde{\epsilon}$ has been empirically set to 1.3: simulations show that, with this choice, the value of λ_2 is always bounded from zero.

Consensus–based rendezvous Connectivity maintenance have been validated in the simulation of a rendezvous application: six single integrator kinematic agents, starting from random initial positions, were supposed to converge to a common point, while avoiding collisions with randomly placed point obstacles. For obstacle avoidance purposes, a repulsive artificial potential field has been added (see e.g. [111]), ensuring that it produces a bounded control action.

Snapshots of the simulations are shown in Figs. 4.9 and 4.10: red stars represent the agents, while blue dots represent randomly placed point obstacles. Fig. 4.9 shows the behavior of the system without the connectivity maintenance control strategy. Conversely,

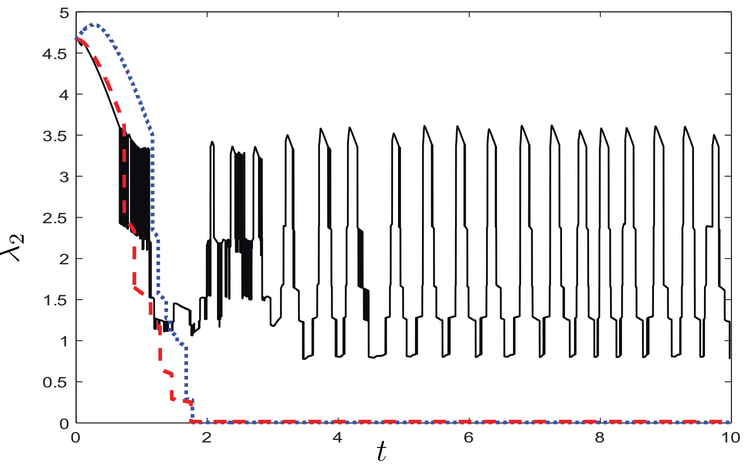


Figure 4.8: Value of λ_2 with a *disconnecting* external controller, *with* the connectivity maintenance controller described in this paper (black solid line), *with* the connectivity maintenance controller described in [101] (blue dotted line), and *without* any connectivity maintenance controller (red dashed line)

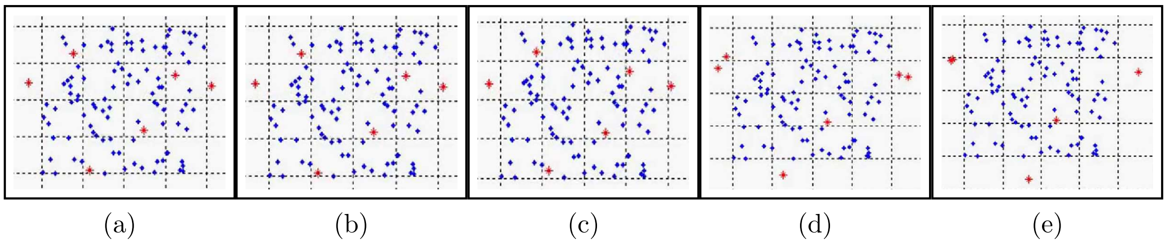


Figure 4.9: Matlab simulation, rendezvous application, WITHOUT connectivity maintenance control strategy: red stars represent the agents, blue dots represent randomly placed point obstacles

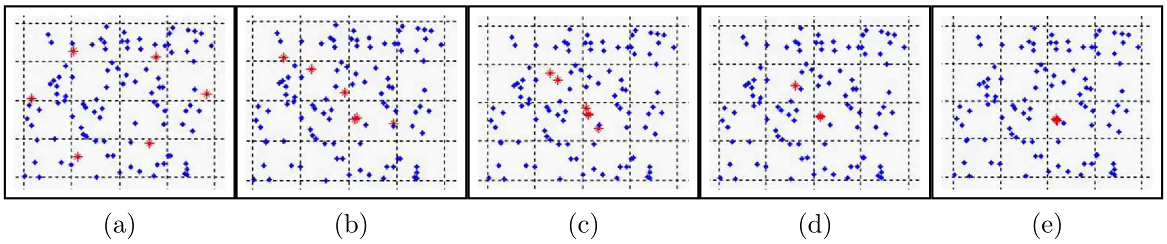


Figure 4.10: Matlab simulation, rendezvous application, WITH connectivity maintenance control strategy: red stars represent the agents, blue dots represent randomly placed point obstacles

Fig. 4.10 shows the behavior of the system with the connectivity maintenance control strategy. After a few seconds, without the connectivity maintenance controller the connectivity is lost: the obstacle avoidance action obstructs the desired movement of some agents, that are thus trapped and lose connectivity with the other ones. As expected, using the connectivity maintenance control action the connectivity of the graph is always preserved, as shows also in Fig. 4.11.

Consensus–based formation control Analogously, the connectivity maintenance algorithm have been validated in the simulation of a formation control application: six single integrator kinematic agents, starting from random initial positions, were supposed to converge to an hexagonal formation, and to move at constant velocity along the x–axis, while avoiding collisions with randomly placed point obstacles. To make the formation move in a desired direction, a common offset has been added to the control law in Eq. (4.90), that describes the desired speed of the barycenter of the formation.

As in the rendezvous case, without the connectivity maintenance controller, the con-

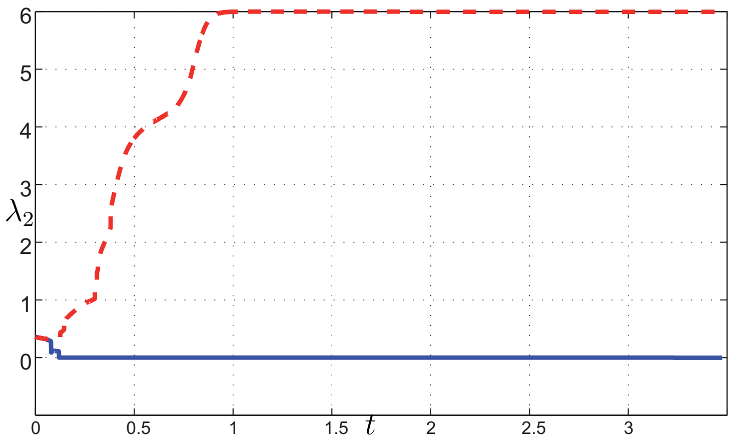


Figure 4.11: Rendezvous simulation results: value of λ_2 *with* (red dashed line) and *without* (blue solid line) the connectivity maintenance controller

nectivity is lost quite soon: the obstacle avoidance action obstructs the desired movement of some agents, that are thus trapped and lose connectivity with the other ones. As expected, using the connectivity maintenance control action the connectivity of the graph is always preserved, as shows also in Fig. 4.12.

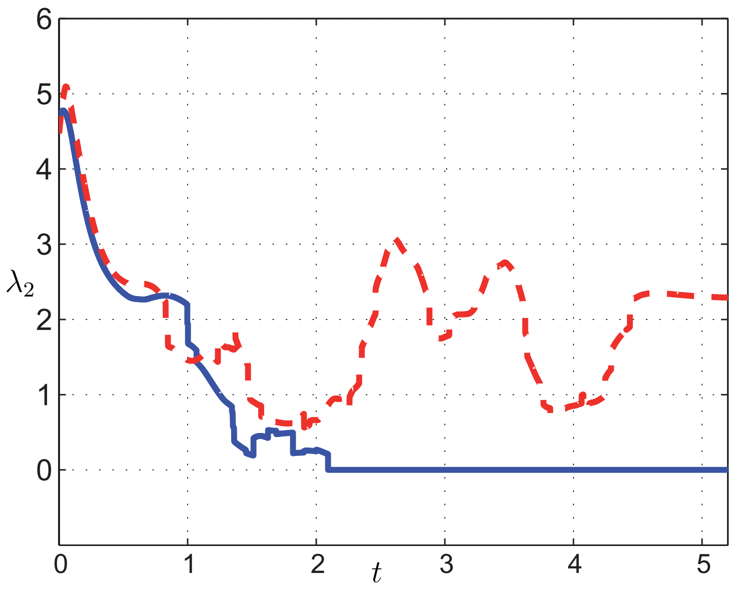


Figure 4.12: Formation control simulation results: value of λ_2 *with* (red dashed line) and *without* (blue solid line) the connectivity maintenance controller

Enhanced connectivity maintenance algorithm, for single integrator kinematic agents

Several Matlab simulations have been carried out in order to compare the enhanced connectivity maintenance control strategy (introduced in Section 4.3.3) with the standard one (introduced in Section 4.3.2).

Simulations have been carried out by considering the following parameters setting

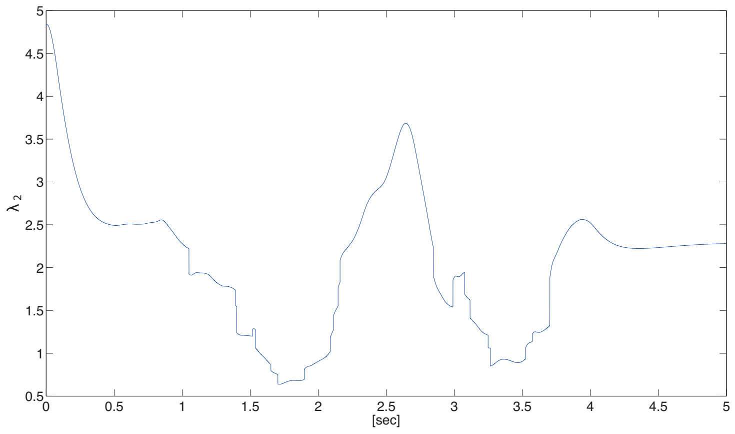


Figure 4.13: Value of λ_2 with the standard connectivity maintenance control strategy presented in this paper

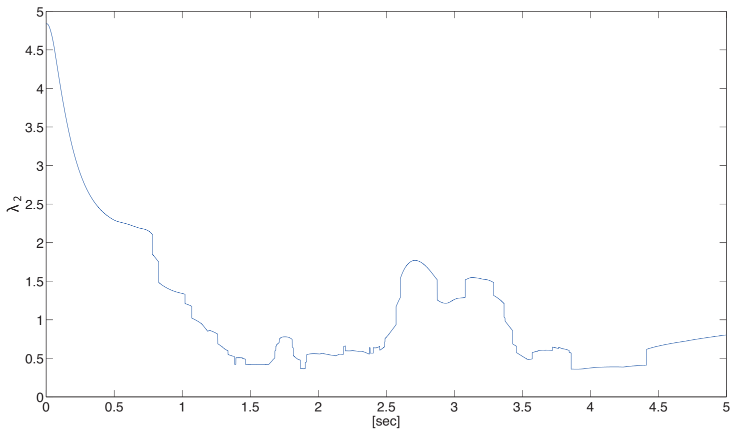
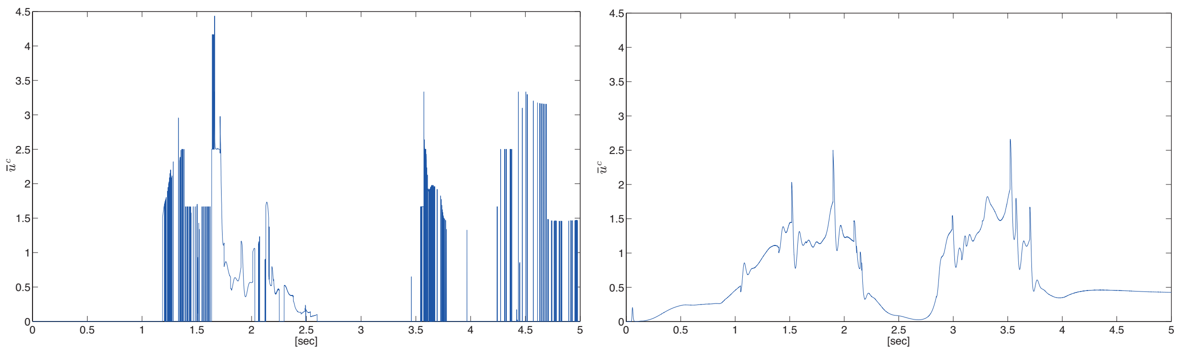


Figure 4.14: Value of λ_2 with the selective connectivity maintenance control strategy presented in this paper

$\{\rho = 10^{-5},\, \delta = 0.8\}$. In the simulation, a multi–robot system composed of 6 agents was involved in a formation control task: starting from random initial positions, they were supposed to converge to an hexagonal formation, and move at constant velocity along the x–axis, while avoiding collisions with randomly placed point obstacles.

In order to point out the advantages of the enhanced control action with respect to the standard one, the two control laws have been implemented within the same setup, that is by considering the same initial positions for the agents and the same positions for the obstacles. Fig. 4.14 and Fig. 4.13 represent the value of the algebraic connectivity over time using the enhanced and stanrdard connectivity maintenance control action, respectively. As expected, the connectivity of the communication graph is always preserved, in both cases.

To compare the two control strategies, Fig. 4.15 represents the average of the absolute value of the connectivity maintenance control action over time, computed over all the agents, that is $\bar{u}^c = \sum_{i=1}^{N} |u_i^c|$. It can be noticed that the introduction of the selective action drastically reduces the number of times the connectivity maintenance control law



(a) Selective control action (b) Standard control action

Figure 4.15: Average over all the agents of the absolute value of the connectivity maintenance control action: with the selective action, the connectivity maintenance control action is often equal to zero

is activated. Indeed, it significantly reduces the interference with the primary task of the multi–robot system, namely the formation control and the obstacle avoidance actions.

In order to carry out a quantitative analysis of the advantage introduced by the selective action, a measurement of the required *control effort* may be defined as the area underneath the curves represented in Fig. 4.15. Data have been acquired during 50 runs of simulations, performed within random setups: initial positions of the agents, as well as the obstacles' positions, have been randomly varied. For each setup, both the standard and the enhanced control law have been implemented. From the statistical analysis of the acquired data, it turns out that the introduction of the selective action drastically reduces the required control effort. In fact, the effort is reduced, on average, by 63.44%, with a standard deviation of 24.85%.

The only drawback in the introduction of the selective action is a slight increase in the instantaneous effort, which can be explained by the discontinuous nature of the selective control action.

Connectivity maintenance for Lagrangian dynamical systems

Matlab simulations have been carried out to validate the connectivity maintenance control strategy for groups of Lagrangian dynamical systems as well. As in the previous examples, the number of agents has been varied, from $N = 3$ to $N = 20$, and they have been placed them in randomly chosen initial positions.

A group of six Lagrangian agents have been simulated during a rendezvous task: six

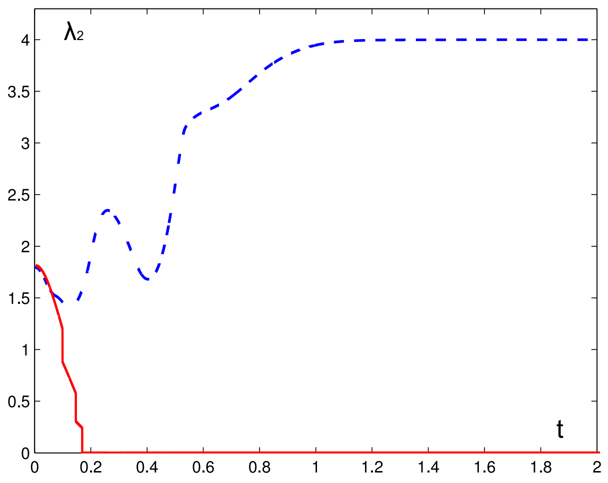


Figure 4.16: Rendezvous for groups of Lagrangian systems: value of λ_2 *with* (blue dashed line) and *without* (red solid line) the connectivity maintenance controller

agents that, starting from random initial positions, were supposed to converge to a common point, while avoiding collisions with randomly placed point obstacles. For obstacle avoidance purposes, a repulsive artificial potential field has been added (see e.g. [111]). As shows in Fig. 4.16, without the connectivity maintenance controller, the connectivity is lost quite soon: the obstacle avoidance action obstructs the desired movement of some agents, that are thus trapped and lose connectivity with the other ones. As expected, using the connectivity maintenance control action the connectivity of the graph is always preserved.

4.5.2 Experiments

Experiments on real robots have been performed within the MORE–pucks experimental platform described in Chapter 2, with group of four E–puck robots [34] moving in a $2.0m \times 1.5m$ arena.

E–puck robots can be described by the differential–drive kinematic model:

$$\begin{cases} \dot{x}_i &= u_i \cos(\phi_i) \\ \dot{y}_i &= u_i \sin(\phi_i) \\ \dot{\phi}_i &= \omega_i \end{cases} \tag{4.131}$$

Experiments have been carried out to evaluate the performance of the connectivity maintenance control algorithm, that has been actually developed for single integrator kinematic agents. To deal with the fact that this model represents a nonholonomic system, the feedback linearization technique presented in [89] has been applied.

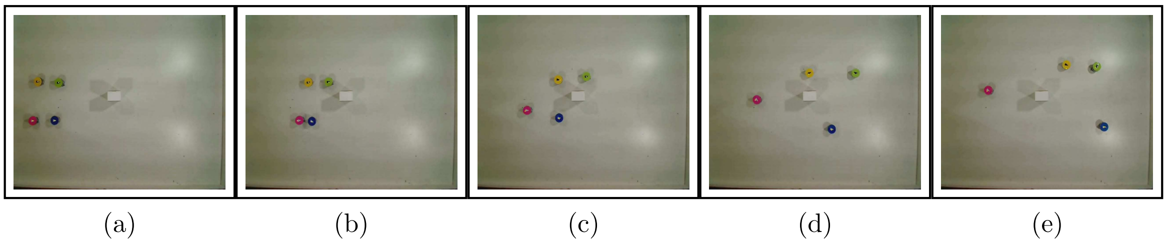


Figure 4.17: MORE–pucks experiment, formation control application, WITHOUT connectivity maintenance control strategy

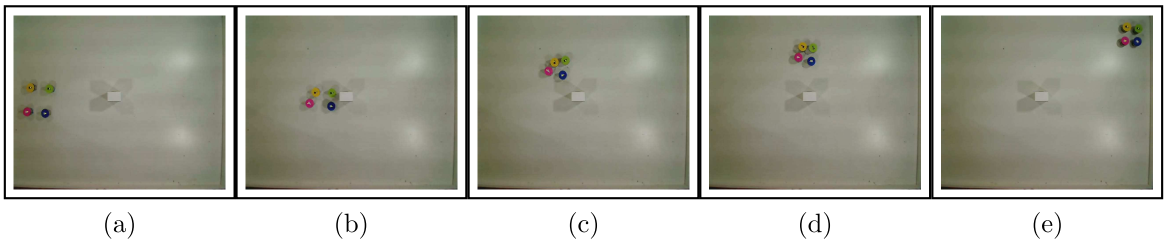


Figure 4.18: MORE–pucks experiment, formation control application, WITH connectivity maintenance control strategy

The connectivity maintenance algorithm has been tested both for rendezvous and formation control applications: in both cases, as expected, the connectivity is always preserved.

Snapshots of a formation control experiments are shown in Figs. 4.17 and 4.18: robots are supposed to create a formation and move through the arena, while avoiding collisions with the obstacle. Fig. 4.17 shows the behavior of the system without the connectivity maintenance control strategy. Conversely, Fig. 4.18 shows the behavior of the system with the connectivity maintenance control strategy. When the obstacle avoidance action is activated, without the connectivity maintenance controller the connectivity is lost: the obstacle avoidance action obstructs the desired movement of one of the robots, that is thus trapped and loses connectivity with the other ones. Conversely, as expected, using the connectivity maintenance control action the connectivity of the graph is always preserved.

4.6 Discussion

In this chapter,a control algorithm has been described that, by means of decentralized estimation of the algebraic connectivity of the communication graph, ensures the maintenance of the connectivity among a group of robots, for any initial condition. This control strategy was first introduced in [29–32].

Analytical proofs have been provided that, by means of this control strategy, the value of the algebraic connectivity of the graph, i.e. λ_2, is bounded from zero, and then the graph is connected. Connectivity maintenance in the presence of estimation errors has been formally proved to be guaranteed. The control strategy has been demonstrated to be effective in the presence of an external (bounded) controller as well.

This connectivity maintenance control strategy has been initially developed for single integrator kinematic agents, and has then be extended for Lagrangian dynamical agents. This extension was first introduced in [33].

Simulations and experiments have been carried out as well, for validation purposes.

Throughout the chapter, several upper–bounds have been defined, some of which depend on the number N of robots in the group. If N is a variable number, i.e. the number of robots can change (e.g. because robots can be added or removed), the actual value of N can be substituted with an upper–bound, given by the maximum number of agents that can be available.

In order to improve the applicability of this control strategy, tighter bounds may be found, in order to make the control strategy less conservative. Hence, it may be feasible to provide a constructive procedure to define the smallest possible bound $\tilde{\epsilon}$ that ensures connectivity maintenance.

To further improve the scope of this connectivity maintenance algorithm, sensing might be considered, instead of explicit communication. In fact, often mobile robots do not communicate explicitly, but they acquire information about the other ones by means of exteroceptive sensors. In this case, the fact that the i–th agent can acquire information from the j–th one does not imply the converse. This communication architecture can be modeled by means of a directed graph.

Chapter 5

Concluding remarks

This thesis deals with distributed control strategies for cooperative control of multi–robot systems. The main results are summarized hereafter.

Specifically, **Chapter 3** describes some results obtained in the field of coordinated motion control strategies. Initially, artificial potential fields are defined for formation control purposes: following the negative gradient of some specifically designed potential fields, robots are driven to create a regular polygon formation. A bijective coordinate transformation is then exploited for obtaining completely arbitrarily shaped formations. This control strategy, first introduced in [21–24], is proved to be asymptotically stable and local minimum free.

Artificial potential fields are subsequently used to solve the coordinated path tracking problem. First introduced in [25, 26], a potential based control strategy is defined to make a group of mobile robots track a path given by an arbitrarily shaped desired curve. This control strategy is a completely decentralized algorithm, since there is no need for any centralized controller or global synchronization.

Formation control problem is then solved exploiting a graph theory based approach. Specifically, as described in [27, 28], weighted graphs are used to drive a group of robots to a predefined configuration while avoiding mutual collisions, by means of a consensus based algorithm. An appropriate edge–weight function has been defined that provably guarantees the convergence to the desired formation, as well as the avoidance of collisions among the robots. The framework is extended for accomplishing the avoidance of collisions among robots and obstacles as well. This control strategy has been experimentally shown to be robust to the presence of communication delays.

Finally, **Chapter 4** deals with global connectivity maintenance. Specifically, as described in [29–32], an estimation procedure is introduced to allow each agent to compute its own estimate of the algebraic connectivity of the communication graph, in a distributed manner. This estimate is then exploited to develop a gradient based control strategy, to ensure the algebraic connectivity of the communication graph always remains positive, as the system evolves. The proposed strategy is implemented initially for single–integrator kinematic agents, and is then extended to Lagrangian dynamical systems, as shown in [33]. The presence of additional external control laws is considered as well.

Appendix A

Background on graph theory

This Appendix provides a brief overview of the main notions on graph theory used throughout the thesis. Since a detailed description of graph theory is out of the scope of this thesis, a few notions will be listed, without going into details. Specifically, no proofs will be provided. Details on these and other notions can be found, for instance, in [18, 112, 113] and references therein.

The main reason why graph theory is considered in this work is the following. Given N mobile robots, the communication architecture among them may be described as a graph. Generally speaking, a graph $\mathcal{G}$ represents the interconnection among a set of *nodes*: if two nodes are interconnected, and *edge* exists among them. The *neighborhood* of a node is defined as the set of its *neighbors*, that is the set of nodes to whom it is connected through an edge.

Hence, in multi–robot systems, each robot is represented as a node of the graph, and the link between two robots is represented as an edge of the graph. In order to represent the communication architecture in multi–robot systems, two different classes of graphs may be adopted: directed graphs and undirected graphs.

- In an *undirected graph* the information exchange is bidirectional: for every couple of nodes i and j, if the $i \to j$ edge exists, then the $j \to i$ edge exists as well. Undirected graphs are thus usually exploited to model explicit bidirectional communication among the robots.

- In a *directed graph* the information exchange is unidirectional: for every couple of nodes i and j, the fact that the $i \to j$ edge exists doesn't automatically imply the existence of the $j \to i$ edge. Directed graphs are thus usually exploited to model unidirectional communication among the robots, that may be based on pure sensing.

Hereafter, undirected graphs will be considered: throughout the thesis, in fact, the possibility for the robots to exploit direct communication has been often assumed.

Hence, the fact that the graph is undirected means that, if the i–th robot can acquire information from the j–th one, the j–th robot can acquire information from the i–th one as well. Let $\mathcal{N}_i$ be the neighborhood of the i–th robot, i.e. the set of robots that can exchange information with the i–th one. The communication graph can be described by means of the adjacency matrix $A \in \mathbb{R}^{N \times N}$. Each element a_{ij} is defined as the weight of the edge between the i–th and the j–th robot, and is a positive number if $j \in \mathcal{N}_i$, zero otherwise. Since undirected graphs are considered, it is possible to assume $a_{ij} = a_{ji}$. The degree matrix of the graph is defined as

$$D = \operatorname{diag}\left(\{d_i\}\right) \tag{A.1}$$

where d_i is the degree of the i–th node of the graph, i.e. $d_i = \sum_{j=1}^{N} a_{ij}$.

The (weighted) Laplacian matrix of the graph is defined as:

$$L = D - A \tag{A.2}$$

The unweighted Laplacian matrix, L_*, is defined as a special case of Laplacian matrix, where all non–zero entries of the adjacency matrix are equal to one.

The Laplacian matrix exhibits some remarkable properties:

1. Let $\mathbf{1}$ be the column vector of all ones. Then, $L\mathbf{1} = \mathbf{0}$.

2. Let λ_i, $i = 1, \ldots, N$ be the eigenvalues of the Laplacian matrix.

   - $\lambda_i \in \mathbb{R}$, $\forall i = 1, \ldots, N$.
   - The eigenvalues can be ordered such that

   $$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \tag{A.3}$$

   - $\lambda_2 > 0$ if and only if the graph is connected. For this reason, λ_2 is defined as the algebraic connectivity of the graph [97].

From the properties described above, it follows that, if the graph is connected, then $\ker(L) = \operatorname{span}(\mathbf{1})$.

Consensus

The *consensus problem* [19] is a well–known and widely studied problem in the field of decentralized control. In networks of agents (or dynamic systems), consensus means to reach an agreement regarding a certain quantity of interest that depends on the state of all agents. Generally speaking, a consensus algorithm is an interaction rule that specifies the information exchange between an agent and all its neighbors.

Consider a group of N single integrator kinematic agents:

$$\dot{z}_i = u_i \quad n = 1, \ldots, N \tag{A.4}$$

where $z_i \in \mathbb{R}$ is the state of the i-th agent. To solve the consensus problem, that is driving all the state variables to a final common value, it is possible to exploit a distributed feedback interconnection, defined as follows:

$$\dot{z}_i = -\sum_{j \in \mathcal{N}_i} w_{ij}\left(z\right)\left(z_i - z_j\right) \tag{A.5}$$

where $w_{ij}\left(x\right)$ are positive edge weight functions.

Let $\mathcal{E}$ be the edge set of the graph $\mathcal{G}$, that is $(i, j) \in \mathcal{E}$ if an edge connects node i and node j. According to this definition, the edge weights exhibit the following property:

$$w_{ij} \neq 0 \text{ if and only if } (i, j) \in \mathcal{E} \tag{A.6}$$

Consider, without loss of generality, a graph with M edges, and let $w \in \mathbb{R}^M$ be the vector containing all the non–zero edge weights of the graph. Hence, the *weight matrix* $W\left(z\right)$ may be defined as follows:

$$W\left(z\right) = \operatorname{diag}\left(w\right) \in \mathbb{R}^{M \times M} \tag{A.7}$$

Let $\mathcal{I} = \left[\iota_{ij}\right] \in \mathbb{R}^{N \times M}$ be the *incidence matrix* of the graph $\mathcal{G}$, defined as follows:

$$\iota_{ij} \begin{cases} 1 & \text{if } (i, j) \in \mathcal{E} \\ -1 & \text{if } (j, i) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{A.8}$$

According to [67], a random orientation of the edges can be considering, when dealing with undirected graphs. Given this definition of the incidence matrix, the Laplacian matrix of the graph $\mathcal{G}$ may be defined with the following alternative formulation [67]:

$$L_* = \mathcal{I} \cdot \mathcal{I}^T \tag{A.9}$$

As shown in [67], the weighted Laplacian matrix may be defined as follows:

$$L = \mathcal{I} \cdot W \cdot \mathcal{I}^T \tag{A.10}$$

Let $z = [z_1, \ldots, z_N]$. The control law in Eq. (A.5) can be rewritten in the following matrix form:

$$\dot{z} = -Lz \tag{A.11}$$

So far, only scalar states have been considered. Consider now the position of each agent as its own state. More specifically, if the position of the i–th agent is n–dimensional, the i–th agent's state is given by $x_i = [x_{i,1}, \ldots, x_{i,n}]^T$. Considering N agents, it is possible to define $x = \left[x_1^T, \ldots, x_N^T\right]^T$.

Therefore, to apply the graph based algorithms defined so far to the multi–dimensional case, the component–wise operator, defined in [67], may be exploited:

$$c\left(x, j\right) = \left(x_{1,j}, \ldots, x_{N,j}\right)^T \in \mathbb{R}^N \quad j = 1, \ldots, n \tag{A.12}$$

The component–wise operator can be then introduced in the control law in Eq. (A.11):

$$\frac{d}{dt} c\left(x, j\right) = -Lc\left(x, j\right) \qquad j = 1, \ldots, n \tag{A.13}$$

The control law in Eq. (A.13) can be rewritten in vector form [67] as follows:

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} w_{ij}\left(x\right)\left(x_i - x_j\right) \tag{A.14}$$

Bibliography

[1] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[2] A. Davids. **Urban search and rescue robots: from tragedy to technology**. *IEEE Intelligent Systems*, **17**(2):81–83, march–april 2002.

[3] Yo. Mei, Y.–H. Lu, Y. C. Hu, and C. S. G. Lee. **Deployment of mobile robots with energy and timing constraints**. *IEEE Transactions on Robotics*, **22**(3):507– 522, june 2006.

[4] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta. **Automatic Deployment of Robotic Teams**. *IEEE Robotics Automation Magazine*, **18**(3):75–86, sept. 2011.

[5] A. Yamashita, T. Arai, J. Ota, and H. Asama. **Motion planning of multiple mobile robots for Cooperative manipulation and transportation**. *IEEE Transactions on Robotics and Automation*, **19**(2):223–237, apr 2003.

[6] M. Hara, M. Fukuda, H. Nishibayashi, Y. Aiyama, J. Ota, and T. Arai. **Motion control of cooperative transportation system by quadruped robots based on vibration model in walking**. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, **3**, pages 1651–1656 vol.3, 1999.

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. **Coordinated multi–robot exploration**. *IEEE Transactions on Robotics*, **21**(3):376–386, june 2005.

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

[9] J. Y. Lee and H. Choset. **Sensor–based exploration for convex bodies: a new roadmap for a convex–shaped robot**. *IEEE Transactions on Robotics*, **21**(2):240–247, april 2005.

[10] J. Sung, H. I. Christensen, and R. E. Grinter. **Sketching the future: Assessing user needs for domestic robots**. In *The 18th IEEE International Symposium on Robot and Human Interactive Communication (RO–MAN 2009)*, pages 153–158, oct. 2009.

[11] F: Yuan, L. Twardon, and M. Hanheide. **Dynamic path planning adopting human navigation strategies for a domestic mobile robot**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3275–3281, oct. 2010.

[12] P. Wurman, R. D'Andrea, and M. Mountz. **Coordinating hundreds of cooperative, autonomous vehicles in warehouses**. *AI Magazine*, **29**(1):9–20, 2008.

[13] D. Ronzoni, R. Olmi, C. Secchi, and C. Fantuzzi. **AGV global localization using indistinguishable artificial landmarks**. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 287–292, may 2011.

[14] N. Wu and M. Zhou. **Deadlock Resolution in Automated Manufacturing Systems With Robots**. *IEEE Transactions on Automation Science and Engineering*, **4**(3):474–480, july 2007.

[15] C. R. Weisbin, J. Blitch, D. Lavery, E. Krotkov, C. Shoemaker, L. Matthies, and G. Rodriguez. **Miniature robots for space and military missions**. *IEEE Robotics Automation Magazine*, **6**(3):9–18, sep 1999.

[16] R. Madhavan. **Robots in Military and Aerospace Technologies [News and Views]**. *IEEE Robotics Automation Magazine*, **17**(2):6, june 2010.

[17] N. Leonard and E. Fiorelli. **Virtual leaders, artificial potentials and coordinated control of groups**. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2968–2973, 2001.

[18] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.

[19] 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.

[20] L. Sabattini, C. Secchi, C. Fantuzzi, and A. Stefani. **Bird's–eye view image for the localization of a mobile robot by means of trilateration**. In *Proceedings of the IFAC SYmposium on Intelligent Autonomous Vehicles*, 2010.

[21] L. Sabattini, C. Secchi, and C. Fantuzzi. **Arbitrarily shaped formations of mobile robots: artificial potential fields and coordinate transformation**. *Autonomous Robots*, **30**(4):385–397, may 2011.

[22] L. Sabattini, C. Secchi, and C. Fantuzzi. **Potential based control strategy for arbitrary shape formations of mobile robots**. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3762–3767, oct. 2009.

[23] L. Sabattini, C. Secchi, and C. Fantuzzi. **Potential Based Control Strategy for Abritrary Shape Formations of Mobile Robots**. In *Convegno SIDRA, Siracusa*, 2009. Poster.

[24] L. Sabattini, C. Secchi, and C. Fantuzzi. **Formation Control and Obstacle Avoidance**. In *Convegno SIDRA, Vicenza*, 2008. Poster.

[25] L. Sabattini, C. Secchi, C. Fantuzzi, and D. de Macedo Possamai. **Tracking of closed-curve trajectories for multi-robot systems**. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 6089–6094, oct. 2010.

[26] L. Sabattini, C. Secchi, and C. Fantuzzi. **Closed–Curve Path Tracking for Decentralized Systems of Multiple Mobile Robots**. *Journal of Intelligent and Robotic Systems*, 2012. (Submitted).

[27] R. Falconi, L. Sabattini, C. Secchi, C. Fantuzzi, and C. Melchiorri. **A Graph–Based Collision–Free Distributed Formation Control Strategy**. In *Proceedings of the IFAC World Congress*, 2011.

[28] R. Falconi, L. Sabattini, C. Secchi, C. Fantuzzi, and C. Melchiorri. **Edge–Weighted Consensus Based Formation Control Strategy With Collision Avoidance**. *Robotics and Autonomous Systems*, 2012. (Submitted).

[29] L. Sabattini, N. Chopra, and C. Secchi. **On Decentralized Connectivity Maintenance for Mobile Robotic Systems**. In *Proceedings of the IEEE Conference on Decision and Control*, 2011.

[30] L. Sabattini, N. Chopra, and C. Secchi. **Distributed control of multi-robot systems with global connectivity maintenance**. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2321–2326, sept. 2011.

[31] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri. **Distributed Control of Multi–Robot Systems with Global Connectivity Maintenance**. *IEEE Transactions on Robotics*, 2012. (Submitted).

[32] L. Sabattini, A. Gasparri, C. Secchi, and N. Chopra. **Enhanced Connectivity Maintenance for Multi–Robot Systems**. In *IFAC Symposium on Robot Control*, 2012. (Submitted).

[33] L. Sabattini, C. Secchi, and N. Chopra. **Decentralized Connectivity Maintenance for Networked Lagrangian Dynamical Systems**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012.

[34] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. C. Zufferey, D. Floreano, and A. Martinoli. **The e-puck, a Robot Designed for Education in Engineering.** In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

[35] **The Player Project**. http://playerstage.sourceforge.net/.

[36] **ROS.org**. http://www.ros.org/.

[37] M. J. Matarić, N. Koenig, and D. Feil–Seifer. **Materials for Enabling Hands–On Robotics and STEM Education**. In *AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, 2007.

[38] J. Borenstein and L. Feng. **Measurement and Correction of Systematic Odometry Errors in Mobile Robots**. *IEEE Transactions On Robotics and Automation*, pages 869–880, 12 1996.

[39] A. T. Samiloglu, Ö. Çayrpunar, V. Gazi, and A. Buğra Koku. **An Experimental Set–up For Multi-Robot Applications**. In *Workshop Proceedings of SIMPAR 2008 International Conference on Simulation, Modeling and Programming for Autonomous Robots*, pages 539–550, 2008.

[40] G. Bradski and A. Kaehler. *Learning OpenCV – Computer Vision with the OpenCV library.* O'Reilly, 2008.

[41] **Qt – A cross–platform application and UI framework**. http://qt.nokia.com/.

[42] R. C. Gonzales and R. E. Woods. *Digital Image Processing 3rd Ed.* Prentice Hall, 2008.

[43] C. Harris and M. Stephens. **A combined corner and edge detector**. In *147–151*, page Alvey vision conference, 1988.

[44] **Boost C++ libraries**. http://www.boost.org/.

[45] **e–puck educational robot**. http://www.e-puck.org/.

[46] **GNU General Public License, Version 3**. http://www.gnu.org/licenses/gpl.html, 2007.

[47] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J. L. Deneuborg, and M. Dorigo. **The cooperation of swarm–bots: physical interactions in collective robotics**. *IEEE Robotics & Automation Magazine*, pages 21–28, 2005.

[48] T. Balch and R. C. Arkin. **Behavior–Based Formation Control for Multirobot Teams**. *IEEE Transactions on Robotics and Automation*, pages 926–939, 1998.

[49] S. W. Ekanayake and P. N. Pathirana. **Artificial Formation Forces for Stable Aggregation of Multi–Agent System**. In *International Conference on Information and Automation*, pages 129–134, 2006.

[50] M. Lindhé, P. Ögren, and K. H. Johansson. **Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1785–1790, 2005.

[51] Y. Liu and K. M. Passino. **Stable Social Foraging Swarm in a Noisy Environment**. *IEEE Transactions on Automatic Control*, pages 30–44, 2004.

[52] T. Balch and M. Hybinette. **Social Potentials for Scalable Multi–Robot Formations**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 73–80, 2000.

[53] R. Bachmayer and N. E. Leonard. **Vehicle Networks for Gradient Descent in a Sampled Environment**. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 112–117, 2002.

[54] L. Chaimowicz, N. Michael, and V. Kumar. **Controlling swarms of robots using interpolated implicit functions**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2487–2492, 2005.

[55] M. A. Hsieh and V. Kumar. **Pattern generation with multiple robots**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2442–2447, 2006.

[56] L. S. Marcolino and L. Chaimowicz. **No Robot Left Behind: Coordination to Overcome Local Minima in Swarm Navigation**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1904–1909, 2008.

[57] L. Barnes, M. A. Fields, and K. Valavanis. **Unmanned Ground Vehicle Swarm Formation Control Using Potential Fields**. In *Mediterranean Conference on Control and Automation*, pages 1–8, 2007.

[58] M. Egerstedt, X. Hu, and A. Stotsky. **Control of Mobile Platforms Using a Virtual Vehicle Approach**. *IEEE Transactions On Automatic Control*, 11 2001.

[59] O. Khatib. **Real–time Obstacle Avoidance For Manipulators and Mobile Robots**. *The International Journal of Robotics Research*, 1986.

[60] M. T. Wolf and J. W. Burdick. **Artificial Potential Functions for Highway Driving with Collision Avoidance**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.

[61] S. Jang, G. Song, and S. K. Hong. **Dynamic Boundary Tracking in Active Sensor Networks**. In *Proceedings of the International Conference on Control, Automation and Systems*, 2007.

[62] Z. Jin and A. L. Bertozzi. **Environmental Boundary Tracking and Estimation Using Multiple Autonomous Vehicles**. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[63] Y. Cao and R. Fierro. **Dynamic Boundary Tracking Using Dynamic Sensor Nets**. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.

[64] M. A. Hsieh, S. Loizou, and V. Kumar. **Stabilization of Multiple Robots on Stable Orbits via Local Sensing**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2312–2317, 2007.

[65] S. Susca, F. Bullo, and S. Martínez. **Synchronization of Beads on a Ring**. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[66] C. Secchi and C. Fantuzzi. **Formation control over delayed communication networks**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 563–568, 2008.

[67] M. Ji and M. Egerstedt. **Distributed Coordination Control of Multiagent Systems While Preserving Connectedness**. *IEEE Transactions on Robotics*, 2007.

[68] D. V. Dimarogonas and K. H. Johansson. **Stability analysis for multi–agent systems using the incidence matrix: quantized communication and formation control**. *Automatica*, 2010.

[69] J. A. Fax and R. M. Murray. **Information Flow and Cooperative Control of Vehicle Formations**. *IEEE Transactions on Automatic Control*, pages 1465–1476, 2004.

[70] W. Ren, R. Beard, and E. Atkins. **Information consensus in multivehicle cooperative control**. *IEEE Control Systems Magazine*, **27**(2), 2007.

[71] R. Olfati–Saber. **Flocking for multi–agent dynamic systems: algorithms and theory**. *IEEE Transactions on Automatic Control*, **51**:401–420, 2006.

[72] T. H. Summers, C. Yu, and B. D. O. Anderson. **Robustness to agent loss in vehicle formations and sensor networks**. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1193–1199, 2008.

[73] R. Falconi, S. Gowal, and A. Martinoli. **Graph Based Distributed Control of Non-Holonomic Vehicles Endowed with Local Positioning Information Engaged in Escorting Missions**. In *IEEE Conf. on Robotics and Automation (ICRA 2010)*, 2010.

[74] F. Han, T. Yamada, K. Watanabe, K. Kiguchi, and K. Izumi. **Construction of an Omnidirectional Mobile Robot Platform Based on Active Dual–Wheel Caster Mechanisms and Development of a Control Simulator**. *Journal of Intelligent and Robotic Systems*, pages 257–275, 11 2000.

[75] G. Oriolo, A. De Luca, and M. Vendittelli. **WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation**. *IEEE Transactions On Control Systems Technology*, pages 835–852, 11 2002.

[76] B. E. Meserve. *Fundamental Concepts of Geometry*. Courier Dover Publications, 1983.

[77] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

[78] K. D. Do. **Formation Tracking Control of Unicycle–Type Mobile Robots With Limited Sensing Ranges**. *IEEE Transactions on Control Systems Technology*, **16**:527–538, 2008.

[79] L. Piege and W. Tiller. *The NURBS Book*. Springer–Verlag, 1995-1997.

[80] A. Tsalatsanis, A. Yalcin, and K.P. Valavanis. **Optimized task allocation in cooperative robot teams**. In *Proceedings of the IEEE Mediterranean Conference on Control and Automation*, pages 270–275, 2009.

[81] B. B. Choudhury and B. B. Biswal. **An Optimized Multirobot Task Allocation**. In *Proceedings of the First International Conference on Emerging Trends in Engineering and Technology*, pages 320–325, 2008.

[82] J. Bellingham, M. Tillerson, A. Richards, and J. P. How. **Multi–task allocation and path planning for cooperative UAVs**. *Cooperative Control: Models, Applications, and Algorithms*, pages 23–41, 2003.

[83] C. Schumacher, P. Chandler, and S. Rasmussen. **Task allocation for wid area search munition**. In *Proceedings of the American Control Conference*, pages 1917–1922, 2002.

[84] R. W. Beard and V. Stepanyan. **Synchronization of information in distributed multiple vehicle coordinated control**. In *In Proceedings of IEEE Conference on Decision and Control*, pages 2029–2034, 2003.

[85] H. L. Choi, L. Brunet, and J. P. How. **Consensus–Based Decentralized Auctions for Robust Task Allocation**. *IEEE Transactions on Robotics*, **25**(4):912–926, 2009.

[86] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. **Market–Based Multirobot Coordination: A Survey and Analysis**. *Proceedings of the IEEE*, **97**(7):1257–1270, 2006.

[87] T. Mercker, D. W. Casbeer, P. T. Millet, and M. R. Akella. **An extension of consensus–based auction algorithms for decentralized, time–constrained task assignment**. In *Proceedings of the American Control Conference*, pages 6324–6329, 2010.

[88] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[89] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, London, UK, 2009.

[90] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie. **Maintaining limited–range connectivity among second–order agents**. In *Proceedings of the American Control Conference*, pages 2134–2129, 2006.

[91] Y. Cao and W. Ren. **Distributed coordinated tracking via a variable structure approach – Part I: consensus tracking. Part II: swarm tracking**. In *Proceedings of the American Control Conference*, pages 4744–4755, 2010.

[92] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Talyor. **Maintaining Network Connectivity and Performance in Robot Teams**. *Journal of Field Robotics*, **25**(1):111–131, 2008.

[93] A. Ajorlou, A. Momeni, and A. G. Aghdam. **A Class of Bounded Distributed Control Strategies for Connectivity Preservation in Multi–Agent Systems**. *IEEE Transactions on Automatic Control*, **55**:2828–2833, 2010.

[94] D. V. Dimarogonas and K. H. Johansson. **Bounded control of network connectivity in multi–agent systems**. *IET Control Theory & Applications*, **4**:1751–8644, 2010.

[95] F. Morbidi, A. Giannitrapani, and D. Prattichizzo. **Maintaining connectivity among multiple agents in cyclic pursuit: A geometric approach**. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 7461–7466, 2010.

[96] G. Hollinger and S. Singh. **Multi–robot coordination with periodic connectivity**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4457–4462, 2010.

[97] M. Fielder. **Algebraic connectivity of graphs**. *Czechoslovak Mathematical Journal*, **23**(98):298–305, 1973.

[98] M. M. Zavlanos, H. G. Tanner, A. Jadbabaie, and G. J. Pappas. **Hybrid Control for Connectivity Preserving Flocking**. *IEEE Transactions on Automatic Control*, **54**:2869–2875, 2009.

[99] M. M. Zavlanos and G. J. Pappas. **Potential fields for maintaining connectivity of mobile networks**. *IEEE Transactions on Robotics*, **23**(4):812–816, 2007.

[100] M. C. De Gennaro and A. Jadbabaie. **Decentralized Control of Connectivity for Multi–Agent Systems**. In *Proceedings of the IEEE International Conference on Decision and Control*, page 3628, 2006.

[101] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar. **Decentralized estimation and control of graph connectivity for mobile sensor networks**. *Automatica*, **46**:390–396, 2010.

[102] S. Fallat and S. Kirkland. **Extremizing algebraic connectivity subject to graph theoretic constraints**. *The Electronic Journal of Linear Algebra*, **3**:48–74, 1998.

[103] Y. Kim and M. Mesbahi. **On maximizing the second smallest eigenvalue of a state–dependent graph Laplacian**. In *Proceedings of the IEEE American Control Conference*, pages 99–103, 2005.

[104] S. Bhattacharya and T. Basar. **Graph–theoretic approach for connectivity maintenance in mobile networks in the presence of a jammer**. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 3560–3565, 2010.

[105] L. N. Trefthen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[106] R. A. Freeman, P. Yang, and K. M. Lynch. **Stability and convergence properties of dynamic consensus estimators**. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 338–343, 2006.

[107] R. Olfati–Saber and R. M. Murray. **Consensus problems in networks of agents with switching topology and time–delays**. *IEEE Transactions on Automatic Control*, **9**:1520–1533, 2004.

[108] R. Ortega, L. Perez, P. J. Nicklasson, and H. Sira–Ramirez. *Passivity–based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*. Springer London, 1998.

[109] R. Kristiansen, P. J. Nicklasson, and J. T. Gravdahl. **Formation Modeling and 6DOF Spacecraft Coordination Control**. In *Proceedings of the IEEE American Control Conference*, pages 4690–4696, 2007.

[110] G. B. Arfken and H.–J. Weber. *Mathematical methods for physicists.* Elsevier, 2005.

[111] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. **Stable flocking of mobile agents, part I: fixed topology**. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2010–2015, 2003.

[112] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics).* Springer, 2005.

[113] N. Biggs, E. Lloys, and R. Wilson. *Graph Theory.* Oxford University Press, 1986.