LONG-DURATION ROBOT AUTONOMY: FROM CONTROL ALGORITHMS TO ROBOT DESIGN

A Dissertation
Presented to
The Academic Faculty

By

Gennaro Notomista

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology

August 2020

LONG-DURATION ROBOT AUTONOMY: FROM CONTROL ALGORITHMS TO ROBOT DESIGN

Approved by:

Dr. Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Anirban Mazumdar
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Wayne Book
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mac Schwager
Department of Aeronautics and Astronautics
Stanford University

Date Approved: July 20, 2020

"Si è sempre meridionali di qualcuno."

— Luciano De Crescenzo, *Così Parlò Bellavista*

To my family

ACKNOWLEDGEMENTS

The work presented in this dissertation would have been very different if not even impossible if not for the support, contributions, discussions, and conversations with a number of people with whom I had the pleasure of working, talking, and sharing a big part of my life during the past four years.

First and foremost, I am deeply honored of having had the opportunity of working with my advisor, Dr. Magnus Egerstedt. His amazing enthusiasm pushed me through the fantastic journey of my PhD, and his philosophy and vision of research—sometimes orthogonal to mine—contributed to broaden my energy spectrum and allowed me to self-propel till the finish line. Thanks to his guidance and advice, I feel I experienced one of the most profound personal growth which will be reflected on my entire future life.

I would like to acknowledge the work of my PhD proposal and thesis committee, comprised of Dr. Wayne Book, Dr. Samuel Coogan, Dr. Seth Hutchinson, Dr. Anirban Mazumdar and Dr. Mac Schwager. I really appreciated their advice that contributed to significantly improve the focus and the quality of this dissertation. In particular, I would like to mention the availability of Dr. Schwager in hosting me one month at Stanford during the Summer 2019: it was a magnificent experience under many different perspectives.

During my stay at Georgia Tech, I had the pleasure of interacting with so many extremely interesting and sharp minds. Listed chronologically and among many others, I will never forget:

- Dr. Erik Verriest's monoids (and bees)
- Dr. Eric Feron's voice, which—I almost don't believe myself saying it—it was missed after he moved to even warmer places of Earth
- Dr. Chris Heil's embarrassment at prooving a triviality like Fatou's lemma, and his passion for math jokes

- Dr. Yorai Wardi's love for good food, good weather, and politics of every single nation ever existed on Earth, as well as his office, which often became a Spirit and Time Chamber and which housed some of the most exciting discussions on control theory I've ever had

- Dr. Seth Hutchinson's smiles and apparently innocent questions during all the meetings we've had, accompanied by songs of always different genre—from classical, to jazz, to soft rock, to Japanese songs for marimba

- Dr. Matthieu Bloch's struggle to convince me that information theory has got any use after all

- Dr. Azadeh Ansari's shared unconditional passion for the brushbots

The journey of my PhD wouldn't have been this exciting without the former and current members of the extended GRITS Lab that I met and shared such a big part of my life with. I will always remember:

- Daniel Pickem, the first person I met in the lab, who taught me that it's possible to do control theory and robot design at the same time

- the first research project and paper with Sebastian Ruf

- the dozens of Chole Bhature I won and lost to Sid Mayya betting on whether the new brushbot design is going to move or the GitHub servers will allow him to push his entire PDF library to an online repository; I'll never forget that, sometimes, electronics can be as exciting as mechanics!

- the voice of Maria Santos—again I don't believe myself saying it—and the papers (and the PhD thesis too!) submitted at the same exact nanoseconds, just a few millisecond before deadlines expired

- the passion and dedication to work of Li Wang

- Paul Glotfelter, who taught me that also control theory and software engineering can be done at the same time
- Ian Buckley's philosophy of coffee and the car trip to reach the first lab retreat: What in tarnation?
- the energy and enthusiasm of Yousef Emam in the late nights and early morning during which we tried to make the SlothBot move, and the several exciting journeys to the Atlanta Botanical Garden
- Anqi Li, her passion for sloths, and all the math courses we took together
- Chris Banks's PC font size, and his turtle participating to all the online meetings of the last few months of quarantine
- Pietro Pierpaoli and Sean Wilson, from whose very different views on robotics research I've learned a great deal
- the late nights spent in Atlanta together with Jeremy Cai making joint experiments with Tokyo Tech work
- Motoya Ohnishi, Federico Celi, Riku Funada, Alessia Benevento, Tatsuya Miyano, and Tatsuya Ibuki, who visited the GRITS Lab, making it an even more vibrant working place
- all the members of the FACTS Lab, Keith Paarporn and Mark Mote
- the passion and efficiency for work of Amber Burke

I would like to thank once more Dr. Book—this time not for academic-related reasons—Mrs. Book, Tad and Danielle Book, who treated my wife, Gabriella, and me like members of their family and let us join all the Easter, Halloween and Thanksgiving celebration we spent in Atlanta. These will be really unforgettable memories.

And last but not least, my most profound thank you goes to my family, my mother, my father and my sister Angelica, who, from too far away—this time, from the other side of the Atlantic Ocean—have always been there, supporting all the choices I made in my life, including the one of pursuing a PhD. None of my accomplishments would have been possible without you. Finally, the greatest thank you of all goes to my wife, Gabriella, who followed me in this journey, supported me in the everyday life, helping me overcome the difficult times during my PhD, and reminding me that there is life beyond robotics. No words will ever be enough: I love you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The transition that robots are experiencing from controlled and often static working environments to unstructured and dynamic settings is unveiling the potential fragility of the design and control techniques employed to build and program them, respectively. A paramount of example of a discipline that, by construction, deals with robots operating under unknown and ever-changing conditions is *long-duration robot autonomy*. In fact, during long-term deployments, robots will find themselves in environmental scenarios which were not planned and accounted for during the design phase. These operating conditions offer a variety of challenges which are not encountered in any other discipline of robotics.

This thesis presents control-theoretic techniques and mechanical design principles to be employed while conceiving, building, and programming robotic systems meant to remain operational over sustained amounts of time. Long-duration autonomy is studied and analyzed from two different, yet complementary, perspectives: control algorithms and robot design. In the context of the former, the *persistification of robotic tasks* is presented. This consists of an optimization-based control framework which allows robots to remain operational over time horizons that are much longer than the ones which would be allowed by the limited resources of energy with which they can ever be equipped.

As regards the mechanical design aspect of long-duration robot autonomy, in the second part of this thesis, the *SlothBot*, a slow-paced solar-powered wire-traversing robot, is presented. This robot embodies the design principles required by an autonomous robotic system in order to remain functional for truly long periods of time, including energy efficiency, design simplicity, and fail-safeness. To conclude, the development of a robotic platform which stands at the intersection of design and control for long-duration autonomy is described. A class of vibration-driven robots, the *brushbots*, are analyzed both from a mechanical design perspective, and in terms of interaction control capabilities with the environment in which they are deployed.

CHAPTER 1
INTRODUCTION

As robots gradually move from research laboratories and industrial settings to less structured and more dynamic environments, new challenges coming from unexpected and unmodeled environmental phenomena arise. This is true even more so when robots are to be deployed on the field for long-term applications, as in the case of environmental monitoring scenarios [1, 2, 3]. Thus, a way of encoding *survivability* [4], intended as the ability to remain alive—in a robotic sense—is needed now more than ever. In this thesis, we study the problem of *long-duration robot autonomy*, which deals with robots deployed over long time horizons, from two different, yet complementary, perspectives: control algorithms and robot design.

Executing tasks over long periods of time is not always possible due to the limited amount of energy that robots can store in their batteries. Even solutions such as rechargeable batteries harvesting solar power are not guaranteed to provide the required amount of energy, due to the variability of the environmental conditions. Moreover, when long-duration autonomy is considered, it is desirable to design robots which are as flexible, adaptable and robust to changing environmental conditions as possible. For this reason, strategies obtained using optimal control techniques are not a viable solution, as they are characterized by the fragility that comes with optimality, i.e. related to the precise model assumptions utilized in the control design [5]. These assumptions, in fact, are very likely to be violated during the long time horizons over which the robots have to remain operational in long-duration autonomy scenarios. Therefore, in the first part of this thesis, we present a control theoretic framework for the *persistification* of robotic tasks, intended as the process of rendering robotic tasks persistent, that is equivalent to rendering robots able to execute the tasks over time horizons much longer than the life of their batteries. This process will

be formally defined and employed to enforce survivibility conditions on the robots in order to let them sustainably execute the tasks for which they are deployed.

With this line of inquiry, the persistification strategy is recognized to be just one aspect of a broader robot control design idea: the *constraint-driven* robot control paradigm. Besides its application to the control of robotic platforms, it is informative to notice that a similar principle has been found to determine the actions of biological organisms. In fact, ecological studies have shown that the constraints imposed by the environment strongly determine the behaviors developed by animals living in it [6]. Inspired by this observation, we ask whether robots can be controlled purely using constraints. This results in what we call a robot-ecological formulation [4], in which robots are *programmed to do nothing*, subject to task and survivability constraints. This concept can be formalized in the following optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}} \ \|u\|^2 \\ &\text{subject to } c_{surv}(x,u) \geq 0 \\ &\qquad\qquad c_{task}(x,u) \geq 0, \end{aligned} \tag{1.1}$$

where u represents the control effort spent by the robots, $c_{surv}(\cdot,\cdot) \geq 0$ is the constraint encoding survivability, and $c_{task}(\cdot,\cdot) \geq 0$ is the constraint corresponding to the task that has to be executed by the robots. The effectiveness of the constraint-based approach is demonstrated by the variety of applications in which it has been employed: from the coordinated control of multi-robot systems [7, 8], to the multi-task prioritization of redundant manipulators [9], from the multi-robot multi-task allocation problem [10, 11, 12] to the learning of robotic tasks [13].

As was discussed so far and will be further elaborated in the following chapters, control theory can allow us to develop strategies suitable for the long-term deployment of robotic systems. *Specification-driven robot design* can facilitate the achievement of this goal. In the second part of this thesis, we show this by presenting the design of two robotic platforms envisioned for long-term environmental monitoring tasks :

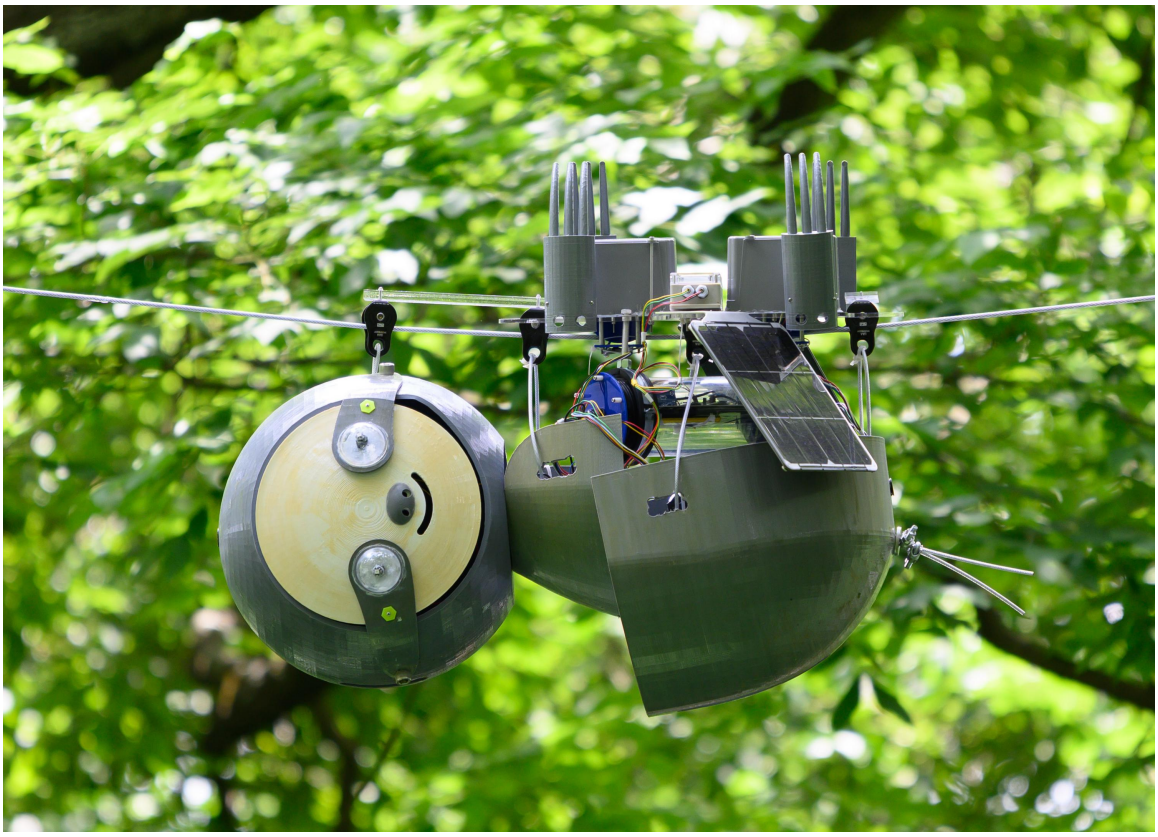


Figure 1.1: The SlothBot in the Atlanta Botanical Garden (Credit: Rob Felt, Georgia Tech).

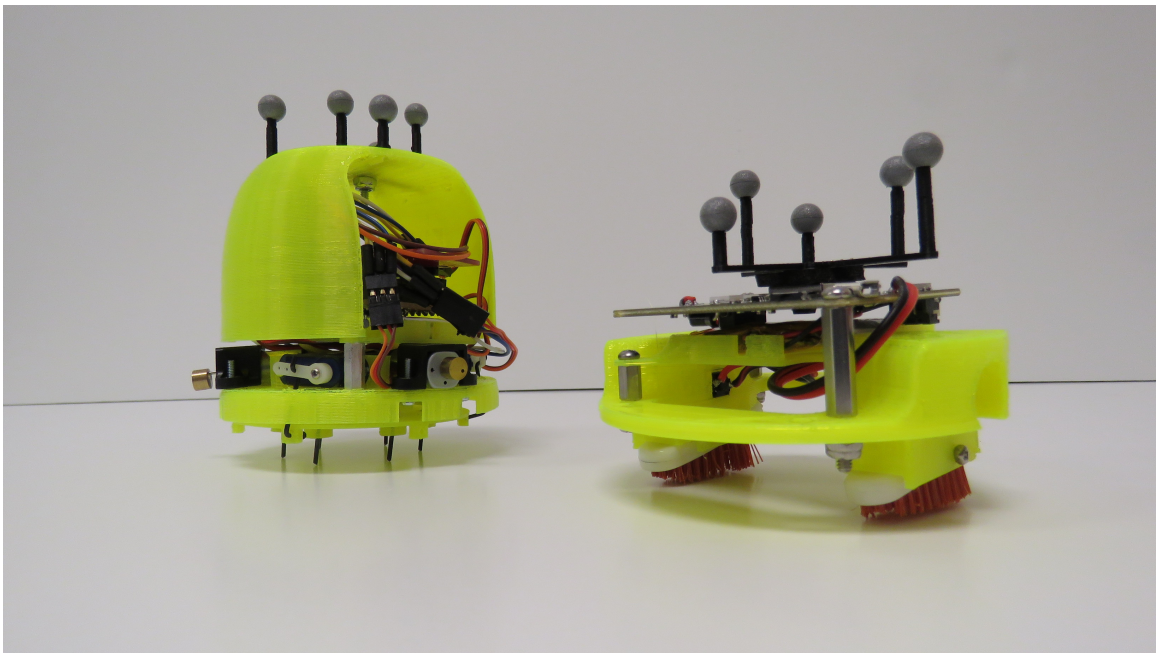


Figure 1.2: Examples of brushbots.

- the *SlothBot* [14], an energy-efficient solar-powered wire-traversing robot (see Fig. 1.1),
- and the *brushbots* [15], a class of vibration-driven robots which leverage elastic elements, the brushes, to achieve robust locomotion and interaction with the environment (see Fig. 1.2).

Owing to their ability to move along cables, wires, and similar infrastructure, wire-traversing robots are characterized by simplicity of motion control, reduced localization error and low energy requirements [16]. These characteristics have allowed them to gain increasing interest in different applications domains, ranging from power line inspection and maintenance [17, 18, 19, 20] to environmental monitoring [21] and agricultural robotics—e.g., plantation growth and health state monitoring [22]. Vibration-driven robots, on the other hand, have the capability of efficiently extracting energy—in the form of mechanical vibration—from the environment in order to locomote [23, 24]. For this reason, they perfectly lend themselves to be employed in ultra-low energy monitoring applications where vibrations can be leveraged as a power source, such as traffic monitoring on suspended bridges and structural health monitoring of mechanical components. Moreover, this design principle suggests the applicability of similar robotic platforms for other kinds of long-term monitoring applications, including human health. The deployment of nano-scale autonomous robots inside the human body might allow the effective realization of an important aspect of what is known as *health-on-demand* system, where targeted measurements to assess the health state can be performed on demand, and drugs can be released in a precise and fully autonomous fashion.

The remainder of this thesis is organized as follows. In the next chapter, the research work carried out in this thesis is contextualized in the current research scenario, and in particular in the fields related to the different studied topics. The design of control algorithms and robotic platforms will be the subjects of Part I and II of this thesis, which present the algorithmic foundations and the design principles of robots envisioned for long-term operations, respectively. In Part I, Chapter 3 is devoted to the development of the *task*

persistification control strategy which allows robots to optimally execute tasks constrained by the amount of remaining available energy. Chapter 4 extends the approach described in Chapter 3 to develop the *constraint-driven control paradigm*, where both tasks to be executed and robot survivability are encoded as constraints within a mininum-energy optimization program. Using the constraint-based formulation formalized in Chapter 4, Chapter 5 presents an *energy-aware task assignment* for heterogeneous multi-robot systems. The algorithm is based on the idea of realizing the allocation of tasks through a different *prioritization* across different robots. In Chapter 6, the energy related to communication in distributed estimation applications is considered in order to design a *communication-constrained distributed estimation* algorithm, where robots with communication bandwidth limited by the available energy are deployed with the goal of estimating an environmental field over a long-time horizon. In Part II, Chapter 7 describes the *SlothBot*, and in particular, two designs proposed to navigate a mesh of wires while switching wire branches in a fail-safe fashion, and to locomote on a single wire, respectively. Moreover, strategies to control the motion of these two robotic platforms for environmental monitoring applications are developed. In Chapter 8, a different perspective on robot design principles for long-duration autonomy is considered, where the *brushbots* are presented as platforms capable of exploiting vibrational energy in order to locomote and perform swarm robotic tasks. At the end of the chapter, a micro-scale version of the brushbots is showcased in an experiment of density control in robotic swarms. Chapter 9 concludes the thesis providing a summary of the main findings as well as directions for future work.

CHAPTER 2
BACKGROUND

In this chapter, we highlight some of the research work carried out in long-duration autonomy, both on the development of control algorithms as well as on the design of robotic platforms. Following the partition of the thesis, in the first part of the chapter, we will concentrate on the algorithmic foundations for rendering robotic task execution persistent, as well as that of allocating multiple tasks to multiple robots in an energy efficient fashion, allowing the task execution over extended amounts of time. In the second part of the chapter, existing robot designs of wire-traversing and vibration-driven robots are analyzed and compared to the SlothBot and the brushbots, respectively.

2.1 Long-Term Execution of Robotic Tasks

The deployment of robots for tasks such as environmental monitoring [25, 26, 27, 28], environment exploration [29, 30, 31] and sensor coverage [32, 33, 34] has been extensively investigated. However, despite the fact that, in most cases, these tasks have to be executed over long time horizons, the limited availability of energy is not directly taken into account. Nevertheless, since low energy density is still a severe limiting factor in many mobile robotic applications, energy-awareness is a necessary feature with which robots have to be endowed [35]. This line of inquiry has been followed in [36], which considers a multi-robot, persistent coverage problem as a variant of the vehicle routing problem. A heuristic algorithm is proposed that is based on the cost-to-go-to-target, which can be adjusted online to take into account detours that pass through refueling stations present in the environment. A different approach is adopted in [37], where a formulation based on Markov decision processes is presented, that is able to ensure persistent surveillance coverage, including communication constraints and sensor failure models.

Energy-aware control policies for persistent surveillance using a team of robots are considered in [38], where an optimization problem is defined in order to trade-off between the coverage mission and the conservation of a desired energy level. This is achieved by transitioning between coverage-directed and charging-directed behaviors, based on the current energy levels. The coverage performances are improved by employing *standby robots* that can be deployed when a robot is docked at a charging station. Similarly, in [39] a solution to the problem of long-duration missions is proposed, which considers the team of robots split into *task robots*, which are in charge of executing tasks, and *delivery robots*, which are deployed with the goal of providing the task robots with the energy resources they require. Also in [40], the strategy consists in making use of a team of robots dedicated to charging tasks (ground mobile docking stations), whose trajectories are planned based on the working robots' trajectories (UAVs), in order to guarantee randezvous and recharge without suspending the operation of the working robots. Limited energy reserve is used as an additional constraint in [41] for a path planning strategy for the optimal deployment of multi-robot teams. A definition of persistency different from the one introduced in this thesis is considered in [42], where environment persistent monitoring is solved by varying the robots' speed along predefined trajectories with the goal of keeping a regenerating environment information field bounded, analogously to what happens to a robot cleaner in an environment in which dust is generated.

In this thesis, we will use Control Barrier Functions (CBFs) in order to develop the persistification framework of robotic taks. CBFs can be used to synthesize controllers that ensure the forward invariance of a set $\mathcal{C}$ of the robot state space. This way, defining $\mathcal{C}$ as the set where the battery energy level of the robots executing the task is always greater than a desired minimum value, the persistification of a task can be formally guaranteed by ensuring the forward invariance of $\mathcal{C}$. One of the first definitions of CBFs was given in [43]; in our work, we use the one introduced in [44]. See [45] for a survey on the subject. Several variations have been introduced in order to employ CBFs with different categories

of nonlinear dynamic systems for different control applications [46, 47, 48, 49, 50, 51].

2.2 Constraint-Driven Control of Robotic Systems

Robots are gradually leaving academic laboratories, e.g. [52, 53], in favor of industrial settings, as in the case of [54], to reach even less structured and more dynamic environments, like agricultural lands and construction sites [1]. This presents new challenges that robots have to face, which come either from unexpected and/or unmodeled phenomena or from changing environmental conditions. These issues become even more pronounced when the robots are deployed on the field for long-term applications, like persistent environment surveiling [2] or plant growth monitoring [3]. Therefore, a way of encoding *survivability* [4], i.e., the ability to remain alive (in a robotic sense), is needed now more than ever. In our work, we introduce a method which can be used to ensure survivability, and which makes use of optimization tools that allow to only minimally influence the task which the robots are asked to perform.

Several solutions have been proposed in order to make robots robust to unknown or changing environmental conditions and to ensure their applicability to unstructured or even hazardous environments [55]. Moreover, in order to let the robots *survive* for as much time as possible, some of the proposed methods entail scheduled periodic maintenance [56], or path optimization with the aim of maximizing the time spent in the field/minimizing the consumed energy [57]. Some other methods employ a power-dependent multi-objective optimization to ensure that the robots execute their task, while maintaining a desired energy reserve, as done in [38]. In both cases, a careful tuning of parameters is required in order to prevent situations in which the robots trade survivability for shorter-term rewards.

As a matter of fact, *goal-oriented* control strategies may not be ideal for long-term applications, where robustness to changing environmental conditions is required. Indeed, control policies obtained using optimal control strategies are characterized by a fragility related to the precise model assumptions [5]. These are likely to be violated during the

long time horizons over which the robots are deployed in the field. For this reason, we adopt a *constraint-oriented* approach, where the survivability of the robots is enforced as a constraint on the robots' task, encoded by the nominal input to the robots. The control input is then synthesized, at each point in time, by solving a constrained optimization problem. Inspired by ecological theories where the actions taken by biological organisms are driven by environmental constraints [6], we ask whether robots can be controlled using constraints only. What this entails is that robots are programmed to do nothing, subject to task and survivability constraints. Thus, what we could call a *robot-ecological* formulation [4] naturally lends itself to be implemented using optimization-based control techniques.

2.3 Energy-Aware Task Allocation

In this section, we briefly review the relevant literature on multi-robot task allocation. For a comprehensive survey on a broader class of task allocation problems, see [58, 59, 60] and references within. In [61], the authors present a framework for assigning heterogeneous robots to a set of tasks by switching between different predefined behavioral routines. To tackle the challenges of computational complexity associated with such discrete assignment-based approaches, market-based methods [62, 63, 64, 65] have been proposed, where robots can split tasks among them via bidding and auctioning protocols. In scenarios where a large number of robots with limited capabilities are present, decentralized stochastic approaches have been developed where allocations are described in terms of population distributions and are achieved by modifying transition rates among tasks [66, 67, 68].

The development of heterogeneous robots, characterized by different availability of features and resources, necessitates the characterization of resource diversity and access within the multi-robot system [69, 70]. In [71, 72], the authors define a community of robot species, each endowed with specific capabilities, and develop an optimization-based framework to allocate sufficient capabilities to each task. This is realized using transition rates, which the robots use to switch between the different tasks. In comparison, the enery-

aware task allocation approach we present in this thesis explicitly models how different feature bundles available to the robots might endow them with capabilities required to execute a task. This has the benefit of enduing the allocation framework with a degree of resilience, as will be demonstrated in the later chapters.

Indeed, adaptivity and resilience are commonly studied aspects of task allocation in multi-robot systems (see, e.g., [73, 74, 75, 76]). Typically, adaptivity is incorporated by defining a time-varying propensity of robots to participate in different tasks. These measures of utility are based on predefined objective functions and aim to capture the effectiveness of the robots at performing tasks in real-time [77]. Such frameworks, however, do not account for drastic unexpected failures in the capabilities of the robots, adversarial attacks, or varying environmental conditions that might affect the operations of the robots. Such considerations point to the question of resilience in multi-robot systems, which has been explored in the context of coordinated control tasks [78], as well as resource-availability in heterogeneous systems [79]. Building up on our previous work, presented in [7, 10, 11], in this thesis we present three distinct novel developments towards the achievement of a resilient task allocation and execution framework. These are:

(i) an optimization-based framework which considers the real-time performance of robots in executing the tasks

(ii) the explicit feature- and capability-based models of robot heterogeneity, which allows for greater flexibility in allocating tasks

(iii) a minimum-energy task execution framework, geared towards long-duration autonomy applications.

2.4 Communication-Constrained Distributed Estimation

An approach to reduce the amount of data transferred between sensor nodes in a network has been presented in [80], where a novelty detection algorithm is employed in an online

learning framework in order to limit the amount of data that needs to be stored and therefore transferred. A similar objective is pursued in [81], where the authors propose a way of generating small data sets that still keep a high proportion of the information contained in the original, large, data-set. General guidelines for choosing the size of subsets of data based on detailed experimental studies are reported in [82]. Finally, with the ultimate goal of decreasing the computational burden of Gaussian Process Regression (GPR) models, massively scalable Gaussian processes are introduced in [83].

Communication constraints are not explicitly considered in the approaches reported above, which do not aim at reducing the amount of data that is to be exchanged between the nodes of the sensor network. The amount of data that a node of a mobile Wireless Sensor Network (WSN) has to transfer significantly affects the amount of consumed energy. With the objective of reducing this energy, [84] consider information theoretical bounds in order to find the minimum number of bits per symbol to be employed in the communication scheme. Similarly, limited communication capabilities in terms of communication range are considered in [85], introducing a Distributed Gaussian Process Regression (DGPR) in which each sensor node only needs to communicate with its neighboring nodes. A similar concept has been further explored in [86], where, however, the problem of keeping the amount of exchanged data between nodes of WSNs bounded has not been specifically investigated.

2.5 Wire-Traversing Robots

Surveys of the state of the art of robots designed to traverse cables are presented in [16] and [87]. In [88] a robot consisting of multiple units allowing wire switching and obstacle avoidance is presented. More recent work on the development of wire-traversing robots can be found in [17, 89, 90, 91]. In [17], the presented robot has folding capabilities that allow it to avoid obstacles. A modular robot that is able to slide on horizontal wires as well as climb on vertical ones is presented in [89]. In [90], the proposed robot uses a caterpillar-

like locomotion strategy that allows it to climb up and move on ropes. In [91], the authors present a robot that is able to locomote using different methods, such as inchworm-like or brachiating motion patterns, which allow it to move fast and avoid occasional obstacles.

The obstacle avoidance feature has been subject of research because the designed robots have been mainly employed in power cables or suspension bridge maintenance. In these applications, since the robots have to adapt to existing infrastructures, the capability of avoiding obstacles along the wires constitutes an essential design requirement. Consequently, the resulting designs feature multiple robotic-arm-like attachments, each of which needs several motors to be fully actuated. The few existing robots capable of switching-wires also rely on similar multi-joint arm mechanisms that are large and complex. Furthermore, relatively little design effort has been put into designing energy efficient robotic platforms capable of staying out in the field for long-term missions. The maintenance operations of bridges and power cables mentioned above are, in fact, usually limited to a few hours. The robots shown in [92], for instance, only have up to 6 hours of autonomy.

Because of the long-term applications targeted by the SlothBot presented in this thesis, we consider two aspects that have not been explicitly taken into account in the state-of-the-art designs, namely energy efficiency and fail-safeness. The definition of fail-safeness adopted in this thesis is the ability of a wire-traversing robot to remain hung on the wire at any point in time given the undetectable failure of one of its actuators. Table 2.1 shows a comparison between the SlothBot and other existing wire-traversing robotic platforms. The metrics considered for the comparison are the wire-switching capabilities, the fail-safeness (as defined above), and the actuation complexity. The SlothBot possesses all the characteristics desirable for the long-term applications mentioned before, and has the minimum number of actuators among the platforms capable of wire-switching.

Table 2.1: Comparison between the SlothBot and existing wire-traversing robot designs.

	Locomotion Principle	Wire-Switching Capability	Fail-safeness	Number of Actuators	Weight (kg)
LineScout [17]	Wheels	No	Yes	—	100
Caterpillar-like robot [90]	Wheels	No	Yes	—	—
SkySweeper [91]	Pulley Arms	No	Yes	3	0.466
Expliner [93]	Wheels	Yes	No	6	60
Modular robot [88]	Wheels	Yes	Yes	16	10
SlothBot	Wheels	Yes	Yes	7	1

2.6 Robot Control Under Motion Constraints

Robots with constrained motion, e.g. those with the ability to move only along pre-designed infrastructures, lend themselves to a large variety of applications, such as environmental monitoring [21] and agricultural robotic tasks [22]. Some of the reasons of their success can be recognized in the following features [21, 91]:

- low energy requirements
- simplicity in the motion control
- small localization errors
- absence of navigation problems even in unknown environments.

However, these advantages are obtained to the detriment of a more complex infrastructure. Nevertheless, there are a lot of applications in which an infrastructure is already present and it can be exploited virtually at no additional cost. An example is power transmission line maintenance [20].

A particular category of constrained motion robots are wire-traversing robots [91, 16]. The approach presented in this thesis focuses on the motion planning and control for this kind of robots whose objective is sensor coverage of the surrounding environment. Wire-traversing robots have already found their application in several domains. In [18] the development of a mobile robot which is able to autonomously navigate on power transmission lines is described. The goal is automating the inspection of power transmission lines and their equipment. Robotics in agriculture and forestry [22] has already experienced an automation process that introduced the use of cable-driven robots whose tasks consist in harvesting fruits and vegetables, dispensing fertilizer and monitoring growth and health of plants. Moving to a different branch, in [94] an algorithm to monitor traffic starting from videos recorded from Skycams ([95]) suggests the viability of wire-traversing autonomous robots for traffic and road network management. In [21, 96] a cable-based robotic platform

is described, whose objective is monitoring the environment and characterizing its phenomena. As also pointed out before, the strength of such a system lies in its overall robustness and reliability, accurate and reproducible motion, long range mobility even in complex environments as well as low energy consumption that enables sustainable operation.

Although the technology for the deployment of wire-traversing robots in the environment is somewhat mature, none of the above-mentioned approaches explicitly deals with the motion planning of the robots on the wires on which they are constrained to navigate. In [97] the concept and the design of a mobile manipulator for autonomous installation and removal of aircraft-warning spheres on overhead wires of electric power transmission lines are presented. [19] describes the development of a mobile robot that can navigate aerial power transmission lines autonomously with the goal of automating inspection of power transmission lines. In [88] a multi-unit structure wire mobile robot is proposed, which allows the robot to transfer to a branch wire and avoid obstacles on the wire.

The motion planning for robots on wires is typically left to general purpose motion planners that use search algorithms on grid maps in order to plan a route to a desired location (see e.g. [98]). The main contribution of the work in this thesis is a solution to the motion planning problem for wire-traversing robots and, in general, for robots constrained to move on grid maps or on curves. This is achieved by including the motion constraints in the formulation of the motion control law. The described concept is applied to a coverage control task, where the robots have to spread in the environment in order to monitor its phenomena. Constrained locational optimization has already been considered in [99], where the author proposes a decentralized gradient projection method in order to obtain the motion control law. Moreover, a hybrid method, which uses both locational optimization and path planning algorithms to generate robots' motion, is presented in [100].

2.7 Vibration-Driven Robots

The principle on which the motion of a brushbot relies is the alternation of stick and slip phases during which the brushes adhere or not to the ground. One of the first applications of the stick-slip mechanism to robot locomotion can be found in [101], where a three-degree-of-freedom micro-robot is presented. Using this principle, in [102], the authors propose an improved, energy-efficient design of a micro robot, together with a control strategy suitable for trajectory tracking. Due to the design simplicity and the resulting robustness, brushbots lend themselves to swarm robotic applications, where groups of robots are utilized to perform coordinated tasks in a decentralized fashion. This idea is explored in [53], where the authors present the Kilobot, a small scale brushbot equipped with an infrared and a light sensor that enable the execution of decentralized swarming algorihtms. Collective behaviors of brushbots are also investigated in [103], where the authors analyze the parameters governing the transition from a disordered motion to an organized collective motion.

As far as the analysis of brush dynamics is concerned, in [23], a model is developed and validated using an experimental robotic platform. Here the authors do not focus on the motion control explicitly, as much as it is done in [104]. In the latter, an omnidirectional stick-slip robot is presented and a way of automatically calibrating it is proposed. A more theoretical analysis is performed in [24], where the derived equations of motion are solved using a heuristic approach in order to obtain analytical formulas for the average velocity of the robot.

In this thesis, we propose a dynamic model for brushbots, which starts from the microscopic analysis of the brushes to culminate in the macroscopic model of the robot. In particular, this model improves the ones which can be found in literature by explicitly taking into account the inertia of the brushes and the effects that it has on the resultant brushbot velocity. Moreover, the derived model will be further validated through the development of a trajectory tracking controller and the implementation of a coordinated control algorithm

for a swarm of brushbots.

To summarize, the main contributions of this work are the following:

(i) we propose a brush model which considers the inertia of the brushes and the contact dynamics of their interaction with the ground

(ii) we analyze and qualitatively characterize different *regimes of operation* for the different models of brushbots developed in the literature

(iii) we present the mechanical design of two brushbots, a fully-actuated platform that can switch between regimes of operations, and a differential-drive-like brushbot, specifically designed for swarm robotics applications

Furthermore, in our related work [105], we build upon the results presented in this thesis and demonstrate the ability of brushbot swarms to achieve collective behaviors using simple local interactions.

Part I

Control Algorithms

CHAPTER 3
PERSISTIFICATION OF ROBOTIC TASKS

In the first part of this thesis, we study the algorithmic foundations of long-duration robot autonomy. In this first chapter, we start by presenting a control framework that enables robots to execute tasks *persistently*, i.e., over time horizons much longer than robots' battery life. This is achieved by ensuring that the energy stored in the batteries of the robots is never depleted. This condition is framed as a set invariance constraint in an optimization problem whose objective is that of minimizing the difference between the robots' control inputs and nominal control inputs corresponding to the task that is to be executed. We refer to this process as the *persistification* of a robotic task. Forward invariance of subsets of the state space of the robots is turned into a control input constraint by using control barrier functions. The solution of the formulated optimization problem with energy constraints ensures that the robotic task is persistent. To illustrate the operation of the proposed framework, we focus in particular on two tasks whose persistent execution is particularly relevant: environment exploration and environment surveillance. We show the persistification of these two tasks both in simulation and on a team of wheeled mobile robots on the Robotarium.

Robotic tasks such as environmental monitoring and exploration, as well as sensor coverage, typically evolve over long time horizons. However, robots employed for these tasks are limited by the amount of energy that can be stored in their batteries. For this reason, we can say that such tasks are not *persistent* as either the robots cannot complete them before their batteries deplete, or they are required to be executed repeatedly and continuously. Although robots can be designed with greater energy capacity to handle longer-duration tasks, hardware solutions will never allow robots to operate perpetually.

The objective of this chapter is to present a control framework that provably guarantees

the *persistent* execution of robotic tasks. This is achieved by minimally modifying the nominal control inputs corresponding to the task that the robots have to execute in order to ensure the continuous execution of the task. As a result, the robots are allowed to freely execute their task whenever they have enough energy stored in their batteries, whereas they are forced to go and recharge their batteries whenever they are running out of energy.

The *persistification* approach, through which a robotic task is rendered persistent, that we present in this chapter, leverages control barrier functions (CBFs) to formulate an optimization problem where the task execution is constrained by the robots' energy level. As formalized in [4] and [7], the *constraint-driven* formulation resulting from the use of CBFs is particularly amenable for long-duration robotic tasks, where *goal-driven* strategies, derived starting from precise model assumptions, do not guarantee enough robustness. The persistification strategy developed in the following extends the work in [106] in order to be able to handle more general robot and energy dynamics. The presented method generalizes to different charging models as well as different robotic tasks. The tasks will be encoded through different nominal inputs to the robots. This allows us to formally guarantee the persistent execution of a large number and variety of robotic tasks.

The remainder of the chapter is organized as follows: in Section 3.1, the models of the robots, the environment, and the energy dynamics are presented, and the problem of persistification of robotic tasks is formulated. In Section 3.2, the control framework required to ensure robotic task persistence is introduced and its application is demonstrated by means of preliminary examples. In particular, Section 3.2.4 discusses the application of the presented framework to the persistification of robotic tasks. Section 3.3 reports the results of the proposed theoretical formulation implemented for two robotic tasks whose persistent execution is particularly relevant, both in simulation and on a team of mobile robots.

3.1 Problem Formulation

The goal of this chapter is the persistification of robotic tasks, i.e., ensuring that the battery energy level of the robots executing the tasks never falls below a minimum value. In this section, we introduce the models used for the robots, the environment in which the robots execute their task, and the robots' energy dynamics. We conclude the section by formalizing the persistification of robotic tasks.

3.1.1 Robot Model

Consider a collection of N robots which are to be deployed to execute a task. The state of robot i, $i = 1, \ldots, N$, is denoted by $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$. We model the robots using the control affine dynamical system:

$$\dot{x}_i = f(x_i) + g(x_i)u_i, \tag{3.1}$$

where $u_i \in \mathcal{U} \subseteq \mathbb{R}^m$ is robot i's input, and f and g are two Lipschitz vector fields. Control affine dynamics arise in many robotic systems, whose models are derived using Euler-Lagrange equations (as observed in [98]), therefore they lend themselves to the description of a large variety of robotic platforms. Throughout this chapter, we will assume that the N robots are homogeneous, i.e., f and g in (3.1) are the same for each robot. This assumption does not compromise the proposed persistification strategy, which can be easily employed in the heterogeneous case too, as will be pointed out in Section 3.2.4.

The robots considered in this chapter are equipped with a rechargeable source of energy, e.g., a battery, and a technology required to recharge it, e.g., solar panels. In the following two subsections, we present a model for the robot energy dynamics which is coupled with the model of the environment in which the robots move. A paramount example of this scenario is that of solar power harvester circuits employed to use solar panels to recharge the batteries of robots.

3.1.2 Environment Model

The environment, i.e., the domain in which the robots are deployed to perform their task, is represented by the compact set $\mathcal{E} \subset \mathbb{R}^p$, with $p = 2$ or $p = 3$, for ground or aerial robots, respectively. The function $\pi : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathcal{E} \subset \mathbb{R}^p$ maps the robot state to its position expressed in a Cartesian reference system defined in the environment $\mathcal{E}$.

Moreover, we consider the time-varying scalar field

$$I : \mathcal{E} \times \mathbb{R}_+ \rightarrow \mathcal{I} \subset \mathbb{R}_+, \tag{3.2}$$

where $\mathcal{I}$ is an interval, defined over the environment, which represents a bounded time-varying non-negative physical quantity (e.g., solar light intensity) associated to each position in $\mathcal{E}$ at each time instant. We insist on I being Lipschitz continuous in its first argument and differentiable in its second argument. The need for these assumptions will be explained in the next section.

3.1.3 Energy Model

State of Charge Dynamics

The State of Charge (SOC) is used to describe the remaining capacity of a battery, and it is therefore a very important parameter to take into account while designing energy-aware control strategies [107]. Its accurate knowledge allows us to design algorithms to protect batteries from overdischarge and overcharge, increasing their life. However, batteries store chemical energy, which cannot be directly measured and has to be, therefore, estimated. The SOC estimation still remains a fundamental challenge, as it hinges on battery models which, in turn, depend on many parameters [108]. A survey of methods which have been proposed to estimate the SOC of batteries can be found in [109, 110]. In the following, we present the basic SOC dynamic model (found, e.g., in [111]) which can be employed in the proposed task persistification strategy to prevent robot batteries from depleting.

Let $SOC_i(t)$ denote the value of the SOC of robot i's battery at time t, C_N its rated capacity, η its coulombic efficiency, and $i_i(t)$ the current flowing from the battery of robot i—$i_i(t) > 0$ if current is flowing out of the battery at time t, $i_i(t) < 0$ if flowing in. Then, the value of $SOC_i(t)$ can be evaluated as

$$SOC_i(t) = SOC_i(t_0) - \frac{1}{C_N} \int_{t_0}^{t} \eta i_i(t) \, \mathrm{d}t, \tag{3.3}$$

where, in general, the efficiency η can be a function of the current $i(t)$, the temperature, the SOC itself, as well as the state of health (SOH) of the battery [111].

In a large variety of robotic applications, it is reasonable to assume that the current supplied by the battery is proportional to the magnitude of the control input $\|u_i\|$ of robot i. This holds, for instance, in cases where electric motors and actuators—for which exerted torque is proportional to absorbed current—are employed for motion and locomotion purposes [112]. Starting from (3.3), and using this assumption, yields the following SOC dynamic model for the battery of robot i:

$$\dot{SOC}_i = f_{SOC}(u_i, t). \tag{3.4}$$

The function f_{SOC} is used to gather the dependencies of the efficiency η on current, temperature, SOC, and SOH, as well as the relationship between current i_i and control input u_i discussed above. In particular, the explicit dependence of f_{SOC} on the time variable t is used to reflect its dependence on the temperature at which the robot battery is operating.

It is informative to notice that the value of SOC and battery voltage have been experimentally observed to be related. In particular battery voltage is a monotonically increasing function of the SOC [113]. Therefore, when the estimation of the SOC described in the previous section is not feasible, the battery voltage can be used as a proxy for the amount of energy stored in the battery of the robots. Starting from the following section, we will talk about the energy level of the battery of the robots, where both SOC estimations and

battery voltage measurements can be used in place of it in the discussion that follows.

Energy Model

Let $E_i \in (0,1) \subset \mathbb{R}$ be the battery energy level of robot i. The charging and discharging behaviors of the battery are modeled by the following dynamic equation:

$$\dot{E}_i = F(x_i, E_i, t) = k\left(w(x_i, E_i, t) - E_i\right), \tag{3.5}$$

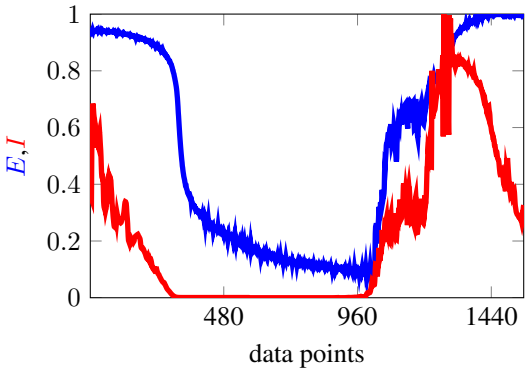
where $k > 0$ and

$$w(x_i, E_i, t) = \frac{1}{1 + \frac{1-E_i}{E_i} e^{-\lambda(I(x_i,t)-I_c)}}. \tag{3.6}$$

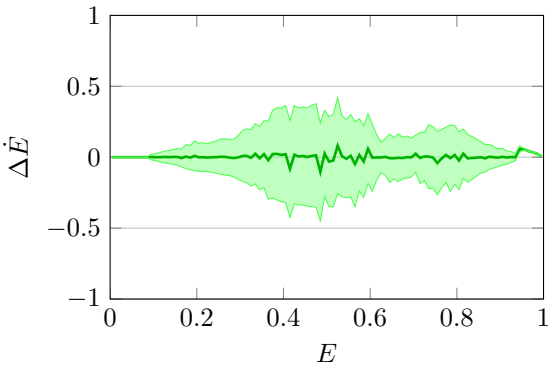
In (3.6), $\lambda > 0$ and $0 < I_c < 1$ are two scalars whose meaning and effect on the energy dynamics will be explained in Remark 3.1, and $I(x_i, t) : \mathcal{E} \times \mathbb{R}_+ \to [0,1]$ is a time-varying scalar-valued function introduced in (3.2). The expression of $F(x_i, E_i, t)$ has been designed in order to model the exponential charging-discharging dynamics of batteries that are used to power robotic platforms in a large number of applications [114].

In [14], the design of a solar-powered robot, the SlothBot, is presented, and data of solar intensity and battery charge collected during the course of a 1-day-long experiment are reported (see Fig. 3.1a). Figure 3.1b shows the difference $\Delta\dot{E}$ between the measured $\dot{E}$ and its value predicted using the energy model (3.5). The thick green line depicts the mean of $\Delta\dot{E}$ and the shaded area denotes the region of one standard deviation around the mean. Moreover, Fig. 3.1c shows the simulated battery charge and discharge curves that can be obtained using (3.5) in the cases when $I > I_c$ and $I < I_c$, respectively. A comparison with Fig. 3.1a indicates that the theoretical model is able to capture the exponential charging and discharging dynamics of real batteries commonly used for robotics applications.

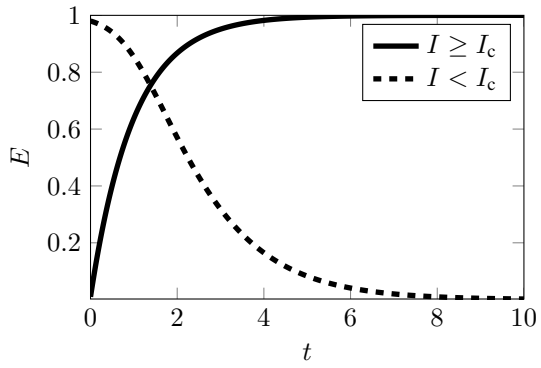
Remark 3.1. *$I(x_i, t)$ can be interpreted as a time-varying power source distributed over the environment $\mathcal{E}$. For instance, in the case where it represents a measure of the solar light intensity at the position x_i and time t, $F(x_i, E_i, t)$ can be used to describe the energy*



(a) Data collected during the course of a 24-hour experiment using a solar-powered robot. E, in blue, and I, in red, are measured battery energy and solar light intensity, respectively.



(b) Comparison between measured and predicted values of $\dot{E}$: $\Delta\dot{E}$ represents the difference between the measured $\dot{E}$ and its value predicted using the model in (3.5). The mean of $\Delta\dot{E}$ is depicted as a thick green line, whereas the shaded area represents the region of one standard deviation from the mean value.



(c) Simulated battery charging and discharging dynamics: a comparison with Fig. 3.1a shows that the model proposed in (3.5) is able to reproduce the true dynamics of a real battery used in robotics applications.

Figure 3.1: Validation of the energy model proposed in (3.5) using data collected during a long-term experiment with a solar-powered robot [14].

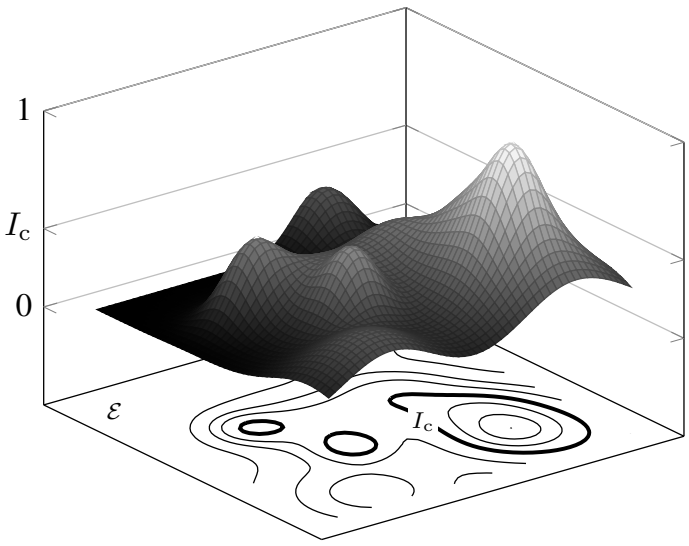


Figure 3.2: Example of the function $I(x_i, t)$ over the environment $\mathcal{E}$ at a given time instant: inside the bold level curve marked with I_c, $\dot{E}_i > 0$, whereas outside $\dot{E}_i < 0$. In other words, the robots can recharge their batteries in the regions bounded by the bold curves.

dynamics of a solar rechargeable battery. With the given dynamics we have that:

- $\dot{E}_i < 0$*, i.e., the battery is discharging, whenever* $I(x_i, t) < I_c$
- $\dot{E}_i > 0$*, i.e., the battery is charging, when* $I(x_i, t) > I_c$
- $\dot{E}_i = 0$*, i.e., the value of* $I(x_i, t) = I_c$ *is such that the generated energy is equal to the energy required by the robot at time* t*.*

Figure 3.2 shows an example of what has been described: the surface plot of the field I *at a given time instant* t *is depicted in grayscale (black to white for values of* I *that go from 0 to 1). Below the surface plot, the contour plot of* I *highlights the level curves where* $I(x_i, t) = I_c$*. Inside the regions bounded by the bold curves, characterized by* $I(x_i, t) > I_c$*,* $\dot{E}_i > 0$*, i.e., the robots can charge their batteries.*

Remark 3.2. *Lumped sources of energy, such as charging stations, can be also modeled using* (3.2)*. Bump-like functions [115] at the locations of the charging stations can be employed to obtain the desired charging behavior, as depicted in Fig. 3.3.*

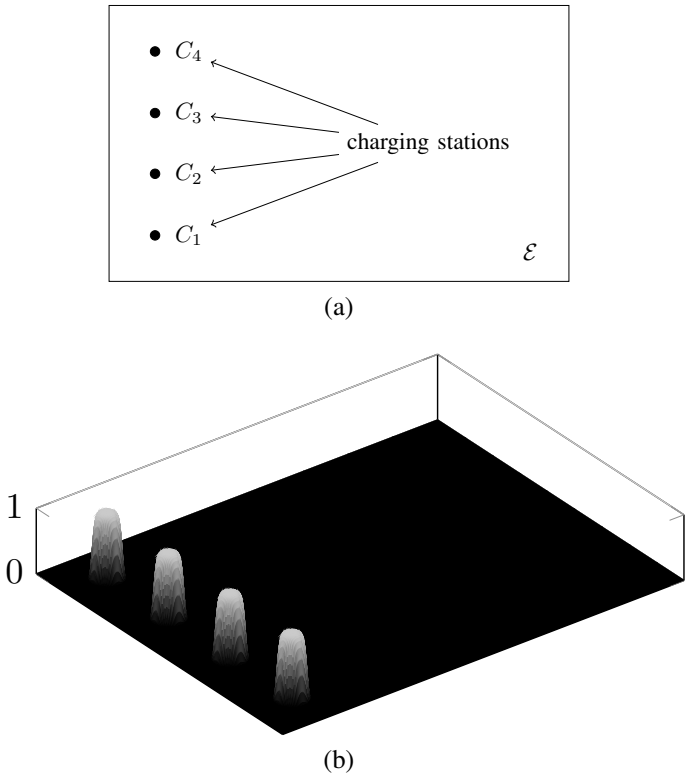


Figure 3.3: Example of the modeling of lumped sources of energy (charging stations) via a suitable $I(x_i, t)$ function. In Fig. 3.3a, a rectangular environment $\mathcal{E}$ is shown, and the positions of four charging stations, denoted by C_1 to C_4, are depicted as black dots. Fig. 3.3b shows the surf plot of $I(x_i, t)$ corresponding to the charging stations of Fig. 3.3a modeled by means of bump-like functions.

Remark 3.3. *The proposed energy model does not depend on the control input u_i of the robot. At first, this can seem too conservative, however in Section 3.2.4 it is shown why this choice was made and how it increases the robustness of the proposed persistification approach, ensuring that the robots will never run out of energy while executing the given task.*

3.1.4 Task Persistification

The compound model of robot and energy dynamics is given by the following set of differential equations:

$$\begin{cases} \dot{x}_i = f(x_i) + g(x_i)u_i \\ \dot{E}_i = F(x_i, E_i, t). \end{cases} \tag{3.7}$$

Indicating the augmented state of robot i by

$$z_i = \begin{bmatrix} x_i \\ E_i \end{bmatrix}, \tag{3.8}$$

the robot model (3.7) can be rewritten in the following control affine form:

$$\dot{z}_i = \hat{f}(z_i, t) + \hat{g}(z_i)u_i, \tag{3.9}$$

where

$$\hat{f}(z_i, t) = \begin{bmatrix} f(x_i) \\ F(x_i, E_i, t) \end{bmatrix} \quad \text{and} \quad \hat{g}(z_i) = \begin{bmatrix} g(x_i) \\ 0 \end{bmatrix}. \tag{3.10}$$

We will use

$$x = \begin{bmatrix} x_1^T, \dots, x_N^T \end{bmatrix}^T \in \mathcal{X}^{Nn}, \quad u = \begin{bmatrix} u_1^T, \dots, u_N^T \end{bmatrix}^T \in \mathcal{U}^{Nm} \tag{3.11}$$

to represent the joint states and inputs of the N robots performing the task to be persistified.

Let the task that has to be executed by the robots be encoded through the nominal input $\hat{u}_i(x, t), \quad i = 1, \dots, N$, or collectively as:

$$\hat{u} : \mathcal{X}^{Nn} \times \mathbb{R}_+ \to \mathcal{U}^{Nm}. \tag{3.12}$$

This modeling choice for the tasks that the robots have to execute is general insofar as

it encompasses both reactive feedback controllers, through the dependence of $\hat{u}$ from the state x, and controllers generated by a high-level planning strategy, through the dependence of $\hat{u}$ from the time t. Examples of such tasks include, for instance, the stabilization of a dynamical system, the coordinated control of multi-robot systems, as well as robot collision avoidance strategies. The execution of robotic tasks using this formulation will be showcased in Section 3.2.4.

Definition 3.4 (Task Persistification)**.** *A task is persistified if the energy E of the N robots deployed to execute it is such that, for all $i \in \{1, \dots, N\}$ and for all t,*

$$E_i(t) \in [E_{min},\ E_{chg}], \tag{3.13}$$

where E_{min} and E_{chg}, with $0 < E_{min} < E_{chg} < 1$, are desired minimum and maximum energy values.

One way to persistify a task is letting the robots execute the input u^* solution of the following optimization program:

$$\begin{aligned} u^*(x,t) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \|u - \hat{u}(x,t)\|^2 \\ \text{subject to } E_i(t) \in [E_{\text{min}},\ E_{\text{chg}}] \quad \forall i \in \{1, \dots, N\} \end{aligned} \tag{3.14}$$

for all $t \geq 0$. At time t, the value $u^*(x,t)$ is the control input closest to nominal input $\hat{u}(x,t)$ encoding the task, satisfying the desired energy constraints.

The task persistification is referred to as the process of turning a task characterized by the nominal input $\hat{u}$ to the persistified task characterized by the input u^*.

Remark 3.5. *Definition 3.4 is not tailored to the specific robot and environment models, which, as a matter of fact, can be quite different from the ones presented above, depending on the particular application that is considered.*

The specific choice of the cost in the optimization program (3.14) *allows us to synthesize*

control signals for the robots, which are as close as possible—in a squared norm sense—to the nominal input $\hat{u}$. Nevertheless, the concept of task persistification prescinds from the particular expression of the cost to minimize, and hinges on the energy constraint

$$E_i(t) \in [E_{min},\ E_{chg}] \quad \forall i \in \{1, \dots, N\},\ \forall t \geq 0. \tag{3.15}$$

3.2 Persistification Framework

The task persistification expressed as the optimization problem (3.14), will be realized by employing control barrier functions (CBFs). In the following subsection, we give a brief introduction to CBFs in their basic form. Then, in Section 3.2.2, we present the extensions that represent the main theoretical contribution of this chapter, namely a framework to deal with time-varying and high relative degree CBFs and control Lyapunov functions (CLFs). The results obtained in this section will be employed in Section 3.2.4 to formulate the optimization program whose solution achieves the persistification of robotic tasks.

3.2.1 Control Barrier Functions

Control barrier functions have been used with the goal of ensuring *safety*, intended as the invariance property of a subset of the state space, the *safe set*.

Definition 3.6 ([45])**.** *Let $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function with 0 a regular value, and $\mathcal{C}$ its zero superlevel set, i.e., $\mathcal{C} = \{x \in \mathbb{R}^n \ :\ h(x) \geq 0\}$. Then, for a control affine system*

$$\dot{x} = f(x) + g(x)u, \tag{3.16}$$

$x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, h is a Control Barrier Function (CBF) if there exists a locally Lipschitz extended class $\mathcal{K}$ function [116] α such that

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x) u + \alpha(h(x))\} \geq 0, \tag{3.17}$$

for all x in the interior of the set $\mathcal{C}$. $L_f h(x)$ and $L_g h(x)$ represent the Lie derivative of h in the directions of the vector fields f and g, respectively.

Starting from this basic definition, in the next sections we will develop tools required for the persistification of robotic tasks.

3.2.2 High Relative Degree CBFs and CLFs

In the previous section, we introduced the robot and energy models, and in Remark 3.3, we have pointed out that the energy dynamics do not depend explicitly on the robot input u_i. This is intended to be a conservative choice which increases the robustness of the persistification strategy. In fact, the rate of charge and discharge of the battery, obtained when $I(x_i, t) = 1$ and $I(x_i, t) = 0$, respectively, are designed to be the rates obtained when the robot input u_i attains its maximum norm. This would correspond, for instance, to the fastest discharge rate obtained when the actuators of the robot are absorbing maximum current. Then, the actual discharge rate will always be slower than the modeled one, increasing, this way, the robustness of the proposed persistification strategy against unmodeled phenomena which can occur in the environment, or unmodeled robot dynamics. Nevertheless, the gained robustness comes at the price of increasing the relative degree of the CBF h, defined as follows.

Definition 3.7 (Relative degree of a CBF, based on [47])**.** *Given the nonlinear system* (3.16)*, with f and g sufficiently smooth vector fields on a domain $\mathcal{D}$, the CBF $h : \mathbb{R}^n \to \mathbb{R}$ has relative degree ρ, $1 \leq \rho \leq n$, in $\mathcal{D}_0 \subset \mathcal{D}$ if the system*

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{3.18}$$

has a relative degree ρ, $\forall x \in \mathcal{D}_0$.

In the following, we give an example in which high relative degree CBFs [47] are

required. This example will be generalized in Theorem 3.9 for arbitrarily high relative-degree CBFs. Theorem 3.9 will be then applied in order to formulate an optimization program which realizes the proposed persistification strategy.

Example 3.8 (Cascade of CBFs)**.** *Let us consider the nonlinear dynamical system in control affine form* (3.16) *and a sufficiently smooth function* $h_1 : \mathbb{R}^n \to \mathbb{R}$ *with relative degree 2 (i.e.,* $L_g h_1(x) = 0$ *and* $L_g L_f h_1(x) \neq 0$*) that defines the superlevel set* $\mathcal{C}_1 = \{x \in \mathbb{R}^n : h_1(x) \geq 0\}$*. To prove the forward invariance of the set* $\mathcal{C}_1$*, we want* h_1 *to be a CBF for which the following must hold:*

$$L_f h_1(x) + \alpha_1(h_1(x)) \geq 0, \tag{3.19}$$

where α_1 *is a continuously differentiable extended class* $\mathcal{K}$ *function, and we used the fact that* h_1 *has relative degree 2. Then, we can define an additional function*

$$h_2(x) = L_f h_1(x) + \alpha_1(h_1(x)) \tag{3.20}$$

whose zero superlevel set is $\mathcal{C}_2 = \{x \in \mathbb{R}^n : h_2(x) \geq 0\}$*. If there exists a positive constant* γ_1 *and a locally Lipschitz extended class* $\mathcal{K}$ *function* α_2 *such that*

$$\sup_{u \in \mathcal{U}} \left\{ L_f^2 h_1(x) + L_g L_f h_1(x)\, u + \gamma_1 L_f h_1(x) + \alpha_2(h_2(x)) \right\} \geq 0, \tag{3.21}$$

then the function h_2 *is a CBF. The condition in* (3.21) *has been obtained using the definition* (3.20) *and employing* $\alpha_1(s) = \gamma_1 s$*. Its expression for an arbitrary high relative degree will be given in the next theorem. The existence of the CBF* $h_2(x)$ *ensures the forward invariance of the set* $\mathcal{C}_2$*, which, in turn, ensures the existence of the CBF* $h_1(x)$*. The forward invariance of the set* $\mathcal{C}_1$ *is thus proved.*

The technique shown in example 3.8 is generalized in the following theorem.

Theorem 3.9. *Given a dynamical system* (3.16)*, a sufficiently smooth CBF* $h_1(x)$ *with relative degree* ρ *and a CBF* $h_\rho(x)$ *which can be evaluated recursively starting from* $h_1(x)$ *using the following equation*

$$h_{n+1}(x) = \dot{h}_n(x) + \alpha_n(h_n(x)), \quad 1 \leq n < \rho, \tag{3.22}$$

with α_n *continuously differentiable extended class* $\mathcal{K}$ *functions, we define the set* $K_\rho(x)$ *as*

$$\begin{aligned} K_\rho(x) = \Big\{ & u \in \mathcal{U} : \; L_f^\rho h_1(x) + L_g L_f^{\rho-1} h_1(x) u \\ & + \sum_{i=1}^{\rho-1} \sum_{\mathscr{C} \in \binom{\rho-1}{i}} \prod_{j \in \mathscr{C}} \frac{\partial \alpha_j}{\partial h_j} L_f^{\rho-i} h_1(x) + \alpha_\rho(h_\rho(x)) \geq 0 \Big\}, \end{aligned} \tag{3.23}$$

where $\binom{\rho-1}{i}$ *is the set of* i*-combinations from the set* $\{1, \ldots, \rho-1\} \subset \mathbb{N}$ *and* α_ρ *is a locally Lipschitz extended class* $\mathcal{K}$ *function. Then, any Lipschitz continuous controller* $u \in K_\rho(x)$ *will render the set* $\mathcal{C}_1 = \{x \in \mathbb{R}^n : \; h_1(x) \geq 0\}$ *forward invariant.*

Remark 3.10. *Employing a cascade of CBFs as shown in Example 3.8 and in Theorem 3.9 is a technique which can be used not only to prove set forward invariance, but also stability of dynamical systems using high relative degree Lyapunov functions, as will be shown in the following. Set forward invariance and stability will be used, in Section 3.2.4, to ensure that the energy stored in the battery of the robots performing a task is never depleted, realizing, this way, the desired task persistification.*

Note that the energy model proposed in Section 3.1.3 is time-dependent, as it depends on the environment model, through the value $I(x_i, t)$ at x_i at time t. As we will use CBFs to ensure the persistent execution of a task, we now extend the notion of CBFs to the case in which the function h that defines the safe set $\mathcal{C}$ explicitly depends on time.

For the nonlinear control affine system (3.16), we wish to ensure the forward invariance of a time-varying set $\mathcal{C}(t) \subset \mathbb{R}^n$ defined by the superlevel set of a function $h : \mathbb{R}^n \times \mathbb{R}_+ \to$

$\mathbb{R}$ as:

$$\mathcal{C}(t) = \{x \in \mathbb{R}^n : \ h(x,t) \geq 0\}. \tag{3.24}$$

Notice that, due to the locally Lipschitz continuity assumption on the system dynamics, the solution $x(t)$ is guaranteed to exist in an interval $[t_0, t_0 + \Delta t_{\max}(x_0)]$, where $x_0 = x(t_0)$, and $\Delta t_{\max}(x_0) \in (0, \infty]$. We then extend the definition of CBFs given in [116] to the time-varying case as follows.

Definition 3.11 (Time-Varying CBFs)**.** *Given a dynamical system* (3.16) *and a set* $\mathcal{C}(t)$ *defined in* (3.24)*, the function* h *is a time-varying CBF defined on* $\mathcal{D} \times \mathbb{R}_+$*, with* $\mathcal{C}(t) \subseteq \mathcal{D} \subset \mathbb{R}^n$*, if there exists a locally Lipschitz extended class* $\mathcal{K}$ *function* α *such that,* $\forall x \in \mathcal{D}$*,* $\forall t \in [t_0, t_0 + \Delta t_{\max}(x_0)]$*,*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h}{\partial t} + L_f h(x,t) + L_g h(x,t)\, u + \alpha(h(x,t)) \right\} \geq 0. \tag{3.25}$$

Starting from the condition in (3.25), we can define the set:

$$K(x,t) = \left\{ u \in \mathcal{U} : \ \frac{\partial h}{\partial t} + L_f h(x,t) + L_g h(x,t)\, u + \alpha(h(x,t)) \geq 0 \right\}. \tag{3.26}$$

The following lemma ensures that the set $\mathcal{C}(t)$, defined in (3.24), is rendered forward invariant by the application of a control input $u \in K(x,t)$. This result will be used in Section 3.2.4 to express the constraints on the energy in (3.14) in terms of the control input u.

Lemma 3.12. *Given a set* $\mathcal{C}(t)$ *defined as in* (3.24)*, if* h *is a time-varying CBF on* $\mathcal{D} \times \mathbb{R}_+$*, then any Lipschitz continuous controller* $u \in K(x,t)$*, where* $K(x,t)$ *is given in* (3.26)*, will render the set* $\mathcal{C}(t)$ *forward invariant, namely* $x(t_0) \in \mathcal{C}(t_0) \implies x(t) \in \mathcal{C}(t), \ \forall t \in [t_0, t_0 + \Delta t_{\max}(x_0)]$.

Remark 3.13. *In case* $\frac{\partial h}{\partial t} = 0$ *and* $L_g h(x,t) = 0$ *we are not able to ensure the existence of a control input such that* (3.25) *holds, condition on which Lemma 3.12 relies. This case*

can be tackled by making use of a cascade of control barrier functions and the result of Theorem 3.9.

So far, we have described the condition in which the robots are executing the assigned task and we want them to keep their energy level E_i within a certain interval $[E_{\text{min}}, E_{\text{chg}}]$. This is realized, employing CBFs, by letting the robots reach regions of the state space where the field I introduced in (3.2) is larger than I_{c}. In these regions, as observed in Remark 3.1, $\dot{E}_i > 0$ and the robots are charging.

The use of CBFs will allow the robots to keep the energy stored in their batteries within the desired interval by synthesizing a controller by solving an optimization problem at each time instant. Nevertheless, although computationally efficient, this point-wise in time approach does not allow the robots to plan their charging strategy. Therefore, in order to make sure that the robots leave the *charging stations*—intended, in a broader sense, as the regions of the state space where $I(x_i, t) \geq I_{\text{c}}$—only when their battery is fully charged, i.e., when $E_i \approx E_{\text{chg}}$, we will make use of control Lyapunov functions (CLFs).

Similarly to what happens with CBFs encoding energy constraints, since the input u_i does not directly show up in the expression of $\dot{E}_i$, a CLF defined to fully recharge the robots' battery will have relative degree higher than 1. Therefore, in the following, we will proceed analogously to what has been done for high relative degree CBFs to handle the case of high relative degree CLFs.

Suppose that, besides the forward invariance of a set, one would like to stabilize the dynamical system (3.16) around the origin $x = 0$ using a CLF. Similarly to what has been done for the CBFs, the existence of a CLF $V : \mathbb{R}^n \to \mathbb{R}$ suggests the definition of the following set:

$$K_V(x) = \{u \in \mathcal{U} : \; -L_f V(x) - L_g V(x)\, u \geq 0\}\,. \tag{3.27}$$

It is easy to see how the choice of a control input $u \in K_V(x)$ will stabilize the system

around $x = 0$.

Remark 3.14. *In case $L_g V(x) = 0$, i.e., the relative degree of the Lyapunov function is greater than 1, we cannot ensure the stability of the origin.*

In Example 3.8 and in Theorem 3.9, a technique for dealing with high relative degree control barrier functions has been introduced: we will proceed here in a similar fashion. We first give an example that shows how to construct high-relative degree CLFs by employing CBFs. Then, we generalize this construction in Theorem 3.16.

Example 3.15 (Constructing high relative degree CLFs using CBFs)**.** *Let us consider the nonlinear dynamical system in control affine form* (3.16) *and a sufficiently smooth function $V : \mathbb{R}^n \to \mathbb{R}$ with relative degree 2, i.e., $L_g V(x) = 0$ and $L_g L_f V(x) \neq 0$. In order for V to be a CLF we must have $-L_f V(x) > 0$. We can then define the CBF $h_1(x) = -L_f V(x)$, and its superlevel set $\mathcal{C}_1 = \{x \in \mathbb{R}^n \colon -L_f V(x) \geq 0\}$, and let $u \in K_2'(x) = \big\{u \in \mathcal{U} : \ -L_f^2 V(x) - L_g L_f V(x)\, u + \alpha(-L_f V(x)) \geq 0\big\}$. where α is a locally Lipschitz class $\mathcal{K}$ function. This way, the set $\mathcal{C}_1$ can be rendered forward invariant. Consequently, the existence of the CLF $V(x)$ guarantees that the origin $x = 0$ is (asymptotically) stable.*

Theorem 3.16. *Consider the dynamical system* (3.16)*, a CLF $V(x)$ with relative degree ρ defined with the objective of stabilizing the system state x to x^*, and a CBF $h_\rho(x)$ which can be evaluated from $V(x)$ using the following recursive formula:*

$$\begin{cases} h_1(x) = -L_f V(x) \\ h_{n+1}(x) = \dot{h}_n(x) + \alpha_n(h_n(x)), \quad 1 \leq n < \rho, \end{cases} \tag{3.28}$$

where α_n are continuously differentiable extended class $\mathcal{K}$ functions. In addition, assume that $\{x^*\}$ is the largest invariant set in $\partial\mathcal{C}_1 = \{x \colon h_1(x) = 0\}$, boundary of the set $\mathcal{C}_1 =$

$\{x\colon h_1(x) \geq 0\}$. *Then, any Lipschitz continuous controller*

$$u \in K'_\rho(x) = \Big\{ u \in \mathcal{U} : \; -L_f^\rho V(x) - L_g L_f^{\rho-1} V(x) u + \sum_{i=1}^{\rho-2} \sum_{\mathscr{C} \in \binom{\rho-2}{i}} \prod_{j \in \mathscr{C}} \frac{\partial \alpha_j}{\partial h_j} (-L_f^{\rho-1-i} V(x)) + \alpha_\rho(h_\rho(x)) \geq 0 \Big\}, \tag{3.29}$$

where α_ρ is a locally Lipschitz class $\mathcal{K}$ function, will asymptotically stabilize the system to $x = x^*$.

3.2.3 Integral Control Barrier Functions

The model of battery SOC dynamics presented in (3.4) depends on the control input u_i, and in general it does so in a non-control-affine fashion. As such, it is easy to see that applying the standard CBF approach would result in differential constraints which are not affine in u_i. As will be shown in the next section, affine input constraints are amenable in order to synthesize robot controllers under real-time constraints at high frequencies commonly used in the control loops of modern robotic platforms. To circumvent this issue, control dependent control barrier functions [117] or the more general formulation of integral control barrier functions (I-CBFs) [118], defined below, can be employed.

Definition 3.17 ([118])**.** *For the system $\dot{x} = f(x, u)$, with corresponding safe set $\mathcal{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ defined as the 0-superlevel set of a function $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ with 0 a regular value: $\mathcal{S} = \{(x, u) \in \mathbb{R}^n \times \mathbb{R}^m \; : \; h(x, u) \geq 0\}$. Then, h is an integral control barrier function (I-CBF) if for any $(x, u) \in \mathbb{R}^n \times \mathbb{R}^m$ and $t \geq 0$:*

$$p(x, u) = 0 \qquad \Rightarrow \qquad d(x, u, t) \leq 0. \tag{3.30}$$

With this definition, the following theorem gives sufficient conditions for the forward invariance of sets defined by an I-CBF.

Theorem 3.18 ([118]). *Consider the control system $\dot{x} = f(x, u)$, with $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, and suppose that there is a dynamically defined controller: $\dot{u} = \phi(x, u, t)$. If the safe set $\mathcal{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ is defined by an integral control barrier function, $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, then modifying the dynamically defined controller to be of the form:*

$$\dot{u} = \phi(x, u, t) + v^*(x, u, t) \tag{3.31}$$

with v^* the solution to the QP:

$$\begin{aligned} v^*(x, u, t) = &\underset{v \in \mathbb{R}^m}{\operatorname{argmin}} ||v||^2 \\ &\text{subject to } p(x, u)^T v \geq d(x, u, t) \end{aligned} \tag{3.32}$$

where

$$p(x, u) := \frac{\partial h}{\partial u}^T (x, u) \tag{3.33}$$

$$d(x, u, t) := -\frac{\partial h}{\partial x}(x, u) f(x, u) - \frac{\partial h}{\partial u}(x, u)\phi(x, u, t) - \gamma(h(x, u)), \tag{3.34}$$

results in safety, i.e., the control system $\dot{x} = f(x, u)$ with the dynamically defined controller (3.31) *results in $\mathcal{S}$ being forward invariant: if $(x(0), u(0)) \in \mathcal{S}$ then $(x(t), u(t)) \in \mathcal{S}$ for all $t \geq 0$.*

There are cases in which a dynamically defined controller $\dot{u} = \phi(x, u, t)$ is available, as, e.g., in the case of tracking and regulation control objectives [119]. In the case when a, possibly time-varying, state feedback nominal controller

$$u = k(x, t). \tag{3.35}$$

is considered, one could set

$$\dot{u} = \underbrace{L_f k(x,u) + \frac{\partial k}{\partial t}}_{=:\phi(x,u,t)} + v^*, \tag{3.36}$$

where v^* is given by (3.32) in order to guarantee the safety of a set $\mathcal{S}$. This choice, however, may cause $u(t)$ to diverge more and more from its nominal expression $k(x,t)$, due to the fact that v^* from (3.32) is the minimizer of the difference between the time derivative of the input functions:

$$\|v\| = \|\dot{u} - \phi(x,u,t)\|, \tag{3.37}$$

i.e. the difference between the derivatives of u and ϕ rather than the input functions themselves. To solve this issue, in the following, we propose a dynamically defined control law which, together with the integral control barrier functions introduced above, allows for the efficient implementation of safety controllers, starting from static state feedback input controllers.

Theorem 3.19. *Consider the system in $\dot{x} = f(x,u)$ and the time-varying nominal feedback controller in* (3.35)*. Consider the integral control barrier function $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defined to ensure the safety of the set $\mathcal{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ defined as its 0-superlevel set. Then, the dynamically defined controller*

$$\dot{u} = \underbrace{L_f k(x,t) + \frac{\partial k}{\partial t} + \frac{\alpha}{2}\left(k(x,t) - u\right)}_{=:\phi_k(x,u,t)} + v^*, \tag{3.38}$$

where v^* is given by (3.32)*, will ensure the safety of the set $\mathcal{S}$, as well as the tracking of the nominal feedback controller* (3.35) *whenever the nominal controller $\phi_k(x,u,t)$ is safe.*

Remark 3.20. *The variable $\dot{u}$ only appears in the software implementation of the safety controller for the system $\dot{x} = f(x,u)$. The input fed to the system is its integral, $u(t)$. Therefore, the value α in the expression of the dynamically defined controller* (3.38) *can*

be chosen arbitrarily large (without introducing numerical errors). As can be noticed in the proof of 3.19, the larger the value of α is, the faster the convergence of the nominal controller u to $k(x,t)$, when no safety-related modifications of $\dot{u}$ are required (i.e., when $v^* = 0$).

Remark 3.21. *Integrating the expression of the dynamically defined controller in* (3.38) *with respect to time, we get:*

$$u(t) = \int_0^t \left(L_f k(x(\tau), u(\tau)) + \frac{\partial k}{\partial t} \right) d\tau + \underbrace{\frac{\alpha}{2} \int_0^t \left(k(x(\tau), t(\tau)) - u(\tau) \right) d\tau}_{\textit{Integral control}}, \tag{3.39}$$

where we explicitly recognize once again the centrality of the integral controller.

Remark 3.22. *A similar construction can be employed when dealing with non-control-affine dynamical systems as well. In fact, in case the dynamical system to control is control affine, the classic construction of an optimization-based controller proposed in [46] results in a convex quadratic program which minimizes the difference between u and $k(x,t)$. Therefore, it lends itself to an efficient online implementation. However, when the dynamics are not control affine, the same construction can lead to non-convex optimization problems, preventing the online synthesis of safety controllers. As will be shown in the next section, by dynamically extending the system, a construction similar to* (3.38) *can be leveraged.*

In the next section, it will be shown how high-relative degree CBFs and CLFs, as well as I-CBFs, can be utilized to enforce constraints on the energy and the SOC of robot batteries in a computationally efficient fashion.

3.2.4 Application to Robotic Tasks

In view of what has been introduced in the previous section, in this section we show that the persistification of robotic tasks, specified in Definition 3.4, can be framed as a constrained optimization problem.

In order for the robots to be able to perpetually execute the given task, their energy $E_i,\ i = 1, \ldots, N$ must be strictly greater than zero at each point in time. Moreover, in order to extend the battery life of the specific types of batteries used in robotic applications, the lower bound for the residual energy should not be too low in order to protect the battery from deep discharge [120]. Considering the robot model (3.7), the constraints to control the residual energy can be formally encoded by the following CBF related to robot i:

$$h_{i1}(z_i) = (E_{\text{chg}} - E_i)(E_i - E_{\text{min}}), \tag{3.40}$$

where E_{chg} and E_{min} are the upper and lower bounds between which we want the energy E_i to be confined, corresponding to charged and depleted battery, respectively.

Following the procedure adopted in Example 3.8, we start by evaluating the time derivative of $h_{i1}(z_i)$:

$$\begin{aligned} \dot{h}_{i1}(z_i, t) &= \frac{\partial h_{i1}}{\partial t} + L_f h_{i1}(z_i) + L_g h_{i1}(z_i)\, u_i \\ &= \begin{bmatrix} \dfrac{\partial h_{i1}}{\partial x_i} & \dfrac{\partial h_{i1}}{\partial E_i} \end{bmatrix} f(z_i, t) + \begin{bmatrix} \dfrac{\partial h_{i1}}{\partial x_i} & \dfrac{\partial h_{i1}}{\partial E_i} \end{bmatrix} g(z_i)\, u_i \\ &= (E_{\text{chg}} + E_{\text{min}} - 2E_i) F(x_i, E_i, t), \end{aligned} \tag{3.41}$$

which does not depend on u_i as the relative degree of $h_{i1}(z_i)$ is 2. Therefore, we define the CBF $h_{i2}(z_i, t)$ as done in (3.20), namely:

$$h_{i2}(z_i, t) = \dot{h}_{i1}(z_i, t) + \gamma_{i1} h_{i1}(z_i), \tag{3.42}$$

in which the locally Lipschitz class $\mathcal{K}$ function has been chosen to be the linear function $\alpha_{i1}(s) = \gamma_{i1} s$, with $\gamma_{i1} > 0$.

Using Theorem 3.9 and Lemma 3.12, we can define the set of control inputs u_i that will render the set $\mathcal{C}_{i1} = \{z_i \in \mathbb{R}^{n+1} :\ h_{i1}(z_i) \geq 0\} = \{E_i \in \mathbb{R} :\ E_{\text{min}} \leq E_i \leq E_{\text{chg}}\}$ forward

invariant:

$$K_{2i}(z_i,t) = \Big\{ u_i \in \mathcal{U} \,:\, \frac{\partial h_{i2}}{\partial t} + L_f^2 h_{i1}(z_i) \\ + L_g L_f h_{i1}(z_i)\, u_i + \gamma_{i1} L_f h_{i1}(z_i) + \gamma_{i2} h_{i2}(z_i,t) \geq 0 \Big\}, \tag{3.43}$$

in which $\alpha_{i2}(h_{i2}(z_i,t)) = \gamma_{i2} h_{i2}(z_i,t)$, with $\gamma_{i2} > 0$, is the locally Lipschitz extended class $\mathcal{K}$ function in (3.21).

For single integrator robot dynamics and the CBFs (3.40) and (3.42), the expressions of $\frac{\partial h_{i2}}{\partial t}$, $L_f^2 h_{i1}(z_i)$ and $L_g L_f h_{i1}(z_i)$ required to evaluate (3.45) are given by:

$$\begin{aligned} \frac{\partial h_{i2}}{\partial t} &= (E_{\text{chg}} + E_{\text{min}} - 2E_i)\frac{\partial F}{\partial w}\frac{\partial w}{\partial I}\frac{\partial I}{\partial t} = \\ &= (E_{\text{chg}} + E_{\text{min}} - 2E_i) k w^2 \frac{1-E_i}{E_i} \lambda e^{-\lambda(I(x_i,t)-I_{\text{c}})}\frac{\partial I}{\partial t} \\ L_f^2 h_{i1}(z_i) &= \Big(-2F(x_i,E_i,t) + \\ &\qquad + (E_{\text{chg}} + E_{\text{min}} - 2E_i)\frac{\partial F}{\partial E_i}\Big) F(x_i,E_i,t) \\ L_g L_f h_{i1}(z_i) &= (E_{\text{chg}} + E_{\text{min}} - 2E_i)\frac{\partial F}{\partial w}\frac{\partial w}{\partial I}\frac{\partial I}{\partial x_i} = \\ &= (E_{\text{chg}} + E_{\text{min}} - 2E_i) k w^2 \frac{1-E_i}{E_i} \lambda e^{-\lambda(I(x_i,t)-I_{\text{c}})}\frac{\partial I}{\partial x_i} \end{aligned} \tag{3.44}$$

Note that, due to the control affine form of (3.9), $u_i \in K_{2i}(z_i,t)$ is an affine constraint in u_i and therefore it can be written as:

$$A_{\text{CBF}i}(z_i,t) u_i \leq b_{\text{CBF}i}(z_i,t), \tag{3.45}$$

with $A_{\text{CBF}i}(z_i,t) = -L_g L_f h_{i1}(z_i)$ and $b_{\text{CBF}i}(z_i,t) = \frac{\partial h_{i2}}{\partial t} + L_f^2 h_{i1}(z_i) + \gamma_{i1} L_f h_{i1}(z_i) + \gamma_{i2} h_{i2}(z_i,t)$.

Remark 3.23. *Recalling the expression of $\dot{E}$ introduced in* (3.5)*, the behavior resulting by enforcing the constraint* (3.45) *is the following: when the battery level E_i is getting close to its minimum value E_{min}, robot i will drive towards areas of the environment $\mathcal{E}$ where the value of the function $I(x_i,t)$ is such that $\dot{E}_i \geq 0$, i.e., robot i starts recharging its battery.*

During operation, the battery of each robot continuously discharges according to the dynamics in (3.5). The mere application of the constraint (3.45) prevents the battery level to go lower than E_{min} or higher than E_{chg}. However, it does not ensure that the battery will be completely charged before robot i leaves areas of the environment where $\dot{E}_i > 0$. This behavior is desirable for two reasons: (i) these kinds of charging/discharging cycles will extend the life of robot batteries [114, 120], and (ii) it is more efficient from the point of view of the time spent by the robots recharging their batteries, allowing them to deviate from the nominal task input for less time. A quantitative justification of the second reason is given in Section 3.3.1.

This behavior can be encoded using a CLF whose objective is that of driving the energy E_i to E_{chg}. We can then define the following CLF:

$$V_i(z_i) = (E_{\text{chg}} - E_i)^2, \tag{3.46}$$

related to robot i, whose time derivative is given by

$$\begin{aligned} \dot{V}_i(z_i, t) &= \frac{\partial V_i}{\partial t} + L_f V_i(z_i) + L_g V_i(z_i)\, u_i \\ &= \begin{bmatrix} \dfrac{\partial V_i}{\partial x_i} & \dfrac{\partial V_i}{\partial E_i} \end{bmatrix} f(z_i, t) + \begin{bmatrix} \dfrac{\partial V_i}{\partial x_i} & \dfrac{\partial V_i}{\partial E_i} \end{bmatrix} g(z_i)\, u_i \\ &= -2(E_{\text{chg}} - E_i) F(x_i, E_i, t). \end{aligned} \tag{3.47}$$

As in the case of $h_{i1}(z_i)$ in the previous section, here $V_i(z_i)$ has relative degree 2 and therefore its time derivative is not a function of the control input u_i. Proceeding as before, we define the CBF $h_{i1}(z_i, t) = -L_f V_i(z_i)$. Its superlevel set $\mathcal{C}(t) = \{z_i \in \mathbb{R}^{n+1} \colon h_{i1}(z_i, t) \geq 0\} = \{z_i \in \mathbb{R}^{n+1} \colon \dot{V}_i(z_i, t) \leq 0\}$ is the set in which the value of the function $V_i(z_i)$ is not increasing. Its boundary $\partial\mathcal{C}(t) = \{z_i \in \mathbb{R}^{n+1} \colon h_{i1}(z_i, t) = 0\}$ is the set where $\dot{V}_i(z_i, t) = 0$ which, if $\dot{E} \neq 0$, coincides with the n-dimensional manifold $\{z_i \in \mathbb{R}^{n+1} \colon E_i = E_{\text{chg}}\}$.

Therefore, by Theorem 3.16, if

$$u_i \in K'_{2i}(z_i,t) = \Big\{ u_i \in \mathcal{U} \,:\, \frac{\partial h_{i2}}{\partial t} - L_f^2 V_i(z_i) - L_g L_f V_i(z_i)\, u_i + \gamma_{i2} h_{i2}(z_i,t) \geq 0 \Big\}, \tag{3.48}$$

with $\gamma_{i2} > 0$, the value of E_i will asymptotically converge to E_{chg}.

For single integrator robot dynamics and the CLF (3.46), the expressions of $\frac{\partial h_{i2}}{\partial t}$, $L_f^2 V_i$ and $L_g L_f V_i$ required to evaluate (3.50) are given by:

$$\begin{aligned}
\frac{\partial h_{i2}}{\partial t} &= -2(E_{\text{chg}} - E_i)\frac{\partial F}{\partial w}\frac{\partial w}{\partial I}\frac{\partial I}{\partial t} = \\
&= -2(E_{\text{chg}} - E_i) k w^2 \frac{1 - E_i}{E_i} \lambda e^{-\lambda(I(x_i,t) - I_{\text{c}})} \frac{\partial I}{\partial t}, \\
L_f^2 V_i(z_i) &= \left(2F(x_i, E_i, t) - 2(E_{\text{chg}} - E_i)\frac{\partial F}{\partial E} \right) F(x_i, E_i, t), \\
L_g L_f V_i(z_i) &= -2(E_{\text{chg}} - E_{\text{min}})\frac{\partial F}{\partial w}\frac{\partial w}{\partial I}\frac{\partial I}{\partial x_i} = \\
&= -2(E_{\text{chg}} - E_{\text{min}}) k w^2 \frac{1 - E_i}{E_i} \lambda e^{-\lambda(I(x_i,t) - I_{\text{c}})} \frac{\partial I}{\partial x_i}.
\end{aligned} \tag{3.49}$$

Note that $u_i \in K'_{2i}(z_i,t)$ is an affine constraint in u_i, and therefore it can be written as:

$$A_{\text{CLF}i}(z_i,t) u_i \leq b_{\text{CLF}i}(z_i,t), \tag{3.50}$$

with $A_{\text{CLF}i}(z_i,t) = L_g L_f V_i(z_i)$ and $b_{\text{CLF}i}(z_i,t) = \frac{\partial h_{i2}}{\partial t} - L_f^2 V_i(z_i) + \gamma_{i2} h_{i2}(z_i,t)$.

In order to combine the CBF constraints (3.45) and the CLF constraints (3.50), letting $z = \left[z_1^T, \ldots, z_N^T \right]^T$, the following nonlinear program can be formulated:

$$\begin{aligned}
u^*(z,t) = \underset{u,\delta}{\operatorname{argmin}} \; & \|u - \hat{u}(x,t)\|^2 + \delta^T \kappa \delta \\
\text{subject to } & \begin{bmatrix} A_{\text{CBF}}(z,t) & 0_N \\ A_{\text{CLF}}(z,t) & -I_N \end{bmatrix} \begin{bmatrix} u \\ \delta \end{bmatrix} \leq \begin{bmatrix} b_{\text{CBF}}(z,t) \\ b_{\text{CLF}}(z,t) \end{bmatrix},
\end{aligned} \tag{3.51}$$

where

$$
\begin{aligned}
A_{\text{CBF}}(z_i,t) &= \operatorname{diag}\left(A_{\text{CBF1}}(z_i,t),\ \ldots,\ A_{\text{CBF}N}(z_i,t)\right),\\
A_{\text{CLF}}(z_i,t) &= \operatorname{diag}\left(A_{\text{CLF1}}(z_i,t),\ \ldots,\ A_{\text{CLF}N}(z_i,t)\right),\\
b_{\text{CBF}}(z_i,t) &= \begin{bmatrix} b_{\text{CBF1}}(z_i,t) \\ \vdots \\ b_{\text{CBF}N}(z_i,t) \end{bmatrix},\ b_{\text{CLF}}(z_i,t) = \begin{bmatrix} b_{\text{CLF1}}(z_i,t) \\ \vdots \\ b_{\text{CLF}N}(z_i,t) \end{bmatrix},
\end{aligned}
\tag{3.52}
$$

$u = [u_1^T, \ldots, u_N^T]$, and I_N and 0_N are $N \times N$ identity and zero matrices, respectively. Moreover, $\delta = [\delta_1, \ldots, \delta_N]^T \in \mathbb{R}^N$ is a vector of relaxation parameters introduced to make the constraints in (3.51) always feasible. The matrix $\kappa = \operatorname{diag}(\kappa_i)$ is a diagonal, positive definite, weighting matrix for δ. Furthermore, the nominal input $\hat{u}(x,t)$ is what encodes the task introduced in Definition 3.4.

Both (3.45) and (3.50) are affine functions of the optimization variables. Moreover, the cost $\|u - \hat{u}(x,t)\|^2 + \delta^T \kappa \delta$ is a convex quadratic form. Hence, (3.51) is a convex quadratic program (QP) and, as such, can be efficiently solved—see, e.g., [121]—and employed to generate controllers in an online fashion as shown, for instance, in [122].

Remark 3.24. *As discussed in Section 3.1, the proposed persistification approach also works when the N robots are not homogeneous, i.e., when they are characterized by different dynamic models. In fact, in this case, once the constraints of the optimization program* (3.51) *have been changed accordingly, the solution $u^*(z,t)$ guarantees the heterogeneous multi-robot system persistently executes the task characterized by the nominal input $\hat{u}$.*

Remark 3.25. *In order to achieve the desired charging behavior, discussed in Section 3.2, κ_i can be made a function of E_i in such a way that robot i charges up to $E_i = E_{chg}$ once it started charging and, at the same time, discharge down to $E_i = E_{min}$ while operating. A candidate mapping $\kappa_i(E_i) : [0,1] \to [0, \kappa_{i,\text{MAX}}]$ is depicted in Fig. 3.4. Note that the function κ_i changes according to the sign of $\dot{E}_i$. This way, the weight of the corresponding relaxation parameter δ_i changes when the energy E_i reaches E_{chg} or E_{min}.*

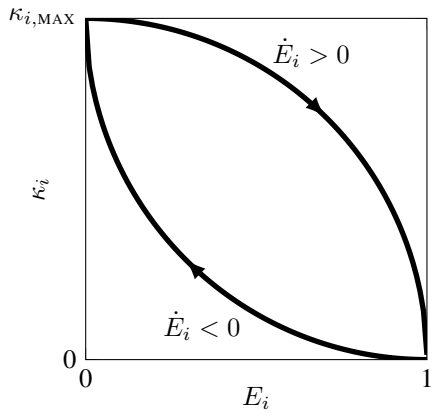


Figure 3.4: Example of the function $\kappa_i(E_i)$ that can be employed in order to let robot i charge its battery up to E_{chg} once it has started charging. The two branches of the curve are labeled with $\dot{E}_i > 0$ and $\dot{E}_i < 0$: at a given value of E_i, the value of κ_i is higher for a robot that is recharging its battery ($\dot{E}_i > 0$) compared to the one of a robot for which $\dot{E}_i < 0$. This way, once a robot starts recharging its battery, the weight of its corresponding component of the vector δ in (3.51) is larger. This ensures that the robot charges its battery until E_{chg}.

We now have the necessary constructions to state the proposition which ensures the task persistification defined in Definition 3.14.

Proposition 3.26. *The control law $u^*(z,t)$, solution of the QP* (3.51)*, makes the robots execute the persistified task corresponding to the task encoded through the control input $\hat{u}(x,t)$.*

In case the battery SOC is to be employed to represent the amount of energy stored in the robot batteries, the following CBF can be used to leverage the SOC dynamics (3.4) in keeping its value above a minimum threshold SOC_{min}:

$$h_x(SOC_i) = SOC_i - SOC_{\text{min}}, \tag{3.53}$$

where SOC_i is the value of the SOC of the battery of robot i. Proceeding similarly to [118], the robot system is dynamically extended by the additional state equation

$$\dot{u}_i = \phi_{\hat{u}_i}(x_i, u_i, t) + v_i, \tag{3.54}$$

where

$$\phi_{\hat{u}_i}(x_i, u_i, t) = L_f \hat{u}_i(x_i, t) + L_g \hat{u}_i(x_i, t) u_i + \frac{\partial \hat{u}_i}{\partial t} + \frac{\alpha}{2} \left(\hat{u}_i(x_i, t) - u \right), \tag{3.55}$$

as in (3.38). Then, one can define the I-CBF

$$h_u(SOC, u_i, t) = \dot{h}_x(SOC_i, u_i, t) + h_x(SOC_i), \tag{3.56}$$

that leads to the following differential inequality—equivalent to $p(x, u)^T v \geq d(x, u, t)$ in (3.32) and affine in v_i—which can be inserted in (3.51) to allow the robots to keep the value of SOC of their batteries above the minimum threshold SOC_{min}:

$$\frac{\partial f_{SOC}}{\partial u_i} v_i + \frac{\partial f_{SOC}}{\partial t} + f_{SOC}(u_i, t) \geq 0. \tag{3.57}$$

In order to fully substitute the optimization variable u_i with v_i in (3.51), a similar strategy has to be adopted to incorporate the overcharge protection using a SOC-dependent CLF.

In the next section, the developed theoretical control framework will be validated by means of simulations and experiments. Two robotic tasks are introduced and the results of their persistent implementation are reported.

3.3 Simulations and Experimental Results

In this section, two robotic tasks, whose persistent application is particularly relevant, are presented to showcase the persistification strategy developed in this chapter. The tasks consist in environment exploration and environment surveillance. Both tasks are typically required to be executed for a long period of time. In the case of environment exploration, the long execution time can be due to the size and/or the dynamic nature of the environment to explore [123, 124]. As regards the environment surveillance, the time-scale of the observed environment phenomena is the factor that determines the length of the task.

Nowadays, longevity is still a limiting factor for the deployment of robotic systems for environment surveillance and monitoring, as discussed in [125]. The persistification of these two tasks through the control framework described in this chapter is discussed in Section 3.3.1 and Section 3.3.2, respectively.

3.3.1 Environment Exploration

The first application that is considered is that of environment exploration. There are many approaches to this task in literature and many solutions have been proposed. Here we consider the one presented in [126], since the result of a trajectory optimization problem provides directly the nominal inputs to the robots. The optimal trajectories are evaluated by minimizing the distance from ergodicity [127]. This results in a trajectory that, instead of maximizing information in a greedy way, distributes information according to its probability density function defined over the environment.

Following what is presented in [126], let us start by defining the ergodic metric $\epsilon = \sum_{k=0}^{K} \Lambda_k \left|c_k - \varphi_k\right|^2$ [127]. where c_k are the time-averaged Fourier coefficients of the trajectory, φ_k are the Fourier coefficients of a spatial distribution of information $\phi : \mathcal{E} \mapsto \mathbb{R}_+$, $\mathcal{E}$ being the environment to explore, and $\Lambda_k = 1/(1+\xi^T\xi)^{\frac{3}{2}}$, with $\xi \in \mathscr{Z} = \{0, 1, \ldots, K-1\} \times \{0, 1, \ldots, K-1\}$, K being the number of employed Fourier basis functions.

By minimizing this ergodic measure at time $\bar{t}$ over a time horizon T, the nominal trajectory $\hat{x}_i(t)$ for $t \in [\bar{t},\ \bar{t}+T]$ of each robot is obtained. Solving this optimization problem at every time instant, in a model predictive control fashion, provides the nominal input $\hat{u}_i$ that is to be executed at time $\bar{t}$ in order to track the trajectory $\hat{x}_i$ [126]. This $\hat{u}_i$ can be plugged in the QP defined in (3.51) allowing, this way, a straightforward application of the persistification framework presented in this chapter to the environment exploration task.

The persistent environment exploration task has been implemented and tested in a simulation environment. For the simulated experiment a planar robot is given the task of exploring an environment $\mathcal{E}$ on which a spatial distribution of information has been defined.

The information is assumed to be distributed according to the Gaussian density function $\phi : x \in \mathcal{E} \subset \mathbb{R}^2 \mapsto e^{-\frac{\|x-x_\mathrm{o}\|^2}{\sigma^2}} \in \mathbb{R}_+$, where, for the experiments, the following values are used: $x_\mathrm{o} = [0, 0]^T$ and $\sigma^2 = 0.1$. The time-varying environment field I is modeled as a mixture of time-varying Gaussians of the following form $I(x, t) = e^{-\|x-M_1(t)x_\mathrm{c}\|^2} + e^{-\|x-M_2(t)x_\mathrm{c}\|^2}$, where $x_\mathrm{c} = [1, 1]^T$ and $M_1(t) = \mathrm{diag}(-1, \sin(2t))$, $M_2(t) = \mathrm{diag}(\sin(2t), 1)$. In the case of robots that are able to exploit solar power to recharge their batteries, this choice of the function I simulates the sunlight intensity that is characterized by a periodic expression over a spatially fixed environment. The other values required to model $\dot{E}_i$ (3.5) in (3.7) are set to $I_\mathrm{c} = 0.85$ and $\lambda = 3$.

The optimization problem aimed at minimizing the ergodic cost metric ϵ is solved offline and the resulting trajectory is given to the robot as a reference for the tracking controller. The input required to track the trajectory is wrapped by the QP (3.51) in such a way that the robot explores the environment while satisfying at the same time the energy constraint that prevents its battery level to go below the lower threshold $E_{\min}$. The result of this controller is a persistified environment exploration.

Figures 3.5a to 3.5e show a sequence of snapshots taken during the course of the environment exploration experiment described above. The contour plot of the function ϕ is depicted as green thin solid lines, while the contour plot of the environment field I is represented by the yellow thin dashed lines. The position that the robot is tracking under the nominal control input is depicted as a black square, whereas its actual position obtained by executing the controller (3.51) is represented by a black circle. Furthermore, nominal and executed trajectories are represented by a gray thick solid line and a red thick dashed line, respectively.

Figure 3.6 compares the probability density function representing the spatial information distribution ϕ (Fig. 3.6a) with the probability density functions for the time-averaged optimized trajectory with energy constraints (Fig. 3.6b) and without energy constraints (Fig. 3.6c). Even though the latter more closely matches the spatial distribution ϕ, it does

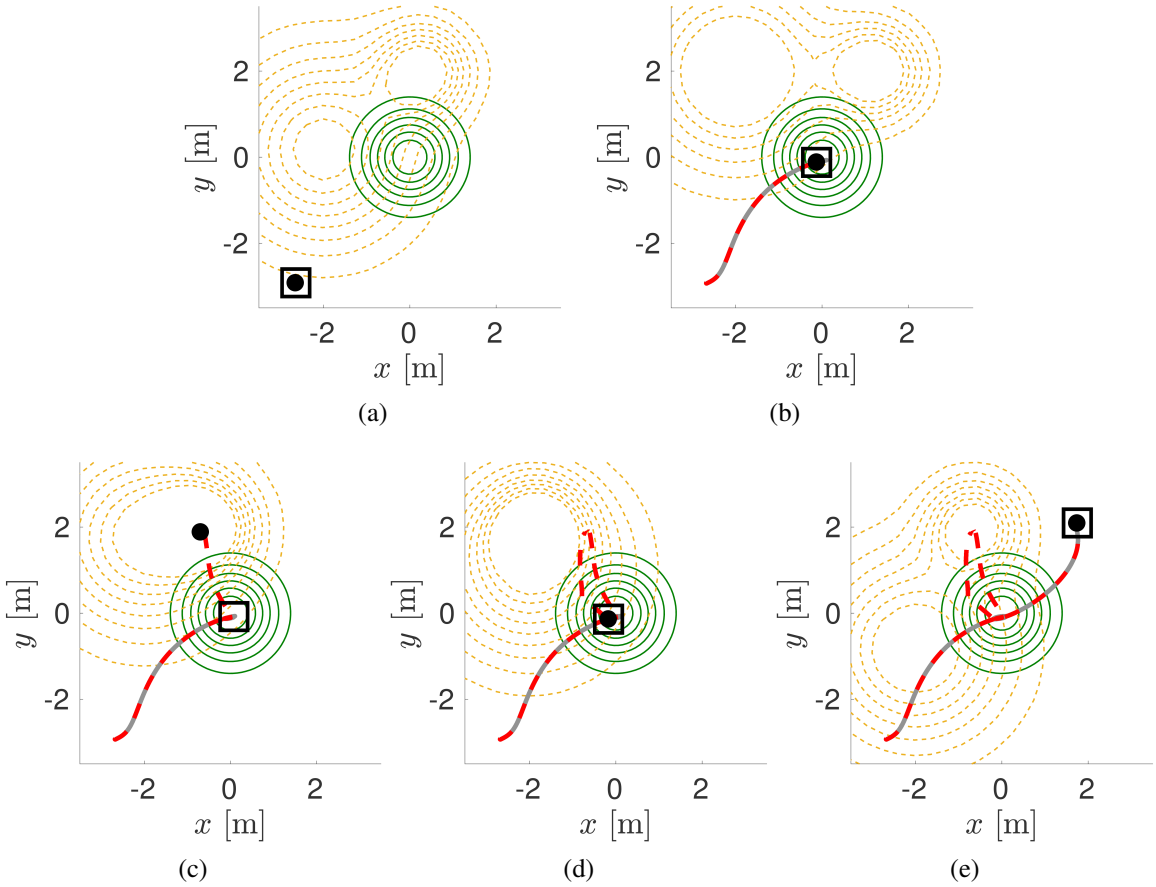


Figure 3.5: Sequence of images recorded during the course of the environment exploration simulated experiment. The contour plots of the information distribution function ϕ and the environment field I are depicted as green thin solid lines and yellow thin dashed lines, respectively. The position tracked by the robot under the nominal control input is represented as a black square, while the actual position of the robot is shown as a black dot. The nominal and actual trajectories are depicted as a gray thick solid line and a red thick dashed line, respectively. In order to persistently explore the environment, the robot follows the nominal input as long as its energy level is high enough. When its battery is depleting, it moves towards regions of the environment where the value of the time-varying field I is such that its energy starts increasing.

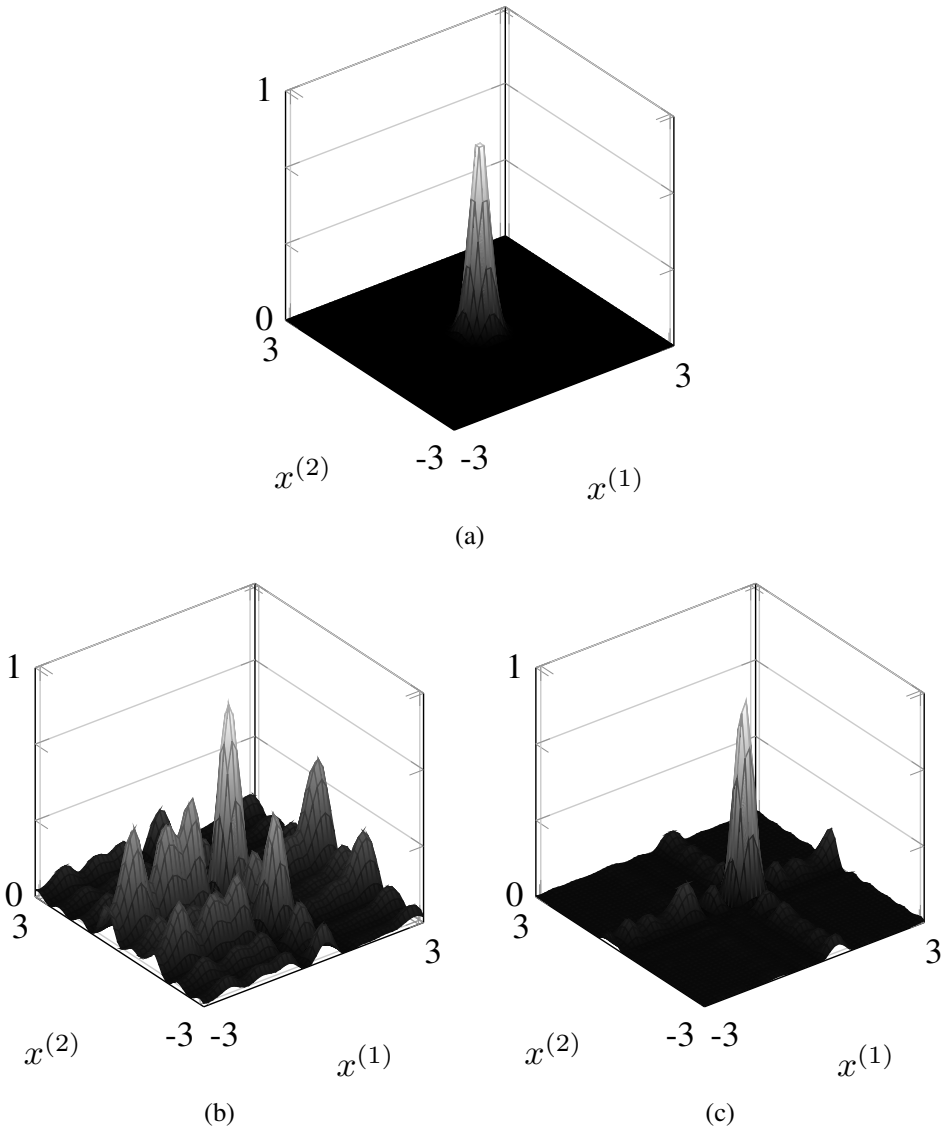


Figure 3.6: Comparison between the spatial probability density function ϕ, in Fig. 3.6a, and the probability density function obtained averaging over time the ergodic trajectory resulting from the implementation of the persistified environment exploration, in Fig. 3.6b; Figure 3.6c depicts the probability density function representing the time-averaged optimized ergodic trajectory obtained without taking into account energy constraints. $x^{(1)}$ and $x^{(2)}$ are the two components of the state vector $x \in \mathcal{E} \subset \mathbb{R}^2$.

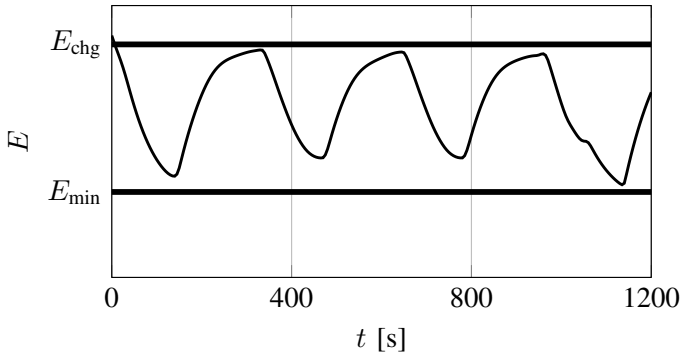


Figure 3.7: Simulated battery level of the robot during the course of the persistified exploration experiment. Employing the persistification strategy presented in this chapter, achieved by executing the control input solution of the QP (3.51), the robot energy (thin line) is constrained within the bounds E_{min} and E_{chg}.

not take into account that the robot has a finite availability of energy.

In Fig. 3.7, the energy level of the robot during the persistified exploration experiment is reported. The application of the control framework presented in this chapter is demonstrated to be successful in persistifying the robotic exploration. This is realized by keeping the robot energy level constrained above a minimum value in an optimal way by means of the QP (3.51). This way, the robot is completely free of tracking the ergodic trajectory given as input to its motion controller, as long as its battery level is above the lower threshold E_{min} and below the upper threshold E_{chg}, depicted as thick solid lines in Fig. 3.7.

In Section 3.2.4, CLF constraints were introduced with the objective of completely recharging the battery of the robots before leaving areas of the environment where $\dot{E} > 0$, referred to as charging stations. This leads to a more efficient task execution in the sense that the robots overall spend less time recharging their batteries. This condition allows them to deviate from the nominal task input of a smaller amount over the course of the experiment. In Fig. 3.8, the value of C at time t represents the integral over the time interval $[0, t]$ of the difference between the nominal control input $\hat{u}$—corresponding to the exploration task—and the input u executed by the robot:

$$C(t) = \int_0^t \|u(\tau) - \hat{u}(\tau)\|^2 \, d\tau. \tag{3.58}$$

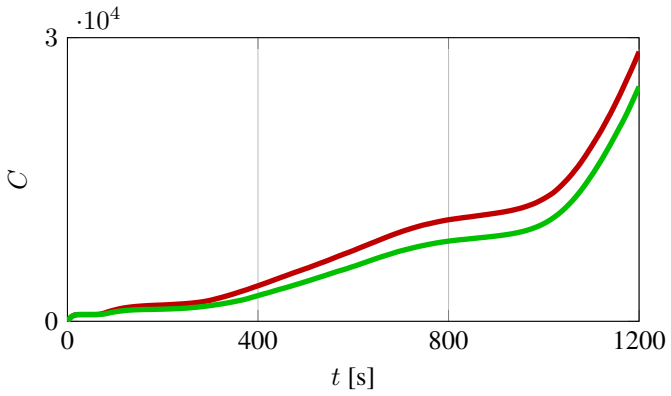


Figure 3.8: Comparison between persistent task execution with and without battery recharging constraints. The green curve depicts the value of C, defined in (3.58), in which the robot input $u(t) = u^*(t)$, solution of (3.51). The red curve has been obtained by letting the robot execute the input $u(t)$ solution of (3.51) from which the constraint (3.50) corresponding to battery recharging has been removed. As a result, the value of C in the latter case is higher than the one in the former, i.e. the robot spends more time deviating from the nominal input $\hat{u}(t)$ in order to visit charging stations in the environment and prevent its energy from depleting.

The green curve is generated letting the robot execute the input $u(t) = u^*(t)$, solution of (3.51). The red curve corresponds to the situation where the robot executes the input solution of (3.51) in which the CLF constraint (3.50) for battery recharging has been removed. As can be seen, the battery recharging constraint allows the robot to visit charging stations less often, so that its input deviates from the nominal one of a smaller amount over the course of the task execution.

3.3.2 Environment Surveillance

The second application that is considered as showcase is environment surveillance. The employment of mobile sensors improves coverage and data gathering performances compared to static sensors, whose positions are determined based on offline optimization algorithms. In this sense, mobility can allow a more efficient estimation of time-varying information fields. However, this comes at the price of higher power consumption. Most of the approaches developed so far assume that the mobile sensors are able to move for an unlimited amount of time. The control framework presented in this chapter can be used to

persistify such surveillance tasks.

The task of environment surveillance can be framed as a sensor coverage control problem, that is an instance of the broader optimal sensor placement problem whose applications can be found in many other disciplines, such as [128]. As in Section 3.3.1, let the map $\phi : \mathcal{E} \to \mathbb{R}_+$ represent a spatial distribution density function. This can be interpreted as a measure of the information spread over the environment $\mathcal{E}$ or the probability that an event can take place at a location $x \in \mathcal{E}$. Moreover, let us define the locational optimization function as in [32]:

$$\mathcal{H}(X, \mathcal{W}) = \sum_{i=1}^{N} \int_{W_i} \sigma(\|x - x_i\|)\phi(x)dx, \tag{3.59}$$

where $X = \{x_1, \ldots, x_N\}$ are the positions of the N robots present in the environment $\mathcal{E}$, $\mathcal{W} = \{W_1, \ldots, W_N\}$ is a partition of $\mathcal{E}$, $\sigma : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing differentiable function describing the degradation in the sensing performances of the robots. Proceeding as in [32], we aim at minimizing $\mathcal{H}(X, \mathcal{W})$ with respect to both X and $\mathcal{W}$. The minimization with respect to the environment partition $\mathcal{W}$ leads to $\mathcal{W} = \mathcal{V} = \{V_1, \ldots, V_N\}$ [128], $\mathcal{V}$ being the Voronoi partition of $\mathcal{E}$ defined by the Voronoi cells

$$V_i = \{x \in \mathcal{E} : \ \|x - x_i\| \leq \|x - x_j\| \ \forall i \neq j\}. \tag{3.60}$$

As far as the minimization with respect to the robot locations X is concerned, considering the single integrator dynamics of the robots, and setting $\sigma(\|x - x_i\|) = \|x - x_i\|^2$ allows us to define the following gradient descent flow to be used as a control law for moving the robots [129]:

$$\hat{u}_i(x_i) = k_p(C_{V_i} - x_i). \tag{3.61}$$

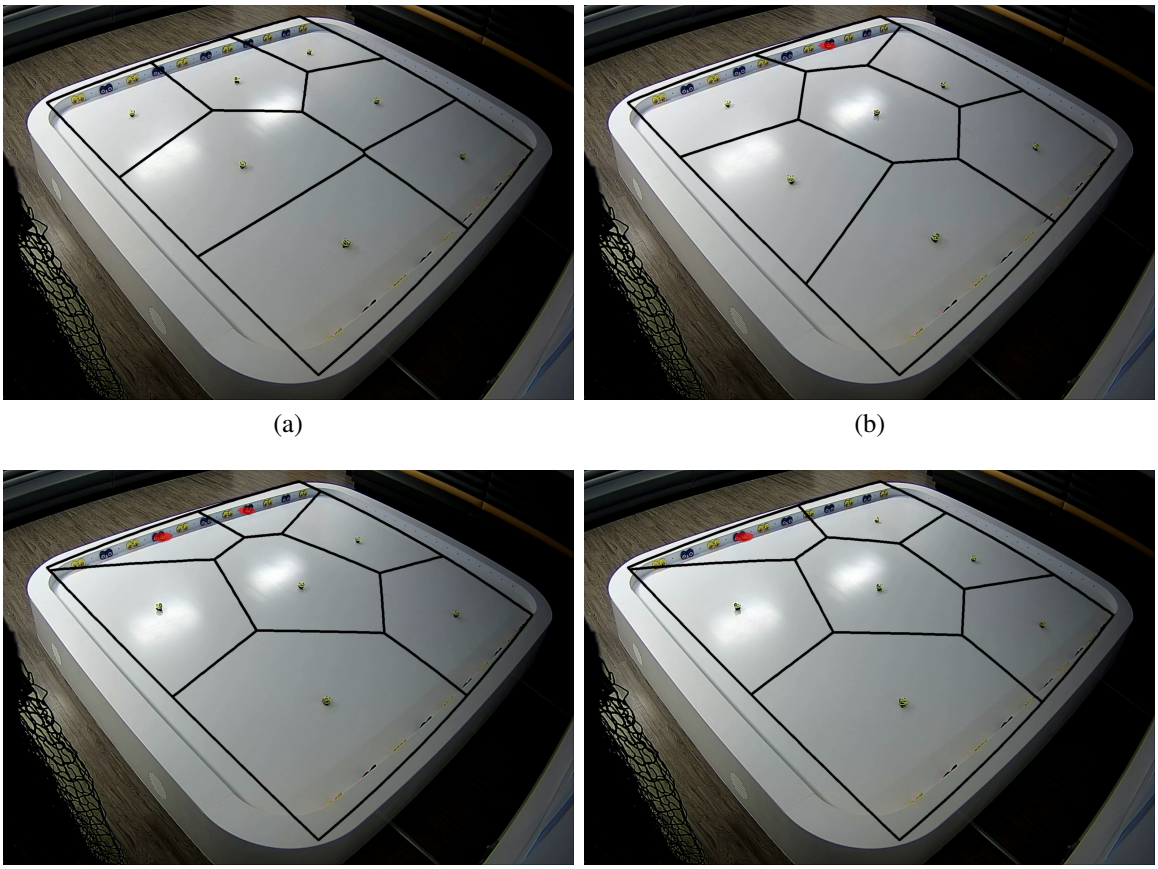
In (3.61), $k_p > 0$ is a proportional gain and C_{V_i} is the centroid of the i-th Voronoi cell defined as $C_{V_i} = \int_{V_i} x\phi(x)dx \big/ \int_{V_i} \phi(x)dx$. In case the map ϕ is time-varying, the ex-

tension presented in [33] and [8] can be employed. The coverage task with $\hat{u}(x) = [\hat{u}_1(x_1)^T, \ldots, \hat{u}_N(x_N)^T]^T$ and $\hat{u}_i(x_i)$ given by (3.61), can be persistified by the implementation of the optimization-based controller solution of the QP in (3.51).

The persistified environment surveillance has been implemented and deployed on the Robotarium, a remotely accessible swarm robotics research testbed [52, 130]. A team of 7 differential-drive robots attempts to cover a $4.5 \times 3.5\,\text{m}$ testbed area by making use of the coverage control introduced above. The Robotarium is endowed with wireless charging stations, allowing the modeling of the charging field I given in (3.2) by means of bump-like functions as depicted in Fig. 3.3. The robot model is a single integrator model, where we can directly control the robots' velocities utilizing the proportional control law given in (3.61). This input is wrapped by the QP (3.51) in order to satisfy the energy constraints. The result is a persistified environment surveillance.

Figures 3.9a to 3.9d show the salient frames of the persistent environment surveillance experiment performed on the Robotarium. The Voronoi cells are superimposed on the frames. The yellow and blue wireless charging station are arranged along one of the edges of the testbed. Following the nominal controller (3.61), the robots perform sensor coverage (3.9a). The actual controller executed by the robots is the solution of (3.51), which allows them to go back and recharge their batteries to prevent the stored energy from going below the minimum desired value E_{min} as shown in 3.9b, 3.9c and 3.9d (the charging stations occupied by robots are marked in red).

In Fig. 3.10, the thin line shows the value of the locational cost defined in (3.59) evaluated during the course of the experiment. Here the information distribution density function ϕ has been set to a constant value, meaning that the information is equally spread over the testbed. The thick line in Fig. 3.10 represents the value of the cost (3.59) obtained if no energy constraints were imposed, in which case, the robots are able to asymptotically reach a centroidal Voronoi configuration, where each robot is in the centroid of its corresponding Voronoi cell. This configuration leads to a local minimum of (3.59). The value of



(a) (b)

(c) (d)

Figure 3.9: Sequence of salient frames extracted from the video of the persistent environment surveillance experiment. A team of 7 small differential-drive robots are deployed to perform persistent sensor coverage of the testbed of the Robotarium [130]. On one edge of the testbed yellow and blue wireless charging stations are arranged, where the robots can recharge their batteries. The environment (charging) field has been modeled similarly to the one shown in Fig. 3.3. The black lines represent the boundaries of the Voronoi cells corresponding to each robot. The sequence of images shows the robots performing coverage under the nominal control input (3.9a), two robots, marked with red circles, going back to the charging stations to recharge their batteries (3.9b to 3.9d) driven by the controller (3.51).

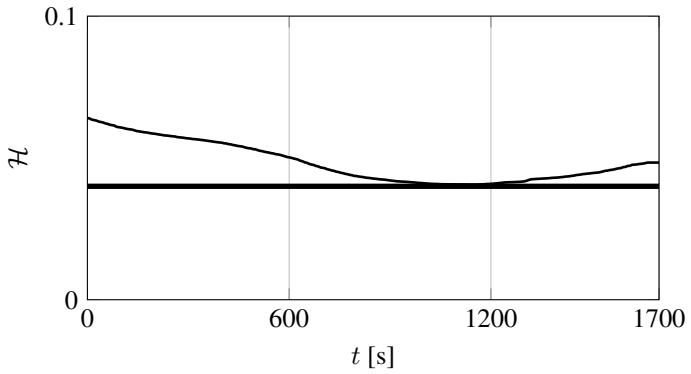


Figure 3.10: Locational cost (3.59) evaluated during the course of a coverage control experiment: the thin line is the cost obtained imposing the energy constraints to the robots, whereas the thick line is the cost obtained assuming infinite availability of energy.

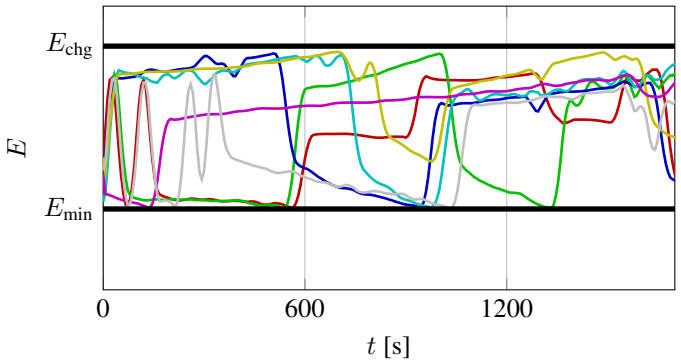


Figure 3.11: Measured battery levels of the 7 robots executing the persistified coverage control experiment: as a result of implementing the input solution of (3.51), the energy levels of all the robots are constrained to be between the values E_{min} and E_{chg}.

$\mathcal{H}$ decreases in correspondence of the situations in which all the available robots are not constrained to charge and are consequently free of following the coverage control input.

Figure 3.11 displays the actual energy stored in the batteries of the robots as measured during the course of the experiment. The inputs to the robots are constrained by (3.45) and (3.50) in such a way that their energies never exceed E_{chg} or go below E_{min}. As a results of the optimization-based formulation, each individual robot is able to follow as closely as possible the input encoding the coverage task as long as its battery level satisfies the imposed constraints.

3.4 Conclusions

In this chapter, we introduced the concept of robotic task persistification, i.e., the process of rendering a robotic task persistent. This allows robots to execute a task over long time horizons, by ensuring that the energy stored in their batteries never gets depleted. The control of the robot energy level is wrapped around the task controller by holistically utilizing control barrier functions, control Lyapunov functions, and integral control barrier functions, combined in a single optimization-based controller which can be efficiently executed in an online fashion. The result of this formulation is a control framework that allows the robots to execute as accurately as possible the assigned task, while simultaneously never depleting the energy in their batteries. This constraint is enforced by expressing the task persistence condition in terms of the forward invariance of a subset of the state space of the robots. The forward invariance property is then ensured by employing control barrier functions, whereas battery recharging is achieved by defining a suitable control Lyapunov function.

The results in Theorem 3.9, Lemma 3.12 and Theorem 3.16 allow us to apply the presented framework to many different kinds of robots endowed with rechargeable sources of energy. In order to be able to efficiently enforce energy constraints, we insist on the robot dynamic model being in control affine form, case that is often encountered in robotic applications. Since the persistified task is obtained as the output of an optimization problem, the robots are free to execute the given task as closely as possible as long as their source of energy is not discharging below a given lower threshold. The persistification strategy has been applied to environment exploration and monitoring tasks, and it has been tested both in simulation and on an team of ground mobile robots.

CHAPTER 4
CONSTRAINT-DRIVEN CONTROL OF ROBOTIC SYSTEMS

The task persistification strategy developed in Chapter 3 allows the execution of tasks described by means of nominal robot control inputs $\hat{u}$—which can be both state and time dependent—for sustained periods of time. This objective is achieved by altering the value of the nominal control input $\hat{u}(x,t)$ as little as possible provided that the desired energy constraints are satisfied. In long-duration autonomy application, it is often desirable to spend as little control effort as possible even to execute a task, as, in such settings, *longevity* and *survivability* take precedence over performances [4]. Nevertheless, this prioritization, is not always taken into account when designing the nominal controller of the robots $\hat{u}$ (see, e.g., examples considered in Chapter 3).

For this reason, in this chapter, we present a reformulation—framed as a constrained optimization problem—of robotic tasks which are encoded by means of a cost function that is to be minimized. The advantages of this approach are multiple. First of all, as discussed above, the constraint-based formulation provides a natural way of enabling long-term robot autonomy applications, where resilience and adaptability to changing environmental conditions are essential. Moreover, under mild assumptions on the cost function encoding the task, the constraint-driven control approach can be applied to multi-robot systems in order to design a decentralized control strategy. Furthermore, finite-time task execution can be achieved, and, in the multi-robot case, this can be done by always leveraging local information only.

An optimization-based control framework which possesses the properties described above is presented in the following. Then, sufficient conditions for turning certain classes of (multi-robot) tasks into constraints within an optimization problem are given. And, finally, we propose an effective task *prioritization* technique obtained by combining hard and

soft constraints, as, e.g., survivability and task execution, respectively. Building up on this task prioritization technique, in the next chapter, a multi-robot multi-task *allocation* algorithm is presented, which is amenable for long-duration autonomy applications. The results of the implementation of the developed constraint-based control framework on a team of ground mobile robots executing a long-term environmental monitoring task conclude the chapter.

4.1 Constraint-Based Control Design

In this chapter, we make use of control Lyapunov functions and control barrier functions introduced in the previous chapter, which will then be used to formulate optimization problems whose solution corresponds to the execution of decentralized coordinated controllers for multi-robot systems.

4.1.1 Finite-Time Stability and Control Barrier Functions

In order to design controllers that allow the execution of multi-robot tasks, we make use of control Lyapunov functions and, in particular, we resort to methods from finite-time stability theory of dynamical systems.

Consider the dynamical system in control affine form

$$\dot{x} = f(x) + g(x)u, \tag{4.1}$$

with $x \in \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$, and f and g locally Lipschitz continuous vector fields. One of the results we will use is given by the following theorem.

Theorem 4.1 (Based on Theorem 4.3 in [131])**.** *Given a dynamical system* (4.1) *and a continuous, positive definite function $V : \mathbb{R}^n \to \mathbb{R}$, a continuous controller u such that*

$$\inf_{u \in \mathcal{U}} \{L_f V(x) + L_g V(x)u + c(V(x))^\gamma\} \leq 0 \quad \forall x \in \mathbb{R}^n, \tag{4.2}$$

where $L_f V(x)$ and $L_g V(x)$ denote the Lie derivatives of V in the directions of f and g, respectively, $c > 0$ and $\gamma \in (0, 1)$, renders the origin $x = 0$ finite-time stable.

Moreover, an upper bound for the settling time T is given by

$$T \leq \frac{1}{c(1-\gamma)} \left(V(x_0)\right)^{1-\gamma}, \tag{4.3}$$

where x_0 is the value of $x(t)$ at time $t = 0$.

To enforce constraints, such as survivability or task execution for multi-robot systems, we employ control barrier functions. These are suitable for synthesizing constraints that can be encoded in terms of set-membership, in order to ensure forward invariance (as seen in Chapter 3) and asymptotic stability, as summarized by the following theorem.

Theorem 4.2 (Safe set forward invariance and asymptotic stability [116])**.** *Given a dynamical system* (4.1) *and a set $\mathcal{S} \subset \mathbb{R}^n$ defined by a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, such that the $\mathcal{S}$ can be defined as the zero-superlevel set of h, i.e.,*

$$\mathcal{S} = \{x \in \mathbb{R}^n : \ h(x) \geq 0\}, \tag{4.4}$$

any Lipschitz continuous controller u such that

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x) u + \alpha(h(x))\} \geq 0 \quad \forall x \in \mathbb{R}^n, \tag{4.5}$$

renders the set $\mathcal{S}$ forward invariant and asymptotically stable, i.e.

$$x(t)|_{t=0} \in \mathcal{S} \ \Rightarrow \ x(t) \in \mathcal{S} \quad \forall t \geq 0 \tag{4.6}$$

$$x(t)|_{t=0} \notin \mathcal{S} \ \Rightarrow \ x(t) \to \in \mathcal{S} \quad \text{as } t \to \infty. \tag{4.7}$$

4.1.2 Minimum-Energy Gradient Flow

In this section, we consider the problem of minimizing a cost function J. We present a method to reformulate the classic gradient flow algorithm using the tools introduced in Section 4.1.1. This allows us to synthesize a constrained optimization program that is equivalent to the minimization of the cost J.

Consider the single integrator dynamical system $\dot{x} = u$, where $x, u \in \mathbb{R}^n$ are the state and the control input, respectively. Assume that the objective consists in minimizing a cost $J(x)$, where $J : \mathbb{R}^n \to \mathbb{R}_+$ is a continuously differentiable function. Applying gradient flow algorithms, a solution to the problem

$$\begin{aligned} &\underset{u(t)}{\text{minimize}}\ J(x) \\ &\text{subject to } \dot{x}(t) = u(t) \\ &\qquad\qquad x(t_0) = x_0, \end{aligned} \tag{4.8}$$

can be found by setting

$$u = -\frac{\partial J}{\partial x}^T (x). \tag{4.9}$$

In fact, with this choice of input, applying chain rule leads to:

$$\dot{J}(x) = \frac{dJ}{dt} = \frac{\partial J}{\partial x}\dot{x} = \frac{\partial J}{\partial x}u = -\left\|\frac{\partial J}{\partial x}\right\|^2 \leq 0. \tag{4.10}$$

We now show that the minimization problem (4.8) can be formulated as a minimum-energy problem that achieves the same objective of minimizing the cost J.

To this end, let us define the barrier function

$$h(x) = -J(x) \tag{4.11}$$

and its zero-superlevel set, i.e. the *safe* set,

$$\mathcal{S} = \{x : \ h(x) \geq 0\} = \{x : \ J(x) \leq 0\} = \{x : \ J(x) = 0\}\,. \tag{4.12}$$

By Theorem 4.2, the differential constraint

$$\dot{h}(x) = \frac{\partial h}{\partial x} u \geq -\alpha(h(x)), \tag{4.13}$$

where α is an extended class $\mathcal{K}$ function, ensures that the set $\mathcal{S}$ is forward invariant when $h(x(0)) \geq 0$ and asymptotically stable when $h(x(0)) \leq 0$, $x(0)$ being the value of the state x at time $t = 0$.

Observation 4.3. *Theorem 4.2 shows the existence of the control Lyapunov function*

$$\begin{aligned} V(x) &= \begin{cases} -h(x) & \text{if } x \notin \mathcal{S} \\ 0 & \text{if } x \in \mathcal{S} \end{cases} \\ &= \begin{cases} J(x) & \text{if } x \notin \mathcal{S} \\ 0 & \text{if } x \in \mathcal{S} \end{cases} \\ &\equiv J(x). \end{aligned} \tag{4.14}$$

Indeed, from (4.10)*, since $J(x) \geq 0$, $\forall x \in \mathbb{R}^n$, one can see that $J(x)$ is a control Lyapunov function. In fact, if x belongs to $\mathcal{X} \subset \mathbb{R}^n$ compact, LaSalle's Invariance Principle ensures that the state will converge to a stationary point of $J(x)$, namely, $x \to x^*$, with $\frac{\partial J}{\partial x}(x^*) = 0$.*

We can now introduce the following optimization problem:

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}} \ \|u\|^2 + \delta^2 \\ &\text{subject to} \ \frac{\partial h}{\partial x} u \geq -\alpha(h(x)) - \delta, \end{aligned} \tag{4.15}$$

$\delta \in \mathbb{R}$, which solves the problem in (4.8), as shown in the following proposition.

Proposition 4.4. *The solution of the optimization problem* (4.15)*, where* $h(x)$ *is given by* (4.11) *and* α *is an extended class* $\mathcal{K}$ *function, solves* (4.8)*, driving the state* x *of the dynamical system* $\dot{x} = u$ *to a stationary point of the cost* J*.*

Corollary 4.5. *Under the same hypotheses as in Proposition 4.4 and* J *such that* $\frac{\partial J}{\partial x} = 0 \Leftrightarrow x = 0$*, the solution of the optimization program*

$$\begin{aligned} &\underset{u}{\text{minimize}} \ \|u\|^2 \\ &\text{subject to} \ \frac{\partial h}{\partial x} u \geq -\alpha(h(x)), \end{aligned} \tag{4.16}$$

solves the problem in (4.8)*.*

In summary, we saw that the expression for u given in (A.11) solves the initial optimization problem (4.8), which can be equivalently solved following the gradient flow of the cost J using (4.9).

We now illustrate that, besides the above-mentioned advantages related to long-term autonomy applications, the formulation in (4.15) can be used to design decentralized cost minimization algorithms that are faster than gradient descent. In the optimization literature, there are plenty of methods that can be employed to improve the convergence speed of gradient flow algorithms (see, e.g., [121]). Nevertheless, these *second order* methods, such as Newton's method or conjugate gradient, suffer from their centralized nature. Only in some cases, this issue can be partially mitigated by resorting to distributed optimization techniques, such as [132, 133]. The above-mentioned methods are all suitable for minimizing a cost function. However, we insist on having a constrained optimization formulation—where we encode cost minimization as a constraint—because of the flexibility and robustness properties discussed above, useful for long-term robot autonomy applications.

The following proposition shows that, using the formulation in (4.15), it is possible to minimize the cost J and to be not just faster than gradient descent, but actually to reach a stationary point in finite time.

Proposition 4.6. *Given the dynamical system $\dot{x} = u$ and the objective of minimizing the cost function J, the solution of the optimization problem*

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}} \quad \|u\|^2 + \delta^2 \\ &\text{subject to } \frac{\partial h}{\partial x} u \geq -c(h(x))^\gamma - \delta, \end{aligned} \tag{4.17}$$

where $h(x)$ is given by (4.11)*, $c > 0$ and $\gamma \in (0, 1)$, will drive the state x to a stationary point of the cost J in finite time.*

Using the results derived in this section, the next section presents a procedure to synthesize decentralized optimization problems whose solutions result in coordinated control of multi-robot systems.

4.2 Constraint-Based Control of Multi-Robot Systems

Local, *scalable*, *safe* and *emergent* are four essential features that decentralized multi-robot coordinated control algorithms should possess [134]. Many algorithms that satisfy these properties have been developed for applications ranging from social behavior mimicking [135], formation assembling [136] and area patrolling [32]. In [134], the authors analyze the common features among these algorithms and discuss their decentralized implementations in robotic applications. In this section, we apply the results derived in Section 4.1 with the aim of obtaining constrained optimization problems equivalent to the decentralized execution of multi-robot tasks.

Consider a collection of N robots, whose position is denoted by $x_i \in \mathbb{R}^d$, $i \in \{1, \ldots, N\}$, where $d = 2$ for planar robots and $d = 3$ in the case of aerial robots. Assume each robot is equipped with an omni-directional range sensor that allows it to measure the relative position of neighboring robots, namely robot i is able to measure $x_j - x_i$, when robot j is within its sensing range. These interactions among the robots are described by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices of the graph, representing the

robots, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges between the robots, encoding adjacency relationships. If $(i, j) \in \mathcal{E}$, then robot i can measure robot j's position. For the purposes of this chapter, we assume that the graph is undirected, namely $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$. In order to obtain decentralized algorithms, we want each robot to act only based on local information, by which we mean the relative positions of its neighbors. By construction, this leads to inherently scalable coordinated control algorithms.

Denoting the ensemble state of the robotic swarm by $x = [x_1^T, \ldots, x_N^T]^T \in \mathbb{R}^{Nd}$, a general expression for the cost that leads to decentralized control laws is given by

$$J(x) = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} J_{ij}(\|x_i - x_j\|), \tag{4.18}$$

where $\mathcal{N}_i$ is the neighborhood set of robot i, and $J_{ij} : \mathbb{R} \to \mathbb{R}$, $J_{ij}(\|x_i - x_j\|) = J_{ji}(\|x_j - x_i\|)$ is a symmetric, pairwise cost between robots i and j. We assume that $J_{ij}(x) \geq 0,\ \forall (i, j) \in \mathcal{E},\ \forall x \in \mathbb{R}^n$, so that $J(x) \geq 0,\ \forall x \in \mathbb{R}^n$. Assuming we can directly control the velocity of robot i, $\dot{x}_i$, we can employ a gradient descent flow policy like (4.9) to minimize J, obtaining

$$u_i = -\sum_{j \in \mathcal{N}_i} \frac{\partial J_{ij}}{\partial \|x_i - x_j\|} \frac{x_i - x_j}{\|x_i - x_j\|} = \sum_{j \in \mathcal{N}_i} w_{ij}(x_j - x_i). \tag{4.19}$$

This is nothing but a weighted consensus protocol, and it is decentralized insofar as the input u_i only depends on robot i's neighbors. The construction shown in Section 4.1 can be then applied to minimize the cost given in (4.18) by formulating the following minimum-energy problem:

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}} \ \|u\|^2 + \delta^2 \\ &\text{subject to } -\frac{\partial J}{\partial x} u \geq -\alpha(-J(x)) - \delta, \end{aligned} \tag{4.20}$$

where $u = [u_1^T, \ldots, u_N^T]^T \in \mathbb{R}^{Nd}$ is the vector of robots' inputs, and a single integrator dynamics, $\dot{x}_i = u_i$, is assumed for each robot. Solving (4.20) leads to the accomplishment

of the task, by which we mean that a stationary point of the cost J has been reached. As explicitly shown by (A.11) in Proposition 4.4, a minimum-energy formulation, initially introduced in [44], allows the robots to move towards lower values of the cost J until the task is accomplished ($u \equiv 0$).

The following proposition gives the expression of the optimization problems whose solutions lead to a decentralized minimization of the cost J in (4.18).

Proposition 4.7 (Constraint-driven decentralized task execution)**.** *Given the pairwise cost function J defined in* (4.18)*, a collection of N robots, characterized by single integrator dynamics, minimizes J in a decentralized fashion, if each robot executes the control input, solution of the following optimization problem:*

$$\begin{aligned} &\underset{u_i,\delta_i}{\text{minimize}} \ \|u_i\|^2 + \delta_i^2 \\ &\text{subject to } -\frac{\partial J_i}{\partial x_i} u_i \geq -\alpha(-J_i(x)) - \delta_i, \end{aligned} \tag{4.21}$$

where $J_i(x) = \sum_{j \in \mathcal{N}_i} J_{ij}(\|x_i - x_j\|)$ and α is an extended class $\mathcal{K}$ function, $\alpha : x \in \mathbb{R} \mapsto \alpha(x) \in \mathbb{R}$, superadditive for $x < 0$, i.e. $\alpha(x_1 + x_2) \geq \alpha(x_1) + \alpha(x_2),\ \forall x_1, x_2 < 0$. If $\alpha(x) = cx^\gamma$, $c > 0$, $\gamma \in (0, 1)$, a stationary point of the cost J is reached in finite time, with the upper bounds on the settling time given in Theorem 4.1.

The structure of the cost function $J(x)$, even though quite specific, allows us to encode a rich set of multi-robot tasks, by carefully choosing the weights w_{ij} as a function of the state x. The following section shows two variations on the cost function which allow a multi-robot system to perform formation control, i.e., assembling particular shapes, and coverage control, consisting in spreading out the robotic swarm in the environment in an optimal way.

4.3 Applications

In this section we recall the expression of the cost J for two specific multi-robot tasks: formation control and coverage control.

4.3.1 Formation Control

In formation control applications, the robots are asked to assemble a predefined shape, specified in terms of inter-agent distances. In order to frame this problem as a cost minimization problem, let J be the formation error

$$J(x) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} \frac{1}{2} (\|x_i - x_j\| - d_{ij})^2 = \sum_{i=1}^{n} J_i(\|x_i - x_j\|), \tag{4.22}$$

where d_{ij} is the desired distance between robots i and j. J measures how far the robots are from assembling the desired formation characterized by the relative distances d_{ij}. $J = 0$ corresponds to the robots forming the desired shape. Note that, as $J_i(x)$ is a sum of squares, $J_i(x) \leq J(x),\ \forall i$, required as a hypothesis in order for Proposition 4.7 to hold.

The gradient the $J_i(x)$ evaluates to

$$\frac{\partial J_i}{\partial x_i} = \sum_{j \in \mathcal{N}_i} \frac{\|x_i - x_j\| - d_{ij}}{\|x_i - x_j\|} (x_i - x_j)^T. \tag{4.23}$$

This can be interpreted as follows: if the distance between robots i and j is smaller than d_{ij}, then the weight $w_{ij} = \frac{\|x_i - x_j\| - d_{ij}}{\|x_i - x_j\|}$ is negative, and the robots experience a repelling effect. Conversely, if the two robots are further than d_{ij} apart, the positive weight w_{ij} will attract one towards the other. The special case in which $d_{ij} = 0,\ \forall i, j$ corresponds to the well-known consensus problem.

The expression of the gradient in (4.23) is decentralized as robot i has to compute relative distances only with respect to its neighboring robots.

4.3.2 Coverage Control

In coverage control, the task given to the robots is that of covering a domain D. Given a coverage performance measure, the robots should spread over the domain in an optimal way. As shown in [137], each robot should be in charge only of a subset of the domain D that, more specifically, is its Voronoi cell, defined as $V_i = \{p \in D : \|p - x_i\| \leq \|p - x_j\| \; \forall i \neq j\}$.

Let us introduce the measure of how bad a domain is being covered:

$$J(x) = \sum_{i=1}^{N} \frac{1}{2}\|x_i - G_i(x)\|^2 = \sum_{i=1}^{n} J_i(x), \tag{4.24}$$

where G_i denotes the centroid of the Voronoi cell V_i. This form is just a reformulation of the locational cost originally introduced in [32]. Taking the derivative of J_i with respect to x_i, required in the optimization problem (4.21), one obtains:

$$\frac{\partial J_i}{\partial x_i} = (x_i - G_i(x))^T \left(I - \frac{\partial G_i(x)}{\partial x_i}\right), \tag{4.25}$$

where I is the identity matrix. Note that, even if $G_i(x)$ virtually depends on the entire ensemble state, x, of the robotic swarm robot i, in order to compute it, only requires information from the robots with which it shares part of the boundary of its Voronoi cells.

The formulation presented in this chapter also allows an exact decentralized implementation of the coverage control with time-varying density functions. In [33], the authors show that the control law

$$u = \left(I - \frac{\partial G}{\partial x}\right)^{-1} \left((G(x,t) - x) + \frac{\partial G}{\partial t}\right) \tag{4.26}$$

minimizes the locational cost

$$\mathcal{H}(x,t) = \sum_{i=1}^{N} \int_{V_i} \|q - x_i\|^2 \phi(q,t) dq. \tag{4.27}$$

$\phi : (q,t) \in D \times \mathbb{R}_+ \mapsto \phi(q,t) \in \mathbb{R}_+$ is a time-varying density function, which specifies the importance of point q at time t. The cost in (4.27) is equivalent to the one defined in (4.24) when the centroids $G_i(x)$ are calculated weighting the points in the domain according to the value of the density function associated to them, as shown in [32].

However, inverting the matrix $I - \frac{\partial G}{\partial x}$ in (4.26) cannot be done in a decentralized fashion. For this reason, in [33], the inverse is approximated by a truncated Neumann series as

$$\left(I - \frac{\partial G}{\partial x}\right)^{-1} \approx I + \frac{\partial G}{\partial x}, \tag{4.28}$$

which, on the contrary, can be evaluated only based on information about neighboring robots.

With the formulation presented in this chapter, instead, by implementing the optimization problem (4.21) in Proposition 4.7, each robot has to solve

$$\begin{aligned} &\underset{u_i, \delta_i}{\text{minimize}} \ \|u_i\|^2 + \delta_i^2 \\ &\text{subject to } -(x_i - G_i(x,t))^T \left(I - \frac{\partial G_i(x,t)}{\partial x_i}\right) u_i \\ &\qquad \geq -\alpha(-J_i(x,t)) - (x_i - G_i(x,t))^T \frac{\partial G_i(x,t)}{\partial t} - \delta_i, \end{aligned} \tag{4.29}$$

which is both exact and decentralized.

We deployed the optimization-based control algorithms with the expressions of the costs J derived in Sections 4.3.1 and 4.3.2 on a real multi-robot system and, in the next section, we show the experimental results. Moreover, the constraint-driven formulation of Section 4.2 is used to achieve long-term environmental monitoring, where the robots are tasked with covering a domain over a time-horizon which is much longer than their battery

life, and during which the robots will also have to avoid collisions with obstacles moving around in the domain.

4.4 Experimental Results

The coordinated control approach presented in this chapter has been tested on the Robotarium [130], a remotely accessible swarm robotics testbed. The Robotarium is populated by small-scale differential-drive robots which can be programmed by uploading code scripts via a web interface.

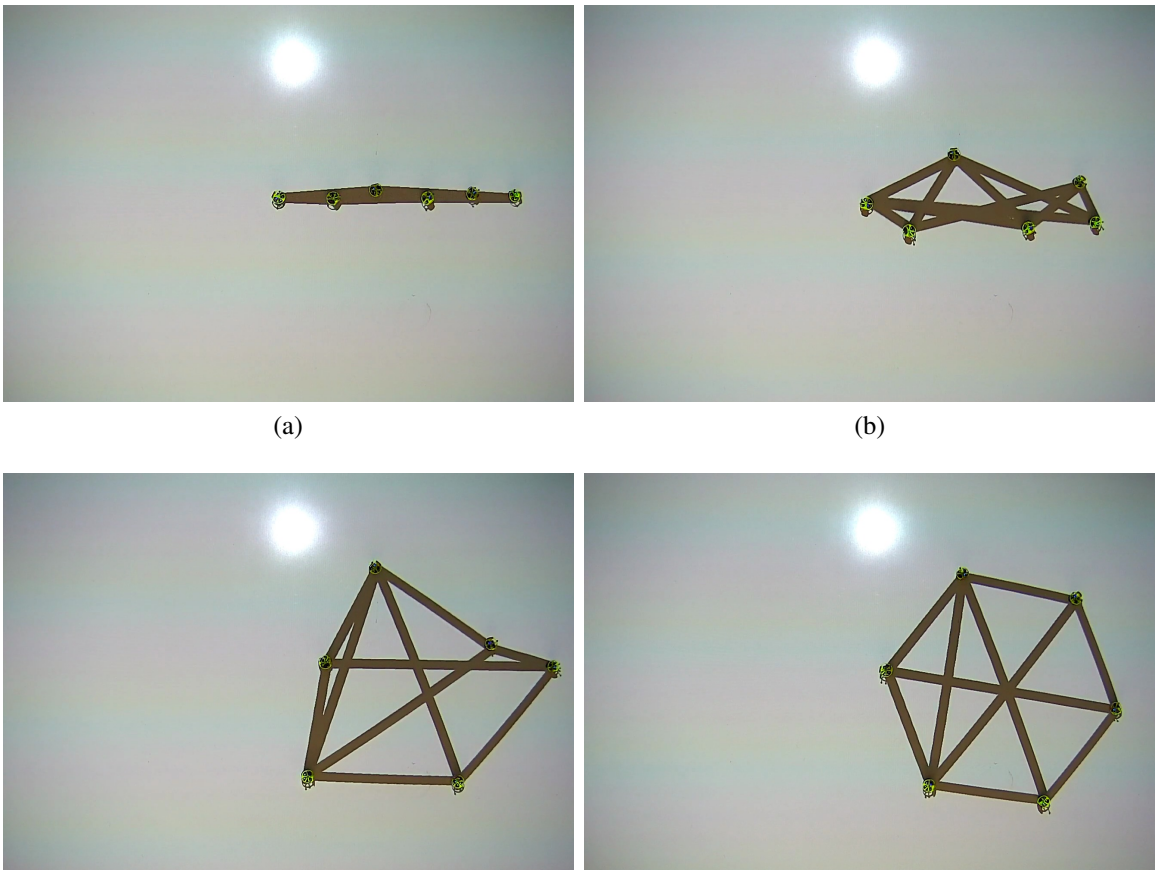
Throughout the chapter, we assumed we can directly control the velocity of the robots, by modeling them using single integrator dynamics. However, a differential-drive robot can be more accurately modeled using unicycle dynamics:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega, \end{cases} \tag{4.30}$$

where $[x, y]^T$ and θ are the robot's position and orientation in the plane, respectively, and v and ω are the linear and angular velocity inputs, respectively. Nevertheless, in [138], it is shown that it is possible to derive a near-identity diffeomorphism that can be used to partially feedback linearize the system (4.30). This way, the unicycle can be abstracted as a single integrator. This is realized through the invertible map

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = R^T(\theta) \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{d} \end{bmatrix} \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix}, \tag{4.31}$$

where $R(\theta)$ is the matrix that rotates vectors in $\mathbb{R}^2$ counterclockwise by an angle θ, and $[\dot{x}_d, \dot{y}_d]^T$ is the velocity in the plane of a point located in front of the unicycle at a distance d from its center. This method is used to control the robots on the Robotarium, by calculating



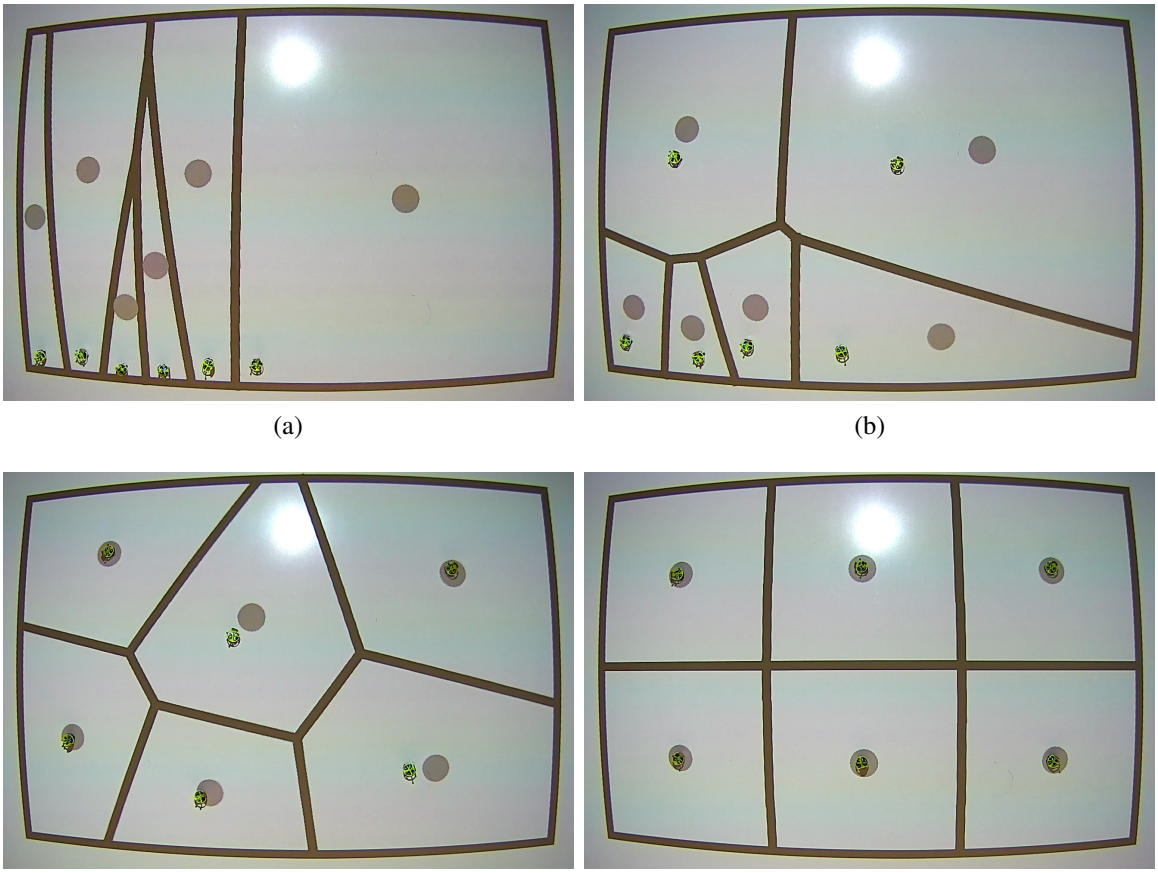
(a) (b)

(c) (d)

Figure 4.1: A team of six small-scale differential-drive robots on the Robotarium executes formation control using (4.22) and (4.23) in the optimization program (4.21). The edges encoding maintained distances between robots are projected onto the testbed.

linear and angular velocities of robot i, v_i and ω_i, from the control input $u_i = [\dot{x}_{d,i}, \dot{y}_{d,i}]^T$, obtained by solving the optimization problems derived in Section 4.2.

Regarding the implementation of the optimization program (4.21) needed for the execution of the tasks presented in the previous section, the function α has been chosen to be $\alpha(x) = \sqrt[3]{x}$, which is an extended class $\mathcal{K}$ function, convex for $x < 0$. This implies it is also superadditive for $x < 0$, as required by the hypotheses in Proposition 4.7.



(a) (b)

(c) (d)

Figure 4.2: A team of six small-scale differential-drive robots performs coverage control of a rectangular area on the Robotarium using (4.24) and (4.25) in (4.21). The Voronoi cells of the robots are projected onto the testbed, together with their centroids, depicted as gray circles.

4.4.1 Formation and Coverage Control

The optimization problem (4.21) has been implemented with the specific expressions of J_i and $\frac{\partial J_i}{\partial x_i}$ given in (4.22) and (4.23) in order to achieve formation control, as explained in Section 4.3.1. A sequence of snapshots recorded during the experiments in the Robotarium is shown in Fig. 4.1: six robots are asked to assemble a hexagon specified through the inter-agent distances d_{ij} in (4.22). The edges corresponding to distances that are maintained are projected down onto the testbed and depicted as black lines in Figures 4.1a to 4.1d.

Similarly, coverage control has been implemented using the constraint-based optimization (4.21) and the expressions of J_i and $\frac{\partial J_i}{\partial x_i}$ in (4.24) and (4.25). The results are shown in Fig. 4.2. Six robots are asked to spread over a rectangular domain. The Voronoi partition of the domain is projected on the testbed. As a result of the optimization program, the robots are moving towards the centroids of their respective Voronoi cells, represented as gray circles in Figures 4.2a to 4.2d.

4.4.2 Combining and Prioritizing Tasks

In this section, we present the application of the proposed constraint-driven coordinated control to long-term environmental monitoring.

The setup of the experiment is as follows. Six robots are asked to monitor an area by performing coverage control. While executing this task, the robots must not run out of energy and must not collide with two dynamic obstacles, embodied by two additional robots moving in the environment. In order to do so, we define constraints that allow the robots to always keep enough residual energy in their batteries and to be always a minimum distance apart from the obstacles.

To accomplish the first goal, we use a method similar to the one developed in Chapter 3 and in [106]. Assuming that the domain is endowed with charging stations, i.e. locations

where the robots can recharge their batteries, let us define the following barrier function:

$$h_{e,i}(x_i, E_i) = E_i - E_{min} - k(\|x_{c,i} - x_i\| - d_{chg})^2, \tag{4.32}$$

where x_i is the position of robot i, E_i is the energy in its battery, E_{min} is the minimum residual energy we want the robots to keep, $x_{c,i}$ is the location of the charging station dedicated to robot i, d_{chg} is the minimum distance from the charging station at which the robots can recharge their batteries (typical behavior of wireless charging technologies), and k is a constant such that $k(\|x_{c,i} - x_i\| - d_{chg})^2$ upper-bounds the energy required to reach a charging station.

As far as obstacle avoidance is concerned, we define, for each obstacle, the following barrier function, which ensures collision-free operations in multi-robot systems [139]:

$$h_{o,i}(x_i) = \|x_i - x_o\|^2 - d_o^2, \tag{4.33}$$

where x_o is the position of the obstacle and d_o is the minimum distance we want the robots to maintain from the obstacle.

Combining the energy constraint $\dot{h}_{e,i} \geq h_{e,i}(x_i, E_i)$ and the obstacle constraint $\dot{h}_{o,i} \geq h_{o,i}(x_i)$ together with the coverage task constraints, the following optimization problem can be formulated:

$$\begin{aligned} \underset{u_i, \delta_i}{\text{minimize}} \ & \|u_i\|^2 + \delta_i^2 \\ \text{subject to} \ & -\frac{\partial J_i}{\partial x_i} u_i \geq -\alpha(-J_i(x)) - \delta_i \\ & \dot{h}_{e,i} \geq h_{e,i}(x_i, E_i) \\ & \dot{h}_{o,i} \geq h_{o,i}(x_i). \end{aligned} \tag{4.34}$$

The variable δ_i in the coverage constraint acts as a relaxation parameter, which allows the constraints related to energy and collisions to be fulfilled. This translates to *trading task execution for survivability*. Consequently, this formulation allows tasks prioritization

obtained by combining hard and soft constraints.

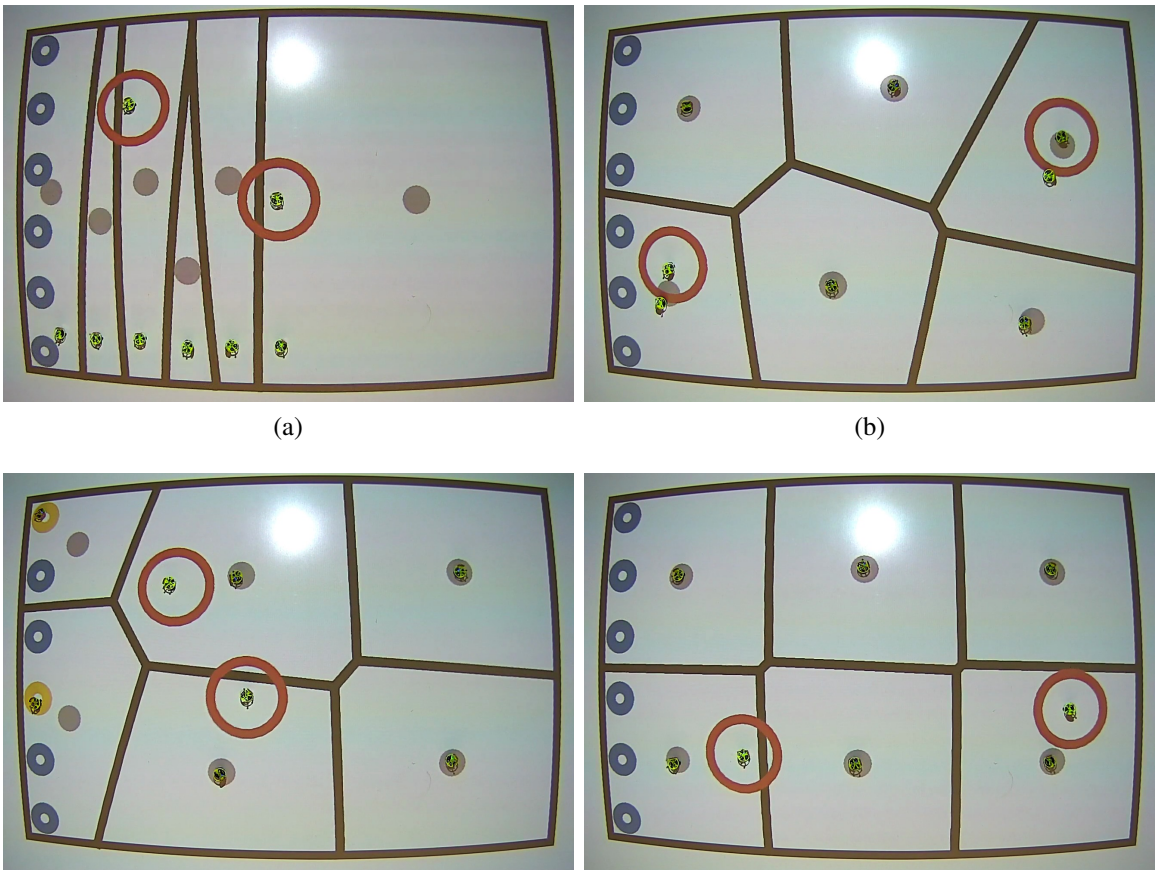
The results of the long-term environmental monitoring experiment are shown in Fig. 4.3. The Voronoi partition generated by the robots is projected down onto the testbed, as in Fig. 4.2. Arranged vertically along the left edge of the domain, there are six charging stations depicted as blue circles that turn yellow when the robots are charging. The two robots circle in red are moving in the environment acting solely as obstacles. The sequence of snapshots shows the robots starting to perform coverage (Fig. 4.3a), two robots avoiding a obstacles (top right and bottom left in Fig. 4.3b), and two robots recharging their batteries (Fig. 4.3c). In Fig. 4.3d the robots have reached a configuration corresponding to a local minimum of the locational cost (4.24). A video of the experiments can be found online [140].

Due to the limited amount of time that each experiment submitted to the Robotarium is allowed to last, we simulate the battery dynamics in such a way that the robots experience multiple charging cycles during the course of a single experiment. Fig. 4.4 shows the energy levels of the robots employed to perform coverage. The minimum desired energy level, E_{min}, and the value corresponding to fully charged battery, E_{chg}, are depicted as black thick lines. Enforcing the energy constraints using (4.32) allows the robots to keep their energy level always above E_{min}.

We have shown how the constraint-driven control formulation can be used to build a minimum-energy optimization problem, whose constraints encode both the task that the robots are asked to perform and the survivability specifications, thus enabling the robust deployment of robots for long-term applications.

4.5 Conclusions

In this chapter, we presented a reformulation of optimization-based robotic tasks in terms of constrained optimization. Identifying a task with a cost function that needs to be minimized, we leverage control barrier functions to synthesize optimization-based controllers



(a) (b)

(c) (d)

Figure 4.3: A team of six robots is tasked with monitoring a rectangular domain on the Robotarium, by performing coverage control as in Fig. 4.2. This time, however, the robots are asked to perform this task over a time horizon which is much longer than their (simulated) battery life. Additionally, two more robots (circled in red) act as obstacles which have to be avoided by the remaining six robots. These execute (4.34) to avoid the obstacles, go and recharge their batteries at the dedicated charging stations (blue circles on the left of the figures that turn yellow when the robots are charging), while always covering the given domain. A video of the experiments is available online [140].

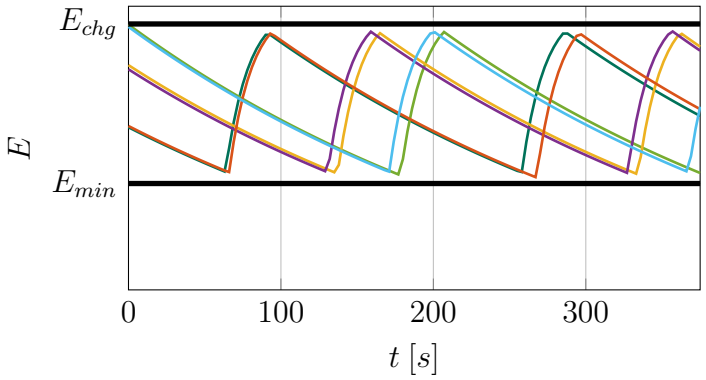


Figure 4.4: Simulated energy levels of the robots tasked with performing persistent coverage (Fig. 4.3). The residual energy is kept above a minimum desired value using (4.32). With the simulated energy dynamics, each robot experiences two charging cycles during the course of the experiment.

that achieve the desired goal—in a decentralized fashion, in the case of multi-robot tasks. The advantages of this approach include its flexibility of encoding a rich variety of tasks and the ease of combining them with different types of constraints. We showed how this flexibility can be used to enforce robot survivability and achieve long-term robot autonomy, where robustness and resilience are indispensable properties that robots have to possess. The effectiveness of the constraint-based approach is demonstrated through a series of experiments using a team of ground mobile robots, culminating in a long-term environmental monitoring application.

Starting from the prioritization technique presented in this chapter based on the constraint-driven robot control, in the next chapter, a multi-robot multi-task *allocation* algorithm is presented. In these settings, tasks are allocated by allowing different robots to prioritize different tasks in a different way.

CHAPTER 5
ENERGY-AWARE TASK ALLOCATION

In the previous chapter, a way of prioritizing tasks is presented, which is based on the distinction between safety-critical and non-safety-critical tasks. In the context of the constraint-driven control framework developed in Chapter 4—where each task is encoded as a constraint within an optimization problem—safety critical are represented by constraints that cannot be relaxed, whereas constraints corresponding to non-safety-critical tasks are relaxed by a slack variable.

In this chapter, we leverage the constraint-driven robot control framework to design a way of prioritize a stack of tasks by enforcing constraints on the slack variables of the tasks. This way, not only is it possible to differentiate between safety-critical and non-safety-critical, but virtually a continuum of priorities can be established. Moreover, once multiple tasks are to be executed by multiple robots, we propose an energy-aware framework for allocating tasks to robots in an online fashion With a primary focus on long-duration autonomy applications, we opt for a *survivability*-focused approach. In this context, an allocation is interpreted as a prioritization of a task over all others by each of the robots. Furthermore, we present a novel model for robot specializations at performing tasks based on features and capabilities of the robots. We leverage these descriptions in the optimization problem to make robot operations *resilient* to situations where environmental conditions make certain features unsuitable to support a capability and when component failures on the robots occur.

Multi-robot task allocation (MRTA) is an active research area in an era where the deployment of multi-robot systems in dynamic and unknown environments is becoming more and more common ([58, 59] and references therein). Many envisioned applications require robots with limited energy resources to operate effectively for long periods of time, neces-

sitating the development of *survivability*-focused energy-aware algorithms for task execution as well as allocation [4, 10]. Similarly, robot *heterogeneity* has received explicit focus within the MRTA literature, as teams equipped with different types of sensors, actuators, and communication devices can enable the execution of a wider range of tasks [61, 73, 71].

Heterogeneity contributes to another desirable property of a MRTA algorithms in the context of long-duration autonomy, namely *resilience*, typically interpreted as the ability of the allocation algorithm to react to component failures on the robots, varying environmental conditions, and other non-idealities in the operating conditions [79]. It should be noted that, in light of such a scenario, the multi-robot task *allocation* problem can be considered as being inextricably linked to the *execution* of the tasks by the robots. This is especially true when considering the deployment of survivability-focused multi-robot systems over long time horizons, where evolving or newly detected environmental features will certainly affect the effectiveness of the task execution.

This chapter presents a dynamic task allocation and execution framework for multi-robot systems which explicitly accounts for the aforementioned survivability and heterogeneity considerations while being demonstrably resilient to robot failures and changes in environmental conditions [12]. To encode heterogeneity, we propose a novel framework for representing the compatibility of robots with tasks in terms of the *capabilities* required to perform the tasks (e.g., flight or high speed) as well as the *features* available on the robots (e.g., a specific type of sensors, actuators, or communication equipment) which support these capabilities. We embed this representation within a constraint-based optimization framework whose solution at each point in time yields (i) a dynamic allocation of tasks to robots through a prioritization scheme, and (ii) control inputs to each robot which ensure the execution of the tasks in accordance with the optimized priorities [10].

Existing task allocation techniques typically define both robots and tasks in terms of the capabilities available on the robots and required to perform the tasks [71, 141]. In contrast, our approach distinguishes between the features available on the robots and the capabilities

that these features enable. We demonstrate that this explicit representation contributes to the resilience of the proposed dynamic task allocation method, leveraging the fact that multiple bundles of robot features can satisfy the same capability. Consequently, dynamic readjustments of Robot-to-Feature and Feature-to-Capability mappings can enhance the resilience of the system by capturing scenarios in which (i) environmental conditions make a certain feature more suitable to support a capability, and (ii) component failures on robots occur, affecting the available features. The pertinent question then becomes: *how can we design a survivability-focused dynamic allocation paradigm based on these descriptions of heterogeneity (at the feature level as well as the capability level) with demonstrably resilient operations?*

Following preliminary work in [7], [10], and [11], we opt for a *constraint-based* approach in this chapter, which encodes the execution and prioritization of tasks as constraints within an optimization problem. Such a formulation has demonstrated higher flexibility and robustness in scenarios where the operating conditions of the robots are only partially known or may change—events which are especially likely when considering long-duration autonomy applications [142]. Similarly to (1.1), as energy considerations are paramount in our framework, the execution of n_t different tasks by a robot can be posed as the following energy-minimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}} \ \|u\|^2 \\ &\text{subject to } c_{\text{task}_j}(x,u) \geq 0, \quad \forall j \in \{1,\dots,n_t\}, \end{aligned} \tag{5.1}$$

where x is the current state of the robot, u is the control effort expended by it, and the inequality $c_{\text{task}_j}(x,u) \geq 0$ denotes a constraint function which enforces the execution of task j. The feasibility of such an optimization problem is ensured by the introduction of a

slack variable $\delta \in \mathbb{R}^{n_t}$:

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}} \ \|u\|^2 + \|\delta\|^2 \\ &\text{subject to } c_{\text{task}_j}(x,u) \geq -\delta_j, \quad \forall j \in \{1,\dots,n_t\}, \end{aligned} \tag{5.2}$$

where $\delta = [\delta_1, \delta_2, \dots, \delta_{n_t}]^T$ is a vector with positive components representing the extent to which the robot can violate the constraints corresponding to each of the tasks. Notice that, compared to what has been presented in Chapter 4, in this formulation each task j is relaxed by its slack variable δ_j.

The applicability of this framework for dynamic task allocation via prioritization is enabled by the observation that relative constraints among the components of δ can allow a robot to perform one task more effectively than others. For instance, if $\delta_m \ll \delta_n, \forall n \neq m$, then the robot will execute task m with priority higher than all the other tasks: this represents an *allocation via prioritization*. Such a prioritization can be then encoded via an additional constraint $K\delta \leq 0$ in (5.2), where the matrix K encodes the relative inequalities among the slack variables.

Within the above described formulation, the allocation problem then consists of designing the matrix K for each robot such that the heterogeneity of the robots—intended as their different ability of performing different tasks—are appropriately accounted for. To this end, we propose a modification of the minimum energy optimization problem presented in (5.2) where priority matrices K are automatically generated. Moreover, by means of an additional constraint, the optimization problem ensures that the minimum amount of capabilities required for the successful execution of each task is met by the allocation.

This formulation yields a mixed-integer quadratic program (MIQP) which not only generates the task allocation of the team (encoded via the prioritization matrices K) but also the control inputs u which the robots can use to execute the tasks. Since MIQPs can be computationally intensive to solve, we further present a mixed centralized/decentralized computational architecture which allows a central coordinator to transmit the task priorities

to each robot. The robots can then solve the simpler convex quadratic program described in (5.2) (with the additional constraint $K\delta \leq 0$) to generate their control inputs in real time.

We demonstrate that non-idealities such as environmental disturbances and/or component failures on the robots can be effectively accounted for in our framework, enabling, this way, resilient task allocation behaviors. It is informative to know how, also in nature, such behaviors emerge from the concepts of survivability and heterogeneity. In fact, these concepts play a central role in ecological studies as well, as highlighted by Bridle and van Rensburg in [143]:

> *For some groups of organisms, we can now integrate genomic data with environmental and demographic data to test the extent to which ecological resilience depends on evolutionary adaptation. Such data will allow researchers to estimate when and where biodiversity within a species has the power to rescue ecological communities from collapse due to climate change and habitat loss.*

Drawing an analogy with the task allocation framework we present in this chapter, the features of the robots (*genomic data*) and the resulting heterogeneity (*biodiversity*) are leveraged to introduce a degree of resilience (*ecological resilience*) into the framework, which results in a natural adaptation of the multi-robot system to failures (*collapse*) due to the dynamic environments in which it operates (*climate change and habitat loss*).

The remainder of the chapter is organized as follows. Section 5.1 introduces the problem formulation and places it within the context of existing literature. Section 5.1.1 develops a novel framework for encoding robot heterogeneity. In Section 5.1.2, we present the main constraint-based minimum energy task allocation paradigm, and demonstrate its resilient capabilities in two distinct failure scenarios. Section 5.3 touches upon the performance guarantees of the developed task allocation paradigm and highlights a mixed centralized/decentralized framework to enable the task allocation and execution. In Section 5.4, we present example use-case scenarios highlighting the resilient allocation and

execution of multiple tasks.

5.1 Problem Formulation

Consider a team of n_r heterogeneous robots which are deployed in an environment and required to execute n_t tasks. Each robot is endowed with a subset of n_f available features (such as camera, LIDAR, and wheels). These features allow the robots to exhibit a subset of n_c capabilities (such as flight and navigation). Certain capabilities can be achieved by multiple combinations of feature bundles, whereas tasks require a given set of capabilities in order to be executed. The successful execution of each task is conditioned upon a minimum number of robots with specific capabilities being allocated to it. In this chapter, we consider *extended set-based tasks* [9], which include tasks whose execution can be encoded as the minimization of a cost function.

Given the above problem setup, the framework presented in this chapter concerns itself with (i) allocating tasks among the robots such that the minimum requirements for each task are met, and (ii) executing the tasks by synthesizing an appropriate control input for the robots. Both these objectives are met while minimizing the control effort expended by the robots. Additionally, we show how the resulting task allocation and execution framework exhibits resilience properties against varying environment conditions and failures on the robots.

5.1.1 Encoding Robot Heterogeneity

The objective of this section is to develop a framework which generates a feasible mapping between robots and their assigned tasks, while explicitly accounting for the heterogeneity in the robots and the different capability requirements of the tasks. We define a novel notion of feasibility based on a newly added feature layer in the description of the robots. Intuitively, a feasible assignment needs to take into account the capabilities needed for the tasks along with the features possessed by the robots. For example, assigning a ground ve-

Table 5.1: Notation

Symbol	Description	Section
n_r	Number of robots	5.1
n_t	Number of tasks	5.1
n_c	Number of capabilities of all robots	5.1
n_f	Number of features of all robots	5.1
$T \in \{0,1\}^{n_t \times n_c}$	Capability-to-Task mapping	5.1.1
$A \in \{0,1\}^{n_f \times n_r}$	Robot-to-Feature mapping	5.1.1
$H_k \in [0,1]^{n_{c_k} \times n_f}$	Feature-to-Capability mapping	5.5
$F \in \mathbb{R}^{n_c \times n_r}$	Robot-to-Capability mapping	5.1.1
$S_i \in \mathbb{R}^{n_t \times n_t}$	Specialization matrix of robot i	5.1.1
$\alpha \in \{0,1\}^{n_t \times n_r}$	Matrix of task priorities	5.2
$\delta_i \in \mathbb{R}^{n_t}$	Task relaxation for robot i	5.1.3
$\delta \in \mathbb{R}^{n_t n_r}$	Vector of task relaxation parameters	5.2
$x_i \in \mathbb{R}^{n_x}$	State of robot i	5.1.2
$u_i \in \mathbb{R}^{n_u}$	Control input of robot i	5.1.2
$x \in \mathbb{R}^{n_x n_r}$	Ensemble state of n_r robots	5.1.3
$u \in \mathbb{R}^{n_u n_r}$	Ensemble control input of n_r robots	5.1.2

hicle to an aerial-surveillance task would be considered an unfeasible assignment. Shown in Fig. 5.1 is an example of the three mappings to be introduced in the next subsections. Starting from the left, we begin by introducing the Capability-to-Task mapping (T) which contains the task specifications. In turn, each of those capabilities requires any one of various feature-bundles to be exhibited. This is captured in the Feature-to-Capability mapping (H_k) through the use of hypergraphs. In subsection 5.1.1, we define the Robot-to-Feature mapping (A) which maps each robot to the set of features it possesses. Finally, we introduce a way to obtain the Robot-to-Capability mapping (F) which is directly used in the task allocation and execution framework. See Table 5.1 for a summary of this notation, as well as the one used throughout the chapter.

Capability-to-Task Mapping

We define the mapping from the set of tasks at hand to their respective capabilities as

$$T \in \{0,1\}^{n_t \times n_c}, \tag{5.3}$$

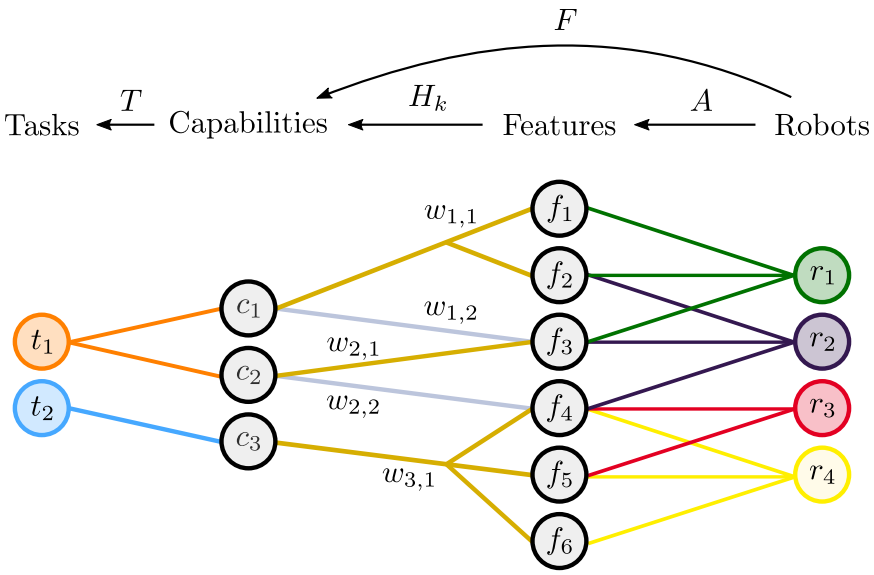


Figure 5.1: Example of scenario including 2 tasks, 3 capabilities, 6 features and 4 robots shown from left to right. The capabilities to features mapping is shown through the gold and silver hyperedges. Note that not all of the hyperedges need to have the same cardinality.

where $T_{tk} = 1$ if and only if task t requires capability k, and n_t and n_c denote the numbers of tasks and capabilities respectively. In Fig. 5.1, we present a graphical representation of the assignment problem consisting of two tasks and three capabilities denoted by t_t and c_k, respectively. The entries of the matrix T are represented by the edges of the mapping from tasks to capabilities.

Robot-to-Feature Mapping

Each robot available for assignment possesses a variety of features. For example, an e-puck's features include an IMU and a CMOS camera [144]. Therefore, we define the following binary mapping from robots to their respective features:

$$A \in \{0,1\}^{n_f \times n_r}, \tag{5.4}$$

where $A_{ij} = 1$ if and only if robot j possesses feature i, and n_r and n_f denote the number of robots and features, respectively. Now that we have defined both the Capability-to-Task and Robot-to-Feature mappings, we have the required constructions to introduce the

Feature-to-Capability mapping.

Feature-to-Capability Mapping

When considering heterogeneous multi-robot systems, it is important to note that two non-identical robots may be able to support the same capability (i.e. robots possessing different sensors and actuators can be interchangeable when it comes to supporting a specific capability). To model this effect, we use the notion of a bipartite hypergraph to define the Feature-to-Capability mapping. A hypergraph is a graph whose edges are not restricted to a cardinality of two. Each hyperedge associates a capability with one of the feature bundles that can support it. The mapping between capabilities and features in the middle of Fig. 5.1 is an example of a bipartite hypergraph. The top edge (colored golden) mapping c_1 to f_1 and f_2 indicates that, together, these two features can support capability c_1. Mathematically, hyperedge k is represented by the following matrix:

$$H_k \in [0,1]^{n_{c_k} \times n_f}, \tag{5.5}$$

where k denotes the capability index, $H_{k,ij} \neq 0$ if and only if feature j belongs to the feature bundle denoted by hyperedge i. n_{c_k} and n_f denote the number of hyperedges incident to capability k and the number of features, respectively.

Mapping Robots to Capabilities

The MRTA algorithm presented in this chapter can be referred to as ST-MR-IA (Single-Task robots, Multi-Robot tasks, Instantaneous Assignment) [58]: in fact, (i) through prioritization, each robot is assigned to a single task, (ii) the tasks can be executed by multiple robots, in a coordinated or independent fashion, and (iii) the allocation of tasks to robots is carried out at each time instant, without planning for future allocations. Previous approaches to solving ST-MR-IA MRTA problems assume knowledge of the direct mapping

from capabilities to robots encoded by a matrix

$$F \in \mathbb{R}^{n_c \times n_r} \tag{5.6}$$

where $F_{kj} \neq 0$ if and only if robot j can support capability k. Therefore, in this subsection, we state the required condition under which robot j can indeed support capability k, and derive the matrix F required by such algorithms based on this condition. Notice that the framework only accounts for a finite number of capabilities and features relevant to the required tasks, so the computation of F remains tractable.

As mentioned above, a capability k can be supported by a number of feature bundles. Consequently, a robot must possess all the features in at least one of the bundles associated with a capability in order to support it. Hence, we say robot i supports capability k if and only if it possesses all the features within a hyperedge associated with capability k. For example, in Fig. 5.1, robot r_1 can support capability c_1 since it possesses features f_1 and f_2 included in the top hyperedge. On the other hand, robot r_3 cannot support capability c_3 since it only possesses features f_4 and f_5 but not f_6. We define the feasibility vector F_k capturing which robots can satisfy capability k as:

$$F_k = \max(\mathrm{kron}_1(H_k A)), \tag{5.7}$$

where kron_1 denotes the shifted Kronecker delta function

$$\mathrm{kron}_1(x) = \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

applied element-wise. The function kron_1 is introduced to eliminate cases where robots have an incomplete portion of the features in a hyperedge. Moreover, the $\max$ operator is intended column-wise, and serves to check whether a robot possesses all the features from

at least one bundle. Note that using the $\max$ operator in the case of a robot satisfying a capability through multiple hyperedges selects only one of those edges, which will become relevant when we introduce weights in the next section.

Shifting our attention to the example from Fig. 5.1, we can compute the feasibility vector F_3 corresponding to c_3:

$$H_3 A = \begin{bmatrix} 0 & 0 & 2/3 & 1 \end{bmatrix} \tag{5.9}$$

whose ij-th component is the proportion of features that robot j possesses belonging to hyperedge i incident to capability k. Therefore, in this case, robot r_3 possesses only $2/3$ of the features in the only bundle associated with c_3, and therefore cannot support that capability. We thus obtain the following feasibility vector for c_3:

$$F_3 = \max(f(H_3 A)) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

As illustrated above, if $F_{k,j} = 1$, then robot j can support capability k. Therefore, by concatenating all the vectors F_k, we obtain the desired linear mapping from capabilities to robots:

$$F = \begin{bmatrix} F_1^T & F_2^T & \dots & F_{n_c}^T \end{bmatrix}^T. \tag{5.11}$$

As such, we can define a feasible assignment as one where all the capabilities required by each task t are at least supported by a given number robots assigned to task t, i.e.

$$\sum_{i \in \mathcal{R}_t} F_{-,i} \geq T_{t,-}, \tag{5.12}$$

where the notation $T_{t,-}$ and $F_{-,i}$ is used to denote the t-th row of T and the i-th column of F. The inequality in (5.12) holds element-wise, and $\mathcal{R}_t$ denotes the set of robots assigned to task t. $T_{t,k} = n$ indicates that at least n of the robots assigned to task t need to exhibit capability k.

Specialization Matrix

To conclude the model of robot heterogeneity used within the task allocation framework proposed in this chapter, we define the requirements for a robot to be considered as a potential candidate for a task. As opposed to [10], where the specialization matrices were assumed to be given, we leverage the above developed feature and capability models to compute the specialization matrix of robot i as follows:

$$S_i = \text{diag}(\mathbb{1}_{n_t} - \text{kron}(TF_{-,i})) \in \mathbb{R}^{n_t \times n_t}, \tag{5.13}$$

where, for $m \in \mathbb{R}^n$, $\text{diag}(m) = M \in \mathbb{R}^{n \times n}$ such that

$$M_{ij} = \begin{cases} m_i & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{5.14}$$

$\mathbb{1}_{n_t}$ is a vector of dimension n_t whose entries are all equal to 1, and $\text{kron}(\cdot)$ denotes the Kronecker delta function ($\text{kron}(0) = 1$ and $\text{kron}(n) = 0,\ \forall n \neq 0$) applied element-wise. In other words, the specialization of robot i towards task j, s_{ij}, is given by

$$s_{ij} = \begin{cases} 1 & \text{if } T_{j,-}F_{-,i} > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

i.e. $s_{ij} = 1$ if robot i exhibits at least one capability required by task j. The motivation behind this choice is two-fold: robots are allowed to combine their capabilities to satisfy a task, and there is no notion of priority between capabilities (i.e. exhibiting capability 1 is more or less crucial than exhibiting capability 2 and 3). The former indicates that if a robot exhibits even a single capability relevant to the task, it may still be able to contribute, whereas the latter indicates that there is no possible ordering of the candidates in terms of specialization.

Finally, as will be shown in Section 5.4, the specialization matrix can be adapted on-the-fly. For example, in the case a robot loses a feature (e.g. its camera is malfunctioning), by removing the edges between the robot and the feature, we can re-compute which capabilities the malfunctioning robot can still exhibit.

5.1.2 Task Execution and Prioritization

This section develops a minimum energy task allocation framework, through prioritization and execution, that explicitly accounts for the heterogeneity of the robots expressed in terms of their capabilities, as well as specifications on the capabilities required to execute each task. Moreover, we demonstrate how the proposed task allocation framework introduces a degree of resilience, allowing the robots to react, for instance, to component failures and, more generally, to unmodeled or unexpected environmental conditions.

As stated in Section 5.1, we consider a team of n_r robots tasked with executing n_t different tasks in the environment. We model the dynamics of each robot $i \in \{1, \dots, n_r\}$ with a control-affine dynamical system:

$$\dot{x}_i = f(x_i) + g(x_i)u_i \tag{5.16}$$

where f and g are locally Lipschitz continuous vector fields, $x_i \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the state of the robot, and $u_i \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the input. Note that, in this chapter, we assume that all robots obey the same dynamics given in (5.16), however, the entire formulation can be extended in a straightforward fashion to the case where individual robots have different dynamics. As done in [9] and in Chapter 4, we use Control Barrier Functions (CBFs) (see [45] and references therein) to encode the set-based tasks that the robots are required to execute. In the following, we briefly recall the definition and the main properties of CBFs as will be used in this chapter to formulate the task prioritization and execution framework.

Definition 5.1 ([45]). *Let $\mathcal{C} \subset \mathcal{D} \subset \mathbb{R}^n$ be the zero superlevel set of a continuously differ-*

entiable function $h\colon \mathcal{D} \to \mathbb{R}$. Then h is a control barrier function (CBF) if there exists an extended class $\mathcal{K}_\infty$ function γ[1] such that, for the control affine system $\dot{x} = f(x) + g(x)u$, $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, one has

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x) u\} \geq -\gamma(h(x)). \tag{5.17}$$

for all $x \in \mathcal{D}$.

The notation $L_f h(x)$ and $L_g h(x)$ are used to denote the Lie derivative of h along the vector fields f and g, respectively. Given this definition of CBFs, the following theorem highlights how they can be used to ensure both set forward invariance and stability [116].

Theorem 5.2 ([45])**.** *Let $\mathcal{C} \subset \mathbb{R}^n$ be a set defined as the zero superlevel set of a continuously differentiable function $h : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$. If h is a control barrier function on $\mathcal{D}$ and $\frac{\partial h}{\partial x}(x) \neq 0$ for all $x \in \partial\mathcal{C}$, then any Lipschitz continuous controller $u(x) \in \{u \in \mathcal{U}\colon L_f h(x) + L_g h(x) u + \gamma(h(x)) \geq 0\}$ for the system $\dot{x} = f(x) + g(x)u$, $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, renders the set $\mathcal{C}$ forward invariant. Additionally, the set $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$.*

The results of this theorem will be used in the remainder of this section to design a control framework that allows a heterogeneous multi-robot system to prioritize and perform a set of tasks that need to be executed.

5.1.3 Constraint-Driven Task Execution

The formulation adopted in this chapter in terms of extended set-based tasks [9] allows us to encode a large variety of tasks: these are tasks characterized by a set, which is to be rendered either forward invariant (usually referred to as *safe* in dynamical system theory [45]), or asymptotically stable, or both. The results recalled above suggest the use of CBFs

[1] An extended class $\mathcal{K}_\infty$ function is a continuous function $\gamma : \mathbb{R} \to \mathbb{R}$ that is strictly increasing and with $\gamma(0) = 0$.

to encode these kinds of tasks. Indeed, CBFs have been successfully used to encode a variety of such tasks for different robotic platforms, ranging from coordinated control of multi-robot systems [10] to multi-task prioritization for robotic manipulators [9]. In particular, in [9] the definition of extended set-based tasks, i.e. tasks which consist in the state x approaching a set (stability) or remaining within a set (safety), is formalized.

As shown in Chapter 4 and in [7], among the extended set-based tasks, there is a class of coordinated multi-robot tasks which are executed through the minimization of a cost function, realized, for instance, by gradient-flow-like control laws [134]. In multi-task multi-robot settings, this framework naturally allows robots to combine multiple constraints, each representing a task, into a single framework. For tasks encoded via CBFs $h_m, m \in \{1, \ldots, n_t\}$, the constraint-based optimization problem for robot i can be written as,

$$\begin{aligned} &\underset{u_i, \delta_i}{\text{minimize}} \ \|u_i\|^2 + \|\delta_i\|^2 \\ &\text{subject to } L_f h_m(x) + L_g h_m(x) u \geq -\gamma(h_m(x)) - \delta_{im} \\ &\qquad\qquad \forall m \in \{1 \ldots n_t\}, \end{aligned} \tag{5.18}$$

where $\delta_i = [\delta_{i1}, \ldots, \delta_{in_t}]^T$ represents the slack variables corresponding to each task being executed by robot i. The tasks encoded by the CBFs $h_m(x)$ are not restricted to be dependent only on the state of robot i, but rather on the ensemble state of the robots $x = [x_1^T, \ldots, x_{n_r}^T]^T$, thus allowing the framework to encompass coordinated multi-robot tasks.

With this framework in place, the slack variables δ_i present a natural way of encoding task priorities for the individual robots. This will be the subject of the next section, where the main task allocation framework is presented.

5.2 Task Allocation Algorithm

In section 5.1.1, we presented a framework to model robot heterogeneity—exhibited in the different suitability that each robot has for different tasks—starting from the lower level concepts of robot features and capabilities. In this section, we leverage the expressiveness of this model in order to render the task prioritization framework, presented in [10] and improved in [11], resilient.

The optimization-based formulation extends the one in (5.18) as follows:

Task allocation optimization problem (MIQP) (5.19)

$$\underset{u,\delta,\alpha}{\text{minimize}} \sum_{i=1}^{n_r} \left(C\|\Pi_i \alpha_{-,i}\|^2 + \|u_i\|^2 + l\|\delta_i\|_{S_i}^2 \right) \tag{5.19a}$$

$$\text{subject to } L_f h_m(x) + L_g h_m(x) u_i \geq -\gamma(h_m(x)) - \delta_{im} \tag{5.19b}$$

$$\Theta\delta_i + \Phi\alpha_{-,i} \leq \Psi \tag{5.19c}$$

$$\mathbb{1}_{n_t}^T \alpha_{-,i} \leq 1 \tag{5.19d}$$

$$F\alpha_{m,-}^T \geq T_{m,-}^T \tag{5.19e}$$

$$n_{r,m,\text{min}} \leq \mathbb{1}^T \alpha_{m,-}^T \leq n_{r,m,\text{max}} \tag{5.19f}$$

$$\|\delta_i\|_\infty \leq \delta_{\text{max}} \tag{5.19g}$$

$$\alpha \in \{0,1\}^{n_t \times n_r} \tag{5.19h}$$

$$\forall i \in \{1 \dots n_r\},\ \forall m \in \{1 \dots n_t\}, \tag{5.19i}$$

where $C, l \in \mathbb{R}_{\geq 0}$ are parameters of the optimization, δ_{max} signifies the maximum extent to which each task constraint can be relaxed, and γ is a continuously differentiable class $\mathcal{K}_\infty$ function. The matrix Π_i is a projection matrix defined in (5.23) to account for the heterogeneous capabilities of the multi-robot system, as explained in detail later.

First of all, as done in Section 5.1.1, the symbols $X_{i,-}$ and $X_{-,j}$ denote the i-th row and the j-th column of the matrix X, respectively. The introduction of the matrix of task priori-

ties $\alpha \in \{0,1\}^{n_t \times n_r}$ in the optimization problem is what determines the prioritization (and, therefore, the allocation) of the tasks for each robot. This is realized through the constraint (5.19c), where the matrices Θ and Φ, and the vector Ψ, enforce constraints among different components of the vectors of task relaxation parameters δ_i. As extensively discussed in [10], the constraint

$$\delta_{in} \geq \kappa\big(\delta_{im} - \delta_{\max}(1 - \alpha_{im})\big), \;\; n \neq m, \tag{5.20}$$

that can be written as (5.19c), realizes the following two implications:

$$\alpha_{im} = 1 \implies \delta_{im} \leq \frac{1}{\kappa}\delta_{in} \quad \forall n \in \{1, \ldots, n_t\} \setminus \{m\}, \tag{5.21}$$

which implies that task m has highest priority for robot i, and

$$\alpha_{im} = 0 \implies \delta_{im} \leq \delta_{\max} + \frac{1}{\kappa}\delta_{in} \quad \forall n \in \{1, \ldots, n_t\} \setminus \{m\}, \tag{5.22}$$

which implies that task m does not have the highest priority for robot i. In fact, in light of constraint (5.19g), no further constraints are enforced on δ_{im}, since $\delta_{\max}$ is the maximum value $|\delta_{im}|$ is allowed to achieve. Notice further that, for the way it is used in (5.2), the optimal value of δ will always be non-negative (see also analyses in [7, 10]).

The constraint (5.19d) is used to ensure that each robot has at most one task to be executed with highest priority, making the task prioritization formulation effectively a task allocation. Notice that, compared to our previous work [10], (5.19d) is here turned from an equality into an inequality constraint. In [10], this constraint was used to ensure that no feasible solution consisted in robots trading off task execution for energy saving. In the presented, enhanced, formulation, this is not necessary anymore thanks to constraint (5.19e)—whose meaning will be described in the following. Consequently, we can now account for situations where no tasks are allocated to some of the robots, implementing, as a matter of fact, the concept of *autonomy-on-demand* in the context of task allocation.

The constraint (5.19e) is what allows us to specify the minimum capabilities required for each task, expressed by the matrices T and F defined in Sections 5.1.1 and 5.1.1, respectively. Moreover, the constraint (5.19f) allows us to enforce the minimum and maximum number of robots required for each task, thus giving a lot of flexibility and versatility to be utilized in many different application scenarios. In Section 5.4, experiments performed on a real multi-robot system will showcase the use of these constraints.

As in our previous works [10] and [11], the cost of the optimization problem (5.19) is composed of 3 terms. The last two terms in (5.19a) correspond to the control effort spent by the robots and the magnitude of the relaxation parameters, respectively. The former is amenable in long-duration autonomy applications: when the robots are required to remain operational over sustained periods of time, minimizing the energy spent in performing a task—which is proportional to control effort—is paramount. The latter, instead, ensures that the tasks to which the robots have been assigned get indeed executed, thanks to constraint (5.19b). The norm of δ_i corresponding to robot i is weighted by the specialization matrix of robot i, S_i. This way, the relaxation variables corresponding to tasks that robot i is not capable of performing (i.e. with a low value of the entry of the specialization matrix) are weighted accordingly less.

Finally, the first term in (5.19a) is introduced to penalize *bad allocations* of tasks to robots, as explained in the following. The matrix Π_i is defined as follows:

$$\Pi_i = I_{n_t} - S_i S_i^{\dagger}, \tag{5.23}$$

where I_{n_t} is the $n_t \times n_t$ identity matrix, and $S_i^{\dagger}$ is the right Moore-Penrose inverse [145] of the specialization matrix S_i of robot i. It is easy to see that Π_i is the projector onto the orthogonal complement of the subspace of specializations possessed by robot i. Assume, for example, that robot i has no specialization at all at performing task k (i.e. $s_{ik} = 0$) and has a non-zero specialization s_{ij} of performing task j, $j \neq k$. Then, its specialization

matrix S_i will be given by:

$$S_i = \text{diag}([s_{i1}, \ldots, s_{i\,k-1}, 0, s_{i\,k+1}, \ldots, s_{in_t}]), \tag{5.24}$$

and

$$\Pi_i = \text{diag}([\underbrace{0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{n_t-k}]). \tag{5.25}$$

Then, the projector Π_i in the cost (5.19a) will contribute to a non-zero cost when the components of α_i corresponding to tasks that robot i has no specialization to perform are not zero, i.e. when robot i has been assigned to a task that it is not able to perform—referred above as a *bad allocation*.

Remark 5.3 (Centralized Mixed-Integer Quadratic Program)**.** *Notice that in* (5.19) *there is a coupling between the robots through the cost as well as the constraints. This means that the task allocation framework has to be solved in a centralized fashion. Moreover, the matrix of task priorities α is integer. This renders* (5.19) *a mixed-integer quadratic program (MIQP). A QP-relaxation approach, as well as ways of solving this framework in a decentralized way, are discussed in [10]. In Section 5.3 of this chapter, we show how the proposed MIQP can be solved in a mixed centralized/decentralized fashion, and we analyze the performances compared to the centralized approach.*

Remark 5.4 (Time-varying and sequential tasks)**.** *Expressing tasks by means of control barrier functions, besides rendering the task execution and allocation particularly amenable for online-optimization-based controllers, allows us to account for time-varying and sequential tasks, comprised by a sequence of sub-tasks, as well. In fact, the time-varying extension of control barrier functions (see, e.g., [142]) can be leveraged to consider tasks which have an explicit dependence on time. In the experimental section, we show how this extension of the proposed task allocation and execution framework can be used to implement state-trajectory-tracking tasks.*

Moreover, thanks to the pointwise-in-time nature of the developed optimization program, tasks can be removed and inserted in a continuous fashion, as demonstrated in [9]. This allows for a flexible implementation of sequential tasks which require the completion of a sub-task before another one can be started, as done in [146]. In the same way, the features and the specialization of the robots towards different tasks can be modified during the execution of the task. In the next subsection, we present an approach to leverage time-varying specialization in order to adapt to disturbances or modeled phenomena in the environment.

The following algorithm summarizes the application of the optimization-based allocation and execution framework to a multi-robot system with heterogeneous capabilities.

Algorithm 1 Task allocation and execution

Require:

Tasks $h_m,\ m \in \{1, \dots, n_t\}$

Mappings F, T

Parameters $n_{r,m,\text{min}}, n_{r,m,\text{max}}, \delta_{\text{max}}, C, l$

1: Evaluate $S_i,\ \forall i \in \{1, \dots, n_r\}$ ▷ (5.13)

2: **while** true **do**

3: Get robot state $x_i, \forall i \in \{1, \dots, n_r\}$

4: Compute robot input $u_i, \forall i \in \{1, \dots, n_r\}$ ▷ (5.19)

5: Send input $u_i, \forall i \in \{1, \dots, n_r\}$ to robots and execute

6: **end while**

We conclude this subsection by showcasing the execution of Algorithm 1 in an explanatory example featuring the use of the allocation constraints described so far.

Example 5.5. *Consider 4 mobile robots moving in a 2-dimensional space, tasked with performing 2 tasks. For clarity of exposition, in this example, we modeled each robot i as single integrator $\dot{x}_i = u_i$—so f and g in* (5.16) *are the zero and identity map, respectively—and each task consists in going to a point of the state space. In Fig. 5.2, the robots are depicted as gray triangles and labeled r_1 to r_4, whereas the locations corresponding to the tasks are labeled t_1 and t_2. The features, capabilities, and task mappings have been set as in Fig. 5.1, where the numerical quantities are given in Section 5.1.1. So, per* (5.13),

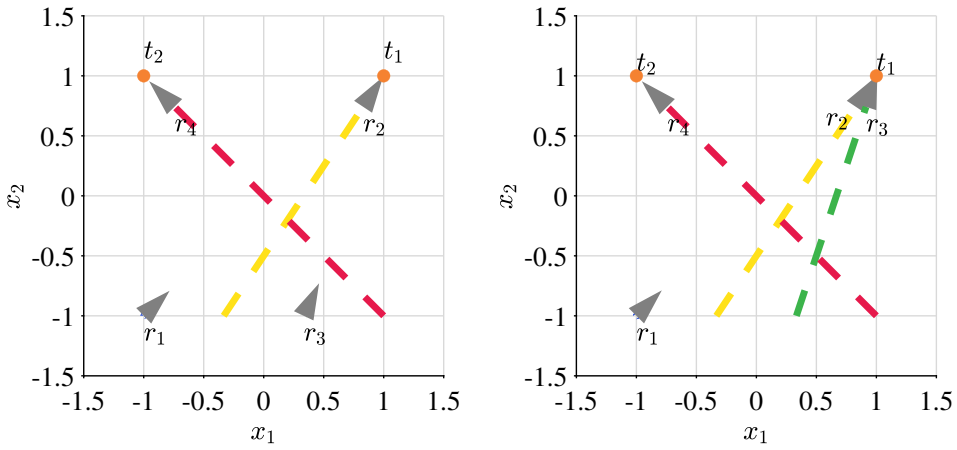


Figure 5.2: Task allocation and execution (Example 5.5). In Fig. 5.2a, 4 robots (gray triangles) have to be allocated to 2 tasks and, as a result of the execution of (5.19), robots r_2 and r_4 are assigned to task t_1 and t_2, respectively, based on their specialization introduced in Fig. 5.1. In Fig. 5.2b, the additional constraint that at least 2 robots are required to execute task t_1 is introduced ($n_{r,1,\text{min}} = 2$), and robot r_3 is picked together with r_2 to perform t_1. The resulting trajectories of the robots are depicted as dashed lines.

robots r_1, r_2 and r_3 are only specialized to perform task t_1, while robot r_4 is specialized to perform both tasks.

For the scenario depicted in Fig. 5.2a, tasks t_1 and t_2 need to be executed and there is no further constraint on the amount of capability or the number of robots required for a task. As a result of the execution of Algorithm 1, the trajectories (red and yellow) show two of the robots performing the two tasks. In particular, robot r_4 is assigned to task t_2, while robot r_2 has been allocated to task t_1 (the only one it can perform). Robots r_1 and r_3 have remained to their initial positions with no task assigned to them, as r_2 and r_4 were already satisfying the task constraints in (5.19).

In the scenario depicted in Fig. 5.2b, instead, $n_{r,1,min} = 2$, i.e. at least 2 robots are required for the execution of task t_1. Driven by the control inputs u_2 and u_3 calculated according to Algorithm 1, robots r_2 and r_3 are assigned to task t_1, while r_4 is assigned to t_2 (as it is the only robot possessing the specialization for it), as can be seen in Fig. 5.2b.

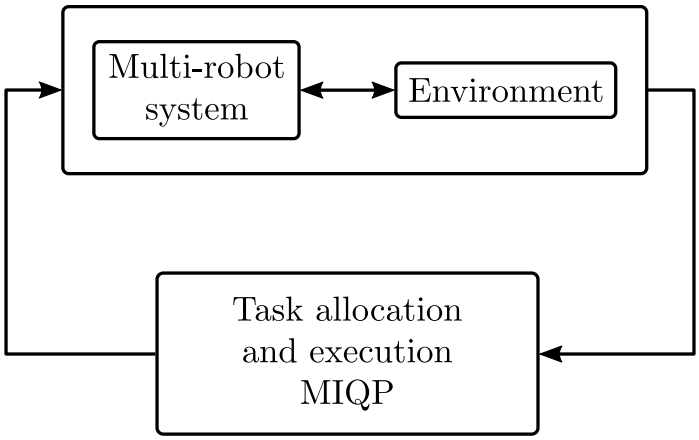


Figure 5.3: The multi-robot system interacting with the environment controlled in feedback by the task allocation and execution optimization program (5.19).

5.2.1 Resilience of the Task Allocation Algorithm

In this subsection, we introduce two distinct methods that render the task allocation and execution framework presented above resilient to environmental disturbances and robot feature failures. To achieve this, we leverage the fact that the optimization problem presented in (5.19) is solved point-wise in time, and thus can be integrated along with online updates to the specializations and capabilities of the robots to construct a feedback loop as depicted in Fig. 5.3.

We begin by introducing an update law aimed at changing the specialization values of the robots based on their measured versus expected progress at completing the tasks they are allocated to. The latter allows the framework to account for exogenous disturbances (i.e. disturbances that are not explicitly modeled or unknown). In the cases when the disturbances are endogenous (e.g. sensor malfunction), we can directly account for them by modifying the specific values in the mappings introduced in Section 5.1.1.

Exogenous Disturbances

In certain deployment scenarios, the specializations of robots towards the tasks might be unknown prior to deployment of the team, or might vary due to changes in the environ-

mental conditions. For the remainder of this chapter, we refer to all such disturbances that cannot be modeled (i.e. cannot be accounted for in F) as *exogenous* disturbances. In these cases, we would like to update the specialization parameters s_{ij} on-the-fly to account for such changes. As described in [11], this is achieved through updating the parameters s_{ij} at each time step k based on the difference between the expected and actual effectiveness of the task allocation and execution framework, where we assume that this difference manifests itself in terms of variations in the dynamical model of the robot. At discretized time intervals $k\Delta t, k \in \mathbb{N}$, let $x_{\text{act}}^{(k)}$ denote the actual ensemble state of the multi-robot system and ${}^{(i)}x_{\text{sim}}^{(k)}$ the ensemble state simulated by robot i assuming it itself obeyed its nominal dynamics with all the other robots being stationary. The simulated states can be then evaluated as follows:

$$ {}^{(i)}x_{\text{sim},j}^{(k)} = \begin{cases} x_{\text{act},i}^{(k-1)} + \Delta x_i^{(k-1)} \Delta t & \text{if } j = i \\ x_{\text{act},j}^{(k-1)} & \text{if } j \neq i, \end{cases} \tag{5.26} $$

where ${}^{(i)}x_{\text{sim},j}^{(k)}$ denotes the j-th component of ${}^{(i)}x_{\text{sim}}^{(k)}$, and $\Delta x_i^{(k-1)}$ is defined as

$$ \Delta x_i^{(k-1)} = f\left(x_{\text{act},i}^{(k-1)}\right) + g\left(x_{\text{act},i}^{(k-1)}\right) u_i^{(k-1)}, \tag{5.27} $$

$u_i^{(k-1)}$ being the input evaluated by solving (5.19) at time $(k-1)\Delta t$. Using ${}^{(i)}x_{\text{sim}}^{(k)}$, robot i can measure its contribution towards the difference between the simulated and the actual progress in the completion of task m at time step k as follows:

$$ \Delta h_{im}^{(k)} = \min\left\{0, h_{im}\left(x_{\text{act}}^{(k)}\right) - h_{im}\left({}^{(i)}x_{\text{sim}}^{(k)}\right)\right\}, \tag{5.28} $$

where $h_{im}\left({}^{(i)}x_{\text{sim}}^{(k)}\right)$ and $h_{im}\left(x_{\text{act}}^{(k)}\right)$ are the simulated and actual values of the CBF corresponding to robot i and task m at time step k, respectively. Note that the $\min$ operator in (5.28) is used to prevent $\Delta h_{im}^{(k)}$ from being positive. This situation may occur due to the coordinated nature of multi-robot tasks, where robot i need not know the actions of its

neighbors, which could result in an unpredictable positive variations of h_{im}.

We assume that the CBF corresponding to each task, h_m, is decomposable into the respective contributions of each robot i, h_{im}. This assumption holds for a large number of coordinated control tasks such as multi-robot coverage control and formation control [134], and allows each robot to assess its own effectiveness at executing a task by measuring $\Delta h_{im}^{(k)}$. In fact, if $\Delta h_{im}^{(k)} < 0$, robot i's actual effectiveness at accomplishing task m is lower than anticipated. Consequently, one can model the evolution of the specialization of robot i at task m according to the following update law:

$$s_{im}^{(k+1)} = s_{im}^{(k)} + \beta \alpha_{im}^{(k)} \Delta h_{im}^{(k)}, \tag{5.29}$$

where $\beta \in \mathbb{R}_{>0}$ is a constant controlling the update rate. Note that the update only occurs for tasks to which the robots are assigned since $\alpha_{im}^{(k)} = 1$ if and only if robot i is assigned to task m at time step k. This update law renders the framework resilient to unknown environmental disturbances by allowing the framework to account for the dynamical variations in the environmental conditions through the updates of the specialization matrix according to the performance of the robots. Although it is not presented in this chapter, in [11] conditions under which the robot specialization lost because of the update law in (5.29) can be recovered over time are shown. Algorithm 2 extends Algorithm 1 developed in the previous section to account for exogenous disturbances.

The following example showcases the use of Algorithm 2 in a simplified scenario with 2 robots, 2 tasks, and an unmodeled, exogenous environmental disturbance.

Example 5.6. *Consider the example depicted in Fig. 5.4. The 2 robots, r_1 and r_2 (shown as gray triangles in Fig. 5.4b, and modeled as 2-dimensional single integrators as in Example 5.5) are asked to execute 2 tasks t_1 and t_2. Their features and capabilities are depicted in Fig. 5.4a: both robots are capable of performing both tasks. Nevertheless, robot r_1 cannot traverse the region of the state space shaded in cyan (unmodeled disturbance), making*

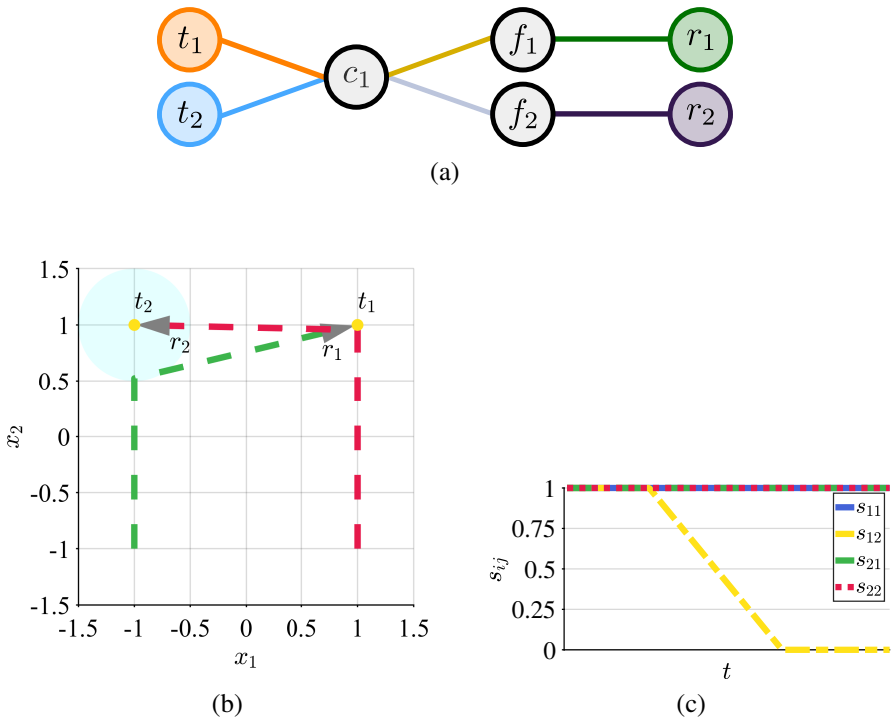


Figure 5.4: Resilience of the task allocation algorithm to exogenous disturbances (Example 5.6). Robots r_1 and r_2 (gray triangles) have to execute tasks t_1 and t_2. They both possess the capabilities to perform both tasks (as pictorially shown in Fig. 5.4a), but r_1 is not capable of traversing the circular cyan-shaded region (representing the exogenous disturbance), rendering practically impossible for it to execute task t_2. Based on (5.19), r_1 is initially assigned to t_2 and r_2 to t_1. As r_1 reaches the cyan zone, it is not able to proceed forward. According to Algorithm 2, per (5.29), s_{12}—the specialization of r_1 to execute t_2, depicted in 5.4b as a function of time t—starts decreasing until it reaches 0. At this point, the allocation evaluated by (5.19) automatically changes and robots r_1 and r_2 are assigned to tasks t_1 and t_2, respectively, fulfilling, this way, the requirement that both tasks need to be executed. The trajectories of the robots resulting by the allocation algorithm are depicted as dashed lines in Fig. 5.4b.

Algorithm 2 Task allocation and execution resilient to exogenous disturbance

Require:

Tasks $h_m,\ m \in \{1, \dots, n_t\}$

Mappings F, T

Parameters $n_{r,m,\text{min}}$, $n_{r,m,\text{max}}$, δ_{max}, C, l

Evaluate $S_i,\ \forall i \in \{1, \dots, n_r\}$ ▷ (5.13)

while true **do**

  Get robot state $x_i, \forall i \in \{1, \dots, n_r\}$

  Compute robot input $u_i, \forall i \in \{1, \dots, n_r\}$ ▷ (5.19)

  Send input $u_i,\ \forall i \in \{1, \dots, n_r\}$ to robots to execute

  for all $i \in \{1, \dots, n_r\}$ **do**

    Evaluate simulated robot state ${}^{(i)}x_{\text{sim}}^{(k)}$ ▷ (5.26)

    for all $m \in \{1, \dots, n_t\}$ **do**

      Evaluate $\Delta h_{im}^{(k)}$ ▷ (5.28)

      Evaluate $s_{im}^{(k+1)}$ ▷ (5.29)

    end for

  end for

end while

the execution of task t_2 impossible for it. By implementing the control obtained by solving (5.19)*, robot r_1 is initially assigned to task t_2, while r_2 is assigned to t_1, as confirmed by the initial vertical segments of the dashed green and red trajectories of the robots. As r_1 reaches the circular cyan region, it is not able to advance anymore and the execution of Algorithm 2 makes its specialization towards task t_2—represented by s_{12}—decrease according to* (5.29)*, as depicted in Fig. 5.4b. When $s_{12} = 0$, the allocation algorithm* (5.19) *swaps the allocation of tasks as robot r_1 is not able to execute task t_2 to any extent anymore. The final allocation satisfies the requirements that both tasks are executed.*

Endogenous Disturbances

We now shift our focus to cases where the disturbances to the model are known to the robots—a condition happening in case of, e.g., sensor malfunction. We refer to this class of disturbances as *endogenous* disturbances for which we account for by directly altering the mappings introduced in Section 5.1.1. Specifically, by leveraging the feature representation, we directly alter the intermediate mappings (i.e. Robot-to-Feature and Feature-to-

Capability mappings) on the fly to reflect such changes. The latter is achieved through modifying the corresponding values in the mappings defined in Section 5.1.1 and re-computing the Robot-to-Capability matrix F and the specialization matrices S_i. Note that F is incorporated in the task allocation framework through the constraint (5.19f), which ensures that the task allocation among the robots reflects the change in F. For example, in case of a feature failure, the Robot-to-Feature mapping matrix A is altered to account for the failure. Moreover, in case of known environmental disturbances, the feature bundle weights W_k is altered for each capability. Following the example from Fig. 5.1, if feature f_4 of robot r_2 malfunctions, we reflect that by altering the original A matrix to

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.30}$$

which is equivalent to removing the edge from robot r_2 to feature f_4 in the hypergraph from Fig. 5.1.

The approach described in this section to cope with endogenous disturbances is summarized in Algorithm 3. To conclude the section, we present a final example to showcase the behavior resulting from the application of Algorithm 3.

Example 5.7. *In this last examples, 2 robots, r_1 and r_2 are considered, which possess the features depicted in Fig. 5.5a to execute 2 tasks, t_1 and t_2, thanks to the capabilities c_1 and c_2. The mappings from features to capabilities to tasks may represent the following scenario. Two robots are endowed with wheels (feature f_1) for mobility (capability c_1), as well as a camera (feature f_2) and a communication module (feature f_3) serving the ability of live streaming (capability c_2). Task t_1 consists in visiting a location of the state space of*

Algorithm 3 Task allocation and execution resilient to endogenous disturbance

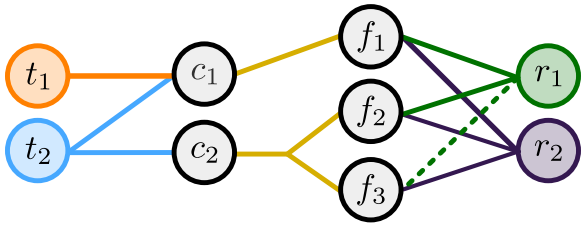
Require:

Tasks $h_m,\ m \in \{1, \dots, n_t\}$

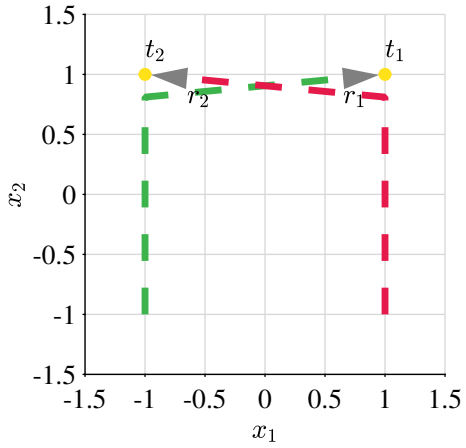
Mappings F, T

Parameters $n_{r,m,\min}$, $n_{r,m,\max}$, $\delta_{\max}$, C, l, β

1: Evaluate $S_i,\ \forall i \in \{1, \dots, n_r\}$ ▷ (5.13)
2: **while** true **do**
3: Get robot state $x_i,\ \forall i \in \{1, \dots, n_r\}$
4: Calculate robot input $u_i,\ \forall i \in \{1, \dots, n_r\}$ ▷ (5.19)
5: Send input $u_i,\ \forall i \in \{1, \dots, n_r\}$ to robots to execute
6: Update matrix A
7: Re-evaluate matrices F and S ▷ (5.7), (5.13)
8: **end while**



(a)



(b)

Figure 5.5: Resilience of the task allocation algorithm to endogenous disturbances (Example 5.7). The 2 robots r_1 and r_2 (gray triangles) are asked to perform tasks t_1 and t_2. Initially, both robots are able to perform both tasks based on their possessed features shown in Fig. 5.5a. Solving (5.19) initially assigns robot r_1 to task t_2 and r_2 to t_1. At a certain time instant, A_{31} transitions from 1 to 0, corresponding to the condition that robot r_1 loses feature f_3 (the dashed edge in Fig. 5.5a is lost). At this point, the constraint (5.19e) forces the task allocation to swap so that r_2, the only robots capable of providing capability c_2 for executing t_2, is assigned to it. This way, the requirement that both tasks are executed by at least one robot are satisfied. The trajectories of the robots are depicted as dashed lines in Fig. 5.5b.

the robots, while task t_2 consists in visiting a location and live streaming a video feed. The endogenous disturbance consists in robot r_1 losing the communication functionality at a certain time instant, compromising its ability of performing task t_2, as it cannot live stream video feed anymore. The dashed edge in Fig. 5.5a is lost, and the robot-to-feature mapping matrix A is modified by setting $A_{31} = 0$.

Fig. 5.5b depicts the trajectories of the robots under the initial allocation of robot r_1 to task t_2 and r_2 to t_1. As $A_{31} = 0$, the matrix F changes according to (5.7)*. Consequently, the constraint* (5.19e) *in the optimization problem* (5.19) *prevents robot r_1 from being allocated to task t_2. Thus, the task allocation swaps in order to be able to perform both tasks, as required.*

5.3 Analysis and Implementation of the Task Prioritization and Execution Algorithm

The definition of the optimization problem as in (5.19) gives rise to two main questions: (i) whether, despite its pointwise-in-time nature, the allocation algorithm generates a stable allocation of tasks among robots, and (ii) whether it can be solved in real time to allocate tasks to robots and synthesize control inputs which allow robots to execute them. As far as (i) is concerned, a stable allocation is the amenable condition under which, with time-invariant parameters of the problem and no exogenous or endogenous disturbances, each robot does not continuously switch between the tasks it executes, but rather is able to accomplish one of them. Regarding (ii), (5.19) is a mixed-integer quadratic program and, as such, solving it in real time might be too computationally intensive.

To address these two issues, in Section 5.3.1 we present results on the convergence of the task prioritization and execution algorithm introduced in the previous section. These results guarantee that the allocation of tasks to a heterogeneous multi-robot system obtained by executing Algorithm 1 will converge, allowing the robots to complete the tasks that have been assigned to them. Moreover, in Section 5.3.2, we present a mixed central-

ized/decentralized implementation of the developed task allocation algorithm which enables its application in online settings.

5.3.1 Analysis of Convergence of the Task Prioritization and Execution Algorithm

In cases where the tasks that the robots are asked to execute are neither coordinated nor time-varying (namely the CBF associated with them does not explicitly depend on the time variable), the following Proposition shows that the application of the task allocation algorithm (5.19) leads to a convergent behavior of the robots, whose states converge to a stable equilibrium point.

Proposition 5.8. *Consider n_r robots modeled by the driftless dynamical system*

$$\dot{x}_i = g(x_i)u_i, \tag{5.31}$$

executing the control input $u_i^{(k)}$ at time k, where $u^{(k)}$ is obtained by solving the task allocation algorithm (5.19) *at time k in order to perform n_t tasks. Assume that the tasks are characterized by the functions $h_1, \ldots, h_{n_t}$ which do not have an explicit dependence on time. Assume further that the tasks are not coordinated, i.e.*

$$h_m(x) = \sum_{i=1}^{n_r} h_{m,i}(x_i) \quad \forall i \in \{1, \ldots, n_t\}, \tag{5.32}$$

where

$$\left\| \frac{\partial h_{m,i}(x_i)}{\partial x_i}^T (x_i) \right\| = \lambda \left(h_{m,i}(x_i) \right), \tag{5.33}$$

λ being a class $\mathcal{K}$ function, and there exist a unique $x_{m,i}^\star$—corresponding to the state at which the task characterized by the function $h_{m,i}$ is completed—such that $h_{m,i}(x_{m,i}^\star) = 0$. If all robots are capable of performing all tasks to a certain extent, i.e. the matrices S_i, $i \in \{1, \ldots, n_r\}$ are positive definite, then the sequences $\{u^{(k)}\}_{k\in\mathbb{N}}$, $\{\delta^{(k)}\}_{k\in\mathbb{N}}$, and $\{\alpha^{(k)}\}_{k\in\mathbb{N}}$, solutions of (5.19)*, converge as $k \to \infty$. In particular, $u^{(k)} \to 0$, $\delta^{(k)} \to 0$.*

The application of the previous results is however restricted to a specific, but nevertheless quite rich, class of tasks. However, in situations where the assumptions of Proposition 5.8 are not satisfied, the following proposition provides us with sufficient conditions to ensure the convergence of the flow of the dynamical system comprised of the multi-robot system, characterized by its nonlinear dynamics, in feedback with the optimization problem embodying the task allocation algorithm (depicted in Fig. 5.3). The similarity between the system in Fig. 5.3 and the Lure's problem [147] suggests us to resort to techniques that have been widely adopted in the stability analysis of such systems, with the aim of studying the convergence of the task allocation algorithm we propose in this chapter. Indeed, in the following proposition a quadratic Lyapunov function is proposed, and conditions to establish the convergence of the task allocation algorithm are given in the form of a linear matrix inequality (LMI) using the S-procedure [148, 149].

Proposition 5.9. *If, for all integers k, there exist positive scalars τ_1, τ_2, τ_3, τ_4 such that*

$$B_0^{(k)} \leq \tau_1 B_1^{(k)} + \tau_2 B_2^{(k)} + \tau_3 B_3^{(k)} + \tau_4 B_4^{(k)}, \tag{5.34}$$

where $B_0^{(k)}$, $B_1^{(k)}$, $B_2^{(k)}$, $B_3^{(k)}$, $B_4^{(k)}$, *and* $c \in \mathbb{R}_{>0}$ *are given by*

$$
\begin{aligned}
B_0^{(k)} &= \begin{bmatrix} cI & \frac{\mathrm{d}\gamma}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}x}g(x^{(k)}) & 0 & 0 & \frac{\mathrm{d}\gamma}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}x}f(x^{(k)}) \\ \frac{\mathrm{d}\gamma}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}x}g(x^{(k)})^T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\mathrm{d}\gamma}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}x}f(x^{(k)})^T & 0 & 0 & 0 & 0 \end{bmatrix}, \\
B_1^{(k)} &= \begin{bmatrix} 0 & 0 & -\frac{1}{2}I & 0 & 0 \\ 0 & 0 & -\frac{1}{2}L_g h(x^{(k)})^T & 0 & 0 \\ -\frac{1}{2}I & -\frac{1}{2}L_g h(x^{(k)}) & -I & 0 & -\frac{1}{2}L_f h(x^{(k)}) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2}L_f h(x^{(k)})^T & 0 & 0 & 0 \end{bmatrix}, \\
B_2^{(k)} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}\bar{\Theta}\bar{\Phi} & 0 \\ 0 & 0 & \frac{1}{2}\bar{\Phi}^T\bar{\Theta}^T & \bar{\Phi}^T\bar{\Phi} & \frac{1}{2}\bar{\Phi}\bar{\Psi} \\ 0 & 0 & 0 & \frac{1}{2}\bar{\Psi}^T\bar{\Phi}^T & 0 \end{bmatrix}, \\
B_3^{(k)} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_\alpha^T A_\alpha & \frac{1}{2}A_\alpha^T b_\alpha \\ 0 & 0 & 0 & \frac{1}{2}b_\alpha^T A_\alpha & 0 \end{bmatrix}, \quad B_4^{(k)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_\delta^T A_\delta & 0 & \frac{1}{2}A_\delta^T b_\delta \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}b_\delta^T A_\delta & 0 & 0 \end{bmatrix},
\end{aligned}
\tag{5.35}
$$

in which 0*'s represent zero matrices of appropriate sizes, then the sequences* $\{u^{(k)}\}_{k\in\mathbb{N}}$, $\{\delta^{(k)}\}_{k\in\mathbb{N}}$, *and* $\{\alpha^{(k)}\}_{k\in\mathbb{N}}$, *solutions of* (5.19) *at time step* k, *converge as* $k \to \infty$.

Despite the flexibility determined by the variety of scenarios encompassed by the optimization-based task allocation formulation presented in this section, its mixed-integer nature does

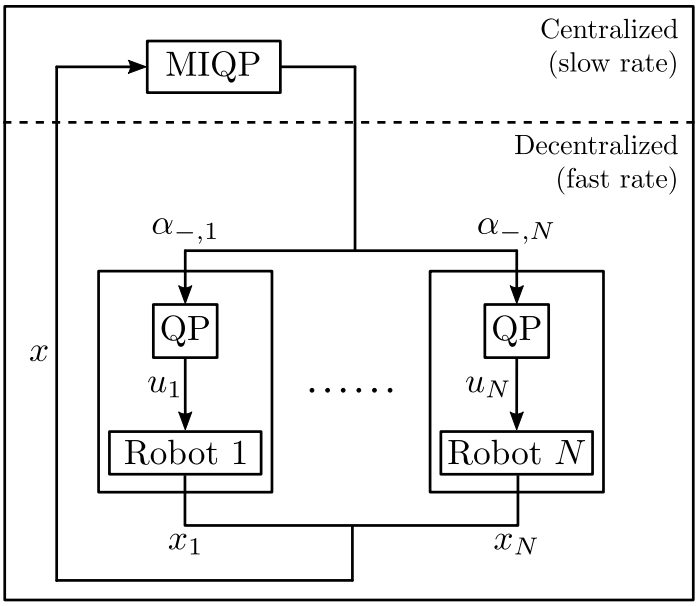


Figure 5.6: A mixed centralized/decentralized architecture to implement the task allocation and execution algorithm. Unlike the MIQP centralized formulation in (5.19), the allocation is solved separately from the execution. The former is evaluated in a centralized fashion based on the states collected from all the robots, and it typically happens at a slower rate due to the computational complexity of mixed-integer programs. The latter is solved by each robot in a decentralized way once the allocation (in terms of $\alpha_{-,i}$) is received by the robots from the central computational unit.

not allow, in most cases, to scale its applicability to a large number of robots [150]. Therefore, it is not always possible to solve the proposed task allocation optimization program (5.19) in an online fashion under real-time constraints. Thus, in the following section, we propose a mixed centralized/decentralized execution strategy which allows the computation of the task prioritization as well as the control inputs required by the robots to execute the tasks to take place in online settings.

5.3.2 Mixed Centralized/Decentralized Implementation of the Task Prioritization and Execution Algorithm

In order to allow the applicability of the proposed task prioritization and execution algorithm to scenarios where a large number of robots have to execute a large number of tasks, in the following we propose an alternative mixed centralized/decentralized formulation.

We then analyze the performance in terms of task allocation and execution compared to the MIQP developed in the previous section.

To this end, the optimization problem (5.19) is solved by a central computational unit which communicates the evaluated matrix of task priorities α to the robots. Then, each robot can solve the following convex quadratic program (QP) in order to compute the control input it requires to execute the task prioritized according to its prioritization vector $\alpha_{-,i}$:

Task execution optimization problem (QP) (5.36)

$$\underset{u_i,\delta_i}{\text{minimize}} \ \|u_i\|^2 + l\|\delta_i\|^2_{S_i} \tag{5.36a}$$

$$\text{subject to } L_f h_m(x) + L_g h_m(x) u_i \geq -\gamma(h_m(x)) - \delta_{im} \tag{5.36b}$$

$$\Theta\delta_i + \Phi\alpha_{-,i} \leq \Psi \tag{5.36c}$$

$$\|\delta_i\|_\infty \leq \delta_{\max} \tag{5.36d}$$

$$\forall m \in \{1 \dots n_t\}. \tag{5.36e}$$

Figure 5.6 summarizes the described mixed centralized/decentralized architecture.

Notice that, if solving the centralized MIQP cannot be done at each time step, by following the mixed centralized/decentralized approach, each robot solves for its control input u_i using an outdated value of its prioritization vector $\alpha_{-,i}$, which is calculated by the central unit using old values of the state x_i of the robots. Depending on the time that the central computational unit takes to solve the MIQP, the difference between the input u_i solution of (5.36) and the one that would have been obtained by solving (5.19) might be different. In the following, we quantify the error that is introduced in the control input u_i by adopting the mixed centralized/decentralized approach, rather than solving the centralized MIQP at each time step.

For notational convenience, we introduce the following mappings. We denote by

$$\Gamma_{\text{MIQP}}\colon \mathbb{R}^{n_x n_r} \to \{0,1\}^{n_t \times n_r}\colon x \mapsto \alpha \tag{5.37}$$

the natural projection of the solution map of (5.19), and by

$$\Gamma_{\text{QP}}\colon \mathbb{R}^{n_x n_r} \times \{0,1\}^{n_t \times n_r} \to \mathbb{R}^{n_u n_r}\colon (x,\alpha) \mapsto u, \tag{5.38}$$

the natural projection of the solution map of (5.36) for all the robots. Moreover, we let $\Gamma(\cdot,\cdot) = \Gamma_{\text{QP}}(\cdot, \Gamma_{\text{MIQP}}(\cdot))$, and denote by $\bar{\Gamma}_{\text{MIQP}}$ the solution map of the QP relaxation of (5.19) projected onto the subspace of allocation vectors α—where $\alpha \in [0,1]^{n_t \times n_r} \subset \mathbb{R}^{n_t \times n_r}$.

Assume that, at time $k\Delta t$, the central unit receives $x^{(k)}$ from the n_r robots, and solves the MIQP (5.19) obtaining $\alpha^{(k)} = \Gamma_{\text{MIQP}}\left(x^{(k)}\right)$. This computation is assumed to take n steps [2], or $n\Delta t$ seconds. At time $(k+n)\Delta t$, the central unit transmits the computed allocation values $\alpha^{(k)}$ to the robots, each of which solves (5.36), and the input to the robots can be expressed as $u^{(k+n)} = \Gamma_{\text{QP}}\left(x^{(k+n)}, \alpha^{(k)}\right)$. This is assumed to take 1 step, or Δt seconds.

We are interested in quantifying the difference between the robot control inputs $u^{(k+n)}$ evaluated by the robots with the old value $\alpha^{(k)}$ and the control input $\hat{u}^{(k+n)}$ that would be evaluated with the current value $\alpha^{(k+n)}$. This difference is given by

$$\begin{aligned}\left\|u^{(k+n)} - \hat{u}^{(k+n)}\right\|_\infty &= \left\|\Gamma\left(x^{(k+n)}, x^{(k)}\right) - \Gamma\left(x^{(k+n)}, x^{(k+n)}\right)\right\|_\infty \\ &= \left\|\Gamma_{\text{QP}}\left(x^{(k+n)}, \Gamma_{\text{MIQP}}\left(x^{(k)}\right)\right) - \Gamma_{\text{QP}}\left(x^{(k+n)}, \Gamma_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_\infty,\end{aligned} \tag{5.39}$$

[2] If stopping criteria are not met, the algorithm times out after $n\Delta t$ seconds.

and the different contributions are explicitly broken down as follows:

$$
\begin{aligned}
&\left\|u^{(k+n)}-\hat{u}^{(k+n)}\right\|_{\infty} \\
=&\left\|\Gamma_{\text{QP}}\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k)}\right)\right)-\Gamma_{\text{QP}}\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_{\infty} \\
\leq&\left\|\Gamma_{\text{QP}}\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k)}\right)\right)-\Gamma_{\text{QP}}\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k)}\right)\right)\right\|_{\infty} \\
&\quad+\left\|\Gamma_{\text{QP}}\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k)}\right)\right)-\Gamma_{\text{QP}}\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_{\infty} \\
&\quad+\left\|\Gamma_{\text{QP}}\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k+n)}\right)\right)-\Gamma_{\text{QP}}\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_{\infty} \\
\leq&L_{\text{QP}}\Big(\left\|\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k)}\right)\right)-\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k)}\right)\right)\right\|_{\infty} \\
&\quad+\left\|\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k)}\right)\right)-\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_{\infty} \\
&\quad+\left\|\left(x^{(k+n)},\Gamma_{\text{MIQP}}\left(x^{(k+n)}\right)\right)-\left(x^{(k+n)},\bar{\Gamma}_{\text{MIQP}}\left(x^{(k+n)}\right)\right)\right\|_{\infty}\Big) \\
\leq&L_{\text{QP}}\Big(n_t^2n_r^3m\Delta\left(\mathscr{A}\left(x^{(k)}\right)\middle|\mathscr{B}\left(x^{(k)}\right)\right) \\
&\quad+L_{\text{MIQP}}\left\|\left(x^{(k+n)},x^{(k)}\right)-\left(x^{(k+n)},x^{(k+n)}\right)\right\|_{\infty} \\
&\quad+n_t^2n_r^3m\Delta\left(\mathscr{A}\left(x^{(k+n)}\right)\middle|\mathscr{B}\left(x^{(k+n)}\right)\right)\Big) \\
\leq&L_{\text{QP}}\Big(n_t^2n_r^3m\Delta\left(\mathscr{A}\left(x^{(k)}\right)\middle|\mathscr{B}\left(x^{(k)}\right)\right) \\
&\quad+L_{\text{MIQP}}\left(\left\|x^{(k)}-x^{(k+1)}\right\|_{\infty}+\left\|x^{(k+1)}-x^{(k+2)}\right\|_{\infty}+\ldots+\left\|x^{(k+n-1)}-x^{(k+n)}\right\|_{\infty}\right) \\
&\quad+n_t^2n_r^3m\Delta\left(\mathscr{A}\left(x^{(k+n)}\right)\middle|\mathscr{B}\left(x^{(k+n)}\right)\right)\Big) \\
\leq&\underbrace{L_{\text{QP}}n_t^2n_r^3m\left(\Delta\left(\mathscr{A}\left(x^{(k)}\right)\middle|\mathscr{B}\left(x^{(k)}\right)\right)+\Delta\left(\mathscr{A}\left(x^{(k+n)}\right)\middle|\mathscr{B}\left(x^{(k+n)}\right)\right)\right)}_{\text{Effect of mixed-integer programming}} \\
&\quad+\underbrace{L_{\text{QP}}L_{\text{MIQP}}L_{\dot{x}}n\Delta t}_{\text{Effect of computation time}},
\end{aligned}
\tag{5.40}
$$

using the sensitivity results in [151]. The notation $\Delta(A)$ in (5.40) denotes the maximum of the absolute values of the determinants of the square submatrices of the matrix A. Moreover, $\mathscr{A}$ and $\mathscr{B}$ denote the matrix and the vector such that the inequality constraints in

(5.19) can be written as

$$\mathscr{A} \begin{bmatrix} u \\ \delta \\ \alpha \end{bmatrix} \leq \mathscr{B}, \tag{5.41}$$

and, provided that the conditions of Theorem 5.2 in [152] hold, L_{MIQP}, L_{QP}, and $L_{\dot{x}}$ are the Lipschitz constants of the mappings Γ_{MIQP}, Γ_{QP}, and the robot dynamics (5.16), respectively.

As expected, the bound on $\left\|u^{(k+n)} - \hat{u}^{(k+n)}\right\|$ in (5.40) is a monotonically increasing function of the number of optimization variables, of the values L_{QP} and L_{MIQP}—which, in turn, depend on the parameters of the optimization problem [152]—of $L_{\dot{x}}$, and of $n\Delta t$, i.e. the time required by the central computational unit to solve the MIQP. In particular, the bound in (5.40) is comprised of two terms: the first one depends on the mixed-integer nature of the allocation algorithm (5.19), while the second one is due to the computation time that the central unit takes in order to solve the allocation optimization. The effect of the mixed-integer programming is the most critical one, as it is proportional to $n_t^2 n_r^3$, and vanishes only when the solution of the MIQP (5.19) is equal to that of its QP relaxation. The term depending on the computational time, instead, vanishes if the MIQP can be solved at each time step.

Remark 5.10 (Communication delays)**.** *Notice that the time to communicate the allocation solution to all the robots, if not negligible, can be added to the quantity $n\Delta t$ to account for the effects of communication delays in the execution of the allocated tasks.*

We conclude this section by summarizing the mixed centralized/decentralized implementation of the proposed task allocation optimization problem in Algorithm 4, which is combined with Algorithms 2 and 3 to obtain an *efficient* implementation of the optimal allocation and execution algorithm *resilient to endogenous and exogenous disturbances*. This combination will be showcased in the next section, where the implementation of the developed allocation and execution algorithm on a real multi-robot platform is presented.

Algorithm 4 Mixed centralized/decentralized implementation of task allocation and execution

Require:

Tasks $h_m,\ m \in \{1, \ldots, n_t\}$

Mappings F, T

Parameters $n_{r,m,\text{min}}, n_{r,m,\text{max}}, \delta_{\text{max}}, C, l$

Evaluate $S_i,\ \forall i \in \{1, \ldots, n_r\}$ ▷ (5.13)

procedure CENTRAL COMPUTATIONAL UNIT

  while true **do**

    Get robots' state $x_i,\ \forall i \in \{1, \ldots, n_r\}$

    Calculate allocation α ▷ (5.19)

    Send allocation $\alpha_{-,i},\ \forall i \in \{1, \ldots, n_r\}$ to robots

    Update matrices S and F if required ▷ Algs. 2, 3

  end while

end procedure

procedure ROBOT i

  while true **do**

    Receive allocation $\alpha_{-,i}$ if ready

    Calculate input u_i and execute ▷ (5.36)

  end while

end procedure

5.4 Experimental Results

The resilient task prioritization and execution framework described in the previous sections has been implemented on a team of mobile robots in the Robotarium [130], a remotely accessible swarm robotics testbed. The scenario of the experiment is depicted in Fig. 5.7. A team of 5 mobile robots, each endowed with a simulated camera system, are deployed in a 3.6×2.4 m rectangular domain and have to perform 2 tasks: task t_1 consists of 1 robot moving along a desired trajectory navigating the environment from a starting point (red circle in Fig. 5.7) to a goal point (red cross in Fig. 5.7); to perform task t_2, 3 robots need to escort the robot executing task t_1 by arranging themselves into a ring around it while simultaneously monitoring a point of interest with their cameras (red star in Fig. 5.7). The physical robots are differential drive robots. In the experiment, we model their motion as

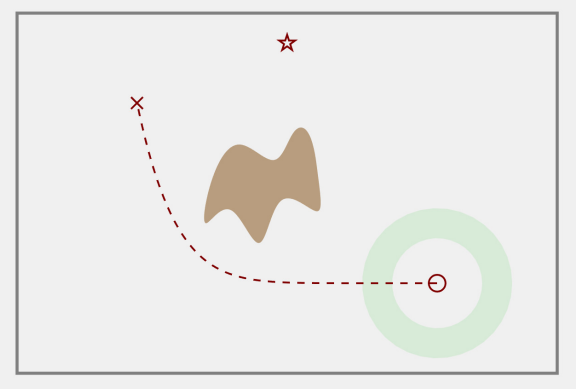


Figure 5.7: Experimental scenario. The robots need to perform 2 tasks: 1 robot has to navigate in the environment to reach a goal point (red cross) following the dashed trajectory, while 3 robots have to escort it by arranging themselves around it (on the green ring) while simultaneously monitoring a point of interest (red star). The brown blob in the middle of the rectangular environment represents a low-friction zone where the motion of ground robots is impeded.

well as that of their cameras using the following single integrator dynamics:

$$\begin{cases} \dot{x}_{i,1} = u_1 \\ \dot{x}_{i,2} = u_2 \\ \dot{x}_{i,3} = u_3, \end{cases} \tag{5.42}$$

where $p_i = [x_{i,1}, x_{i,2}]^T \in \mathbb{R}^2$ represents the position of robot i, and $x_{i,3} \in [0, 2\pi]$ is the orientation of its camera. $u_1, u_2, u_3 \in \mathbb{R}$ are the velocity inputs to the robot and to the camera.

Task t_1 is realized by tracking a predefined trajectory, while task t_2 is achieved by implementing a weighted coverage control [32, 8] in order to arrange the robots on the green ring in Fig. 5.7. The two tasks are encoded by the following two CBFs, respectively:

$$h_{1,i}(x,t) = -\|p_i - \hat{p}(t)\|^2 \tag{5.43}$$

$$h_{2,i}(x,t) = -\|p_i - G_i(x)\|^2 - \|x_{i,3} - \angle(p^* - p_i)\|^2, \tag{5.44}$$

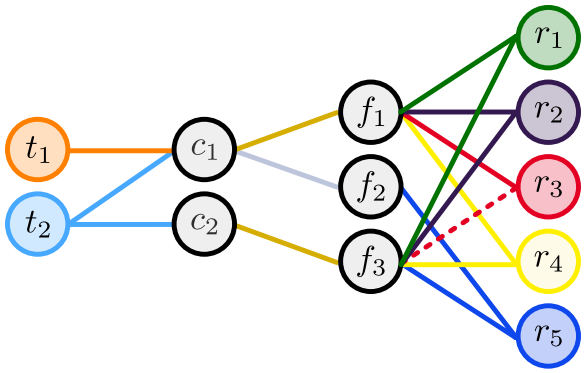


Figure 5.8: Robots, features, capabilities and tasks mappings used for the experiment on the Robotarium. The features are wheels to locomote on the ground (f_1), propellers to locomote in the air (f_2), and a camera (f_3). The resulting capabilities are locomotion (c_1) and monitoring of a point of interest (c_2). Tasks consist of navigating the environment to reach a goal point (t_1), escorting the robot navigating the environment by arranging around it and monitoring a point of interest (t_2).

where $\hat{p} : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ is the desired trajectory (dashed line in Fig. 5.7), $p^* \in \mathbb{R}^2$ is the position of the point of interest to monitor (red star in Fig. 5.7), $\angle(p^* - p_i)$ denotes the angle formed by the vector $p^* - p_i$ with the horizontal coordinate axis, and $G_i(x)$ is the centroid of the Voronoi cell corresponding to robot i. In order to achieve the desired arrangement of robots performing task t_2 around the robot performing task t_1, the centroids G_i have been evaluated as follows:

$$G_i(x) = \frac{\int_{\mathcal{V}_i} p_i \phi(p_i) \mathrm{d}p_i}{\int_{\mathcal{V}_i} \phi(p_i) \mathrm{d}p_i} \in \mathbb{R}^2 \tag{5.45}$$

where $\mathcal{V}_i$ is the Voronoi cell of robot i, $\phi(p_i)$ is the function

$$\phi(p_i) = e^{-k(\|p_i - \hat{p}(t)\|^2 - r^2)^2}, \tag{5.46}$$

with $k \in \mathbb{R}_{>0}$ and r being the radius of the green circle in Fig. 5.7 (see [32] for details on coverage control). These two parameters have been set to $k = 100$ and $r = 0.4$.

Moreover, in order to be able to perform the prescribed tasks, the robots need certain features which allow them to exhibit the capabilities required by the two tasks. The mappings employed for the experiments are depicted in the bipartite graph in Fig. 5.8. The

available features are wheels to locomote on the ground (f_1), set of propellers to fly (f_2), and a camera (f_3). The capabilities required to perform the given tasks are mobility (c_1) and monitoring (c_2). The former is supported by features f_1 or f_2, while the latter by f_3. To perform task t_1, only c_2 is required, while both capabilities are required for t_2. Finally, robots r_1 to r_4 are each endowed with wheels and a camera, while r_5 is able to fly—depicted in the Robotarium experiment by projecting down the shape of a quadcopter at its location—and possesses a camera.

Moreover, since 1 robot is required to be assigned to task t_1 and 3 robots to t_2 for all times, the following parameters have been set for the experiment:

$$T = \begin{bmatrix} 1 & 0 \\ 3 & 3 \end{bmatrix} \tag{5.47}$$

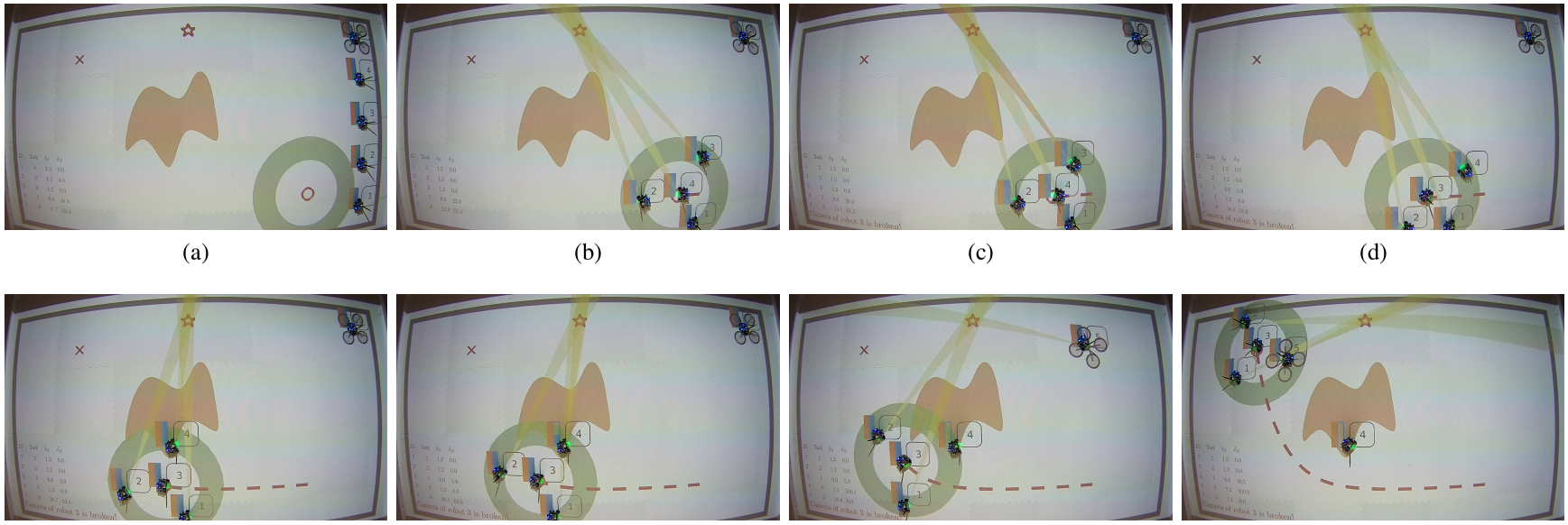
$$n_{r,1,\text{min}} = n_{r,1,\text{max}} = 1 \tag{5.48}$$

$$n_{r,2,\text{min}} = n_{r,2,\text{max}} = 3. \tag{5.49}$$

Furthermore, the remaining parameters of (5.19) have been set to: $C = 10^6$, $l = 10^{-6}$, $\gamma\colon s \mapsto 5s$, $\kappa = 10^6$, $\delta_{\text{max}} = 10^3$.

The total duration of the experiment is 80 seconds. During this time span, the resilience of the allocation algorithm to both endogenous and exogenous disturbances is tested. At time $t = 15$ s, the feature f_3 of robot r_3 is lost (endogenous disturbance), depicted by the dashed red edge on the hypergraph in Fig. 5.8. Moreover, in the middle of the environment, a region of low friction is present (brown blob in Fig. 5.7). This prevents the robots endowed with wheels from moving (exogenous disturbance).

In Fig. 5.9, snapshots recorded during the course of the experiment are shown. The robots start on the right of the rectangular environment (Fig. 5.9a). The task prioritization and execution framework results in the following allocation: r_4 is allocated to t_1 and therefore has to navigate the environment to reach the red cross, while r_1, r_2 and r_3 are assigned



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

(e) (f) (g) (h)

Figure 5.9: Snapshots recorded during the course of the experiment on the Robotarium [130]. The scenario is that depicted in Fig. 5.7, where the robots are initially arranged along the right side of the rectangular environment (Fig. 5.9a). The robot ID is projected down onto the Robotarium testbed at the top right corner of each robot. The capabilities of each robot to perform task t_1 and t_2 are depicted as vertical progress bars at the top left corner of each robot, using the same color codes as in Fig. 5.8, i.e. orange for t_1 and blue for t_2. Solving (5.19) results in the following initial allocation: r_4 is allocated to t_1—i.e. it has to navigate the environment to reach the red cross—and r_1, r_2 and r_3 are allocated to t_3—i.e. they need to escort r_4 during its mission. In Fig. 5.9b, r_1, r_2 and r_3 are arranged around r_4 and have pointed their cameras at the red star (the field of view of the cameras are depicted as a yellow triangle projected onto the Robotarium testbed). In Fig. 5.9c, the endogenous disturbance takes place: the camera of r_3 breaks—this event is represented by the the field of view of its camera becoming red. As r_3 is not able to perform the monitoring required for t_2, the constraints (5.19e) and (5.19f) result in r_3 swapping its allocation with r_4 (Fig. 5.9d). In Fig. 5.9e, the exogenous disturbance is encountered: r_4 is not able to move away from the simulated low-friction zone (brown area in the middle of the environment). Thanks to the update law (5.29), the specialization of r_4 to perform t_2 drops to 0 (in Fig. 5.9f, the blue progress bar corresponding to task t_2 next to r_4 is emptying). The task allocation recruits r_5 to perform t_2, while r_4 is not assigned to any task (Fig. 5.9g). In Fig. 5.9h, the robot team has successfully completed both tasks as desired: 1 robot has reached the goal point (red cross) while being escorted at all times by 3 more robots. In the video of the experiment available at `https://youtu.be/fdfYID7u72o`, it is also possible to see, at the bottom left of the frames, a table containing current allocated task and values of the components of δ_i for each robot over the course of the experiments.

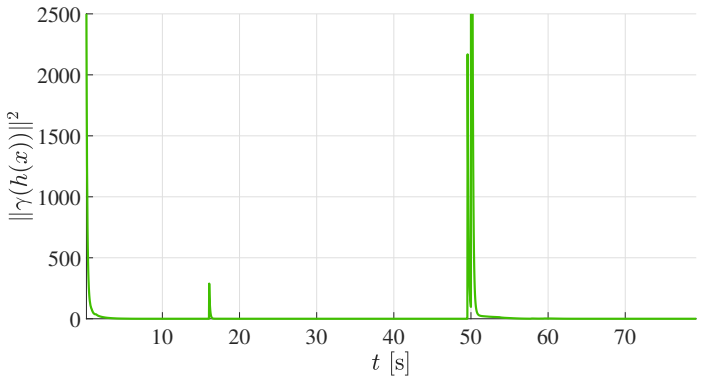


Figure 5.10: Trajectory of the value of the Lyapunov function (A.30) recorded over the course of the Robotarium experiments. At the beginning of the experiment, it decreases as the robots perform the assigned tasks. The endogenous and exogenous disturbances at $t = 15$ s and $t = 50$ s, respectively, make the value of the Lyapunov function jump to higher values, which are promptly decreased by the execution of the tasks by the robots, owing to the stability guarantees given in Proposition 5.9.

to t_3 and thus need to escort r_4 during its mission. Using coverage control, they arrange themselves around r_4 and point their cameras—whose field of view is depicted through a yellow beam projected down onto the Robotarium testbed—at the red star (Fig. 5.9b). At $t = 15$ s, the camera of r_3 breaks (Fig. 5.9c). Therefore, it cannot keep on executing t_2. The constraints (5.19e) and (5.19f) result in r_3 swapping its allocation with r_4 (Fig. 5.9d). Around $t = 50$ s, one of the robots, specifically r_4, encounters the low-friction zone, and, as a result, its motion is impeded (Fig. 5.9e). The update law (5.29) makes the specialization of r_4 towards task t_2 drop (depicted as progress bars next to r_4 in Fig. 5.9f). When the specialization of r_4 towards task t_2 reaches 0, the task allocation driven by the cost in (5.19) changes once again to adapt to the unexpected environmental conditions: r_5 is recruited to perform t_2 while r_4 is relieved of its duty (Fig. 5.9g). The last snapshot (Fig. 5.9h) shows the robot team successfully accomplishing both tasks as desired: 1 robot has reached the goal point (red cross) while being escorted at all times by 3 more robots.

Another way of appreciating the resilience of the task allocation algorithm consists in observing the trajectory of the Lyapunov function (A.30) defined in Proposition 5.9, which is depicted in Fig. 5.10. At the beginning of the experiment, the value of the Lyapunov

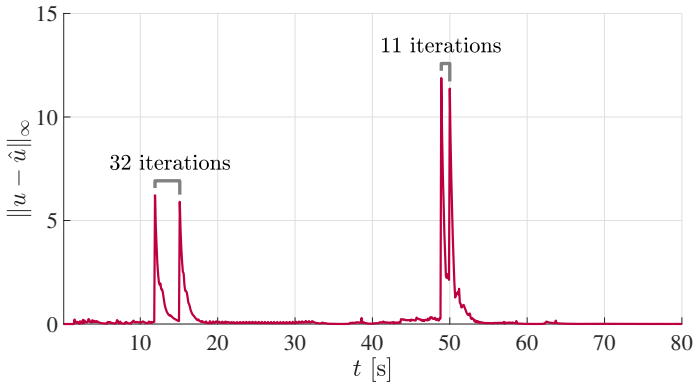


Figure 5.11: Comparison between simulations of centralized (5.19) and mixed centralized/decentralized (consisting of (5.19) and (5.36)) implementations of the optimization-based task allocation in terms of difference between robot inputs. Without the presence of endogenous or exogenous disturbances, the difference between the inputs $\hat{u}$—obtained employing a centralized approach—and u—synthesized using a mixed centralized/decentralized strategy—is close to 0. The difference peaks around the times of the endogenous and exogenous disturbances: this phenomenon is due to the delay introduced by the time required to solve the MIQP (5.19). The allocation changes 34 and 11 iterations later in the case of endogenous and exogenous disturbances, respectively. This effect due to the computation time is highlighted in (5.40).

function V decreases as the robots perform the assigned tasks. The endogenous disturbance at $t = 15$ s makes the allocation swap: by means of the stability properties highlighted in Proposition 5.9, the allocation algorithm makes the robots perform forward progress towards the accomplishment of the tasks—which results in a decrease of the Lyapunov function for $t > 15$ s. Towards the end of the experiment, a similar situation is observed where the exogenous disturbance of one of the robots incapable of moving anymore results in a change of the task allocation. Again, owing to the aforementioned stability properties, the execution of the tasks makes the Lyapunov function decrease again towards 0 after a jump due to the allocation swap.

To conclude, as observed in Section 5.3, the developed task prioritization and execution framework would not be realizable in realistic scenarios unless a mixed centralized/decentralized strategy is implemented. In the Robotarium experiment, two communicating processes have run in parallel: one responsible for solving the task allocation optimization problem (5.19), and one with the objective of synthesizing the controller

for the robots given the task allocation, using (5.36). To show the difference between the implementation of a purely centralized allocation strategy versus a mixed centralized/decentralized one, both have been simulated and the results in terms of difference between robot inputs are reported in Fig. 5.11. From the graph, it is clear that without the effect of disturbance, the difference between the inputs $\hat{u}$—obtained employing a centralized approach—and u—synthesized using a mixed centralized/decentralized strategy—would be very close to 0. The peaks around the times of the endogenous and exogenous disturbances are due to the fact that in the mixed centralized/decentralized case, there is a delay of n times steps (the effect of computation time in (5.40)) in recomputing the task allocation. In fact, solving the MIQP (5.19) takes on average 100 steps required to solve the QP (5.36), using the MATLAB CVX library [153] and the Gurobi solver [154].

5.5 Conclusions

In this chapter, we presented an optimization-based task prioritization and execution framework that achieves a resilient and energy-aware task allocation strategy for heterogeneous multi-robot systems. The approach lies its foundations on a proposed decomposition of the ability of the robots at performing tasks into features, capabilities and specialization of the robots. Moreover, the approach builds up on the notion of set-based tasks, where each task executed by the robots is characterized by a set encoded using a control barrier function. These modeling choices allow us to prioritize tasks by considering the different specialization that different robots have at performing different tasks, effectively realizing a *heterogeneous* task allocation. Furthermore, the optimization-based and pointwise-in-time nature of the task allocation algorithm contribute to foster its *resilience* properties.

We showed ways to achieve resilience with respect to endogenous disturbances (failure of a robot caused by loss of features) as well as exogenous disturbances (caused by unmodeled phenomena in the environment) which leverage the reactive nature of the formulation. Moreover, we demonstrated how the formulation allows us to specify both the number of

robots and the amount of capabilities required to perform a certain task. This way, thanks to the energy-awareness of the algorithm, robots which are not required to perform tasks are not utilized. Nevertheless, they can potentially be recruited at any point in time, achieving, this way, autonomy-on-demand in the context of task allocation. The effectiveness of the proposed approach has been showcased through a mixed centralized/decentralized implementation of the developed task allocation strategy on a team of mobile robots, undergoing both endogenous and exogenous disturbances.

CHAPTER 6

COMMUNICATION-CONSTRAINED DISTRIBUTED ESTIMATION

In the previous chapters, the energy spent by the robots to execute the assigned tasks has been supposed to be proportional to the control effort u. This assumption, as argued extensively in Chapter 3, is reasonable whenever the robots control input serve to actuate electric motors or actuators, or any other type of mechanical device, where the power is proportional to the instantaneous control effort (such as applied torque and absorbed current). Nevertheless, there are applications in which *mobility* is not the primary, or main, source of energy consumption. In scenarios where robots are deployed in the field to collect data and provide an estimate of environmental fields (e.g. temperature, pressure, air quality, or occupancy map), the power requirements for sensing and communication are not negligible compared to the one for mobility.

With the goal of deploying multi-robot systems for long-term estimation applications, in this chapter, we address the problem of *distributed estimation of spatial fields using mobile sensor networks with communication constraints* [155]. These constraints consist of a maximum communication bandwidth which limits the amount of data that can be exchanged between any two nodes of the network at each time instant. An algorithm to select the most significant data to be transferred between neighboring sensor nodes is developed starting from derived analytical error bounds. Moreover, the motion of the network nodes is controlled using a coverage control algorithm with the objective of minimizing the estimation uncertainty of each of the nodes. Finally, the results of the implementation of the developed communication constrained distributed estimation algorithm on a team of ground mobile robots in the Robotarium are reported, and the algorithm performance is evaluated both in terms of estimation accuracy of a simulated spatial field, and of the amount of data transferred. In the remainder of the chapter, the term sensor network will be used in place

of multi-robot system, as in the considered scenarios, the robots are mainly viewed as *sensing units* or *sensor nodes* which move in the environment to collect data and communicate with neighboring nodes.

Mobile wireless sensor networks (WSNs) are widely employed in applications of environmental monitoring, which typically involve spatial field estimation tasks (see, e. g., the survey in [156]). Generally, a mobile sensor network consists of a large number of sensor nodes which, in their basic configuration, are equipped with computation, mobility, sensing, and communication units. These units are responsible for performing the basic tasks for which the sensor nodes are designed: move in the environment and explore it, collect measurement data, process them and communicate them either to a central unit or to their neighboring nodes.

One of the main obstructions to achieving long-term deployment of mobile wireless sensor networks is energy management. While mobility is the main source of energy consumption, in most applications, communication is significantly more energy-consuming than computation, as recognized in [157]. Thus, in long-term distributed estimation tasks—whereby each node is supposed to build an estimate of an environment field by means of local interactions with its neighbors only—the energy employed for communication needs to be explicitly taken into account. In this chapter, we address this issue through the following two contributions:

(i) the errors introduced by the approximations adopted in a distributed estimation framework versus a centralized one are quantified;

(ii) the results of this analysis are leveraged to develop an algorithm to select the most significant data that need to be transferred between neighboring sensor nodes in a WSN.

The remainder of the chapter is organized as follows. In the next section, Gaussian process regression is briefly introduced, as it will be used in Sections 6.2 and 6.3 to develop

a communication constrained distributed Gaussian process regression (DGPR) algorithm. In Section 6.4, the results of the implementation of the proposed distributed estimation framework on a real multi-robot system are presented.

6.1 Gaussian Process Regression

In this chapter, we use Gaussian process regression as the basic framework for spatial field estimation [158]. Gaussian processes are flexible and exhibit good generalization properties thanks to the lack of any underlying model of the process to estimate. In this section, we briefly recall the Gaussian process regression and introduce a way of rendering the estimation process distributed, requiring each node of a WSN to transfer only a fixed amount of data with its neighboring nodes.

From the definition given in [158], a Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. Let $D \subseteq \mathbb{R}^d$ be the input space of a scalar-valued function $f : D \to \mathbb{R}$. Then, f is a GP if, for any index set $J \subset \mathbb{N}$ and $x = \{x_i\}_{i \in J}$, $x_i \in D \; \forall i \in J$, one has that $f(x) = \{f(x_i)\}_{i \in J}$ are Gaussian distributed. A GP is completely specified by its mean function, $\mu : D \to \mathbb{R}$, and its covariance function, also called kernel function, $k : D \times D \to \mathbb{R}$. The value $\mu(x_i)$ is the mean of $f(x_i)$, for $x_i \in D$, whereas $k(x_i, x_j)$ is the covariance between $f(x_i)$ and $f(x_j)$ for $x_i, x_j \in D$. Adopting the same notation used for multivariate Gaussian distributions, we can write $f \sim \mathcal{GP}(\mu, k)$, with which we mean that, for a given $x = [x_1, \ldots, x_n]$, $x_i \in D \; \forall i$, we have $f(x) \sim \mathcal{N}(\mu(x), k(x, x))$. In order to give rise to a valid covariance matrix $k(x, x)$, the function k needs to be symmetric and positive definite.

Spatial field estimations using GP regression are performed using GPs as a prior probability distribution over functions describing the field that is to be estimated. Once the measurements of f at the points $x = [x_1, \ldots, x_M]$, $f(x) = [f(x_1), \ldots, f(x_M)]$, have been performed, one can calculate the mean and the covariance of the posterior probability distribution of the value of the function f at a test point $x^\star \in D$, denoted by $f(x^\star) \mid f(x)$.

This calculation has to be performed by each node of a WSN and is carried out by using the marginalization properties of multivariate Gaussian distributions, resulting in the following expression (see, e. g., [158]):

$$f(x^\star) \mid f(x) \sim \mathcal{N}\Big(k(x^\star, x)^T k(x, x)^{-1} f(x), \\ k(x^\star, x^\star) - k(x^\star, x)^T k(x, x)^{-1} k(x^\star, x)\Big). \tag{6.1}$$

6.1.1 Compactly Supported Kernel Functions and Distributed Gaussian Process Regression

The naive computation of the conditional probability (6.1) at inference time requires $\mathcal{O}(M^3)$ operations, where M is the number of observations, in order to invert the covariance matrix $k(x, x)$ ([158]). For this reason, in practice, the exact implementation cannot handle problems with more than a few thousands observations. To overcome this computational limitation of GPR, a variety of solutions have been proposed such as [80, 81, 82, 83], as discussed above (see [159] for a unifying framework for sparse approximations in Gaussian regression models).

The approximation technique we consider in this chapter in the context of distributed estimation consists in sparsifying the covariance matrix $k(x, x)$ by making use of compactly supported (CS) kernel functions. This way, GPR can be performed in a distributed fashion. A distributed approach that leverages CS kernel functions to allow spatial estimation using mobile sensor networks is introduced in [85]. In this work, the authors allow the sensor nodes to transfer all the measurements they have collected to their neighbors. In the next section, we present a way of selecting and communicating the data that are most relevant to the neighbors of a sensor node to improve its estimation. Before that, we give a brief overview of CS covariance functions that are used in GPR and introduce the specific one that is used in this chapter.

In [160], the author provides sufficient conditions for positive definiteness of radial basis functions with compact support. Using the derived conditions, a series of positive

definite and CS radial functions, known as Wu's polynomials, can be produced. In the context of spatial estimation for interpolating large datasets, [161] show that tapering a covariance matrix with an appropriate CS positive definite function can significantly reduce the computational burden while still leading to asymptotically optimal estimations. The benefits introduced by CS covariance functions in terms of computational efficiency in spatial prediction and data interpolation are recognized also in [162]. A constructive way of obtaining CS kernels using functions known as mollifiers—smooth functions with compact support—is presented in [163], where the objective is once again that of significantly reducing the computational complexity inherent in GPR. Finally, the use of CS Radial Basis Fucntion (RBF) kernels for computational improvements and memory reduction in function estimation is investigated in [164].

Due to their universality—the property of approximating continuous functions on compact sets with arbitrary accuracy ([165])—in this chapter we employ CS Gaussian RBF kernels ([166]). These are obtained by mollifying the Gaussian RBF kernel

$$k : (x_1, x_2) \in D \times D \mapsto e^{-\frac{\|x_1 - x_2\|^2}{\sigma^2}} \in \mathbb{R}, \tag{6.2}$$

where σ is a parameter of the function, by multiplying it by the following CS kernel:

$$k_{\mathrm{c}} : (x_1, x_2) \in D \times D \mapsto \max\left\{0, \left(1 - \frac{\|x_1 - x_2\|}{l}\right)^{\nu}\right\} \in \mathbb{R}. \tag{6.3}$$

In (6.3), the parameters l and ν need to satisfy the conditions $l > 0$ and $\nu > (d+1)/2$, where d is the dimensionality of the vectors x_i, in order to ensure positive definiteness of k_c. The product of the kernels in (6.2) and (6.3),

$$\hat{k} : (x_1, x_2) \in D \times D \mapsto k(x_1, x_2) k_{\mathrm{c}}(x_1, x_2) \in \mathbb{R}, \tag{6.4}$$

is a CS kernel function and l is a parameter known as the effective range. The meaning

of the effective range can be understood observing that $\|x_i - x_j\| \geq l \Rightarrow \hat{k}(x_i, x_j) = 0 \; \forall x_i, x_j \in D$. In the context of WSNs, this implies that two measurements taken at the points x_i and x_j are uncorrelated, namely they do not influence each other. Therefore, measurement points outside the effective range are not required to perform inference using DGPR. Thanks to this property, Gaussian process regression performed using CS kernel functions lends itself to be employed in distributed estimation applications. At the same time, however, the lack of infinite support of the kernel function determines an estimation error when compared to a centralized approach. Quantifying and analyzing this error, with the objective of classifying data to be exchanged between neighboring sensor nodes, is the subject of next section.

6.2 Distributed Gaussian Process Regression Error Analysis

The sparsification of the covariance matrix of a GP obtained by using CS kernel functions is exploited to formulate a distributed version of GPR in [85]. The algorithm proposed by the authors relies on communication between neighboring sensors in order to exchange collected measurements. If the estimation task takes place over long time horizons, or if the mobile sensor networks employed to perform it have energy constraints, reducing the communication burden is an important factor for the successful execution of the estimation task. This is because, after mobility, communication is the most energy-consuming task in many mobile sensing applications ([157]).

With the objective of deploying mobile sensor networks over long time horizons, in this section, a way of ranking the data to be transferred is presented, which can be used to select only the most significant data that will be transferred between neighboring sensor nodes. In Section 6.2.1 we obtain bounds on the error that is introduced by approximating a kernel function using a CS version of it, whereas in Section 6.2.2 the estimation difference due to the fact that each node of the sensor network has a different subset of the measurement data is estimated.

To this end, consider a mobile sensor network with N sensors deployed in a 2D environment $X \subseteq \mathbb{R}^2$ in order to estimate an environment field $f : D \to \mathbb{R}$, with $X \subseteq D \subseteq \mathbb{R}^2$, such as temperature, light intensity, and concentration of a chemical substance. We denote by $x_i \in X$ the position of a sensor nodes and by $y_i \in \mathbb{R}$ the observation made by the sensor at position x_i. Note that the subscript i does not refer to a specific sensor node; instead, x_i and x_j denote just two different points in X where two measurements have been performed. Moreover, we denote by k the Gaussian RBF kernel and by $k(x_i, x_j)$ its value computed as in (6.2). Similarly, $\hat{k}$ indicates the CS version of k, whose value $\hat{k}(x_i, x_j)$ is computed as in (6.4).

6.2.1 Approximation Using Compactly Supported Kernel Functions

Let $\{(x_m, y_m)\}_{m\in\{1,\dots,M\}}$ be the set of measurements collected by a mobile sensor of the WSN: $\{y_m\}_{m\in\{1,\dots,M\}}$ are the measured values at locations $\{x_m\}_{m\in\{1,\dots,M\}}$. We want to analyze the effect of using a CS kernel function instead of a kernel with infinite support. Given a point $x^\star \in X$ and letting $x = [x_1, \dots, x_M]$ and $y = [y_1, \dots, y_M] = [f(x_1), \dots, f(x_M)]$, we denote by $y^\star$ and $\hat{y}^\star$ the means of the conditional probabilities $f(x^\star) \mid y$ obtained using the kernel functions k and $\hat{k}$, respectively.

We now aim at finding a relationship between the estimation difference $|y^\star - \hat{y}^\star|$ and the difference between the kernel functions. To this end, using (6.1), one can write: $|y^\star - \hat{y}^\star| = \left|k(x^\star, x)^T k(x, x)^{-1} y - \hat{k}(x^\star, x)^T \hat{k}(x, x)^{-1} y\right|$. For sake of notational compactness, we use the following conventions: $K_{xx} = k(x, x)$, $K_{x^\star x} = k(x^\star, x)$, $\hat{K}_{xx} = \hat{k}(x, x)$, $\hat{K}_{x^\star x} = \hat{k}(x^\star, x)$. Using the definitions of vector and matrix l^2-norms, and the fact that $0 < k(x_1, x_2) \leq 1 \; \forall x_1, x_2 \in D$, one can show that $|y^\star - \hat{y}^\star| \leq 2 \left\|K_{xx}^{-1}\right\| \left\|\hat{K}_{xx}^{-1}\right\| \|y\| \, M^{\frac{3}{2}} \delta$, where $\delta = \sup_{x_i, x_j \in D} \left|k(x_i, x_j) - \hat{k}(x_i, x_j)\right|$.

If the measurements $\{(x_m, y_m)\}_{m\in\{1,\dots,M\}}$ are linearly independent, the positive definite covariance matrix $k(x, x)$ is non-singular ([167]), which, in this case, is equivalent to the fact that its minimum eigenvalue $\lambda_{\min}(k(x, x))$ is strictly positive. Hence, using properties

of symmetric and positive definite matrices, one obtains:

$$|y^\star - \hat{y}^\star| \leq \frac{2}{\lambda_{\min}(K_{xx})\lambda_{\min}(\hat{K}_{xx})} \|y\| M^{\frac{3}{2}} \delta < \infty. \tag{6.5}$$

Thus $|y^\star - \hat{y}^\star| \to 0$ as $\delta \to 0$. This means that the distributed estimation performed by the mobile sensor nodes of a WSN using CS kernel functions is close to the centralized estimation obtained by using a kernel function with infinite support as long as their effective range is large. This concept is formalized in the following.

Since k and $\hat{k}$ are radial basis functions, we can define the following two functions that depend only on the quantity $r = \|x_i - x_j\| \; \forall x_i, x_j \in D$:

$$\mathcal{K}(r) = e^{-\frac{r^2}{\sigma^2}}, \quad \hat{\mathcal{K}}(r) = e^{-\frac{r^2}{\sigma^2}} \max\left\{0, \left(1 - \frac{r}{l}\right)^\nu\right\}. \tag{6.6}$$

$\mathcal{K}$ belongs to the space of continuous functions vanishing at infinity

$$C_0(\mathbb{R}) = \left\{f \in C^0(\mathbb{R}) : \lim_{x \to \pm\infty} = 0\right\}, \tag{6.7}$$

whereas $\hat{\mathcal{K}}$ belongs to the space of compactly supported continuous

$$C_c(\mathbb{R}) = \left\{f \in C^0(\mathbb{R}) : f \text{ has compact support}\right\}. \tag{6.8}$$

$C_c(\mathbb{R})$ is a dense proper subspace of $C_0(\mathbb{R})$ with respect to the uniform norm $\|f\|_u = \sup_{x \in \mathbb{R}} |f(x)|$, as shown in [168]. Therefore, $\forall f \in C_0(\mathbb{R})$ and $\forall \varepsilon > 0$, there exists a sequence of functions $\{f_n\}_{n \in \mathbb{N}} \in C_c(\mathbb{R})$ and $N > 0$ such that, $\forall n > N$, $\|f_n - f\|_u < \varepsilon$. This means that any function $f \in C_0(\mathbb{R})$ can be approximated with arbitrary accuracy using compactly supported functions $f_n \in C_c(\mathbb{R})$. The approximation of functions obtained by using series of CS functions can be exploited in the context of DGPR as explained in the following.

Taking the sequence of compactly supported functions

$$\hat{\mathcal{K}}_n(r) = \left\{ e^{-\frac{r^2}{\sigma^2}} \max\left\{0, \left(1 - \frac{r}{n}\right)^{\nu}\right\}\right\}_{n\in\mathbb{N}}, \tag{6.9}$$

we have that, as $n \to \infty$, $\delta = \|\hat{\mathcal{K}}_n - \mathcal{K}\|_u \to 0$. Therefore, in order to minimize the estimation error $|y^\star - \hat{y}^\star|$ due to the use of CS kernel functions, the objective of the sensor nodes of a WSN is that of maximizing the effective range l. Note that for a sensor node, with M collected measurements denoted by $\{(x_m, y_m)\}_{m\in\{1,\dots,M\}}$, the effective range l is bounded by $\max_{r,s\in\{1,\dots,M\}} \|x_r - x_s\|$. In conclusion, for a spatial estimation task, this result means that it is desirable to have measurement locations that are as far apart as possible in space. This way, the effective range, which is a parameter of the CS kernel used for the GPR, can be increased and, consequently, the estimation error will be reduced. The following proposition summarizes what has been derived so far.

Proposition 6.1. *Let $\mathcal{GP}_1$ be a GPR model that employs the infinitely supported Gaussian RBF kernel function in* (6.2)*. Define $\mathcal{GP}_2$ as the GPR model that uses as kernel function the compactly supported version of* (6.2)*, given by* (6.4)*. Provided that the two models, $\mathcal{GP}_1$ and $\mathcal{GP}_2$, are built using the same dataset $\{(x_m, y_m)\}_{m=1,\dots,M}$, the estimation difference $|y^\star - \hat{y}^\star|$ at a point $x^\star$ is linearly bounded by $\|\hat{\mathcal{K}}_n - \mathcal{K}\|_u$, where $\mathcal{K}$ and $\hat{\mathcal{K}}_n$ are defined as in* (6.6) *and* (6.9)*, respectively.*

The following Corollary shows how DGPR which employs CS kernels generalize to full GPR when the effective range l of the kernel goes to infinity.

Corollary 6.2. *Under the same conditions of Proposition 6.1, the estimation difference $|y^\star - \hat{y}^\star| \to 0$ as the effective range $l \to \infty$.*

These results will be employed in Section 6.3 to develop an algorithm used by the nodes of the sensor network in order to select the data that need to be transferred to the neighboring nodes, given a maximum communication bandwidth.

6.2.2 Approximation Using Different Sets of Data

The difference between a centralized and a distributed approach for spatial field estimations can be interpreted in terms of different sets of measurements as follows. If all sensor nodes transferred all the data they have collected to a central unit, the entire set of measurements would be available to a single computational unit that would be able to perform a full GPR. In a distributed framework, instead, each sensor node can be seen as a computational unit that has available only a subset of the entire set of measurements.

Therefore, we quantify the error introduced by only having available a subset of the measurement data. In order to do that, we proceed as follows. Using the same notation adopted in the previous subsection, we let $\{(x_m, y_m)\}_{m\in\{1,\dots,M\}}$ be the measurement data available to a sensor node. We assume that an additional measurement (x_{M+1}, y_{M+1}) becomes available, and we define $x = [x_1, \dots, x_{M+1}]$, $y = [y_1, \dots, y_{M+1}]$, $\tilde{x} = [x_1, \dots, x_M]$ and $\tilde{y} = [y_1, \dots, y_M]$. We aim at quantifying the difference $|y^\star - \tilde{y}^\star|$ between the estimations at a given point $x^\star \in X$ obtained incorporating or not, respectively, the new measurement. More specifically, we want to find an upper bound for $|y^\star - \tilde{y}^\star|$ with the objective of transferring only those measurements that might lead to a significant change in the estimation.

Adopting the same notational shortcuts introduced in the previous subsection, and denoting by χ the location x_{M+1}, we can proceed as follows.

Formally[1] defining

$$\bar{K}^{-1}_{\tilde{x}\tilde{x}} = \begin{bmatrix} K^{-1}_{\tilde{x}\tilde{x}} & 0 \\ 0 & 0 \end{bmatrix}, \tag{6.10}$$

one has that

$$|y^\star - \tilde{y}^\star| \leq \left\|K^T_{x^\star x}\right\| \|y\| \underbrace{\left\|K^{-1}_{xx} - \bar{K}^{-1}_{\tilde{x}\tilde{x}}\right\|}_{\Delta K}, \tag{6.11}$$

[1] $\bar{K}^{-1}_{\tilde{x}\tilde{x}}$ is not the inverse of any matrix.

where the quantity ΔK can be further simplified as follows:

$$\left\|K_{xx}^{-1} - \bar{K}_{\tilde{x}\tilde{x}}^{-1}\right\| = \left\| \begin{bmatrix} \mathcal{X} & \mathcal{Y} \\ \mathcal{Y}^T & \mathcal{Z} \end{bmatrix} \right\| \leq \max\{\|\mathcal{X}\|, \|\mathcal{Z}\|\} + \|\mathcal{Y}\| \tag{6.12}$$

where

$$\begin{aligned} \mathcal{X} &= K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi} \left(K_{\chi\chi} - K_{\tilde{x}\chi}^T K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\right)^{-1} K_{\tilde{x}\chi}^T K_{\tilde{x}\tilde{x}}^{-1} \\ \mathcal{Y} &= -K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi} \left(K_{\chi\chi} - K_{\tilde{x}\chi}^T K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\right)^{-1} \\ \mathcal{Z} &= \left(K_{\chi\chi} - K_{\tilde{x}\chi}^T K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\right)^{-1}, \end{aligned} \tag{6.13}$$

and their norms satisfy

$$\begin{aligned} \|\mathcal{X}\| &\leq \|K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\|^2 \|\mathcal{Z}\| \\ \|\mathcal{Y}\| &\leq \|K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\| \|\mathcal{Z}\| \\ \|\mathcal{Z}\| &= \left| \left(K_{\chi\chi} - K_{\tilde{x}\chi}^T K_{\tilde{x}\tilde{x}}^{-1} K_{\tilde{x}\chi}\right)^{-1} \right|. \end{aligned} \tag{6.14}$$

Then, using the Woodbury and the Shermann-Morrison formulas, the following upper bound for $\|\mathcal{Z}\|$ can be obtained:

$$\|\mathcal{Z}\| \leq \frac{\left|\frac{1}{K_{\chi\chi}}\right| + \left(\frac{\|K_{\tilde{x}\chi}\|}{K_{\chi\chi}}\right)^2 \|K_{\tilde{x}\tilde{x}}^{-1}\|}{1 - \left(\frac{\|K_{\tilde{x}\chi}\|}{K_{\chi\chi}} \|K_{\tilde{x}\tilde{x}}^{-1}\|\right)^2}. \tag{6.15}$$

In the next section, expression (6.15) will be used in order to decide whether the datapoint (x_{M+1}, y_{M+1}) should be transferred between neighboring sensor nodes or not.

The result obtained in this section is summarized by the following proposition.

Proposition 6.3. *Let $\mathcal{GP}_1$ and $\mathcal{GP}_2$ be two GPR models built using the infinitely supported Gaussian RBF kernel function* (6.2)*. Let $\{(x_m, y_m)\}_{m=1,\ldots,M+1}$ be a set of measurement data. Then, the estimation difference $|y^\star - \tilde{y}^\star|$ at a point $x^\star$, defined in* (6.11)*, is bounded by a monotone increasing function of $\|K_{\tilde{x}\chi}\|$.*

Proof. From (6.11), $|y^\star - \tilde{y}^\star| \leq \alpha \Delta K$, $\alpha \in \mathbb{R}$, $\alpha > 0$. Moreover, from (6.12), $\Delta K \leq$

$\max\{\|\mathcal{X}\|, \|\mathcal{Z}\|\} + \|\mathcal{Y}\|$. As derived above, the functions that bound their norms are all monotone increasing functions of $\|K_{\tilde{x}\chi}\|$. Hence, $\max\{\|\mathcal{X}\|, \|\mathcal{Z}\|\} + \|\mathcal{Y}\|$ is a monotone increasing function of $\|K_{\tilde{x}\chi}\|$, from which the result follows. □

The bound introduced in Proposition 6.3 allows us to estimate the difference $|y^\star - \tilde{y}^\star|$ by using the scalar quantity $\|K_{\tilde{x}\chi}\|$, without the need of computing the estimate $y^\star$ using the entire data set x. This result, together with the one of Proposition 6.1, will be leveraged in the next section to define two algorithms required to implement the communication constrained DGPR proposed in this chapter.

6.3 Communication Constrained Distributed Gaussian Process Regression

6.3.1 Maximization of the Effective Range

As stated in Corollary 6.2, in order to reduce the error due to the use of CS kernel functions, the effective ranges of the sensor nodes have to be increased. From now on, we will need to differentiate between two sensor nodes in a WSN, the receiver and the sender: we use superscripts i and j to refer to the former and the latter, respectively.

Let $\{(x_m^{(i)}, y_m^{(i)})\}_{m\in\{1,\dots,M\}}$ be the M measurement data points stored by the receiver node i. Because of the previous argument, when exchanging data with its neighbors, it is desirable that a measurement data point (x_{M+1}, y_{M+1}) is received and incorporated if

$$\max_{m\in\{1,\dots,M\}} \|x_m^{(i)} - x_{M+1}\| > l_i, \tag{6.16}$$

l_i being the effective range of node i.

Let $j \in N_i$, where $N_i \subset \{1, \dots, N\}$ is the index set of the neighbors of node i. We require that the sender node j transfers a data point to node i only if (6.16) holds. Therefore, node j has to know the locations $\{x_m^{(i)}\}_{m\in\{1,\dots,M\}}$ of the measurement data points of node i. In order for node i to communicate to node j the locations of its data points, these can be compressed by computing the minimum volume ellipsoid that encloses all

the data points $\{x_m^{(i)}\}_{m\in\{1,\dots,M\}}$. This can be done efficiently as shown, for instance, in [169]. Moreover, [169] show that scaling the minimum area enclosing ellipse about its center of a factor $1/d$, d being the dimension of the measurement data points, results in an ellipse that is completely inside the convex hull of the data points. In $\mathbb{R}^2$, denoting by $A_e^{(i)} \in \mathbb{R}^{2\times 2}$ the matrix encoding length and directions of the axes of the minimum-area ellipse corresponding to the measurement data points of sensor node i, and by $c_e^{(i)} \in \mathbb{R}^2$ the center of the ellipse, we define

$$\bar{x}_e = \max_{(x-c_e^{(i)})^T A_e^{(i)} (x-c_e^{(i)})=1} \|x - x_{M+1}\|, \tag{6.17}$$

and

$$\bar{x} = \max_{x\in\{x_m^{(i)}\}_{m\in\{1,\dots,M\}}} \|x - x_{M+1}\|. \tag{6.18}$$

Then, we can quantify the accuracy in the approximation of the measurement data points by means of the minimum area enclosing ellipse as follows ([170]):

$$\big|\, \|\bar{x}_e - x_{M+1}\| - \|\bar{x} - x_{M+1}\| \,\big| \leq \frac{d-1}{d}\sqrt{\lambda_{\max}(A_e^{(i)})}, \tag{6.19}$$

where $\lambda_{\max}(A_e^{(i)})$ is the maximum eigenvalue of $A_e^{(i)}$.

After receiving $A_e^{(i)}$, $c_e^{(i)}$ and l_i from node i, node j can decide whether to transfer its data to node i or not. Moreover, in case it needs to transfer data, given a maximum amount of data that can be transferred, it can rank the data to be transferred according to the bound (6.16) on the resulting l_i, as described in Algorithm 5.

6.3.2 Maximization of the Novelty of Measurement Data Points

In Section 6.2.2 a bound for $|y^\star - \tilde{y}^\star|$ has been derived. In the following, we briefly recall the notion of feature space in the context of GPs, which will be used to formulate an algorithm to select the measurement data that each sensor node has to transfer to its

Algorithm 5 Selection of data to transfer — Part 1

Require: datasets D_i, D_j of neighboring nodes i, j
Ensure: sorted measurement data of sensor j
  procedure NODE i
    $[A_e^{(i)}, c_e^{(i)}] \leftarrow$ minimumAreaEnclosingEllipse(D_i)
    transfer $[A_e^{(i)}, c_e^{(i)}]$ to node j
  end procedure
  procedure NODE j
    receive $[A_e^{(i)}, c_e^{(i)}]$ from node i
    $d = [\,]$
    for d_j in D_j **do**
      $d_{\max} \leftarrow$ maximumDistance$\Big(d_j, [A_e^{(i)}, c_e^{(i)}]\Big)$
      $d \leftarrow$ append($d_{\max}$)
    end for
    sort(D_j) ▷ according to d
  end procedure

neighbors according to the bound (6.15).

So far, we presented what is known as the function-space view of a GP. In the feature-space view of a GP, the function f to be estimated is expressed as $f = \phi(x)^T w$, where $\phi : D \subseteq \mathbb{R}^d \to D' \subseteq \mathbb{R}^n$ maps the inputs x to an n-dimensional, $n \leq \infty$, inner product space, the feature space. The variable w denotes a vector of weights to be estimated. In this framework, one can show that the the covariance function can be expressed as the inner product $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathbb{R}^n}$ in $\mathbb{R}^n$ ([158]). The matrix $K_{\tilde{x}\chi} = k(\tilde{x}, x_{M+1})$ in (6.15) can be then expressed as $K_{\tilde{x}\chi} = \langle \phi(\tilde{x}), \phi(x_{M+1}) \rangle_{\mathbb{R}^n}$. Since, according to Proposition 6.3 the function on the right-hand side of (6.15) is a monotone increasing function of $\|K_{\tilde{x}\chi}\|$, using the Cauchy-Schwarz inequality leads to:

$$\|\mathcal{Z}\| \leq \frac{\left|\frac{1}{K_{\chi\chi}}\right| + \frac{\|K_{\tilde{x}\tilde{x}}^{-1}\|}{K_{\chi\chi}^2} \|\phi(\tilde{x})\|^2 \|\phi(x_{M+1})\|^2}{1 - \left(\frac{\|K_{\tilde{x}\tilde{x}}^{-1}\|}{K_{\chi\chi}}\right)^2 \|\phi(\tilde{x})\|^2 \|\phi(x_{M+1})\|^2}. \tag{6.20}$$

Now, as in the case of different kernels described in Section 6.3.1, we want the sender node j, neighbor of the receiver node i, to transfer only data that significantly influence the prediction $\tilde{y}^\star$. For this reason, node j would have to know the data points that node i has.

However, as in the previous section, we do not want node i to transfer all its data points to node j. In view of what has been derived in (6.20), if node i transfers the values $\|\bar{K}_{\tilde{x}\tilde{x}}^{-1}\|$, $\|\phi(\tilde{x})\|^2 \in \mathbb{R}$, node j can evaluate what are its measurement data points that can more significantly influence the prediction $\tilde{y}^\star$of node i, and transfer them only. Alternatively, as before, if there is a maximum number of data points that can be transferred, node j can rank its data according to the influence that they can have on the prediction $\tilde{y}^\star$. The ranking-based transferring strategy described in this section is summarized in Algorithm 6.

Algorithm 6 Selection of data to transfer — Part 2

Require: datasets D_i, D_j of neighboring nodes i, j
Ensure: sorted measurement data of sensor j
  procedure NODE i
    transfer $\left[\|\phi(\tilde{x})\|, \|K_{\tilde{x}\tilde{x}}^{-1}\|\right]$ to node j
  end procedure
  procedure NODE j
    receive $\|\phi(\tilde{x})\|$, $\|K_{\tilde{x}\tilde{x}}^{-1}\|$ from node i
    $d = [\,]$
    for d_j in D_j **do**
      $b \leftarrow$ evaluateBound(d_j,$\|\phi(\tilde{x})\|$, $\|K_{\tilde{x}\tilde{x}}^{-1}\|$) ▷ (6.11), (6.12), (6.20)
      $d \leftarrow$ append(b)
    end for
    sort(D_j) ▷ according to d
  end procedure

6.3.3 Information-Entropy-Based Sensor Motion Control

Following the approach in [85], we employ an area coverage control algorithm to move the mobile sensors in the environment in which they are deployed. The approach presented by [32] lends itself to accomplish this objective. The locational cost

$$\mathcal{H}_i(x_i) = \int_X \|x_i - q\|^2 \varphi_i(q) \mathrm{d}q \tag{6.21}$$

defined for each sensor node i, $i = 1, \ldots, N$, can be minimized moving towards the weighted centroid ρ_i of X (see [32]). As done in [85], the weighting function $\varphi_i(q)$ is set to

$$\varphi_i(q) = \log \det(k(q,q) - k(q, x^{(i)})^T k(x^{(i)}, x^{(i)})^{-1} k(q, x^{(i)})), \tag{6.22}$$

where $x^{(i)}$ is the data collected by sensor node i. Assuming that it is possible to directly control the velocity of each sensor node, $\dot{x}_i$, the decentralized motion control law

$$\dot{x}_i = \gamma(\rho_i - x_i), \quad i = 1, \ldots, N \tag{6.23}$$

$\gamma > 0$ being a control gain, minimizes the locational cost (6.21), and lets each node visit regions of the environment where the variance of its estimation is higher. This allows it to collect more data in those regions, which, in turn, has the effect of reducing the variance of its estimation. In [85], this strategy is shown to minimize the information entropy of the Gaussian random variable representing the spatial field f to estimate, conditioned on the observations taken by each sensor node in the environment. The combination of motion and communication strategies described in Sections 6.3.1 and 6.3.2 is described in Algorithm 7, executed by each sensor node i.

Algorithm 7 Communication constrained DGPR

Require: nodes' positions x_i, datasets D_i, control gain $\gamma > 0$, maximum number of data points N_{max}
Ensure: communication constrained DGPR
  for i in $\{1, \ldots, N\}$ **do**
    compute ρ_i ▷ [32]
    $\dot{x}_i \leftarrow \gamma(\rho_i - x_i)$
    move with velocity $\dot{x}_i$
    for sensor node j neighbor of sensor node i **do**
      $D_i \leftarrow$ sorted data ▷ Algorithms 5 and 6
      transfer first N_{max} data points from D_i to sensor node j
    end for
  end for

Algorithms 5, 6 and 7 allow a wireless sensor network to perform distributed estimation

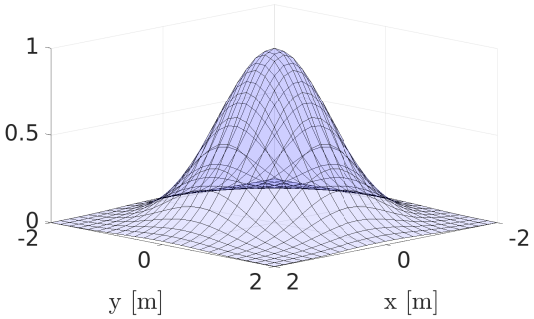


Figure 6.1: Surf plot of an example of environment field simulated over the Robotarium testbed, which has to be estimated by the network of mobile sensors.

of a spatial field by exchanging only a limited amount of data between each other at each point in time. This approach lends itself to be employed in long-term distributed estimation applications, where communication requires a non-trivial amount of energy.

6.4 Experimental Results

The communication constrained DGPR algorithm developed in the previous section has been deployed on a team of 16 ground mobile robots in the Robotarium ([130]), a remotely accessible swarm robotics testbed. Here, environment fields consisting in mixtures of Gaussian surfaces (such as the one depicted in Fig. 6.1) have been simulated, together with the sensor measurements collected by the robots. By varying the environment field to estimate, as well as the initial positions of the robots in the environment, several experiments have been performed. In the following, the results in terms of root mean square (RMS) error are compared to a centralized estimation, in which all robots are able to communicate all collected data to a centralized computational unit.

Figure 6.2 shows how the RMS error changes over time during the course of one of the experiments performed in the Robotarium. Setting the maximum communication bandwidth—expressed in terms of maximum number of data points exchanged between any two neighboring sensor nodes—6 different experiments have been performed for the same environment field. The results show how increasing the communication bandwidth

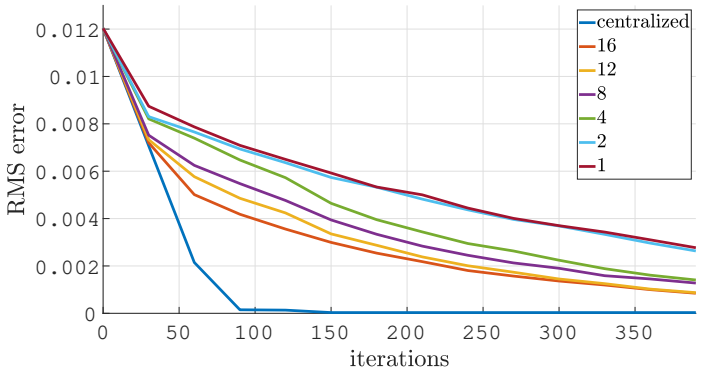


Figure 6.2: Plot of RMS error over time for one of the experiments performed in the Robotarium. The different curves show how the RMS error decreases over time as the sensor nodes exchange data between each other, as a function of the maximum number of data points exchanged at each point in time by any two sensor nodes (see legend). The blue curve at the bottom represents the centralized approach where data collected by all robots are gathered by a central computational unit, which is able to perform full GPR. As can be seen, the higher the communication bandwidth—in terms of number of data points exchanged—the faster the decrease of the RMS error towards the centralized lower bound.

leads to a faster convergence of the estimation to the lower bound defined by the centralized GPR (blue line in Fig. 6.2). For slowly-varying environment fields that take place over long time horizons, it is thus worthwhile saving energy by decreasing the maximum number of data points that can be communicated between neighbors, at the cost of decreasing the speed of the estimation convergence.

In Fig. 6.3, the effective range l_i is depicted for 16 ground mobile robots employed in one of the field estimation experiments. At iteration 120, the sensor nodes have collected enough data so they can start exchanging them with the neighboring nodes. The graphs show how Algorithms 5 and 6 are effective at selecting the data points to be exchanged in order to increase l_i for each robot and consequently decrease the RMS estimation error.

Finally, Figures 6.4a to 6.4f show snapshots of the video of one of the experiments performed in the Robotarium. The environment field to be estimated is overlaid in blue on the testbed, whereas the estimation of the field performed by one of the sensor nodes is shown in orange. Following Algorithm 7, the sensor nodes (ground mobile robots) move in the environment and exchange data to increase their effective range (yellow circle). As can

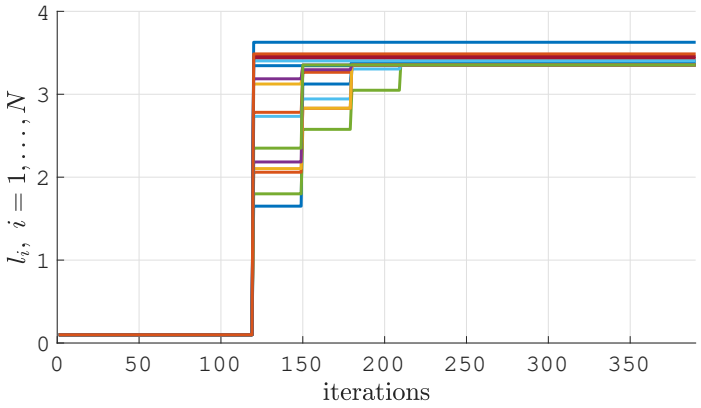
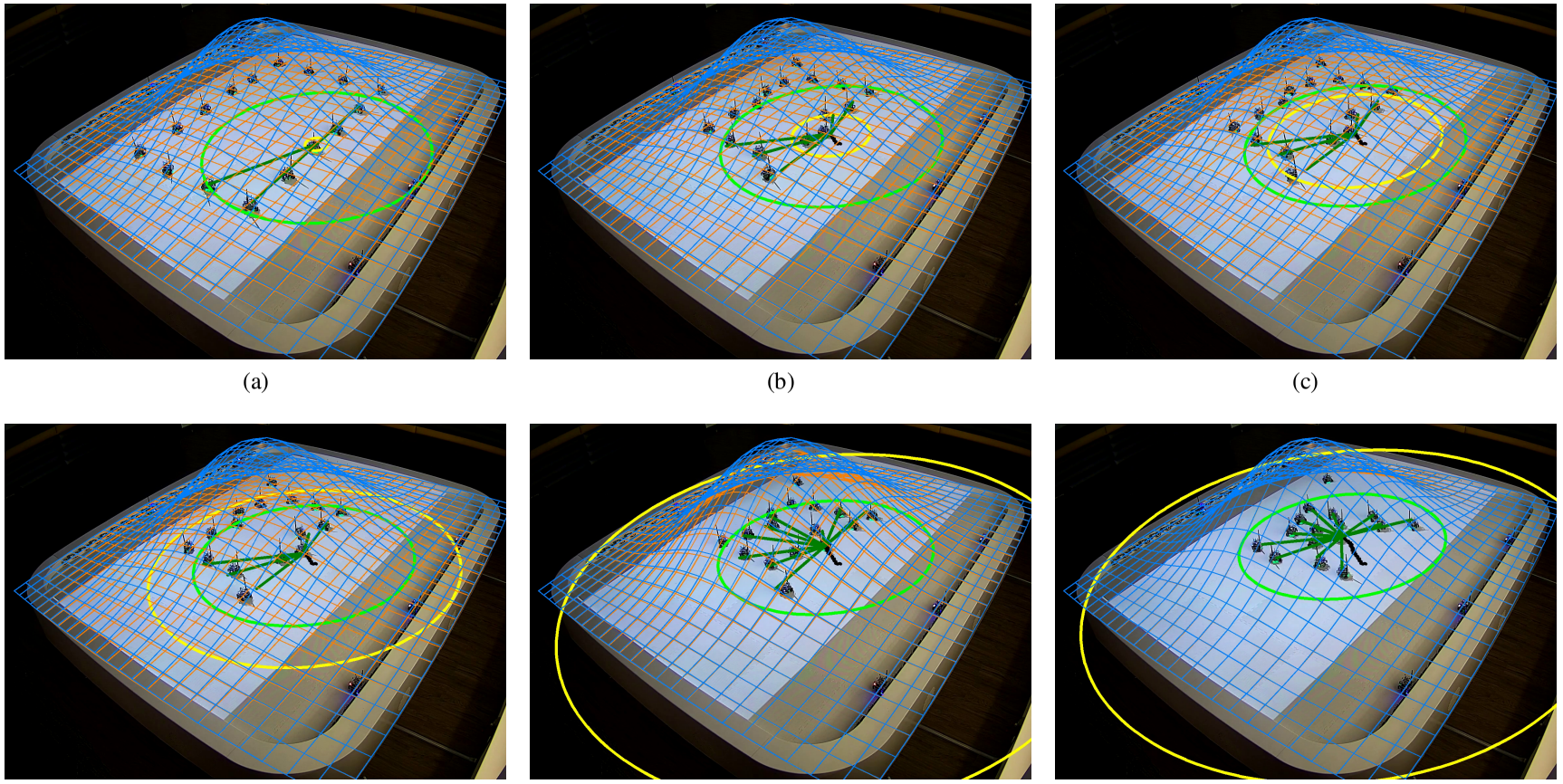


Figure 6.3: Effective range l_i of the 16 ground mobile robots recorded over the course of one of the experiment conducted on the Robotarium. The sensor nodes start communicating between each other once enough data has been collected (around 120 iterations). The graphs show how the selection of exchanged data according to Algorithms 5 and 6 allows the sensor nodes to quickly increase their effective range and correspondingly decrease the RMS estimation error (cf. Fig. 6.1).

be seen, during the course of the experiment, the estimated field (orange) approaches the ground truth (blue) as the node collects and exchanges data with its neighbors. This way, an accurate estimate of the environment field is obtained using a distributed and communication constrained algorithm.

6.5 Conclusions

In this chapter, we proposed a solution to communication constrained distributed Gaussian process regression. The main objective is that of enabling long-term deployment of mobile wireless sensor networks for spatial field estimation. Since severe limitations on the battery life of sensor nodes are caused by communication, we addressed the problem of estimating spatial fields in a distributed manner while explicitly imposing communication constraints. The proposed approach is based on the derivation of theoretical bounds on the estimation error introduced by distributed algorithms. Given a maximum communication bandwidth, we proposed an algorithm to select the most significant data to be transferred. The performance of this algorithm have been demonstrated, both in terms of estimation accuracy and amount of data transferred, on an team of mobile robots.



(a) (b) (c)

(d) (e) (f)

Figure 6.4: Snapshots from the video of the experiments performed on the Robotarium. The estimation of a simulated environment field (blue surf plot) performed by one of the 16 mobile robots employed in the experiment is depicted. Its trajectory is shown as a thick black line, its communication and effective ranges are shown in green and yellow, respectively. As can be seen, during the course of the experiment, the error between the true environment field and its estimate (orange surf plot) decreases, as the effective range increases. The video of the experiment is available online at the link `https://youtu.be/6vTcnh4wsZU`.

Part II

Robot Design

CHAPTER 7
THE SLOTHBOT

This chapter opens the second part of this thesis devoted to studying the deign principles of robots envisioned for long-term deployment. In Part I, we referred to the *persistent* operations of robots intended as the scenario in which robots are to remain functional for time periods much longer than their battery life. In this Part II, we present two robot designs conceived to enable and facilitate long-term operations.

In this chapter, we start by presenting the SlothBot, a solar-powered wire-traversing robot envisioned for long-term environmental monitoring applications. The SlothBot is a slow-paced and energy-efficient robot—hence its name—capable of moving on a mesh of wires by switching between branching wires. Unlike ground mobile robots or aerial robots employed in environmental monitoring applications, the use of wire-traversing robots allows for longer-term deployment because of the significantly lower energy consumption. Wire-traversing, coupled with the use of solar panels, facilitates the self-sustainability of the SlothBot. Locomotion and wire-switching maneuvers are performed in a fail-safe fashion, inasmuch the robot is always firmly attached to the wires, even when switching between branching wires. This is achieved by employing a two-body structure featuring an actuated decoupling mechanism. In this chapter, we show the design and the motion control of the SlothBot, together with the results of long-term monitoring experiments.

Wire-traversing robots are able to move along cables, wires, and similar infrastructure. Due to their wire-traversing capability, these robots are suitable for applications in agricultural robotics [22], environmental monitoring [21] or maintenance in hazardous places, as in the case of power line inspection [17]. The latter has been the main catalyst for the development of wire-traversing robots, see e.g. [18, 19, 20]. The geometric and mechanical design of wire-traversing robots varies a lot among the existing mechanisms developed

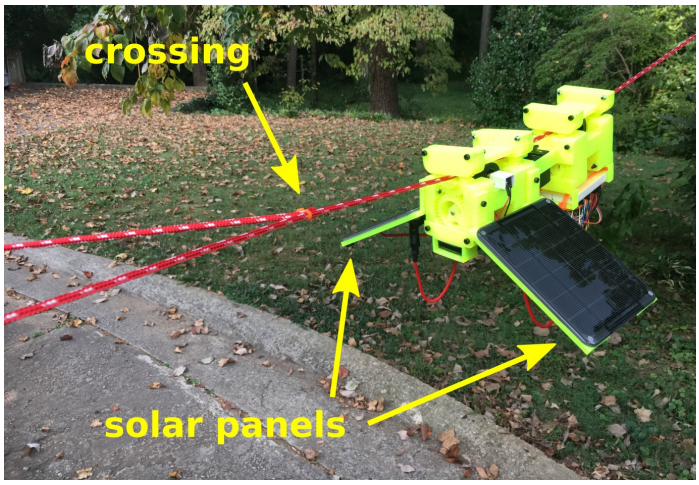


Figure 7.1: The SlothBot is a lightweight, solar-powered, minimally-actuated, wire-traversing robot, capable of switching between branching wires and envisioned for long-term environmental monitoring applications.

over the past 20 years, as discussed in [16]. Nevertheless, common features shared, to a certain extent, among many of the architectures are: (i) simplicity of the system design and, consequently, of the motion control; (ii) reduced localization errors and navigation complexity; (iii) low energy requirements. These characteristics have allowed wire-traversing robots to gain increasing interest in different applications domains.

The mechanical design of the SlothBot has been conceived to be simple and compact, while at the same time allowing the robot to switch between branching wires and to remain safely attached to them even during the switching maneuvers. In order to reduce the maintenance efforts and to minimize the risk of failures, robots targeting long-term applications should, in fact, be fail-safe and as simple as possible. We refer to *long-term* tasks as tasks that occur over extended periods of time and, in particular, which require multiple battery charges. In this sense, quadcopters, which are used for agriculture robotics applications, are not suitable for long-term tasks because of the high power they require to remain operational, which forces them to visit charging stations (which could be locations where batteries are recharged or swapped) [22].

We envision the deployment of the SlothBot for long-term environmental monitoring tasks, required, for example, in agricultural robotics applications. In order to move in an

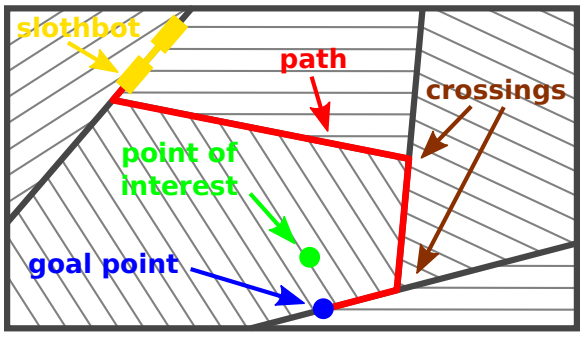


Figure 7.2: Example of monitoring applications in agricultural robotics using a wire-traversing robot. Crosshatched areas represent different crops in a field. The robot (depicted in yellow) has to collect measurements at the point marked in green. It, therefore, traverses the wires (on the red path), overcoming crossings, until it reaches the blue point, closest to the green location.

agricultural field, the SlothBot has to be able to traverse a mesh of wires and, therefore, to overcome crossingsure 7.2 represents an idealized agricultural field. The crosshatched areas represent different crops in the field, while the thick black lines show a mesh of wires that go across the field. The objective of the robot deployed in the field is that of monitoring phenomena that take place over long time scales, such as crop growth. For this task to be successfully completed, energy efficiency and fail safeness are required features, so that maintenance and risk of failure are minimized. Moreover, as mentioned before, in order to be able to move on the mesh of wires across the field, the robot has to traverse intersections of wires. For this to be possible, the robot has to have the capability of switching between different wire branches.

To summarize, the objectives that drove the design and development of the SlothBot are:

- energy efficiency
- wire-switching capability
- fail-safeness.

Among the existing wire-traversing robotic platforms, the ones that fulfill all the listed

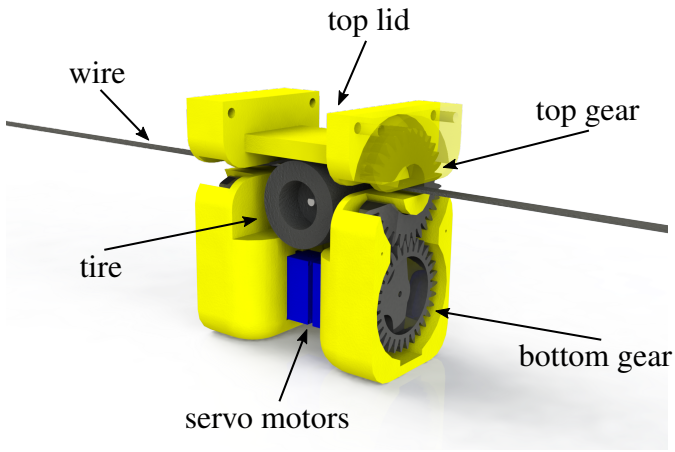


Figure 7.3: One of the two bodies of the SlothBot, with the nomenclature used in the chapter. Part of the top lid has been made transparent to be able to fully see the top gear.

requirements exhibit complex designs that are, generally, less energy-efficient and not easy to control, requiring careful motion planning.

The remainder of the section is organized as follows. Section 7.1 describes the multi-body prototype of the SlothBot and, in particular, the locomotion principle, the wire-switching mechanism, and the hardware architecture. Section 7.2 presents an improved single-body design which has been deployed in the Atlanta Botanical Garden starting from June 2020. Section 7.3 reports the results of long-term environmental monitoring experiments performed with the multi-body and single-body SlothBot designs. Finally, the last two sections of the chapter, Sections 7.4 and 7.5, propose motion control strategies that are particularly suitable for controlling the SlothBot to move on a mesh of wires [171] and on a single wire [172], respectively.

7.1 Multi-Body Mechanical Design

The SlothBot is composed of two bodies connected by an actuated hinge, as seen in Fig. 7.1. Each body, depicted in Fig. 7.3, houses a driving motor connected to a rim on which a tire is mounted. The use of wheels for locomotion is simple, energy-efficient and makes the SlothBot safer when compared to brachiating robots. The switching maneuver is made

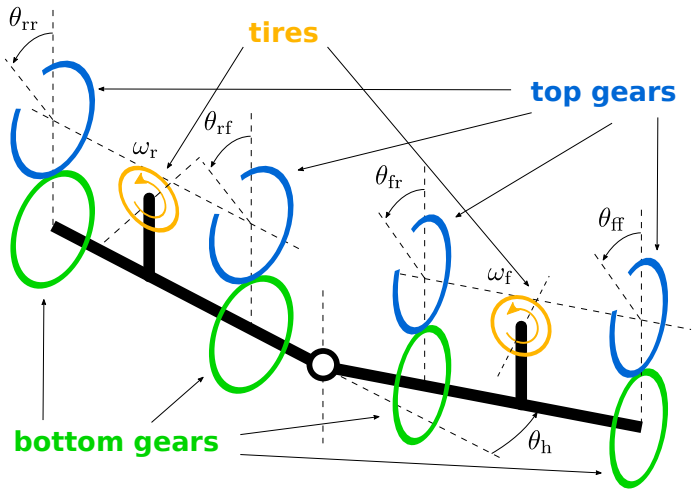


Figure 7.4: Kinematic scheme of the SlothBot highlighting its degrees of freedom. θ_{ff}, θ_{fr}, θ_{rf}, θ_{rr} are the angles of which the front and rear top gears of the front and rear body, respectively, can rotate. These degrees of freedom are actuated by 4 servo motors. θ_h is the relative angle between the two bodies of the SlothBot, and is also actuated by a servo motor. ω_f and ω_r are the speeds of the 2 DC motors that move the SlothBot.

possible through four pairs of spur gears. Each pair is stacked vertically with the top gear having a circumferential gap of 20°. The four gears with such a gap will be referred to as C-shaped gears throughout the section. The bottom gear is driven by a servo-motor and allows orienting the C-shaped gear's gap to three different positions. These positions are top, left and right, which correspond to the robot going straight, turning right and turning left, respectively. This novel wire-switching mechanism minimizes the required actuation to only one servo-motor per gear pair, thus significantly increasing the simplicity and compactness of the design. The fail-safeness of the SlothBot is guaranteed through the use of the two bodies connected by a hinge: this consists of a rotational joint, whose axis lies in the longitudinal plane of the robot, and it is actuated by a servo-motor. The servo-motor ensures the alignment of the bodies with respect to the branches they are traversing. The switching maneuver itself will be explained in more detail in Section 7.1.2. Figure 7.4 depicts the scheme of the robot highlighting its degrees of freedom. Relative to existing designs that are capable of wire-switching, thanks to its locomotion principle and wire-switching mechanism, the SlothBot is more energy- efficient as well as fail-safe.

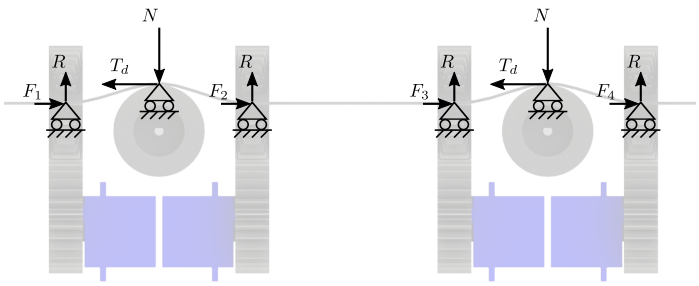


Figure 7.5: Diagram of forces acting on the SlothBot. The small triangles represent hinge/cart supports. In the background the skeleton of the SlothBot is shown, highlighting the contact points between its bodies and the wire. T_d is the force generated by the motor torque τ. F_i, $i = 1, \ldots, 4$, are the reaction forces due to friction. N and R are the force and the reaction due to the weight of robot and payload.

7.1.1 Locomotion Principle

The wire that the SlothBot has to traverse is compressed between the tires and the top lids of both bodies. This lets the friction force remain high enough to allow the tires to move the robot. Figure 7.5 depicts the forces involved in the dynamic equilibrium of the robot. The triangular supports represent hinges/carts capable of reacting with the forces drawn in the figure. The horizontal and vertical equilibrium equations reduce to:

$$\sum_{i=1}^{4} F_i = 2T_\mathrm{d}, \qquad 2N = 4R, \tag{7.1}$$

where T_d is the force generated by the motor used for locomotion, $F_i,\ i = 1, \ldots, 4$ are the horizontal reaction forces exerted by the body due to friction, and N and R are the forces and the reactions of the constraints, respectively, due to the weight of the robot and of the payload. Starting from (7.1), we can calculate the motor torque required to carry a payload of a given mass. Let m be the mass of the payload the robot has to move around. Examples of payloads can be sensor modules or hard drives used to store collected measurements. The minimum value of motor torque τ provided by each motor that is required to move the

robot along the wire is given by:

$$\tau = T_{\mathrm{d}} r = 2Fr = 2\mu Rr = \frac{(m+M)g}{2}\mu r, \tag{7.2}$$

where r is the effective rolling radius of the tires [173], $F = \frac{T_{\mathrm{d}}}{2}$ (from (7.1) assuming reaction forces equally distributed among the 4 support points), μ is the friction coefficient between the wires and the robot body, R the normal reaction force exerted by the supports (Fig. 7.5), M is the mass of the robot, and g is the acceleration due to gravity.

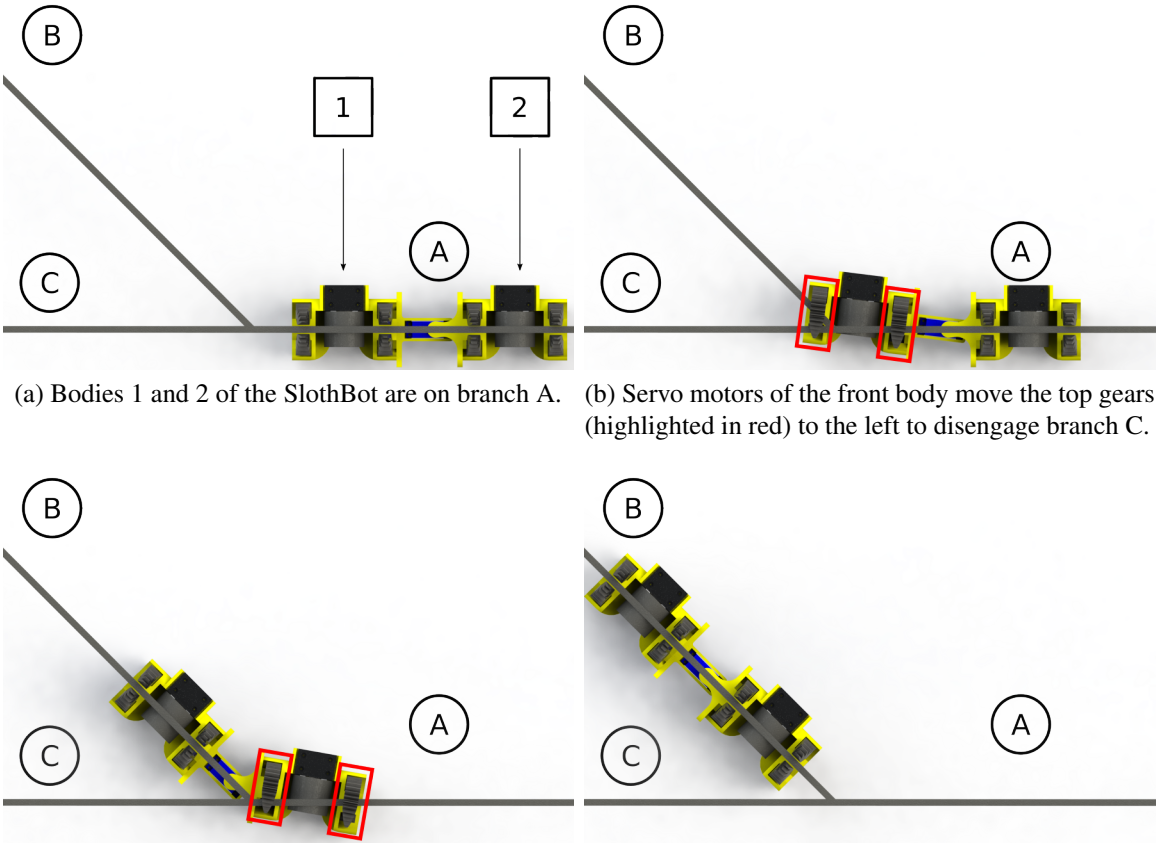
While moving on a mesh of wires, the SlothBot has to overcome wire crossings by switching to different wire branches. The switching maneuver is the subject of the next section.

7.1.2 Fail-Safe Wire Switching

Switching between different wires is required in all applications where the robot is constrained to move on a mesh of wires. As discussed in the Introduction, there have been several solutions proposed to the wire-switching problem for wire-traversing robots. The mechanism we propose in this section is robust against failures of the actuators and of the actuators' controller. More specifically, we designed the SlothBot in such a way that it firmly remains on the wire in case the actuators fail during the wire switch, or in case the motors are actuated at the wrong time. This is a key feature for robotic systems designed for long-term monitoring tasks. In fact, as the probability of failures increases with time, a way of at least mitigating unsafe consequences of these failures is required. The switching method presented in this section fulfills this requirement.

Fig. 7.6a to Fig. 7.6d show the sequence of actions performed by the SlothBot to switch to a different wire branch:

- Fig. 7.6a: Both bodies of the SlothBot are on the same wire, indicated by branch A, the servo motors keep all the gaps of the C-shaped gears straight up, holding the top



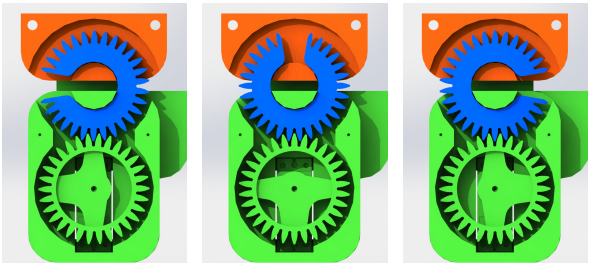
(a) Bodies 1 and 2 of the SlothBot are on branch A.

(b) Servo motors of the front body move the top gears (highlighted in red) to the left to disengage branch C.

(c) The rear body performs the same maneuver that the front body performed in 7.6b.

(d) Both bodies of the SlothBot are on branch B.

Figure 7.6: Simulated wire-switching maneuver: the SlothBot switches from branch A to branch B. The top lids, that ensure that the wire is in contact with the tires, have been hidden to make the orientation of the top gears visible.



(a) (b) (c)

Figure 7.7: The switching mechanism for the SlothBot. The red components of the robot always remain above the wires, while the green components are confined to stay below them. The C-shaped blue gear allows the red and green parts to be held together, while, at the same time, allowing the wires to disengage from the robot during wire-switching maneuvers.

lids on the wire while not allowing the wire to disengage. The objective is switching from branch A to branch B. 3

- Fig. 7.6b: Body 1 is at the junction between the wires, its C-shaped gears are both open allowing the branch C to disengage from it.
- Fig. 7.6c: Body 2 performs the same maneuver that Body 1 completed in Fig. 7.6b, disengaging from branch C and moving onto branch B.
- Fig. 7.6d: Both bodies are on the same wire, and the SlothBot successfully switches from branch A to branch B.

The described switching maneuver is made possible by the use of the C-shaped top gears, with a circumferential gap of 20°. In Fig. 7.7 the operation of these gears is shown in more detail. In order to be able to switch between branching wires, the parts of the robot that are above and below the wires, depicted in red and green, respectively, have to be disconnectable. In fact, referring to the sequence in Figures 7.6a to 7.6d, the robot that has to switch from branch A to branch B, has to cross over branch C. Thus, at some point in time, opening a gap between the red and green parts is required. Nevertheless, in order to keep the robot hung to the wires at any point in time, parts above and below the wires have to remain connected.

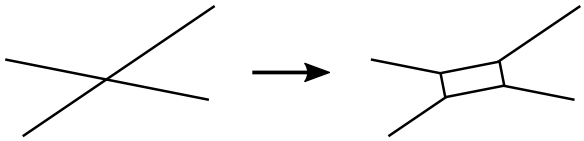


Figure 7.8: Example of turning one 4-way crossing into a sequence of four 3-way crossings. This modification is required since the SlothBot is only able to traverse 3-way crossings.

In order to accomplish what has been described, an actuated decoupling mechanism is proposed. This consists of a train of two spur gears per side per body of the SlothBot. One gear is mounted on the servo motor (green gears in Fig. 7.7); the other one (blue gears in Fig. 7.7) has a C-shape that allows the top lid to be held for any orientation of the gap (i.e. for any value of the angles θ_{ff}, θ_{fr}, θ_{rf}, θ_{rr} of Fig. 7.4). At the same time, the shape of this gear allows wire branches to disengage from the left (Fig. 7.7a) and from the right (Fig. 7.7c) of the robot. While the robot is driving straight on a wire, the C-shaped gears are oriented as in Fig. 7.7b.

Because of the fail-safeness constraint on the switching maneuver, the SlothBot does not have the ability of traversing crossings with more than 3 branching wires. However, this is not a substantial limitation, since any crossing can be turned into a sequence of 3-way crossings as shown in Fig. 7.8. Moreover, from Fig. 7.6 it is clear that the SlothBot cannot traverse crossings when the turning angle is smaller than 90°. However, this situation can always be avoided by performing a two-step maneuver in which two obtuse-angle crossings are overcome instead of one acute-angle crossing. Nevertheless, with the technique shown in Fig. 7.8, all the resulting crossing angles can be made strictly larger than 90°. Furthermore, the inability of the SlothBot to traverse crossings with more than 3 wire branches allows the synthesis of a motion control law that will be particularly suitable for navigating over meshes of wires, as will be explained further in Section 7.4.

7.1.3 Hardware Architecture

A prototype of the SlothBot has been realized using rapid prototyping technologies. All the main components are 3D printed using standard PLA material. The printing time of

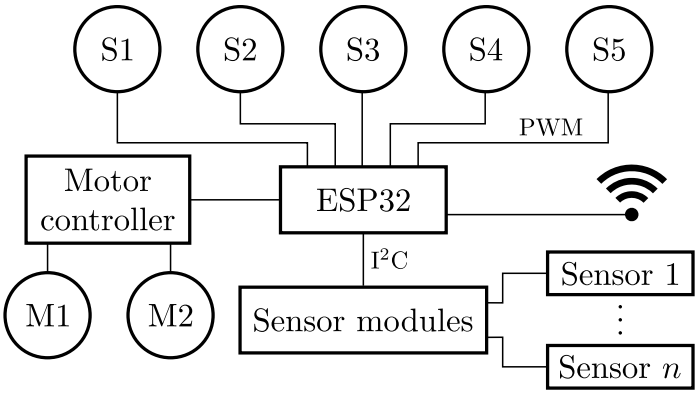


Figure 7.9: Hardware architecture of the multi-body design of the SlothBot. The main processing unit, ESP32, controls the DC motors, M1 and M2 through a motor controller, and the servo motors, S1, S2, S3, S4, and S5, using PWM signals. The unit can communicate with several sensors using the available I^2C bus. Moreover, the ESP32 is Wi-Fi enabled, thus allowing remote monitoring by handling requests of sensor data via a locally hosted web server.

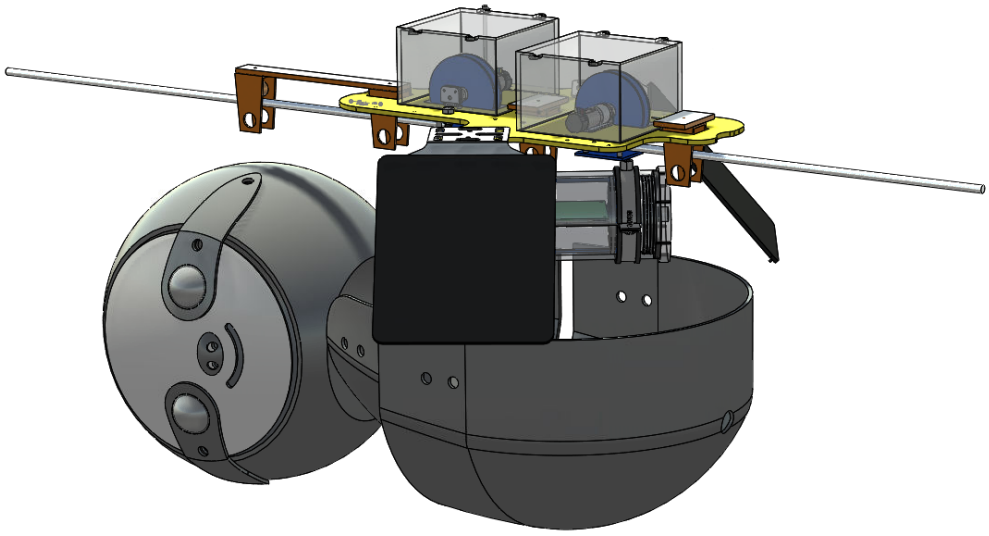
an entire SlothBot is about 30 hours using a commercial 3D printer. Assembly time is 30 minutes since all the other components are off-the-shelf. The realized prototype of the SlothBot is 25.5 cm long, 11.2 cm high (13.5 cm with solar panels) and 6 cm wide (31.2 cm with solar panes). The 2 motors are Micro Metal Gear motors, operating at 6V, with 1000:1 reduction ratio, maximum speed of 32 rpm and maximum torque of 0.88 Nm, which allow the SlothBot to move at a maximum speed of 5 cm/s. The servo motors used to rotate the spur gears are standard 9-gram servo motors with an operating voltage of 5V and a maximum torque of 0.16 Nm. The servo motor used to actuate the hinge between the two bodies, required during the wire-switching maneuver, has a maximum torque of 1.52 Nm while operating at 5V. The robot is powered by a rechargeable 7.4V 1000 mAh LiPo battery. Two solar panels (visible in Fig. 7.1), mounted on the sides of the SlothBot, are used to recharge the battery when the light intensity is high enough. A maximum power point transfer (MPPT) solar charging circuit is used to regulate the charging current based on the solar cell characteristics. This way, the power efficiency, expressed as the ratio between the power that is transfered to the battery and the power received from the sun, is maximized at each time instant.

Figure 7.9 shows the hardware architecture of the SlothBot. The main processing unit onboard the SlothBot is the ESP32, an IoT-enabled microcontroller. The microchip directly controls all the 5 servo motors, and, through a dedicated motor controller, the 2 DC motors that drive the tires. The SlothBot is designed to carry sensors for environmental monitoring applications. In Section 7.3 we show the results of a 1-day experiment during which the SlothBot measured its environment's temperature and luminosity. The ESP32 connects to the sensors using the I^2C protocol, usually available on sensor data acquisition boards. Moreover, the microcontroller hosts a web server which handles requests of sensor data by a client running on a desktop computer responsible for storing the collected measurements, thus enabling remote environmental monitoring. See Appendix for list of components and cost breakdown of the SlothBot prototype.

7.2 Single-Body Mechanical Design

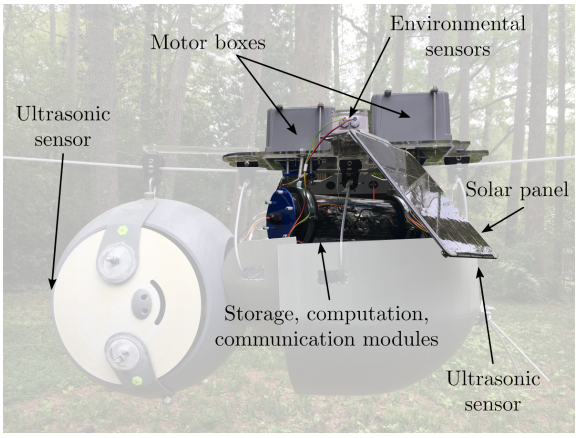
In this section, a modified design of the SlothBot presented in the previous section is described. The major difference is the fact that the multi-body structure has been substituted by a single-body one where the wheels used for locomotion are mounted on top of the wire where the robot moves. Figure 7.10 shows pictures of the computer-aided design (CAD) model and the real robotic system. A sloth-like shell has been designed and realized using additive manufacturing techniques for decorative purposes (see Fig. 7.12). In fact, in June 2020, this version of the SlothBot has been deployed in the Atlanta Botanical Garden, as shown in Figures 7.11a to 7.11c.

As this SlothBot only moves on a single wire, the wire-switching mechanism presented in the previous section is not required anymore. Consequently, the motors mounted to the wheels arranged on top of the wires can be permanently connected to the power source position below the wire for stability purposes. Except for the motors moving the wheels to achieve locomotion with a similar principle to the one explained in the previous section, all the electronic components are housed in a waterproof tube. The electronic board is depicted



(a)

(b)



(c)

Figure 7.10: Single-body design of the SlothBot. In Fig. 7.10a, the computer-aided design (CAD) model is shown, with its physical realization being depicted in Fig. 7.10b. Figure 7.10c highlights the main components of this single-body SlothBot. The decorative shell is being designed for the deployment of the SlothBot in the Atlanta Botanical Garden, shown in Fig. 7.11 (see details in Figures 7.12a and 7.12b).

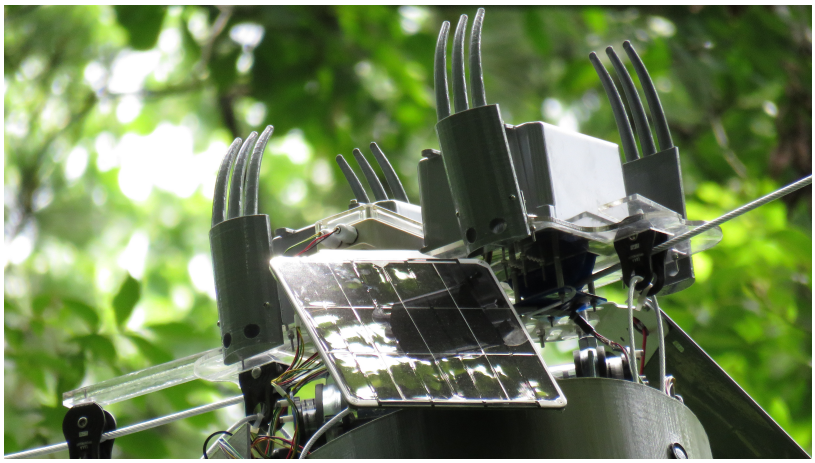
(a)

(b)

(c)

Figure 7.11: The SlothBot deployed in the Atlanta Botanical Garden in June 2020.

(a)



(b)

Figure 7.12: Details of the decorative shell that has been designed for the deployment of the SlothBot in the Atlanta Botanical Garden. The eyes of the robot (visible in Fig. 7.12a) consist of 2 RGB LEDs protected by a transparent acrylic dome, whose color and intensity are independently controlled to relay information about the robot battery status.

as a green plate in a transparent cylinder in Fig. 7.10, and it is positioned right below the yellow plate where motors are mounted. The latter can be seen in Fig. 7.10 attached to the blue wheels inside the transparent boxes on top of the yellow plate. Two solar panels are mounted one on each side of the robot: one is used to power the entire electronic circuit, while the other one serves as backup or to power additional application-specific circuitry the robot can carry during deployment.

7.2.1 Hardware Architecture

The architecture of the single-body SlothBot, shown in Fig. 7.13 is significantly changed compared to the one described in Fig. 7.9 related to the multi-body robot design. This allows the robot to activate and deactivate its individual hardware modules separately. The main computational unit consists of a Raspberry PI endowed with an external real time clock (RTC). This feature allows the board to adapt its operation to different times of the day and to different seasons. Moreover, the Raspberry PI has wireless connection capabilities with which it is able to communicate with a base station. This is used, for example, to log the data collected by the SlothBot, eliminating the need of carrying storage media—which would unnecessarily increase the weight and the power consumption of the robot—on board.

As the Raspberry PI requires about 500 mA of current to operate, a Teensy 3.2 is mounted on the SlothBot as well. This microcontroller only needs 80 mA nominal current to operate, and it is therefore responsible for the power management of the robot. It communicates with the Raspberry PI through serial through which the latter can send requests to be turned off and on to the former. Moreover, and equally importantly, the Teensy interfaces directly with all sensors and actuators which are mounted on the SlothBot. These are: 2 DC motors, 3 environmental sensors (measuring air properties, air quality, and light, respectively), 2 ultrasonic sensors—used to avoid obstacles present on the wire on which the SlothBot is moving—LEDs, and 2 power sensors—to monitor the power flowing in or

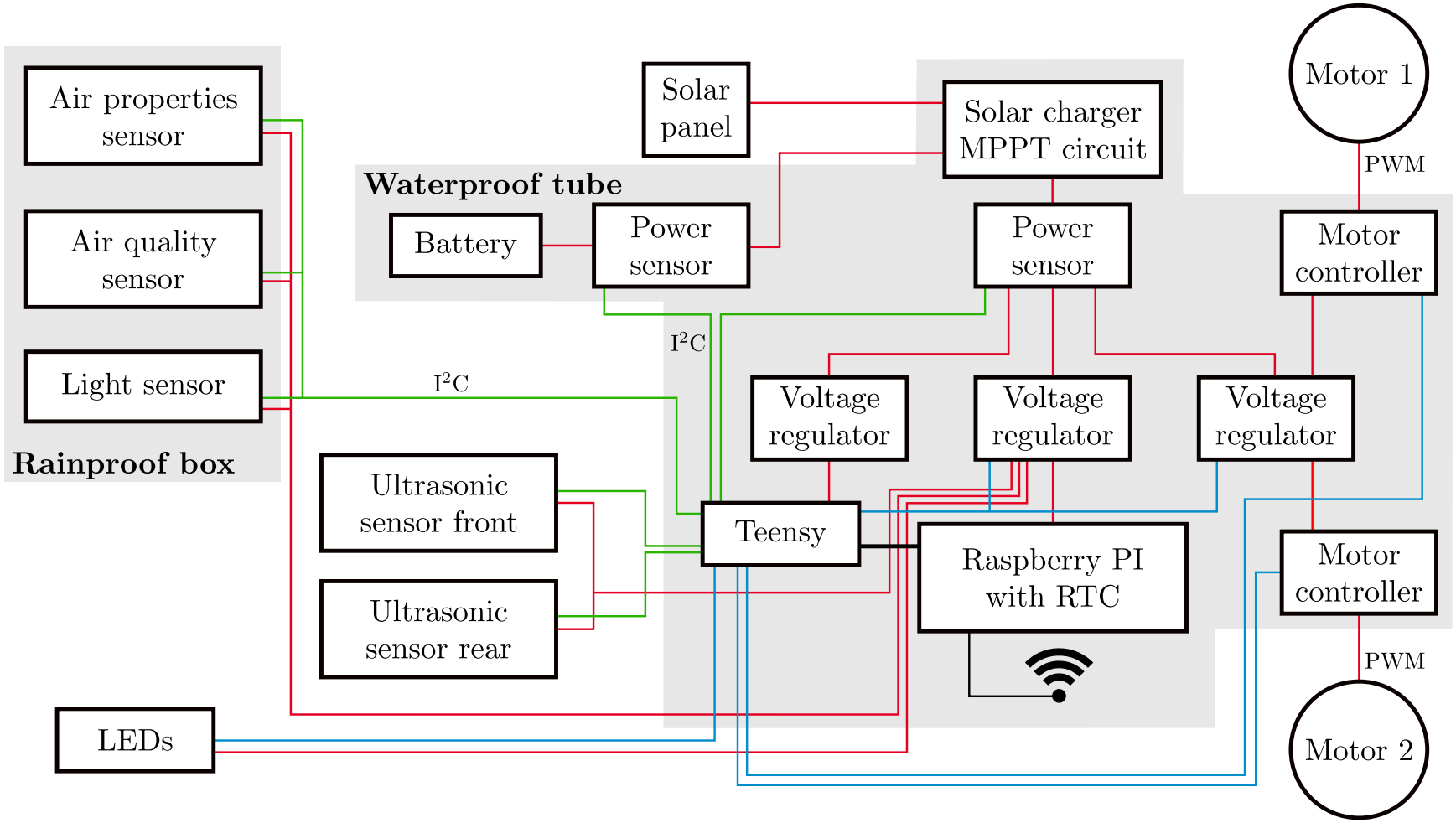


Figure 7.13: Hardware architecture of the single-body design of the SlothBot. The main processing unit is a Raspberry PI running a Linux operating system with wireless connection capabilities. It is endowed with a real time clock (RTC) in order to allow it to adapt its operating modes based on the time and date. As this computational module is more energy consuming compared to a simple microcontroller, it is supported by a Teensy 3.2, responsible for power management and interface with sensors and actuators, with which it communicates through a serial protocol. The shaded regions in the figure represent rainproof and waterproof enclosures where the components are mounted. LEDs, ultrasonic sensors, and motors are also waterproofed but they are mounted in containers separate from the one where the main electronics board is housed. The power is provided by a solar panel which charges a 10000 mAh LiPo battery though a MPPT solar charger circuit similar to the one mounted on the multi-body SlothBot version. The color code of the connection is as follows: red for power-carrying connections, blue and green for signal-carrying connections (blue is output from and green input to the Teensy, respectively), while the black connection between the Raspberry PI and the Teensy denotes a two-way serial connection.

out of the battery, as well as the one absorbed by the entire circuit.

The modular design can be subdivided into three groups, each characterized by its power source, the 3 voltage regulators block in Fig. 7.13. Through these voltage regulators, the Teensy can switch off entire sections of the circuit. In particular, the Raspberry PI, the LEDs and the environmental sensors can be deactivated, significantly reducing the power consumption of the SlothBot. The same holds for the motors, whose controller boards can be activated only when there is the need for the SlothBot to move. Through the capabilities described above, an intelligent power management logic can be devised in order to let the SlothBot remain operational for truly long time periods.

7.2.2 Software Architecture

In order to let the SlothBot execute tasks (e.g. environmental monitoring) persistently, techniques developed in Part I of this thesis can be leveraged, like, for example, the task persistification framework described in Chapter 3. For the task persistification strategy to work, it is assumed throughout Chapter 3 that a solution to the optimization problem guaranteeing the persistent execution of tasks exists (i.e. the QP (3.51) is feasible). Nevertheless, there are cases in which this is not the case. Consider, for example, the case of a solar-power robot deployed in an environment characterized by unfortunate weather conditions, where no light intensity is present to recharge its batteries. In the scenario, the survivability of the solar-powered robot is compromised. For this reason, to manage the operation of the SlothBot, we adopted a robot-programming paradigm called behavior-based programming [174].

The hybrid automaton describing the behaviors of the SlothBot is shown in Fig. 7.14, where each behavior (depicted as circles) is controlled by dedicated hardware and software modules. The symbols t_X are used to denote the times spent in state X, whereas Δt_X represents the desired time interval to be spent by the SlothBot in state X. Moreover, the values Adequate charge$^+$ $>$ Adequate charge$^-$ are used to avoid Zeno executions of

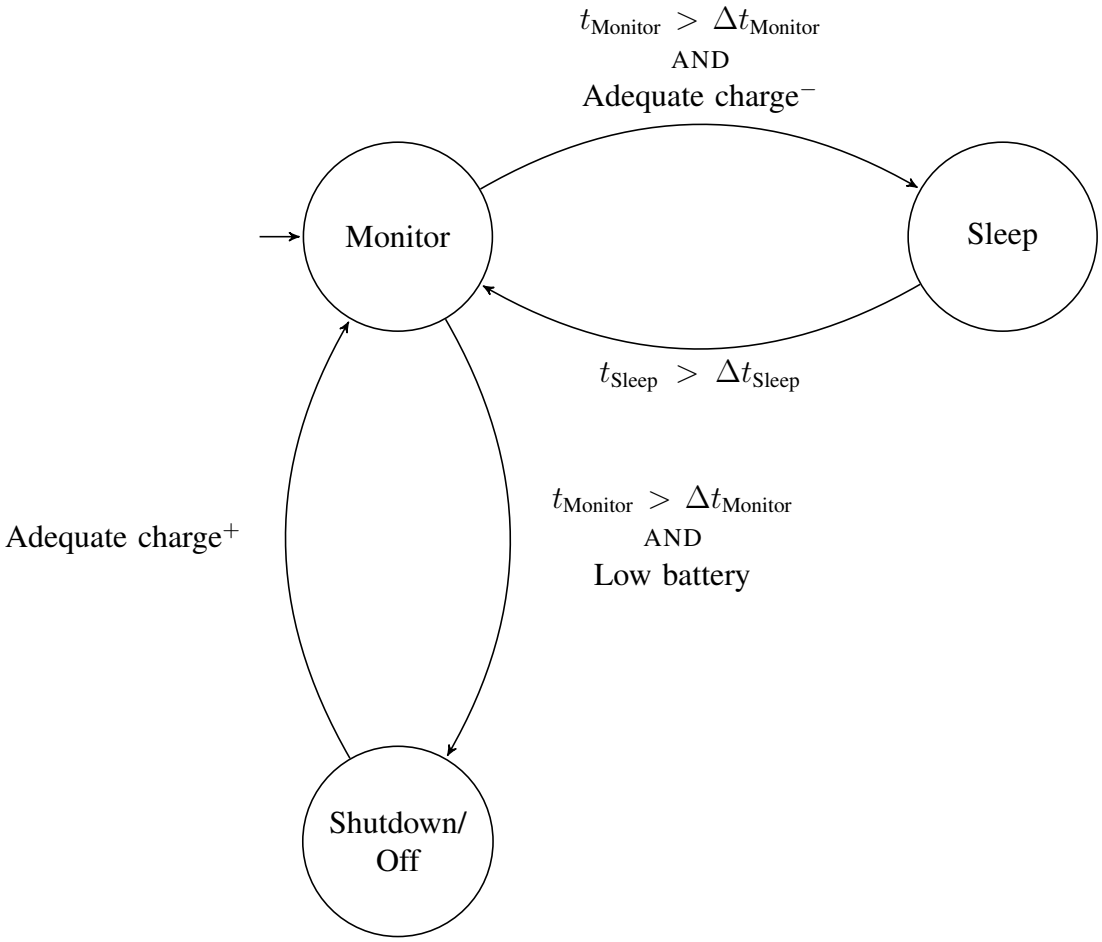


Figure 7.14: Hybrid automaton describing the behaviors of the SlothBot. Each behavior (Monitor, Sleep, Shutdown/Off is managed by dedicated hardware and software modules. In particular, the SlothBot starts in Monitor state. From here, based on time or sensor readings, the Raspberry PI can send messages to the Teensy (see Fig. 7.13) in order to be deactivated—by going to Sleep of Shutdown/Off mode—and reactivated based on a time or sensor readings—in the case of the Sleep or Shutdown/Off modes, respectively.

the automaton [175] when transitioning between the Monitor, Sleep, and Shutdown/Off behaviors.

The transitions are based on time or sensors readings and are determined by high-level logic software modules executed on the Raspberry PI and Teensy boards. Based on time ($\Delta t_{\text{Monitor}}$) or sensor readings (remaining energy in the battery), the former sends messages through serial to the latter (see Fig. 7.13) to make the SlothBot transition into Sleep mode—where Raspberry PI, environmental sensors and motors are shutdown until a pre-decided amount of time, Δt_{Sleep}, has elapsed—or to Shutdown/Off mode—where Raspberry PI, environmental sensors and motors are shutdown until enough battery charge is measured.

7.3 Experimental Data

To show the effectiveness of the design of the SlothBot, we performed two long-term monitoring experiments with the multi-body and the single-body designs, respectively, during which we left the robots in the environment for 24 hours and 2 months, respectively. Note that a battery life can keep the robot operational for about 12 hours (multi-body design) and 2 days (single-body design), without any motors being activated. For this reason, these classify as a long-term experiments.

Figure 3.11 in Chapter 3 shows the battery voltage and light intensity recorded over the course of a 1-day long experiment performed using the multi-body SlothBot, while the results of the deployment of the single-body SlothBot for a period of 2 months—during which the picture in Fig. 7.10c has been taken—are reported in Fig. 7.15. The latter are in terms of battery voltage and current flowing in and out of the battery. In the week April 13-20, for instance, the SlothBot underwent 3 charging cycles, noticeable because of the negative current measured by the power sensor connected to the battery (see Fig. 7.13).

In the next two sections, motion control strategies for robots moving on a mesh of wires and on a single wire, respectively, are developed. These control strategies lend themselves to be employed to control the multi-body and single-body SlothBot, respectively, presented

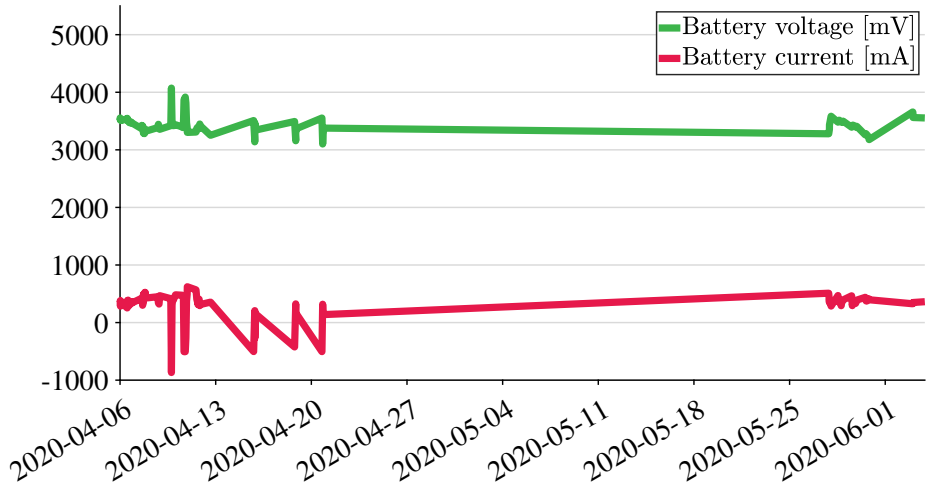


Figure 7.15: Battery voltage and current data measured by the power sensor connected to the battery (see Fig. 7.13) during the course of a 2-month-period outdoor deployment of the single-body SlothBot (see Fig. 7.10c). During the week April 13-20, one can see how the SlothBot recharged its battery three times thanks to its solar panels. The voltage increases towards the maximum nominal value of 3.7 V, with negative current values representing currents flowing in the battery.

in this first part of the chapter.

7.4 Motion Control on Wire Meshes

In this and the following section, we present methods to allow wire-traversing robots to spread across the environment where they are deployed, while being constrained to move on a mesh of wires or on a single wire. Coverage control is used as an algorithm to implement environmental monitoring by controlling the robots to arrange themselves optimally in the environment. Nevertheless, the techniques to adapt coverage control to robots that move on wires can be employed for other types of planning and control algorithms in presence of such motion constraints.

In this section, we consider the coverage control problem for a team of wire-traversing robots. The two-dimensional motion of robots moving in a planar environment has to be projected to one-dimensional manifolds representing the wires. Starting from Lloyd's descent algorithm for coverage control, a solution that generates continuous motion of the robots on the wires is proposed. This is realized by means of a Continuous Onto Wires

(COW) map: the robots' workspace is mapped onto the wires on which the motion of the robots is constrained to be. A final projection step is introduced to ensure that the configuration of the robots on the wires is a local minimizer of the constrained locational cost. An algorithm for the continuous constrained coverage control problem is proposed and it is tested both in simulation and on a team of mobile robots.

In Section 7.4.1, after a brief overview on the Lloyd's algorithm for coverage control for multi-robot systems, the definition of the constrained coverage control problem is introduced and a solution to it is proposed. In Section 7.4.3 the main results of the section are used to synthesize a motion controller for the robots on the wires in order to solve the constrained coverage control problem. In Section 7.4.5, the results of the deployment of the proposed algorithm on a team of mobile robots are reported.

7.4.1 Coverage on Wires

In this section we first introduce the notation that will be used throughout the chapter and then derive the results on constrained coverage control. These will be used in Section 7.4.3 to derive the motion control law to be applied to a group of wire-traversing robots.

Locational Optimization

Let $X \subsetneq \mathbb{R}^2$ be a closed and convex polygon and ∂X its boundary. Let $p_1, \ldots, p_N \in \mathbb{R}^2$ denote the locations of N robots moving in the space X. We further assume that the motion of the N robots can be modeled using the single integrator dynamics:

$$\dot{p}_i = u_i, \tag{7.3}$$

where $u_i \in \mathbb{R}^2$ is the control input of robot i. Define

$$\mathcal{J}(p_1, \ldots, p_N) = \sum_{i=1}^{N} \int_{V_i} \|x - p_i\|^2 \, dx \tag{7.4}$$

as the locational optimization function [32]. $\mathcal{V}(p_1, \ldots, p_N) = \{V_1, \ldots, V_N\}$ is called a Voronoi partition of the polygon X, whose i-th Voronoi cell V_i corresponding to robot i is defined as:

$$V_i = \{x \in X : \ \|x - p_i\| \leq \|x - p_j\| \ \forall j \neq i\}. \tag{7.5}$$

The integrand function in the expression of $\mathcal{J}(p_1, \ldots, p_N)$ is an increasing function of the Euclidean norm $\|\cdot\|$ and it describes the degradation of the sensing performances of the robots.

The Lloyd's descent algorithm is given by the following motion control law for robot i:

$$u_i = k_{\mathrm{p}}(\rho_i - p_i), \tag{7.6}$$

where $k_{\mathrm{p}} \in \mathbb{R}_+$, and $\rho_i \in \mathbb{R}^2$ is the centroid of the Voronoi cell V_i. It is derived by solving the following minimization problem using gradient descent:

$$\min_{p_1, \ldots, p_N} \mathcal{J}(p_1, \ldots, p_N). \tag{7.7}$$

In [32] the set of critical points of $\mathcal{J}(p_1, \ldots, p_N)$ has been demonstrated to be the set of centroidal Voronoi configurations on X where the location of each robot, p_i, is the centroid of the Voronoi cell V_i.

Constrained Locational Optimization

In order to describe the constrained motion of the robots on the wires, we define the following function that identifies the i-th wire:

$$g_i : x \in X \mapsto a_i^T x + b_i \in \mathbb{R}, \tag{7.8}$$

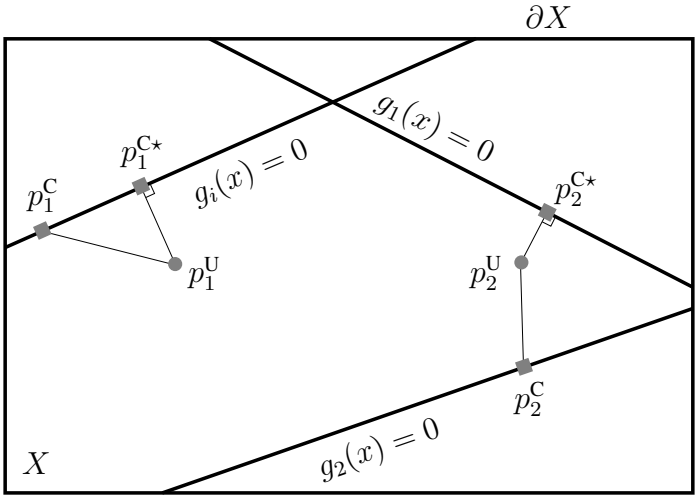


Figure 7.16: Robots' workspace X with the i-th wire defined by $g_i(x) = 0$. The points p_1^{U} and p_2^{U} (gray circles) are the solution of the unconstrained locational optimization problem; the points p_1^{C} and p_2^{C} belong to the set $\mathcal{G}$, while $p_1^{\mathrm{C}\star}$ and $p_2^{\mathrm{C}\star}$ (gray squares) are solutions of the minimization problem (7.11)

where $a_i \in \mathbb{R}^2$ and $b_i \in \mathbb{R}$ for $i = 1, \ldots, N_{\mathrm{w}}$, N_{w} being the number of wires present in the environment. The wires are then identified as the set:

$$\mathcal{G} = \{x \in X : \ g_i(x) = 0 \text{ for some } i \in \{1, \ldots, N_{\mathrm{w}}\}\} \cup \partial X, \tag{7.9}$$

where we assume that the boundary of X, denoted by ∂X, can also be traversed by the robots. The wire-traversing constraint can be formalized as:

$$p_i \in \mathcal{G} \quad \forall i \in \{1, \ldots, N\}. \tag{7.10}$$

Since the robots are constrained to move on wires, the problem we aim at solving is a constrained version of (7.7). The integrand function in (7.4) is non-decreasing therefore, in order to minimize (7.7), we want to minimize the distance from p_i while remaining on the wires. This can be done by solving the following minimization problem for each robot:

$$\min_{p_i^{\mathrm{C}} \in \mathcal{G}} \left\| p_i^{\mathrm{C}} - p_i^{\mathrm{U}} \right\|, \tag{7.11}$$

where $p_1^{\mathrm{U}}, \ldots, p_N^{\mathrm{U}}$ are solutions of (7.7). The superscripts U and C used in (7.11) distinguish the positions of the unconstrained robots from those of the wire-constrained robots (see Fig. 7.16).

The following theorem establishes the equivalence between the minimization problem (7.7) with the wire-traversing constraints (7.10) and the minimization problem (7.11). We say that two minimization problems are equivalent if they have a common local minimizer.

Theorem 7.1. *Given the locational optimization function $\mathcal{J}(p_1, \ldots, p_N)$ defined in* (7.4)*, the set $\mathcal{G}$ defined in* (7.9) *and $p_1^U, \ldots, p_N^U$, solutions of* (7.7)*, the following minimization problems are equivalent:*

$$\begin{aligned} &\underset{p_1,\ldots,p_N}{\text{minimize}}\ \mathcal{J}(p_1, \ldots, p_N) \\ &\text{subject to } p_1, \ldots, p_N \in \mathcal{G} \end{aligned} \tag{7.12}$$

$$\min_{p_i^C \in \mathcal{G}} \left\| p_i^C - p_i^U \right\|, \quad i = 1, \ldots, N. \tag{7.13}$$

Remark 7.2. *The equivalence established in Theorem 7.1 allows us to solve* (7.13) *instead of* (7.12)*. What this entails is that, instead of solving a constrained minimization problem, we can solve an unconstrained minimization problem and project its solution onto the constraints' set. Moreover, since we have the motion control law* (7.6) *that solves the minimization problem* (7.7)*, we can proceed by just projecting it onto the wires.*

Remark 7.2 points out the advantages of solving an unconstrained optimization problem followed by a projection of the solution onto the constraints set (as in (7.13)), over solving a constrained minimization problem like (7.12). However, the set of constraints, $\mathcal{G}$, is the union of affine sets; in fact, each wire is defined as the set $\{x \in X : \ g_i(x) = 0\} = \{x \in X : \ a_i^T x + b_i = 0\}, \ i \in \{1, \ldots, N_{\mathrm{w}}\}$. So, $\mathcal{G}$ is not convex. As such, the solution of (7.13)

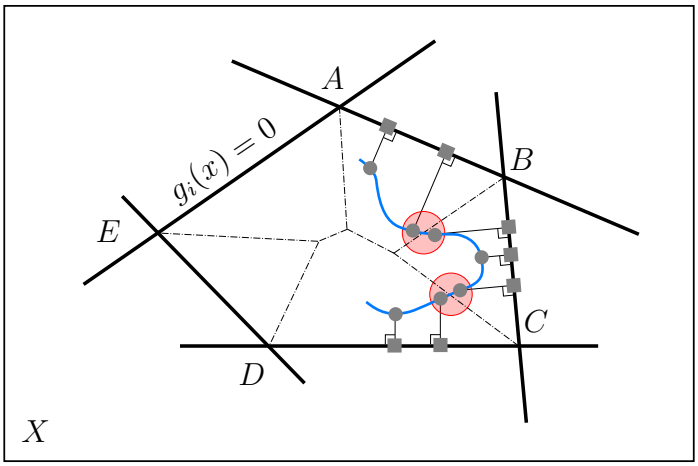


Figure 7.17: The areas shaded in red highlight regions where the operator that projects points p_i^{U} of the workspace X onto the closest wire is discontinuous. The wires are depicted as thick black lines, while the dash-dot lines represent the medial axis of the polygon $ABCDE$ formed by the wires. The gray circles are points of the blue trajectory that are mapped to the gray squares onto $\mathcal{G}$, the set of wires

requires optimization methods for non-convex problems. Even though the latter can be also solved efficiently (see [121]), the main objective of the constrained coverage control problem is that of generating a motion control law to be executed by the robots on the wires. A solution to (7.13), i.e. an *orthogonal projection* onto the wires (as proposed in [99]), does not fulfill this requirement. In Fig. 7.17 an example of a discontinuous projection on the wires is shown. In particular, the set of discontinuity points coincides with the set of points that have more than one closest point on the set $\mathcal{G}$. It follows that the *medial axes* (or the topological skeletons) of the polygons bounded by the wires, defined as the boundaries of the Voronoi diagrams of the edges of the polygons, are the sets of discontinuity points for the projected motion of the robots onto the wires.

7.4.2 From Projection to Mapping

In this section we describe a method to relax the conditions imposed by Theorem 7.1 in the interest of producing a continuous motion of the points p_i. This will be done by using a Continuous Onto Wires (COW) mapping to the set $\mathcal{G}$ defining the constraints.

Let

$$M : p \in X \subsetneq \mathbb{R}^2 \mapsto p^M \in \mathcal{G} \subsetneq \mathbb{R}^2 \tag{7.14}$$

be the operator that maps the robot workspace X to the set of wires $\mathcal{G}$, where p^M does not necessarily solve the program:

$$\min_{x \in \mathcal{G}} \|x - p\| . \tag{7.15}$$

In order to find a suitable expression for such a mapping, elements from complex analysis will be required. For this reason they are recalled in the following.

First of all, let us define the following isomorphism between $\mathbb{R}^2$ and $\mathbb{C}$:

$$\mathcal{I} : \begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{R}^2 \mapsto a + \iota b \in \mathbb{C}, \tag{7.16}$$

where ι is the imaginary unit. Moreover, in the following we will use $\Re(\cdot)$ to denote the operator that extracts the real part of a complex number. With abuse of notation, we denote with $\widetilde{X} = \mathcal{I}(X) \subsetneq \mathbb{C}$ the image of the robots' workspace X through the isomorphism (7.16). Let $\widetilde{p}_1, \ldots, \widetilde{p}_N \in \widetilde{X} \subsetneq \mathbb{C}$ be the robots' positions in the complex plane. Similarly, we can define $\widetilde{g}_i$ and $\widetilde{\mathcal{G}}$.

The following results will be used in the definition of a mapping (7.14).

Definition 7.3 (Conformal map [176])**.** *Let X_1 and X_2 be two open subsets of $\mathbb{C}$. A map $f : X_1 \to X_2$ is said to preserve angles if for every two differentiable curves $\gamma_1 : t \in [-\epsilon, \epsilon] \subsetneq \mathbb{R} \mapsto c \in \mathbb{C}$ and $\gamma_2 : t \in [-\epsilon, \epsilon] \subsetneq \mathbb{R} \mapsto c \in \mathbb{C}$, where $\gamma_1(0) = \gamma_2(0) = c^\star$, the angle formed by their tangents at $c^\star$ is the same as the angle formed by the tangents of the mapped curves $f \circ \gamma_1$ and $f \circ \gamma_2$ at $f(c^\star)$. A conformal map from X_1 to X_2 is a differentiable bijection that preserves angles.*

With this definition, we can now state the following theorem.

Theorem 7.4 (Riemann mapping theorem [177])**.** *Let $X \subsetneq \mathbb{C}$ be a simply connected region*

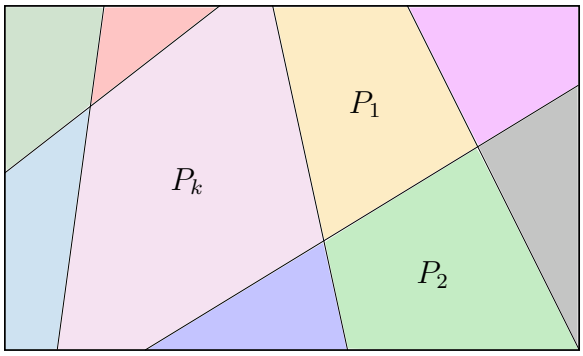


Figure 7.18: The polygonal tessellation induced by the wires consists of all closed and convex polygons P_k as they results from the intersection of half planes

of the complex plane, and let $x \in X$. Then, there is a unique conformal map $f : X \to \mathbb{D}$, where $\mathbb{D}$ is the unit disc, such that $f(x) = 0$ and $f'(x) = 0$.

From this theorem the following corollary can be proven.

Corollary 7.5. *Two simply connected regions of the complex plane, $X_1, X_2 \subsetneq \mathbb{C}$, are homeomorphic.*

Fact 7.6. *The wires defined by the set $\widetilde{\mathcal{G}}$ induce a polygonal tessellation of $\widetilde{X}$. The resulting polygonal areas P_k are closed and convex as they come from the intersection of half planes (see Fig. 7.18).*

Using the result of Corollary 7.5, we can construct a mapping from each of the polygons of the polygonal tessellation introduced in Fact 7.6 onto their boundaries, i.e. the wires. This can be realized as follows.

Let $P_k \subseteq \widetilde{X} \subsetneq \mathbb{C},\ k = 1, \ldots, K$ be the K polygons resulting from the polygonal tessellation defined by the wires $\widetilde{\mathcal{G}}$, and let $G_k \in \mathbb{C},\ k = 1, \ldots, K$ be their corresponding centroids. Let $l_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C},\ j = 1, \ldots, J$ denote the J sides of the polygon P_k, and $p_{kj}^{(1)} \in \mathbb{C}$ and $p_{kj}^{(2)} \in \mathbb{C}$ the two endpoints of the side l_{kj}. Note that $\widetilde{\mathcal{G}}$, the subset of the complex plane isomorphic to $\mathcal{G}$ through (7.16), indicated with abuse of notation by $\mathcal{I}(\mathcal{G})$, can be defined as:

$$\widetilde{\mathcal{G}} = \mathcal{I}(\mathcal{G}) = \bigcup_{k=1}^{K} \bigcup_{j=1}^{J} l_{kj}. \tag{7.17}$$

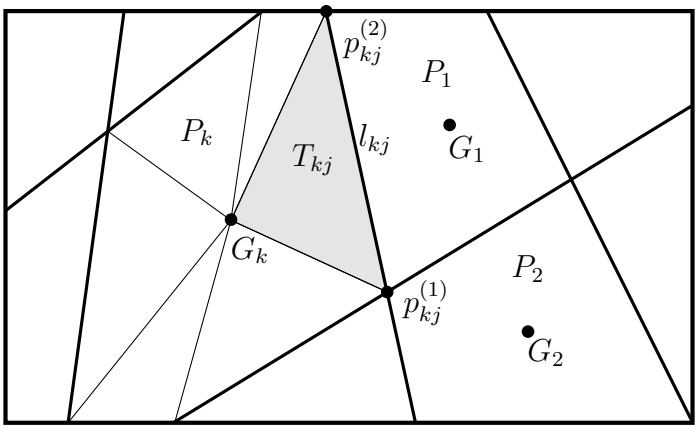


Figure 7.19: Quantities used in the formulation of a COW map $\widetilde{M}$

Let us define $T_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C}$ as the triangle with vertices $p_{kj}^{(1)}$, $p_{kj}^{(2)}$ and G_k. Fig. 7.19 shows all the quantities that have been just introduced.

With these premises, let us define the mappings m_{kj} as follows:

$$m_{kj} = \begin{cases} T_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \to l_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \\ \\ \widetilde{X} \setminus T_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \to \{0\} \subsetneq \mathbb{C}. \end{cases} \tag{7.18}$$

So, the mapping

$$\widetilde{M} : x \in \widetilde{X} \subsetneq \mathbb{C} \mapsto \sum_{k=1}^{K} \sum_{j=1}^{J} m_{kj}(x) \in \widetilde{\mathcal{G}} \subsetneq \mathbb{C} \tag{7.19}$$

transforms the robot workspace $\widetilde{X} \subsetneq \mathbb{C}$ to the set of wires $\widetilde{\mathcal{G}} \subsetneq \mathbb{C}$ in the complex plane.

Remark 7.7. *Let $x \in \mathbb{C}$ and $p \in \mathbb{R}^2$ such that $\mathcal{I}(p) = x$. By the properties of the isomorphism* (7.16) *and by the definitions of the mappings* (7.14) *and* (7.19)*, we characterize M by the following equations:*

$$\begin{aligned} \widetilde{M}(x) &= \mathcal{I}\left(M\left(\mathcal{I}^{-1}(x)\right)\right) \\ M(p) &= \mathcal{I}^{-1}\left(\widetilde{M}\left(\mathcal{I}(p)\right)\right). \end{aligned} \tag{7.20}$$

Now the expression of m_{kj} is left to define. In order to do so, let us introduce a particular

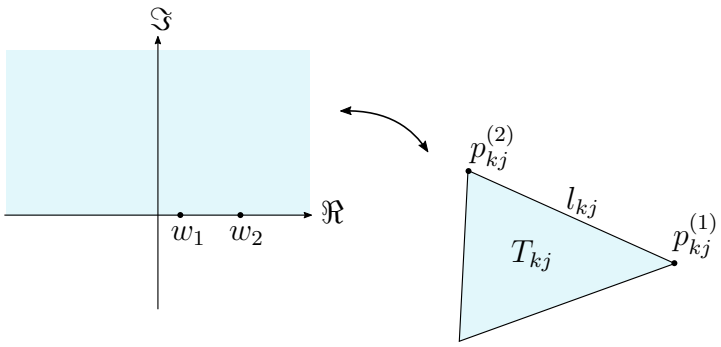


Figure 7.20: Schwar-Christoffel mapping between the upper-half plane $\mathbb{H}$ and the triangular region T_{kj} of the complex plane. The prevertices w_1 and w_2 are mapped to the vertices of the triangle $p_{kj}^{(1)}$ and $p_{kj}^{(2)}$, respectively

conformal mapping.

Definition 7.8 (Schwarz-Christoffel mapping [178])**.** *A Schwarz-Christoffel mapping is a conformal mapping from the upper half-plane* $\mathbb{H} = \{x \in \mathbb{C} : \ \Re(x) \geq 0\}$ *(the canonical domain) to a region* $\mathbb{P}$ *of the complex plane bounded by a polygon (the physical domain). Its expression is given by:*

$$f : x \in \mathbb{H} \mapsto x_0 + c \int_{x_0}^{x} \prod_{j=1}^{J-1} (\chi - w_j)^{\alpha_j - 1} \, d\chi \in \mathbb{P}, \tag{7.21}$$

where $x_0, c \in \mathbb{C}$ *are two constants that translate, rotate and scale the polygon that bounds* $\mathbb{P}$*,* J *is the number of sides of the polygon,* α_j *is the interior angle at the* j*-th vertex of the polygon and* $w_j \in \mathbb{R}, \ j = 1, \ldots, J-1$ *are called the prevertices and have the property of being mapped to the vertices of the polygon. In case of triangular domains, i.e.* $J = 3$*, the prevertices can be arbitrarily set to any location on the real axis.*

Definition 7.9. *Let*

$$f_{kj} : \mathbb{H} \to T_{kj} \tag{7.22}$$

be the Schwarz-Christoffel mapping between the upper-half plane and the triangular region T_{kj} *defined above. The real axis is mapped to the boundary of* T_{kj} *and the prevertices* w_1 *and* w_2 *are mapped to* $p_{kj}^{(1)}$ *and* $p_{kj}^{(2)}$*, respectively (see Fig. 7.20). Because of the fact that*

for triangular physical domains the prevertices can be arbitrarily chosen on the real axis, we assume that their value does not depend on the indexes k and j.

The following theorem defines a COW map, i.e. a continuous and onto mapping, from the robots' workspace $\widetilde{X}$ to the set $\widetilde{\mathcal{G}}$ defined in (7.17).

Theorem 7.10. *Let f_{kj} be the Schwarz-Christoffel mapping defined by* (7.22)*, w_1 and w_2 the prevertices of the mapping f_{kj}; let $p_{kj}^{(1)}$, $p_{kj}^{(2)}$ and G_k the vertices of the triangular boundary of the region T_{kj}. The mapping $\widetilde{M}$ defined in* (7.19) *where m_{kj} is given by the following onto mapping:*

$$m_{kj}(x) = \begin{cases} p_{kj}^{(1)} & \text{if } \Re\left(f_{kj}^{-1}(x)\right) \leq w_1 \\ f_{kj}\left(\Re\left(f_{kj}^{-1}(x)\right)\right) & \text{if } w_1 < \Re\left(f_{kj}^{-1}(x)\right) < w_2 \\ p_{kj}^{(2)} & \text{if } \Re\left(f_{kj}^{-1}(x)\right) \geq w_2, \end{cases} \tag{7.23}$$

defines a continuous mapping from the robots' workspace $\widetilde{X} \setminus \left(\bigcup_{k=1}^{K} \{G_k\}\right) \subsetneq \mathbb{C}$ to the set $\widetilde{\mathcal{G}} \subsetneq \mathbb{C}$ defined in (7.17)*.*

Remark 7.11. *Solving an optimization problem such as* (A.41) *performs a projection operation in the physical domain $\mathbb{P}$ of a Schwarz-Christoffel mapping. The same result is obtained in the canonical domain $\mathbb{H}$ by means of the operator $\Re(\cdot)$.*

7.4.3 Motion Control

The COW map $\widetilde{M}$ defined by Theorem 7.10 allows the direct derivation of a motion control law to be executed by each of the robots on the wires. The resulting motion is continuous and inherently takes into account the constraints defined by the wires.

Mapped Gradient Descent

As the motion control law derived in this section are to be applied to all the robots without distinction, the subscript i will be dropped from now on.

Note also that all the quantities used in the following are complex numbers, subsets of the complex plane and complex-valued functions of complex variables. Due to the isomorphism (7.16), the formulation in $\mathbb{C}$ and that in $\mathbb{R}^2$, even though formally different, are substantially equivalent.

The COW mapping derived in Theorem 7.10 that transforms the domain $\widetilde{X}$ into $\widetilde{G}$ is denoted by:

$$x^{\widetilde{M}} = \widetilde{M}(x), \tag{7.24}$$

where $\widetilde{M}$ is defined in (7.19). Differentiating (7.24), one obtains:

$$\dot{x}^{\widetilde{M}} = \frac{\partial \widetilde{M}}{\partial x}\dot{x} = \frac{\partial \widetilde{M}}{\partial x}k_{\mathrm{p}}(\widetilde{\rho} - x), \tag{7.25}$$

where $\widetilde{\rho} = \mathcal{I}(\rho)$ and $\dot{x} = k_{\mathrm{p}}(\widetilde{\rho} - x)$ comes from (7.6).

As far as the expression of $\frac{\partial \widetilde{M}}{\partial x}$ is concerned, starting from (7.19), it can be written as:

$$\frac{\partial \widetilde{M}}{\partial x} = \sum_{k=1}^{K}\sum_{j=1}^{J}\frac{\partial m_{kj}}{\partial x}. \tag{7.26}$$

It has to be noticed that, since the operator $\Re(\cdot)$ is not differentiable, $\frac{\partial m_{kj}}{\partial x}$ is not well-defined. However, since we are interested in deriving a motion control law for the robots on the wires, we actually need only the directional derivative of $\widetilde{M}$ and m_{kj} along the wires, denoted by $\partial_{\widetilde{\mathcal{G}}}\widetilde{M}$ and $\partial_{\widetilde{\mathcal{G}}}m_{kj}$, respectively. Consequently, we need the directional derivative of $\Re$ only along the real axis. The latter is well-defined and it is identically equal to 1. Thus,

we can write:

$$\partial_{\widetilde{\mathcal{G}}} m_{kj} = \begin{cases} 0 & \text{if } \Re\left(f_{kj}^{-1}(x)\right) \leq w_1 \\ m'_{kj} & \text{if } w_1 < \Re\left(f_{kj}^{-1}(x)\right) < w_2 \\ 0 & \text{if } \Re\left(f_{kj}^{-1}(x)\right) \geq w_2 \end{cases}, \tag{7.27}$$

where m'_{kj} is given by

$$\begin{aligned} m'_{kj} &= \left(\frac{\partial f_{kj}}{\partial x} \circ \Re\left(f_{kj}^{-1}(x)\right)\right)\left(\frac{\partial \Re}{\partial x} \circ f_{kj}^{-1}(x)\right)\frac{\partial f_{kj}^{-1}}{\partial x} = \\ &= \left(\frac{\partial f_{kj}}{\partial x} \circ \Re\left(f_{kj}^{-1}(x)\right)\right)\left(1 \circ f_{kj}^{-1}(x)\right)\frac{\partial f_{kj}^{-1}}{\partial x} \\ &= \left(\frac{\partial f_{kj}}{\partial x} \circ \Re\left(f_{kj}^{-1}(x)\right)\right)\frac{1}{\frac{\partial f_{kj}}{\partial x}}, \end{aligned} \tag{7.28}$$

where

$$\frac{\partial f_{kj}}{\partial x} = \prod_{i=1}^{2}(x - w_i)^{\alpha_i - 1}, \tag{7.29}$$

and all the quantities used here and specified in Definition 7.8 are specific for the triangular region T_{kj}.

The theorem stated below follows directly from the derivation of (7.25).

Theorem 7.12. *Let $x_i = \mathcal{I}(p_i),\ i \in \{1, \ldots, N\}$ be the positions of N robots expressed as points of the complex plane $\mathbb{C}$. Let*

$$\widetilde{\mathcal{J}}(x_1, \ldots, x_N) = \mathcal{I}\left(\mathcal{J}\left(\mathcal{I}^{-1}(x_1), \ldots, \mathcal{I}^{-1}(x_N)\right)\right) \tag{7.30}$$

be the locational cost defined based on (7.4)*. The motion control law*

$$\dot{x}^{\widetilde{M}} = \partial_{\widetilde{\mathcal{G}}}\widetilde{M}\ \dot{x}_{\widetilde{\mathcal{G}}}, \tag{7.31}$$

where the subscript $\widetilde{\mathcal{G}}$ indicates quantities mapped onto the wires, applied to robots whose motion is constrained to be in the set $\widetilde{\mathcal{G}} = \mathcal{I}(G) \subsetneq \mathbb{C}$ defined in (7.17)*, solves the con-*

strained optimization problem:

$$\begin{aligned} &\underset{x_1,\ldots,x_N}{\text{minimize}}\ \widetilde{\mathcal{J}}(x_1,\ldots,x_N) \\ &\text{subject to } \widetilde{M}(x_i) = x_i \quad \forall i \in \{1,\ldots,N\}. \end{aligned} \tag{7.32}$$

7.4.4 From Mapping to Projection

In Section 7.4.2, in order to derive a motion control law that ensures a continuous motion of the robots on the wires, we relaxed the constraints imposed in the optimization problems defined in Theorem 7.1. Now we propose an algorithm that restores those constraints and ensures the best coverage quality that is achievable when the robots are constrained to move on wires.

Let us start stating the following result.

Fact 7.13. *Employing the Euclidean distance as a metric to measure the distance* $\mathrm{d}(x, l)$ *between a point* x *and a segment* l*, required for the evaluation of Voronoi partitions, the Voronoi cells determined by the edges of a convex polygon are convex polygons themselves. This also means that the medial axis of a convex polygon consists of all straight lines.*

This allows the definition of conformal mappings, similar to f_{kj} introduced in (7.22), between the upper-half plane $\mathbb{H}$ and the convex and polygonal Voronoi cells related to polygon P_k.

With the objective of mapping the point $x \in X$ to its closest wire, each triangular region T_{kj} of polygon P_k is continuously deformed to its corresponding Voronoi cell

$$V_{kj} = \{x \in P_k : \ \mathrm{d}(x, l_{kj}) \leq \mathrm{d}(x, l_{k\bar{j}}) \quad \forall \bar{j} \neq j\}, \tag{7.33}$$

with which it shares the side l_{kj}. See Fig. 7.21a to Fig. 7.21e.

The optimization problem (7.32) is solved using gradient descent that results in the control law (7.25) applied to each robot. Let us define τ_f as the time instant at which the

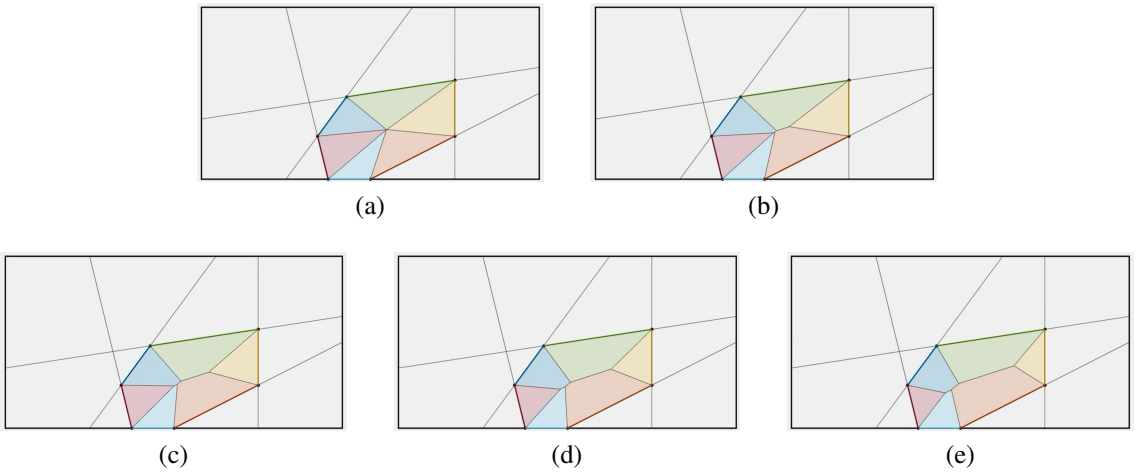


Figure 7.21: Continuous deformation of the triangular regions T_{kj} in 7.21a to the corresponding Voronoi cells V_{kj} in 7.21e

velocities given by $k_{\mathrm{p}}(\widetilde{\rho} - x)$ become sufficiently small for all the robots. For a given time interval $[\tau_f, \tau_f + \tau]$, let

$$\mathcal{D}_{kj} : t \in [\tau_f, \tau_f + \tau] \mapsto T_{kj}^{(t)}, \tag{7.34}$$

with

$$\mathcal{D}_{kj}(\tau_f) = T_{kj}^{(\tau_f)} = T_{kj} \tag{7.35}$$

$$\mathcal{D}_{kj}(\tau_f + \tau) = T_{kj}^{(\tau_f + \tau)} = V_{kj}, \tag{7.36}$$

be the deformation operator that transforms the region T_{kj} into the Voronoi cell V_{kj} during the time interval $[\tau_f, \tau_f + \tau]$.

Following the definition in (7.19), let us define the following COW map at time $t \in [\tau_f,\ \tau_f + \tau]$:

$$\widetilde{M}^{(t)} : x \in \widetilde{X} \subsetneq \mathbb{C} \mapsto \sum_{k=1}^{K} \sum_{j=1}^{J} m_{kj}^{(t)}(x) \in \widetilde{\mathcal{G}} \subsetneq \mathbb{C}, \tag{7.37}$$

where

$$m_{kj}^{(t)} = \begin{cases} T_{kj}^{(t)} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \to l_{kj} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \\ \widetilde{X} \setminus T_{kj}^{(t)} \subsetneq \widetilde{X} \subsetneq \mathbb{C} \to \{0\} \subsetneq \mathbb{C}. \end{cases} \tag{7.38}$$

The velocity $\dot{x}^{\widetilde{M}^{(t)}}$ is evaluated using (7.31) where $\widetilde{M}^{(t)}$ is used in place of $\widetilde{M}$.

Remark 7.14. *By definition of Voronoi cell* (7.33)*, once the transformation* (7.34) *is completed, i.e.* $t = \tau_f + \tau$*, the COW map* $\widetilde{M}^{(\tau_f+\tau)}$ *transforms each point* x *to a point belonging to* $\widetilde{\mathcal{G}}$ *on the closest wire. Therefore, the robots can execute gradient descent on the wire on which they are at time* $t \geq \tau_f + \tau$ *in order to get to the positions that minimize* (7.30)*.*

Algorithm 8 Continuous Constrained Coverage Control

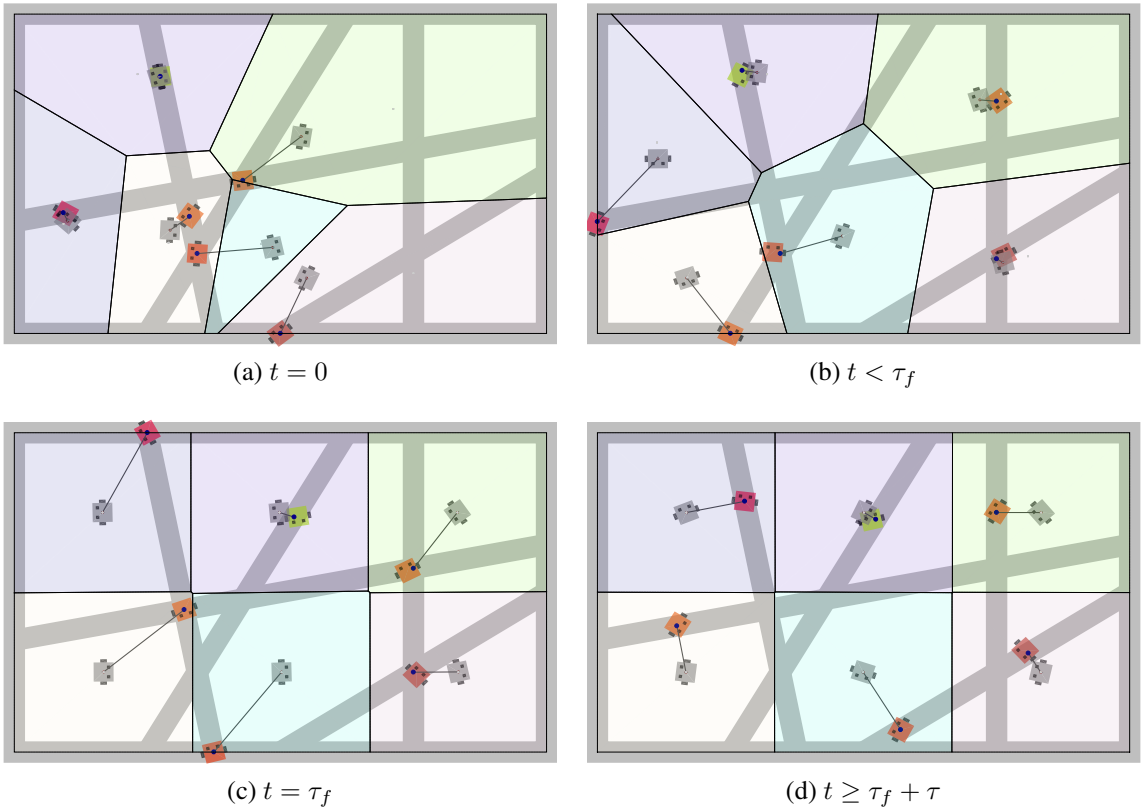
Require: x (robot initial position), $\widetilde{M}$, $\widetilde{M}^{(t)}$, $\mathcal{D}_{kj}$
Ensure: Continuous Constrained Coverage Control
  while $|k_{\mathrm{p}}(\widetilde{\rho} - x)| \geq \epsilon$ **do**
    $\widetilde{u} \leftarrow$ compute $\dot{x}^{\widetilde{M}}$
    execute $\widetilde{u}$
  end while
  for $t = \tau_f \rightarrow \tau_f + \tau$ **do**
    for all adjacent regions T_{kj} **do**
      $T_{kj}^{(t)} \leftarrow \mathcal{D}_{kj}(t)$
      $m_{kj}^{(t)} \leftarrow$ compute $m_{kj}^{(t)}$
    end for
    $\widetilde{M}^{(t)} \leftarrow \sum_{k=1}^{K} \sum_{j=1}^{J} m_{kj}^{(t)}\,(x)$
    $\widetilde{u} \leftarrow$ compute $\dot{x}^{\widetilde{M}^{(t)}}$
  end for
  execute gradient descent on the current wire

Algorithm 8 outlines the motion control strategy executed by each robot on the wires. The resulting behavior is depicted in Fig. 7.22a to Fig. 7.22d.

Based on the derivation of Algorithm 8, we can state the following theorem.

Theorem 7.15. *Algorithm 8 solves the following constrained optimization problem:*

$$\begin{aligned} &\underset{x_1,\ldots,x_N}{\text{minimize}}\ \widetilde{\mathcal{J}}(x_1,\ldots,x_N) \\ &\text{subject to } x_1,\ldots,x_N \in \widetilde{\mathcal{G}}. \end{aligned} \tag{7.39}$$



(a) $t = 0$ (b) $t < \tau_f$

(c) $t = \tau_f$ (d) $t \geq \tau_f + \tau$

Figure 7.22: Motion of the robots under coverage control constrained by the wires resulting by the application of Algorithm 8. The thick lines are the wires that constrain the motion of the colored robot. The gray robots move according to the control law derived from the minimization of the locational cost (7.4). The colored area represent the Voronoi cells (7.5) related to the gray robots. Each gray robot is linked to the corresponding colored robot on the wires to which it is mapped

7.4.5 Experimental Results of Simulated Robots on Wire Meshes

The algorithm to execute the continuous constrained coverage control described in Algorithm 8 has been deployed on a swarm of ground mobile robots on the Robotarium, a remotely accessible swarm robotics testbed [52], where the robots have been artificially constrained to move on wires.

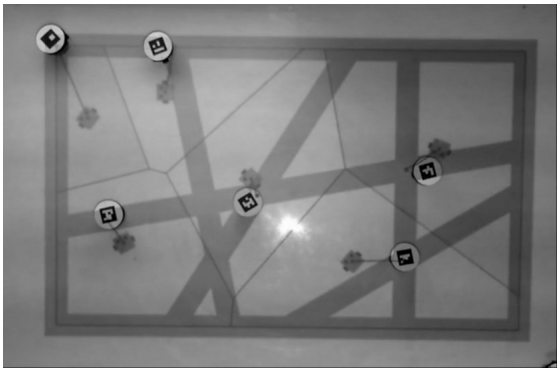
The algorithm has been implemented in MATLAB and submitted through the Robotarium web interface[1] in order to be executed on the real robots. Fig. 7.23a to Fig. 7.23e show images taken from the video recorded during the experiments. An overhead projector visualizes the wires on which the robots are constrained (thick gray lines). The virtual robots projected on the testbed are linked to the real ones by means of the mappings $\widetilde{M}$ and $\widetilde{M}^{(t)}$ and they move in order to minimize the unconstrained locational cost (7.4). In Fig. 7.23a the robots are initialized to random positions on the wires. In Fig. 7.23b and Fig. 7.23c the robots execute the control law (7.31) moving on the wires until the velocities of the projected robots are below a minimum threshold (Fig. 7.23d). At this point the deformation (7.34) is performed, and executing gradient descent on the wires on which the robots are at time $\tau_f + \tau$ brings them to the final positions that solve the constrained locational optimization problem (7.39).

7.5 Motion Control on Single Wire

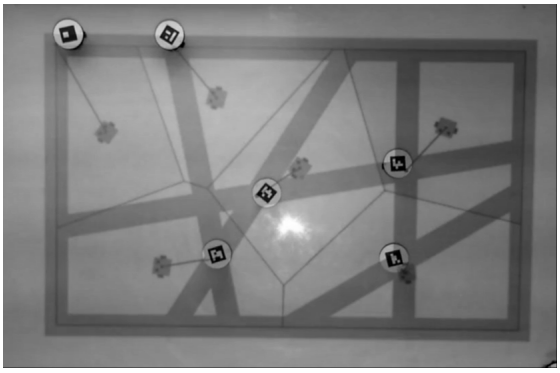
In the previous section, we presented a control strategy to allow robots moving on a mesh of wires to spread optimally in the environment in which they are deployed in order to monitor it. This method is suitable for the multi-body SlothBot presented in Section 7.1, which is able to switch wire branches and therefore successfully locomote on a mesh of wires.

In this section, we consider a constrained coverage control problem for a team of mobile robots which are asked to provide sensor coverage over a two-dimensional domain, while

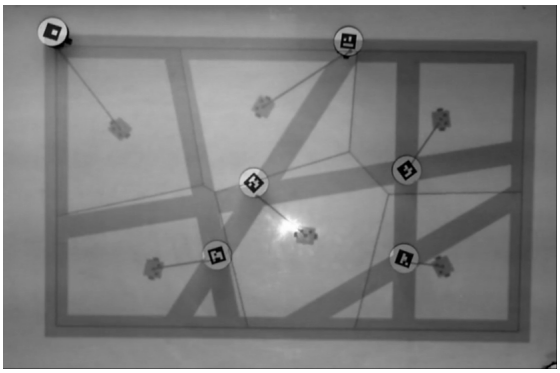
[1] http://www.robotarium.org/



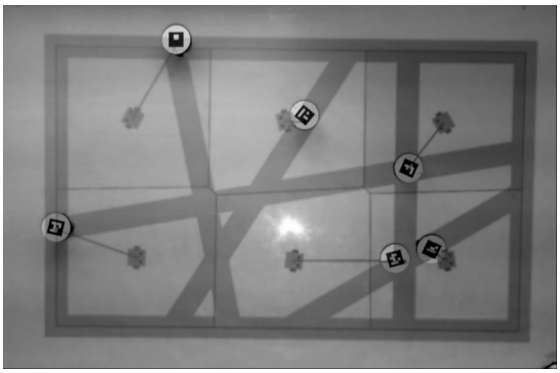
(a) Beginning of the experiment ($t = 0$ seconds)



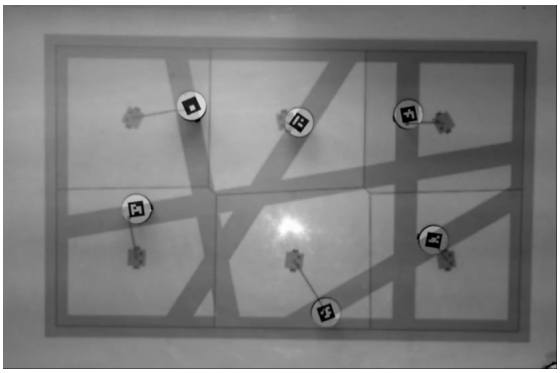
(b) $t = 23$ seconds



(c) $t = 41$ seconds



(d) $t = \tau_f = 80$ seconds



(e) $t = 99$ seconds

Figure 7.23: Algorithm 8 is deployed on a team of robots on the Robotarium. An overhead projector is visualizing information related to the experiment: the thick lines are the wires on which the real robots are constrained to move, the motion of the the projected robots is determined by solving the minimization problem (7.7), the thin lines are the boundary of the Voronoi cells (7.5). As in Fig. 7.22, the projected robots are linked with the robots on the wire to which they are mapped

being constrained to only move on a curve. This, in turn, is a motion control strategy tailored to the single-body SlothBot design, described in Section 7.1, which only moves on a single wire, without the possibility of switching wire branch.

This constrained coverage control problem is tackled by defining a modification to the locational cost for unconstrained coverage control (7.4), which incorporates the constraints. Moreover, a convex relaxation is proposed which allows us to efficiently minimize a convex approximation of the cost using a decentralized strategy. The resulting algorithm is implemented on a team of mobile robots whose motion is artificially constrained to a curve.

In this section, we introduce a coverage control algorithm that addresses the problem of how to cover a closed and convex planar domain, $\mathcal{D} \subset \mathbb{R}^2$, with a team of N agents constrained to move on a smooth curve[2]. We consider the coverage objective of the team with respect to the density function in the two-dimensional environment as in [32], but reformulate the locational cost to include the constraint that confines the robots to move on a one-dimensional manifold. A convex relaxation is then introduced such that the problem can be solved efficiently in a decentralized fashion.

In Section 7.5.2 we present the formulation of the coverage problem for planar robots that are constrained to move on curves. A decentralized algorithm to minimize a locational cost is proposed, which leverages a convex relaxation of the cost. Section 7.5.5 features the application of the derived algorithm to a team of ground mobile robots tasked with environment surveillance.

7.5.1 Constrained Locational Optimization

The employment of wire-traversing robots for environmental monitoring applications can be advantageous in terms of motion planning and control as well as energy requirements. In order to perform monitoring tasks while being constrained to move on wires, a constrained locational optimization problem can be defined.

[2] X, used in the previous section to denote $\mathcal{D}$, is utilized in this section to represent the ensemble state of the robots constrained to move on a curve, as formally defined later

In this section, we illustrate the problems that may arise while trying to minimize the cost in (7.4) in the case of robots constrained to move on a curve defined in the domain $\mathcal{D}$.

To this end, let

$$c : s \in I \subset \mathbb{R} \mapsto p \in \mathcal{D} \subset \mathbb{R}^2 \tag{7.40}$$

be an arc-length parameterized, simple, regular and $C^2(I^\circ)$ curve, i.e. a curve for which $\frac{dc}{ds} \neq 0 \; \forall s \in I$, and twice continuously differentiable in the interior of the interval I. We define the set

$$\mathscr{C} = \{c(s) : \; s \in I\} \subset \mathcal{D} \subset \mathbb{R}^2 \tag{7.41}$$

as the set of points belonging to the curve. We will refer to c and $\mathscr{C}$ as the *curve*, making the distinction clear when required. Thus, the constrained locational optimization problem can be expressed as follows:

$$\begin{aligned} &\min_P \; \mathcal{H}(P) \\ &\text{s.t. } p_i \in \mathscr{C} \;\; \forall i \in \mathcal{N}, \end{aligned} \tag{7.42}$$

or,

$$\min_S \; \mathcal{H}_c(S), \tag{7.43}$$

where $\mathcal{H}_c(S) = (\mathcal{H} \circ c)(T) = \sum_{i \in \mathcal{N}} \int_{\mathcal{V}_i} \|c(s_i) - q\|^2 \phi(q) \, \mathrm{d}q$ and $S = \{s_1, \ldots, s_N\} \subset I \times \ldots \times I$ s.t. $c(s_i) = p_i \; \forall i \in \mathcal{N}$. Assuming the robots can move according to single integrator dynamics on the curve, solving (7.43) using gradient descent leads to the following decentralized control law:

$$\dot{s}_i = -\frac{\partial \mathcal{H}_c}{\partial s_i}(T) = \kappa \frac{dc}{ds}^T (s_i) \, (\rho_i - c(s_i)) \in \mathbb{R}, \tag{7.44}$$

$\kappa > 0$ a proportional constant, where chain rule has been leveraged to express $\frac{\partial \mathcal{H}_c}{\partial s_i}$ in terms of $\frac{\partial \mathcal{H}}{\partial p_i}$.

Then, the evolution in time of the cost $\mathcal{H}_c$ is given by:

$$\begin{aligned}\dot{\mathcal{H}}_c &= \frac{\partial \mathcal{H}_c}{\partial s_i}\dot{s}_i = -\kappa \left|\frac{\partial \mathcal{H}_c}{\partial s}\right|^2 \\ &= -\kappa \left|\frac{dc}{ds}^T (s_i)\,(\rho_i - c(s_i))\right|^2 \leq 0.\end{aligned} \tag{7.45}$$

The expression in (7.45) vanishes in one of the following cases:

(i) $c(s_i) = \rho_i$,

(ii) $\dfrac{dc}{ds}(s_i) = 0$, or

(iii) $\rho_i - c(s_i) \perp \dfrac{dc}{ds}(s_i)$.

In the first case, the position of robot i coincides with the centroid of its Voronoi cell $\mathcal{V}_i$. The second case, by the regularity assumption on the curve c, by definition, can never occur. The third case, however, tells us that the speed of robot i on the curve is zero when the curve is orthogonal to the line segment joining the position of robot i and the centroid of its Voronoi cell. This condition, typical of the projected gradient descent algorithm (7.44), highlights the problem suffered by the constrained locational cost optimization. Figure 7.24 schematically depicts an example of what has been discussed in this section.

In the next section, we derive an algorithm that overcomes this problem by solving a convex approximation of the constrained optimization problem (7.43).

7.5.2 Constrained Coverage Control

The discussion in Section 7.5.1 shows that, due to the non-convexity of the cost and the constraints, a gradient descent policy, albeit decentralized, can drive the robots to a stationary point corresponding to a high locational cost. In this section, we show that the non-convexity of the problem is caused by the shape of the curve and that not all curves will

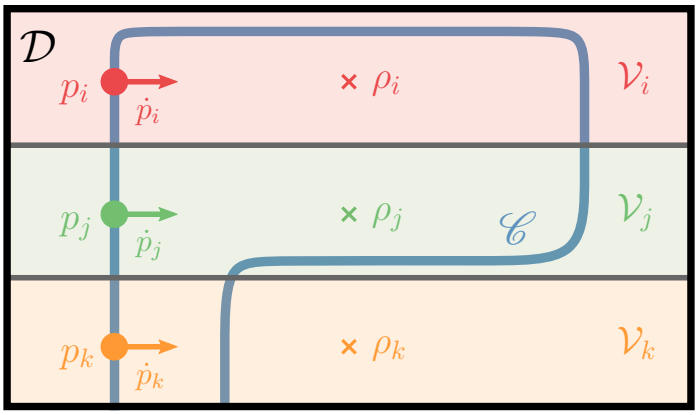


Figure 7.24: Poor spatial allocation of the robots obtained by executing projected gradient descent to minimize the locational cost (7.4). The robots at positions p_i, p_j and p_k are controlled to go to the centroids of the corresponding Voronoi cells, ρ_i, ρ_j and ρ_k. Since the tangent to the curve is orthogonal to the velocity vector (depicted as colored arrows), the robots have reached a local minimum of (7.43).

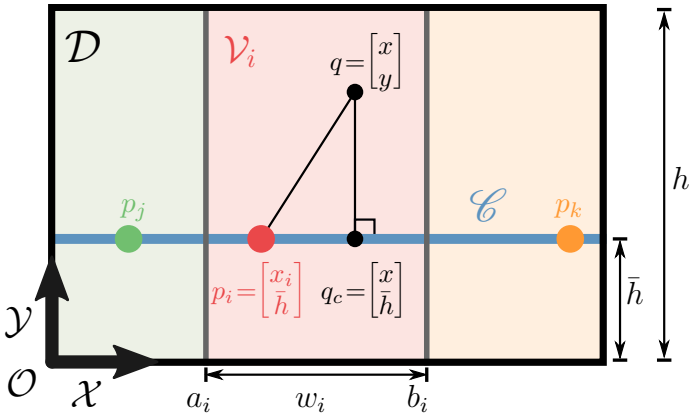


Figure 7.25: Example of coverage on a straight line. The depicted quantities are used in (7.47) to derive the one-dimensional locational cost equivalent to (7.4).

result in non-convex problems. Indeed, under certain circumstances, (7.43) can actually be a convex problem. This gives insight on how to formulate the locational optimization problem for robots constrained to move on generic curves, as will be shown in the next section.

Motivation

Let us consider the scenario depicted in Fig. 7.25. The domain $\mathcal{D}$ is a rectangle and the curve c is a straight line parallel to one of its sides. Assuming a constant density function

$\phi(q) = 1 \; \forall q \in \mathcal{D}$, we can rewrite the cost (7.4) as follows:

$$\begin{aligned}\mathcal{H}(P) &= \sum_{i\in\mathcal{N}} \int_{\mathcal{V}_i(P)} \|p_i - q\|^2 \, \mathrm{d}q \\ &= \underbrace{\sum_{i\in\mathcal{N}} \int_{\mathcal{V}_i(P)} \|p_i - q_c\|^2 \, \mathrm{d}q}_{①} + \underbrace{\sum_{i\in\mathcal{N}} \int_{\mathcal{V}_i(P)} \|q - q_c\|^2 \, \mathrm{d}q}_{②},\end{aligned} \tag{7.46}$$

where $p_i = [x_i, \bar{h}]^T$, $q = [x, y]^T$ and $q_c = [x, \bar{h}]^T$ is the projection of q onto the curve $\mathscr{C}$. Solving the integrals ① and ②, yields:

$$\begin{aligned}① &= \sum_{i\in\mathcal{N}} \int_0^h \int_{a_i}^{b_i} |x_i - x|^2 \, \mathrm{d}x \, \mathrm{d}y = h \sum_{i\in\mathcal{N}} \int_{a_i}^{b_i} |x_i - x|^2 \, \mathrm{d}x \\ &= h\mathcal{H}_c(X), \\ ② &= \sum_{i\in\mathcal{N}} \int_0^h \int_{a_i}^{b_i} |y - \bar{h}|^2 \, \mathrm{d}x \, \mathrm{d}y = \sum_{i\in\mathcal{N}} w_i \int_0^h |y - \bar{h}|^2 \, \mathrm{d}y \\ &= \sum_{i\in\mathcal{N}} w_i \left(\frac{\left(h - \bar{h}\right)^3}{3} - \frac{\bar{h}^3}{3} \right) = C,\end{aligned} \tag{7.47}$$

where a_i and b_i are the x coordinates of the vertical segments of the boundaries of the Voronoi cell $\mathcal{V}_i$ induced by the positions of the robots, $w_i = |b_i - a_i|$, $\bar{h}$ is the y position of the curve $\mathscr{C}$ (see Fig. 7.25), $X = \{x_1, \ldots, x_N\}$, and C is a constant. Thus, $\mathcal{H}(P) = h\mathcal{H}_c(X) + C$, and, therefore, $\min_P \; \mathcal{H}(P) \;\sim\; \min_X \; \mathcal{H}_c(X)$, i.e. the two minimization problems are equivalent in the following sense:

$$\begin{aligned}&\left\{ \begin{bmatrix} x_1^* \\ h \end{bmatrix}, \ldots, \begin{bmatrix} x_N^* \\ h \end{bmatrix} \right\} = \arg\min_P \; \mathcal{H}(P) \\ &\Leftrightarrow \\ &\{x_1^*, \ldots, x_N^*\} = \arg\min_X \; \mathcal{H}_c(X).\end{aligned} \tag{7.48}$$

In [179], it is shown that in case the domain $\mathcal{D}$ is one-dimensional and the density func-

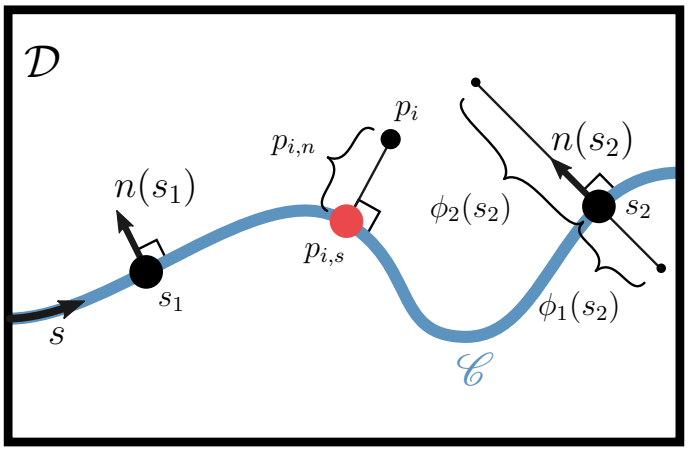


Figure 7.26: Reference system $\{s, n(s)\}$ and other quantities used in the derivation of the constrained locational cost $\mathcal{H}_c(P)$.

tion ϕ is log-concave, conditions satisfied by $\mathcal{H}_c(X)$ (characterized by a constant density function), the minimization of the locational cost (7.4) is a convex problem. Therefore, gradient descent methods can be used to synthesize a decentralized control law that will drive the robots to a configuration corresponding to the global minimum of (7.4).

In the following section, we will extend the construction introduced in this section to curves that are not necessarily straight lines.

Formulation for Generic Curves

Similarly to what has been done in Section 7.5.2, in this section we introduce a system of coordinates, using which we evaluate the integrals that show up in the locational cost $\mathcal{H}(P)$. Figure 7.26 depicts the domain $\mathcal{D}$ with a curve c, parameterized using the arc length $s \in I_s \subset \mathbb{R}$. The system of coordinates $\{s, n(s)\}$, in which the points in the domain will be expressed, consists of the curvilinear abscissa, s (analogous to x in Section 7.5.2), and the normal to the curve at s, $n(s)$ (analogous to y in Section 7.5.2). Figure 7.26 shows an example of such coordinates for a generic point p_i of the domain.

Proceeding as in Section 7.5.2, using the system of coordinates just defined, we can

write:

$$\begin{aligned}\mathcal{H}(P) &= \sum_{i\in\mathcal{N}} \int_{\mathcal{V}_i(P)} \|p_i - q\|^2 \,\mathrm{d}q \\ &\leq \sum_{i\in\mathcal{N}} \int_{S_i} \int_{-\phi_1(s)}^{\phi_2(s)} \left(|p_{i,s} - s|^2 + |p_{i,n} - n|^2\right) \,\mathrm{d}s \,\mathrm{d}n \\ &= \mathcal{H}_c(P),\end{aligned} \tag{7.49}$$

where $p_i = [p_{i,s}, p_{i,n}]^T$, $S_i = \{s \in I_s : \; c(s) \in \mathcal{V}_i\}$ is a union of closed intervals, corresponding to the curve segments that lie in the Voronoi cell $\mathcal{V}_i$, and the functions ϕ_1 and ϕ_2, whose values at s_2 are shown in Fig. 7.26, will be defined in the following. $\mathcal{H}_c(P)$ in (7.49) is entirely analogous to (7.47) and, as a matter of fact, its natural generalization to non-straight curves.

Since the goal is that of minimizing the upper bound of $\mathcal{H}(P)$ in (7.49), we make the following assumption on the curve c in order to bound the same integral from below too, and have a well-defined optimization problem.

Assumption 7.16. *The curve c is such that:*

$$\begin{cases} k_1 = \max\limits_{u,v\in\mathscr{C}} \left(\widehat{uv}^2 - \|u - v\|^2\right) < \infty \\ k_2 = \max\limits_{\substack{u\in\mathscr{C} \\ v\in\mathcal{D}}} \left(\|u - v\|_c^2 - \|u - v\|^2\right) < \infty, \end{cases} \tag{7.50}$$

where $\widehat{uv}$ denotes the arc length between the two points u and v on the curve, whereas $\|u - v\|_c^2 \triangleq \widehat{uv_c}^2 + \|v - v_c\|^2$, v_c being the point on the curve closest to v. The symbol $\|\cdot\|_c$ is an abuse of notation, since it does not define a norm (nor even a metric) as triangle inequality does not hold.

Note that $k_2 \geq k_1 \; \forall u \in \mathscr{C}, \; \forall v \in \mathcal{D}$.

Under Assumption 7.16, the following two inequalities hold:

$$\mathcal{H}_c(P) \geq \mathcal{H}(P) + k_1|\mathcal{D}| \quad \text{and} \quad \mathcal{H}_c(P) \leq \mathcal{H}(P) + k_2|\mathcal{D}|$$

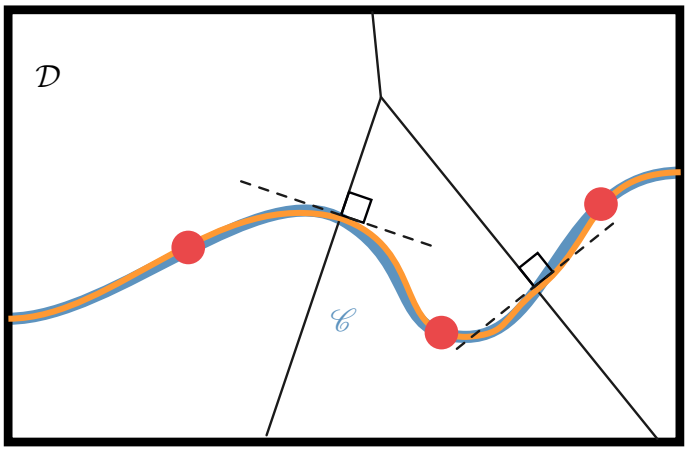


Figure 7.27: An example of deformation of the curve c in order to fulfill Assumption 7.17. The actual curve is depicted in blue, while its deformation is shown in green. The thin black lines are the Voronoi cells corresponding to the robots at the locations specified by the red dots. The dashed lines are the tangents to the green curve at the intersection points with the Voronoi cells' boundary, to which they are orthogonal.

Therefore, we can write:

$$-\infty < \mathcal{H}_c(P) - k_2|\mathcal{D}| \leq \mathcal{H}(P) \leq \mathcal{H}_c(P) - k_1|\mathcal{D}| < \infty, \tag{7.51}$$

i.e. the cost $\mathcal{H}(P)$ is bounded from above and below, as desired.

Now, in order to rigorously define ϕ_1 and ϕ_2, we need to introduce an additional assumption on the curve $\mathscr{C}$.

Assumption 7.17. *The curve $\mathscr{C}$ intersects the boundary of the domain $\mathcal{D}$ and of the Voronoi cells $\mathcal{V}_i$, $i \in \mathcal{N}$ at right angle, i.e. at the intersection points, the tangent to the curve is orthogonal to the Voronoi cells' boundary.*

Remark 7.18. *Any smooth curve $c(s)$ can be continuously modified (e.g. using bump functions [115]) in an arbitrarily small neighborhood of the intersection points, in order to satisfy the condition stated in Assumption 7.17, while still remaining smooth. Figure 7.27 shows an example of such a modification. Moreover, note that this construction does not need to happen on the physical curve on which the robots are constrained, since it is only required to be able to calculate the control inputs to the robots.*

Under Assumption 7.17, the following Fact holds.

Fact 7.19. *Let*

$$\Phi(t) = \Big\{ p \in \mathcal{V}_i \mid \|p - c(t)\| \leq \|p - c(s)\| \forall s \in I_s \text{ s.t. } c(s) \in \mathcal{V}_i \Big\} \subset \mathbb{R}^2. \tag{7.52}$$

Then, under Assumption 7.17, $|\Phi(t)| = 0$. *Moreover,* $\Phi(t)$ *is a segment of straight line orthogonal to the curve* $\mathscr{C}$ *at* $c(t)$.

Fact 7.19 is required in order for $\phi_1(s)$, $\phi_2(s)$ and consequently the integrals in (7.49), to be well-defined.

From Fact 7.19 it also follows that $\Phi(t) \subset \text{span}\{n(t)\}$. Therefore, we can use the following arc-length-parameterized line segment to describe the set $\Phi(t)$:

$$\phi_t : l \in \mathbb{R} \mapsto c(t) + l\, n(t) \in \mathbb{R}^2. \tag{7.53}$$

We are now ready to define $\phi_1(s)$ and $\phi_2(s)$ as follows:

$$\begin{aligned} \phi_1 : s \in I_s \subset \mathbb{R} \mapsto & \max_l \Big\{ \|\phi_s(l) - \phi_s(0)\| \mid \\ & \qquad l \leq 0,\ \phi_s(l) \in \mathcal{V}_i,\ i \text{ s.t. } s \in S_i \Big\} \in \mathbb{R}_+ \\ \phi_2 : s \in I_s \subset \mathbb{R} \mapsto & \max_l \Big\{ \|\phi_s(l) - \phi_s(0)\| \mid \\ & \qquad l \geq 0,\ \phi_s(l) \in \mathcal{V}_i,\ i \text{ s.t. } s \in S_i \Big\} \in \mathbb{R}_+. \end{aligned} \tag{7.54}$$

Since $\Phi(t)$ is a line segment in $\mathbb{R}^2,\ |\Phi(t)| = 0$, the integrals in (7.49) have a geometric meaning that is entirely analogous to that of (7.47) in Section 7.5.2.

Observing that $p_{i,n} = 0$, we can simplify the expression of (7.49) as follows:

$$\begin{aligned} \mathcal{H}_c(P) &= \sum_{i \in \mathcal{N}} \int_{S_i} \int_{-\phi_1(s)}^{\phi_2(s)} \left(|p_{i,s} - s|^2 + |p_{i,n} - n|^2 \right) \mathrm{d}s\, \mathrm{d}n \\ &= \sum_{i \in \mathcal{N}} \int_{S_i} \left(|p_{i,s} - s|^2 \phi(s) + \bar{\phi}(s) \right) \mathrm{d}s, \end{aligned} \tag{7.55}$$

where

$$\begin{aligned}\phi(s) &= \phi_1(s) + \phi_2(s)\\ \bar{\phi}(s) &= \frac{\phi_1(s)^3 + \phi_2(s)^3}{3}.\end{aligned} \tag{7.56}$$

As mentioned above, in case of one-dimensional domain and log-concave density function, the locational cost minimization is a convex problem [179]. The density functions $\phi(s)$ and $\bar{\phi}(s)$, however, depend on the domain $\mathcal{D}$, on the position of the curve $\mathscr{C}$ in the domain and, at each time instant, on the position of the robots along the curve, through the boundaries of the Voronoi cells. Therefore, in general, $\phi(s)$ is not a log-concave function.

In the next Section, Proposition 7.22, will give the expression of a convex relaxation of the one-dimensional coverage problem, given any, not necessarily log-concave, density function. Using this, we will derive a decentralized algorithm that minimizes $\mathcal{H}_c(P)$ in (7.55).

7.5.3 Convex Relaxation of Constrained Coverage Control Formulation for Generic Curves

In order to formulate the sought convex relaxation problem, we start observing the following.

Observation 7.20. *If $\mathcal{I} \subset \mathbb{R}$ is compact, $\{\mathcal{V}_1, \dots, \mathcal{V}_N\}$ is the Voronoi partition of $\mathcal{I}$ generated by the points $P = \{p_1, \dots, p_N\} \in \mathcal{I}^N$, then*

$$\mathcal{H}(P) = \sum_{i \in \mathcal{N}} \int_{\mathcal{V}_i(P)} |p_i - q|^2 \phi(q)\, \mathrm{d}q \tag{7.57}$$

is measurable, hence square-integrable, on $\mathcal{I}^N$.

Let $\mathcal{F} = \{f \mid f : \mathbb{R} \to \mathbb{R}_+\}$ be the space of functions that map real numbers to positive real numbers. Then, let us define

$$\begin{aligned}\mathscr{F} &: \mathcal{F} \to L^2(\mathcal{I}^N)\\ &: \theta \mapsto \sum_{i \in \mathcal{N}} \int_{\mathcal{V}_i(P)} |p_i - q|^2 \theta(q)\, \mathrm{d}q\end{aligned} \tag{7.58}$$

as the mapping that associates to each function $\theta \in \mathcal{F}$ the coverage cost with density function θ, that belongs to $L^2(\mathcal{I}^N)$ as shown in Observation 7.20. In (7.58), $p_i, q \in \mathcal{I} \subset \mathbb{R}$, $\mathcal{I}$ compact and $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ is the Voronoi partition of $\mathcal{I}$ generated by $P = \{p_1, \ldots, p_N\} \in \mathcal{I}^N$. Using this notation, $\mathcal{H}(P)$ in (7.57) can be expressed as $\mathscr{F}(\phi)$.

Now, letting

$$\mathcal{C} = \{f \in \mathcal{F} \mid f \text{ is concave}\} \subset \mathcal{F} \tag{7.59}$$

be the set of concave functions that map from $\mathbb{R}$ to $\mathbb{R}_+$, we are ready to state the convex relaxation problem as follows. The convex cost, obtained deriving a convex relaxation of (7.55), is given by the solution to the following program:

$$\min_{\substack{\theta \in \mathcal{C} \\ \theta > \phi}} \|\mathscr{F}(\theta) - \mathscr{F}(\phi)\|^2_{L^2(\mathcal{I}^N)}. \tag{7.60}$$

We insist on θ being larger than ϕ since, this way, $\mathscr{F}(\theta) \geq \mathscr{F}(\phi)\ \forall P$, and we preserve the upper bound on the two-dimensional locational cost initially stated in (7.49).

The last notion we need in order to formulate the expression of a convex relaxation of the one-dimensional coverage control problem is given in the following definition.

Definition 7.21 (Concave envelope)**.** *Let $f : X \to \mathbb{R}$ be a real-valued function defined over the non-empty convex set $X \subset \mathbb{R}^n$. The function $g : X \to \mathbb{R}$ is the concave envelope of f over X, denoted by* $\mathrm{conc}(f)$ *if*

(i) g *is concave over* X

(ii) $g(x) \geq f(x)\ \forall x \in X$

(iii) $g(x) \leq h(x)\ \forall x \in X$ *for any* h *concave s.t.* $h(x) \geq f(x)\ \forall x \in X$.

Proposition 7.22. *The following is a convex relaxation of the problem of minimizing the one-dimensional locational optimization* (7.57)*:*

$$\min_P\ \mathscr{F}(\mathrm{conc}(\phi)). \tag{7.61}$$

With the result stated in Proposition 7.22, a decentralized algorithm that allows the robots to minimize the cost defined in (7.55) is derived in the following section.

7.5.4 A Decentralized Algorithm for Constrained Coverage Control

Starting from (7.55), we can write:

$$\begin{aligned}\mathcal{H}_c(P) &= \sum_{i\in\mathcal{N}} \int_{S_i} |p_{i,s} - s|^2 \phi(s)\, \mathrm{d}s + \sum_{i\in\mathcal{N}} \int_{S_i} \bar{\phi}(s)\, \mathrm{d}s \\ &= F(P,\phi) + G(P)\end{aligned} \tag{7.62}$$

By Proposition 7.22, the problem of solving

$$\min_P \ \mathcal{H}_c(P) = \min_P \ \left(F(P,\phi) + G(P)\right) \tag{7.63}$$

can be relaxed into

$$\min_P \ \left(F(P,\mathrm{conc}(\phi)) + G(P,\psi)\right), \tag{7.64}$$

where

$$F(P,\mathrm{conc}(\phi)) = \sum_{i\in\mathcal{N}} \int_{S_i} |p_{i,s} - s|^2 (\mathrm{conc}(\phi))(s)\, \mathrm{d}s \tag{7.65}$$

is convex, and

$$G(P,\psi) = \sum_{i\in\mathcal{N}} \int_{S_i} \left(|p_{i,s} - s|^2 \psi(s) + \bar{\phi}(s)\right)\, \mathrm{d}s, \tag{7.66}$$

with $\psi(s) = \phi(s) - (\mathrm{conc}(\phi))(s)$, is differentiable.

The following theorem provides with an algorithm for solving optimization problems where the cost function is the sum of two terms: one convex (but not necessarily differentiable) and the other differentiable (but not necessarily convex).

Theorem 7.23 (Theorem 3.1 in [180])**.** *Consider the minimization problem:*

$$\min_x \ h(x) = \min_x (f(x) + g(x)), \tag{7.67}$$

where $f : \mathbb{R}^n \to (-\infty, +\infty]$ is a convex, but not necessarily differentiable, function and $g : \mathbb{R}^n \to (-\infty, +\infty]$ is a function which is continuously differentiable on an open set containing the domain of f, but g need not be convex. Then, the following algorithm is designed to generate critical points of h:

Step 1: Set $k = 0$ and initialize $x^{(k)}$ with x_0.

Step 2: Solve the convex optimization problem

$$\tilde{x}^{(k)} = \arg\min_x \left(f(x) + \left.\frac{\partial g}{\partial x}\right|_{x=x^{(k)}} x \right). \tag{7.68}$$

Step 3: Find

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)} d^{(k)}, \tag{7.69}$$

with $\lambda^{(k)} > 0$, such that

$$h\left(x^{(k+1)}\right) \leq h\left(x^{(k)} + \lambda^{(k)} d^{(k)}\right), \tag{7.70}$$

where

$$d^{(k)} = \tilde{x}^{(k)} - x^{(k)}. \tag{7.71}$$

Step 4: Check for convergence:

$$\|d^{(k)}\| < \epsilon, \tag{7.72}$$

where ϵ is some prescribed positive number.

Step 5: k = k + 1

The functions $F(P, \mathrm{conc}(\phi))$ and $G(P, \psi)$ in (7.64), satisfy the conditions of $f(x)$ and $g(x)$ in Theorem 7.23. Therefore the previous algorithm can be used to find stationary points of $\mathcal{H}_c(P)$.

The calculation of $\frac{\partial G}{\partial P}$ is decentralized, hence the value of $\tilde{P}^{(k)}$ can be obtained in a decentralized fashion using (7.68). The same holds for Step 3. A more careful analysis is required to understand whether the inequality (7.70) can be checked in a decentralized way. Given the expressions of $F(P, \mathrm{conc}(\phi))$ and $G(P, \psi)$ in (7.65) and (7.66), we can write $\mathcal{H}_c(P) = \sum_{i \in \mathcal{N}} \mathcal{H}_{c,i}(P)$, therefore $\mathcal{H}_c(P) \geq \mathcal{H}_{c,i}(P) \geq 0 \; \forall i \in \mathcal{N}$. Thus, if each robot ensures that $\mathcal{H}_{c,i}\left(p_i^{(k+1)}\right) \leq \mathcal{H}_{c,i}\left(p_i^{(k+1)} + \lambda_i^{(k)} d_i^{(k)}\right)$, then $\mathcal{H}_c\left(P^{(k+1)}\right) \leq \mathcal{H}_c\left(P^{(k)}\right)$, which means that condition (7.70) can be checked in a decentralized fashion.

Assumption 7.24. *As it is the case of the unconstrained coverage control in* (7.4)*, it is assumed that the robots know the density function over their domain.*

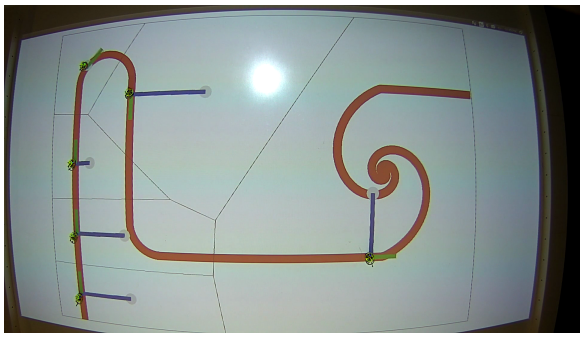
Algorithm 9 summarizes what has been derived in Sections 7.5.3 and 7.5.4, by describing the decentralized strategy adopted by the robots to perform constrained coverage control.

Algorithm 9 Constrained Coverage Control

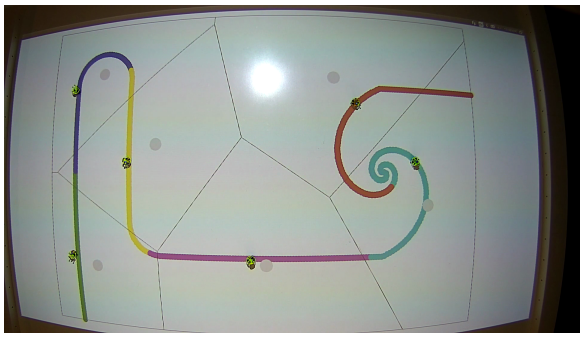
Require: $\epsilon > 0, \gamma > 0$
1: initialize $k = 0$
2: initialize $p_i^{(k)}$ to robot i's initial position
3: **repeat**
4: measure $p_j^{(k)}, \; j \in \mathcal{N}_i$
5: build Voronoi cell $\mathcal{V}_i$
6: deform curve to be orthogonal to the boundary of $\mathcal{V}_i$
7: calculate $p_i^{(k+1)}$ and $d^{(k)}$ ▷ (7.68), (7.69), (7.71)
8: execute $u_i = \gamma\left(p_i^{(k+1)} - p_i^{(k)}\right)$
9: $k \leftarrow k + 1$
10: **until** $\|d(k)\| < \epsilon$

7.5.5 Experimental Results of Simulated Robots on Single Wire

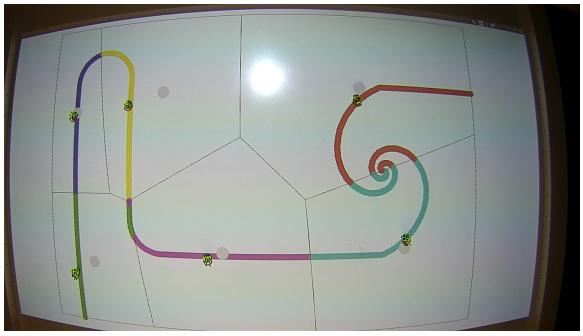
Algorithm 9 has been implemented and tested on a team of mobile robots on the Robotarium, a remotely-accessible swarm-robotics testbed [130]. A team of 6 small-scale differential drive robots has been tasked with covering a 2.8m×2m rectangular area. The curve



(a) The robots perform projected gradient descent ending up in a position where the vector that joins their position to that of the centroid of their Voronoi cell (blue vector) is orthogonal to the tangent to the curve (green vector).



(b) Performing one-dimensional coverage control, the domain that is considered in the coverage problem formulation is the (one-dimensional) curve only. Therefore, this does not take into account that the curve is in a two-dimensiona environment.



(c) The application of Algorithm 9results in a spatial allocation of the robots correposnding to a lower locational cost. This can be also seen in Fig. 7.29.

Figure 7.28: Snapshots of the experiments conducted on the Robotarium recorded using an overhead camera. Figure 7.28a to 7.28c compare the spatial allocation of a team of 6 small-scale differential drive robots obtained using three different coverage control algorithms. The Voronoi cells (black thin lines), together with their centroids (gray dots), are superimposed on the three plots.

on which the robots are constrained to move has been projected down onto the testbed using an overhead projector present in the Robotarium. Figure 7.28 shows the results of the application of the proposed algorithm and compares them to those obtained adopting two other strategies to perform two-dimensional sensor coverage while constraining the robots to move on a curve. The black thin lines represent the boundary of the Voronoi cells computed using, as generators, the position of the robots in the (two-dimensional) domain. The centroids of these cells are shown in gray: the closer the robots are to the centroid of their corresponding Voronoi cell, the smaller the locational cost is. In Fig. 7.28a the curve on which the robots move is depicted in red, whereas in Figures 7.28b and 7.28c, it is painted using a different color for each Voronoi cell (S_i in the expression of $\mathcal{H}_c(P)$ in (7.55)) on the (one-dimensional) curve.

More in detail, Fig. 7.28a depicts the spatial allocation which the robots achieve by running a projected gradient descent strategy. This has been implemented as follows: the coverage control problem has been solved as if there were no constraints on the robots' motion. Then, the input velocities, calculated by minimizing the locational cost using gradient descent, are projected onto the curve. Figure 7.28a shows the final configuration whereby, at each robot's location, the tangent to the curve (depicted in light green) is orthogonal to the unconstrained robot's velocity (represented in dark green). This configuration highlights the issue of this naive solution to the constrained coverage control, which causes the robots not to be able to move away from a very poor spatial distribution.

A second solution to the considered consists in performing sensor coverage of the curve itself. With this, we mean that the robots are asked to cover a one-dimensional manifold, i.e. the curve. Figure 7.28b shows the outcome of the implementation of this strategy. As observed before, the resulting optimization problem is convex. However no information about the two-dimensional surrounding environment is used to optimize the locations of the robots. Therefore, although a global minimum of the one-dimensional locational cost defined as in (7.57) can be reached using simple gradient descent, the shape and the position

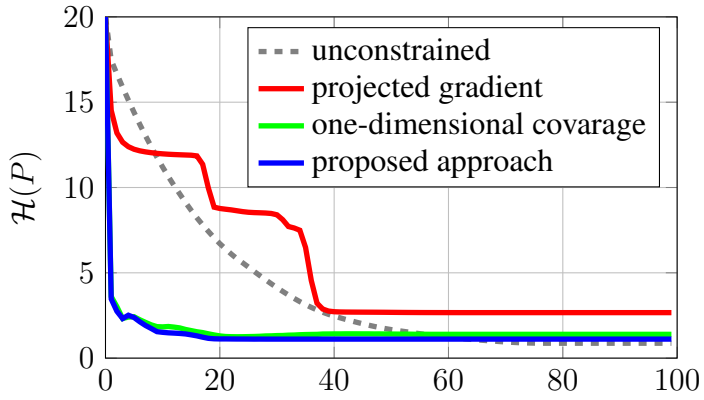


Figure 7.29: Comparisons of the location cost $\mathcal{H}(P)$ (7.4) obtained using projected gradient descent, one-dimensional coverage and the proposed approach in Algorithm 9.

of the curve in the environment heavily influence the coverage quality.

Finally, Fig. 7.28c depicts the configuration that the robots achieve by running Algorithm 9. The algorithm is decentralized as each robot can evaluate its control input only based on local information. Moreover, as the derived cost is convex, it is guaranteed to converge to the global minimum. Furthermore, as shown in Fig. 7.29, the achieved locational cost is smaller than the other methods, being higher only than the unconstrained case (where no motion constraints where imposed to the robots which where completely free to move in the environment), that has been included just as a reference. This result depends on the particular expression of the curve c and on its position in the domain $\mathcal{D}$. In the case shown in Fig. 7.28, the curve was designed to highlight the problems that projected gradient descent and one-dimensional coverage algorithms have. Nevertheless, in general, both projected gradient descent and one-dimensional coverage suffer from the problems discussed above, which make them undesirable for applications.

7.6 Conclusions

In this chapter, we presented two designs of the SlothBot: (i) a multi-body design that allows the robot to traverse a mesh of wires by switching between different branches, and (ii) a single-body design which has been deployed in the Atlanta Botanical Garden in June

2020. Compared to the state-of-the-art designs, both the locomotion and, in particular, the wire-switching maneuvers of the multi-body robotic platform are executed using a simpler actuation mechanism, which, nevertheless, results in fail-safe executions, a characteristic amenable for long-term operations.

Moreover, we presented two control algorithm suitable to control wire-traversing robots, which are able to switch between wire branches or not, respectively. The algorithms are a modification of the coverage control strategy to optimally distribute a multi-robot system in an environment in order to monitor it. Nevertheless, the same approaches can be employed in order to execute different types of tasks with robots which are constrained to move on wires.

Starting from coverage control derived for robots moving in a planar environment, in the case of robots which can switch wires, the resultant two-dimensional motion is mapped, in a continuous fashion, onto one-dimensional manifolds which represent the set of wires. The main contribution is the derivation of a *continuous* motion control law that is to be executed by the robots on the wires in order to minimize the constrained locational cost. This is realized by defining a Continuous Onto Wires (COW) map that continuously maps the robot workspace onto the wires on which the robots are constrained to move. A final projection step ensures that the locational cost subject to the motion constraints is minimized. The motion that results by the application of the derived algorithm minimizes the constrained locational cost, thus solving the constrained coverage control problem.

In the case of robots which cannot switch wires, like the single-body SlothBot design, we proposed a modification to the locational cost defined for unconstrained coverage problems in order to take into account the constraint introduced by the curve on which the robots move. Moreover, we developed a convex relaxation to efficiently, even though approximately, solve the constrained coverage control problem. The results of the implementation of the devised control strategies on a team of mobile robots (artificially constrained to move on wires) showed that the proposed approach outperforms naive projected gradient descent

and one-dimensional coverage control.

CHAPTER 8
THE BRUSHBOTS

The previous chapter described the SlothBot, a solar-power wire-traversing robot envisioned for long-term environmental monitoring applications. That is an example of design of a robotic platform which harvest solar energy, stores in its battery, and uses it for motion, sensing, communication, and computation. This way, a higher-level planning logic as well as lower-level controllers can be generated depending on sensor readings and pre-planned algorithms. In this final chapter, we make a step forward and examine the design of robots for long-duration autonomy from a different perspective. The analysis in this chapter is devoted to the study of the locomotion principles of a specific class of vibration-driven robots: the brushbots. The advantage of this types of robots is that they can passively—i.e. without the need of storing energy, computing and synthesizing control signals, as in the case of the SlothBot, or other classical robotic platforms—leverage energy harvested from the environment in order to locomote. In particular, as will be discussed in more detail at the end of the chapter, this robot design lends itself to be miniaturized and used in long-term monitoring applications (including environmental and structural health monitoring).

In the case of the SlothBot, the control design and the mechanical design were kept separate, although always serving the same purpose, i.e. letting the SlothBot remain operational for as long time as possible. The mechanical design of the brushbot, on the other hand, is more intimately connected to the control design. As a result, the former has to be explicitly taken into account in the development of the latter. In this chapter, in a bottom-up fashion, we start by deriving dynamic models of the brushes and we discuss the conditions under which these models can be employed to describe the motion of brushbots. Then, we present two designs of brushbots: a fully-actuated platform and a differential-drive-like one. The former is employed to experimentally validate both the developed theoretical

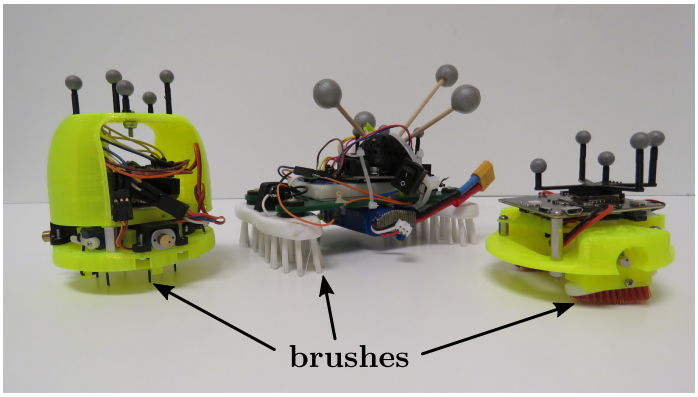


Figure 8.1: Examples of brushbots: metallic rods, brushes and toothbrushes are employed to convert energy of vibrations into directed locomotion.

models and the devised motion control algorithms. Finally, a coordinated-control algorithm is implemented on a swarm of differential-drive-like brushbots in order to demonstrate the design simplicity and robustness that can be achieved employing a vibration-based locomotion strategy.

Many diverse robot locomotion modalities have been the subject of analyses and design studies carried out so far (see, e. g., Part B of [181]). Some of these modalities are biologically inspired, as, for instance, in the case of batbots [182], brachiating robots [183], or robotic bees [184]. Others, instead, are the result of attempts to enhance human capabilities, the most remarkable example of which being the *wheel* [185].

The reasons driving the efforts to understand different robot locomotion strategies can be summarized in the following ones:

(i) improving the efficiency and the quality of human life (which led to the invention of the wheel)

(ii) developing general design principles for more complex robotic platforms (as, e.g., in [186, 187])

(iii) understanding the underlying operating mechanisms (see, e.g., [188, 189]).

This chapter presents a theoretical and experimental study of a particular class of vibration-driven robots: the brushbots (Fig. 8.1). The brushbot is a robot that employs elastic ele-

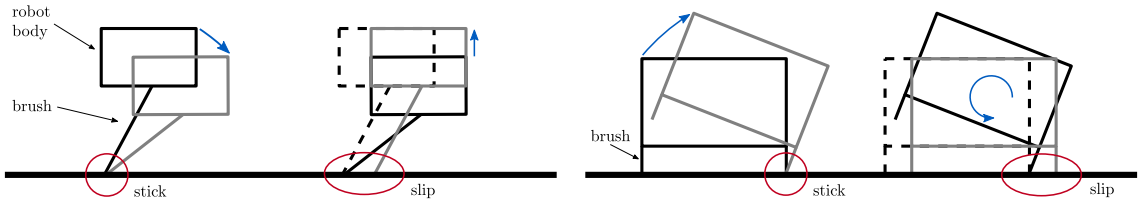
ments, referred to as *brushes*, to convert the energy of a vibration source into directed locomotion. The subset of the brushbots on which we focus our study is that of planar robots moving on a smooth surface. The reasons for studying this kind of robot and locomotion mechanism include all the ones mentioned above. First, we will develop a model for the brushbots and their locomotion strategy; second, we will provide the theoretical foundations required to further analyze more complex systems (as, for instance, [105]); and, finally, we will demonstrate that the results of this study can be leveraged in many robotic applications, ranging from low-energy environment monitoring [190] to medical robotics [191], in order to refine the system design, improve performance and optimize execution.

In next section, we start by developing a dynamic model suitable for brushbots which is experimentally validated using a fully-actuated brushbot, in Section 8.2, and a swarm of differential-drive-like brushbots, in Section 8.3.

8.1 Modeling of Vibration-Based Locomotion

To understand the use of brushes for locomotion, let us start by developing a microscopic dynamic model of the brushes themselves. The attempts at describing the brush dynamics using different physical models have been multiple. Depending on the considered robot design, the resulting developed models are fundamentally different. In this chapter, we identify two regimes in which a brushbot can operate, and we analyze them in detail in Sections 8.1.1 and 8.1.2. Section 8.1.3 discusses the range of parameters under which there two regimes are valid by highlighting the factors which cause a brushbot to operate in one regime rather than the other.

As in many types of locomotion, brushbots move by exploiting friction [192]. The source of energy for the system is given by vibration motors: these can be in the form of piezoelectric actuators as well as eccentric rotating mass motors. In the latter, a mass is mounted with an eccentricity with respect to the axle of a DC motor; when rotating, the



(a) *Regime I*: robots operating in this regime are characterized by a high flexibility of the brushes. The vibrations are modeled as alternating vertical forces which deform the brushes during the stick phase and pull the robot up during the slip phase. At this moment, the friction reduces proportionally to the reduction of normal force, allowing the brush to slide and the robot to step forward.

(b) *Regime II*: light robots with stiff brushes can operate in this regime. As the flexibility of the brushes cannot be exploited, locomotion is achieved by the sequence of two rigid body rotations happening in sequence. The first one in the stick phase and the second in the slip phase, during which the brushbot experiences a net displacement.

Figure 8.2: Two regimes of operation of the brushbot: locomotion is achieved by exploiting vibrations in two different ways, depending on the physical characteristics of the robot.

mass produces a rotating centrifugal force which induces vibration in the robot body on which the motor is mounted. This kind of vibration motors are the ones considered in this chapter. The produced vibrations are transformed into net motion by alternating a *stick* phase and a *slip* phase.

Fig. 8.2 shows the sequence of stick-slip phases for two regimes in which brushbots can operate. Fig. 8.2a depicts *regime I*: on the left (stick phase), a schematic representation of the brushbot moves from the position depicted in black to the one depicted in gray. This is obtained by deforming the long flexible brush. On the right of the figure, the slip phase is shown: here the brush slides on the ground until the robot reaches the position depicted in gray. At this point, the robot has experienced a net displacement towards the right compared to the initial position (dashed contour). During these two steps, the robot body always remains parallel to the ground and the motion is achieved thanks to the deformation of the brush.

In Fig. 8.2b, *regime II* is depicted. This regime is characterized by the fact that stiff short brushes do not deform, but rather act as pivot points for the robot to rotate. During the stick phase (on the left of the figure), the robot body rotates about a pivot point, whereas, in the slip phase (on the right), the robot body rotates back to its initial orientation while

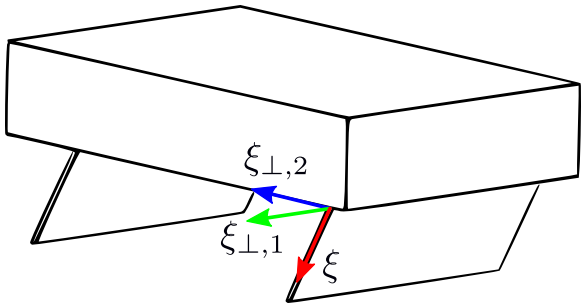


Figure 8.3: A brushbot with plate-like brushes with the brushes reference frame $\xi\xi_{\perp,1}\xi_{\perp,2}$. The resulting second area moment of the cross section of the bristles is higher about the $\xi_{\perp,2}$ axis than about $\xi_{\perp,1}$. The higher the difference between the two second area moments, the more realistic Assumption 8.3 is.

sliding towards the right. The two regimes can be more or less predominant depending on the physical characteristics of the robot, as will be discussed in Section 8.1.3. In the following two sections, we derive the dynamic models for brushbots operating in the two described regimes.

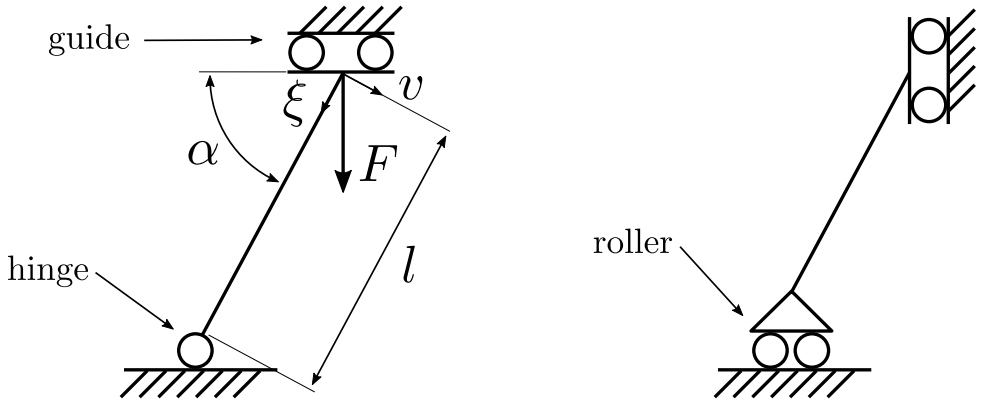
8.1.1 Model for Regime I

The main factors that make brushbots operate in regime I rather than II are weight and brush stiffness. Regime I is characterized by a lower brush stiffness (more deformable brushes) and/or a heavier robot body. The following assumptions are used for the derivation of the brush dynamic model in this regime.

Assumption 8.1. *During operations in regime I, the brushes are always in contact with the ground. The heavier robot body, in fact, does not allow the centrifugal force generated by vibration motors to lift the robot from the ground.*

Assumption 8.2. *The body of the brushbot always remains parallel to the ground. This is justified by the fact that the inertia of the body does not allow big rotations at the frequencies at which the vibration motors are typically actuated.*

Assumption 8.3. *The deformation of the brushes is planar. Indeed, the inclination of the brushes has the effect of reducing their equivalent stiffness in one direction. More precisely,*



(a) Model for the stick phase.

(b) Model for the slip phase.

Figure 8.4: Beam model employed to analyze the dynamics of the brush during the stick and slip phases. v represents the displacement of the beam in the direction orthogonal to the beam axis ξ. Compare with the qualitative motion depicted in Fig. 8.2.

referring to Fig. 8.3, the fact that the brushes are rotated around axis $\xi_{\perp,1}$ with respect to the ground makes their equivalent stiffness in the plane $\xi\xi_{\perp,2}$ smaller than the one in the plane $\xi\xi_{\perp,1}$. This will be theoretically derived later in this section.

We employ the Euler-Bernoulli beam model (see, e. g., [193]) to analyze the motion of each brush. Figures 8.4a and 8.4b show the structural scheme used to model stick and slip phases, respectively. During the stick phase, the constraints are a guide at the top (where the brushes connect to the robot) and a hinge at the bottom (at the contact with the ground). In the slip phase, a horizontal translational degree of freedom for the interaction with the ground is added by using a roller support in place of the hinge. This allows the tip of the brush in contact with the ground to slide.

The Euler-Bernoulli beam model allows us to evaluate the deformed shape of the brush,

as well as its equivalent stiffness, by solving the following boundary value problem:

$$\begin{cases} EIv'''' = 0 \\ EIv'''|_{\xi=l} = F\cos\alpha \\ EIv''|_{\xi=l} = 0 \\ v'|_{\xi=0} = 0 \\ v|_{\xi=0} = 0. \end{cases} \tag{8.1}$$

Here, v represents the displacement of the beam in the direction orthogonal to the beam axis ξ, v' is used to denote $dv/d\xi$, $F = m\omega^2 r\sin(\omega t)$ is the centrifugal force produced by an eccentric rotating mass motor which rotates a mass m, at speed ω, mounted with an eccentricity r with respect to the motor axle. E and I are the Young modulus and the second area moment about $\xi_{\perp,1}$ of the beam. l and α are the length and inclination of the beam, respectively. The solution to (8.1) is given by

$$v(\xi) = \frac{F\cos\alpha}{6EI}\xi^3 - \frac{Fl\cos\alpha}{2EI}\xi^2. \tag{8.2}$$

So, the displacement v of the robot body at the tip of the brush can be evaluated as:

$$|v(l)| = \frac{Fl^3\cos\alpha}{3EI}. \tag{8.3}$$

During the slip phase (Fig. 8.4b), the robot body moves upwards, reducing the horizontal force due to friction which acts on the brush tip. The net horizontal displacement can be calculated as follows (see Fig. 8.5):

$$\begin{aligned} \delta = \overline{P_2P_3} &= \overline{P_1P_3} - \overline{P_1P_2} = l\cos(\alpha-\vartheta) - l\cos\alpha \\ &= l\cos\left(\alpha - \frac{m\omega^2 r l^2\cos\alpha}{3EI}\right) - l\cos\alpha, \end{aligned} \tag{8.4}$$

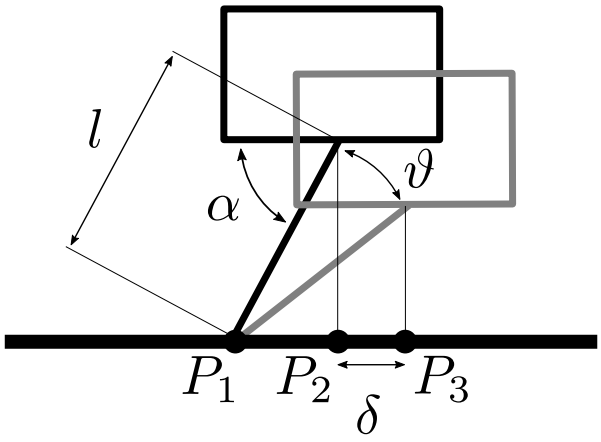


Figure 8.5: The net displacement of the brushbot, δ, is evaluated based on the angle ϑ induced by the force F (see Fig. 8.4a) and the geometric characteristics of the brush.

where $\overline{P_iP_j}$ denotes the length of the segment joining points P_i and P_j, and the expression for ϑ is obtained by observing that, under the small-angle approximation, $|v(l)| = l\vartheta$ (). Considering the fact that the robot experiences a displacement of δ per full rotation of the motor, the ground speed of the robot, v_r, can be obtained as follows:

$$v_r = \frac{\delta}{\Delta t} = \frac{\omega}{2\pi}\left(l\cos\left(\alpha - \frac{m\omega^2 r l^2\cos\alpha}{3EI}\right) - l\cos\alpha\right), \tag{8.5}$$

where ω is the angular velocity of the motor.

For the study of the oscillating brush dynamics, we use the lumped-parameter model depicted in Fig. 8.6 with

$$k_\vartheta = \frac{3EI}{l^2\cos\alpha} \tag{8.6}$$

$$I_\vartheta = \frac{M_b l^2}{2}, \tag{8.7}$$

being the stiffness and the inertia relating the force F and the angle ϑ, and M_b denotes the mass of the brush.

Assumption 8.4. *The inclination angle of the brushes $\alpha \in (0, \pi/2)$, i. e. the brush is*

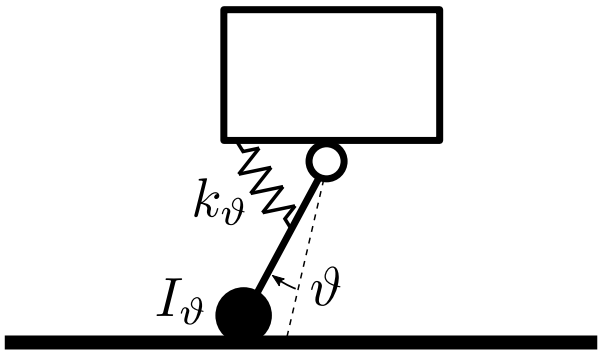


Figure 8.6: Lumped-parameter model used to analyze the dynamics of the brushes: the equivalent stiffness k_ϑ and inertia I_ϑ, given in (8.6) and (8.7), determine the spring-mass-like response of the brush angle ϑ as a result of the force F in Fig. 8.4a.

neither horizontal nor vertical.

Under this assumption, k_ϑ is well-defined.

Observation 8.5. *The expression of k_ϑ in* (8.6) *indicates that equivalent stiffness of the brushes increases with an increase of the angle α. In the limit case: $k_\vartheta \to \infty$ as $\alpha \to \pi/2$. This reflects the fact that, if brushes are perpendicular to the ground, no net displacement can be achieved. Moreover, by the insight gained using the Euler-Bernoulli model, we can see that Assumption 8.3 becomes more realistic as the second area moment around $\xi_{\perp,1}$, which we denoted by I, becomes smaller with respect to the one around $\xi_{\perp,2}$ (see Fig. 8.3).*

In the analysis of the dynamic effects introduced by the inertia of the brushes, we start by calculating the time that the brushes take, during the slip phase, to go back to the rest position from the configuration reached at the end of the stick phase (see Fig. 8.2). Taking into account their inertial effects, the brushes can be modeled as the following second-order

system:

$$\begin{cases} I_\vartheta \ddot{\vartheta} + k_\vartheta \vartheta = 0 \\ \vartheta(0) = \bar{\vartheta} \\ \dot{\vartheta}(0) = 0, \end{cases} \tag{8.8}$$

whose solution is given by $\vartheta(t) = \bar{\vartheta}\cos(\omega_n t)$, where

$$\omega_n = \sqrt{\frac{k_\vartheta}{I_\vartheta}} = \sqrt{\frac{6EI}{M_b l^4 \cos\alpha}} \tag{8.9}$$

is the natural frequency of the brush. The time to go back to the rest position is the earliest time at which $\vartheta(t) = 0$, i. e. $\omega_n t = \kappa\pi/2$. So, the earliest time instant $\bar{t}$ at which the brushes come back to the undeformed configuration is given by:

$$\bar{t} = \kappa \frac{\pi}{2\omega_n}\bigg|_{\kappa=1} = \frac{\pi}{2}\sqrt{\frac{M_b l^4 \cos\alpha}{6EI}}. \tag{8.10}$$

Stiffer (larger EI), shorter (smaller l), less inclined (smaller α), lighter brushes (smaller M_b) lead to a faster response to vibrations (smaller $\bar{t}$).

While the vibration motor is rotating, the slip phase occurs if the friction between the brush and the ground is not enough to prevent the brush from sliding. The transition from the stick phase to the slip phase is triggered by a reduction of the force acting on the robot and normal to the ground due to centrifugal acceleration of the unbalanced rotating mass. Therefore, a quarter of period of revolution of the motor is the time the brushes have to move forward during the slip phase. Thus, to maximize the net displacement of the robot, we want to achieve a motor speed ω such that

$$\bar{t} = \frac{1}{4}T = \frac{1}{4}\frac{2\pi}{\omega}, \tag{8.11}$$

where T is the period of revolution of the motor. Solving (8.11) for ω yields:

$$\frac{\pi}{2}\sqrt{\frac{M_b l^4 \cos\alpha}{6EI}} = \frac{\pi}{2\omega^\star} \quad \Leftrightarrow \quad \omega^* = \sqrt{\frac{6EI}{M_b l^4 \cos\alpha}} = \omega_n. \tag{8.12}$$

Thus, not surprisingly, if the motor speed matches the natural frequency of the brushes ω_n, the displacement of the robot is maximized. This can be also seen by considering the model in (8.8) with a non-zero input force:

$$I_\vartheta \ddot{\vartheta} + k_\vartheta \vartheta = m\omega^2 r \sin(\omega t)\cos\alpha^1, \tag{8.13}$$

whose forced solution is given by

$$\vartheta(t) = \frac{m\omega^2 r \sin(\omega t)\cos\alpha}{\omega_n^2 - \omega^2}\sin(\omega t) = \hat{\vartheta}(\omega)\sin(\omega t). \tag{8.14}$$

According to the model (8.13), the amplitude of the brush oscillations, $|\hat{\vartheta}(\omega)| \to \infty$ as $\omega \to \omega_n$. In practice, there are damping effects which will reduce the oscillation amplitude to a finite value. However, notice also that the model derived in this section holds under Assumpion 8.1. Therefore, it cannot be used to analyze the motion of the brushbot in case ω is such that $m\omega^2 r \sin(\omega t) > Mg$, where Mg is the weight of the robot, M being its mass. At this point, the robot starts transitioning towards regime II which will be explained in the following section.

8.1.2 Model for Regime II

The model for the second regime in which brushbots can operate predicts the robot motion under the following assumption.

Assumption 8.6. *The robot body and the brushes are rigid bodies. This entails that brushes are not deformable.*

[1] Despite their expressions, $I_\vartheta \ddot{\vartheta}$ and $k_\vartheta \vartheta$ are not torques, but rather forces.

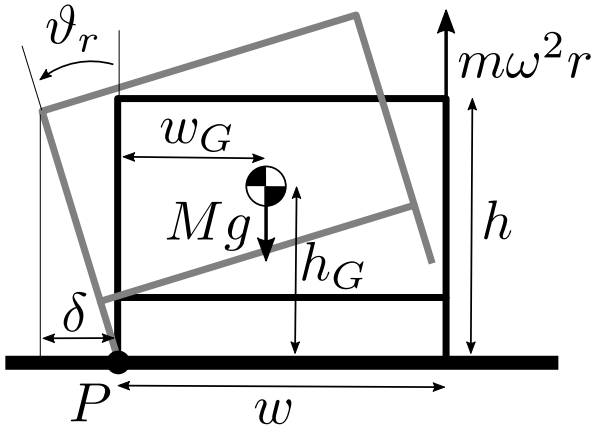


Figure 8.7: Motion of the brushbot during the stick phase in regime II. The inclination angle of the brushbot body, ϑ_r, accelerates about the point P under the effect of vibrations and gravity, through the moments generated by the forces $m\omega^2 r$ and Mg, respectively.

Similar to what has been discussed for regime I, also in this second regime we have the alternation of stick and slip phases as shown in Fig. 8.2b. The difference with the previous case lies in the fact that the effect of the deformation of the brushes is not significant and, therefore, can be neglected. In order to model the motion of the brushbot, by Assumption 8.6, we can write the following rigid body motion equations for a brushbot operating in regime II:

$$I_P \ddot{\vartheta}_r = m\omega^2 r \sin(\omega t) w - Mg w_G, \tag{8.15}$$

where I_P is the rotational inertia about point P shown in Fig. 8.7, where the quantities w, w_G and the gravitational force acting on the robot body are depicted. In order to simulate the interaction with the ground, the constraint $\vartheta_r \geq 0$ has been enforced. At the point of impact on the ground, we assume $\dot{\vartheta}_r = 0$ and $\ddot{\vartheta}_r = 0$. Trajectories of angular position $\vartheta_r(t)$, velocity $\dot{\vartheta}_r(t)$ and acceleration $\ddot{\vartheta}_r(t)$, are shown in Fig. 8.8, together with the resulting displacement x of the robot on the ground. The maximum absolute value of ϑ_r, which is denoted by $|\hat{\vartheta}_r|$, is the one which determines the displacement δ of the robot, given by:

$$\delta = h \sin |\hat{\vartheta}_r|. \tag{8.16}$$

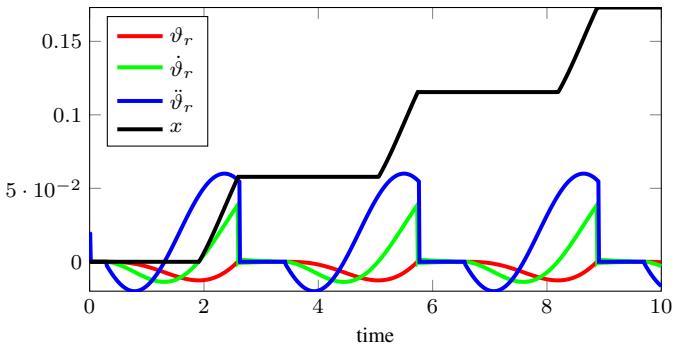


Figure 8.8: Simulation results of the sequence of stick-slip phases of regime II (as depicted in Fig. 8.2b) obtained by solving (8.13). Trajectories of the angle ϑ_r, its first and second time derivatives are reported. The robot position x, depicted in black, shows its ability to locomote in regime II.

Observation 8.7. *The rotation of the robot body is neglected for the development of an analytical model for regime I, because the flexibility of the brushes prevails on the rigid rotation of the robot body. Here, on the other hand, the brushes are assumed to be rigid, therefore, the rotation angle of the robot body has the most significant effect. Nevertheless, due to the inertia of the robot, the centrifugal force generated by the unbalanced mass of the motors is not able to rotate the robot body by more than a few degrees. For this reason, we can introduce a small-angle approximation in* (8.16) *and express the robot velocity as a function of* $|\hat{\vartheta}_r|$ *as*

$$v_r = \frac{\delta}{\Delta t} \approx \frac{\omega h |\hat{\vartheta}_r|}{2\pi}. \tag{8.17}$$

8.1.3 Range of Applicability of the Models

In [103] and [102], two vibration-driven robots which work in regime I and regime II, respectively, are presented. The fundamental differences between these robots are related to their weight and the brushes they employ to transform vibrations into motion. In the following, we discuss the physical characteristics of brushbots which cause the models for regimes I and II to be able to describe more or less accurately the robot motion.

(i) *Rigidity of the brushes.* Expressed in terms of EI in (8.1), the rigidity of the brushes proportionally influences the equivalent stiffness (8.6) and therefore the natural fre-

quency (8.9). A high rigidity, however, means also a small displacement $v(l)$ in (8.3). In practice this means that a stiffer robot moves very little per each revolution of the motor, although it is able to vibrate more at faster frequencies, as indicated by (8.14). For this reason, brushbots equipped with stiffer brushes are more likely to operate in regime II.

(ii) *Mass of the robot.* The influence of the mass of the robot is recognizable in the effect it has on the inertia I_P used in (8.15) that the robot exhibits with respect to rigid rotations around axes that lie in the plane in which the robot moves. Therefore, at a constant power produced by the motors, robots operating in regime II typically have smaller masses compared to the ones operating in regime I. This, in fact, results in smaller inertias which allow the robots to quickly respond to alternating input forces. In the case of robots operating in regime I, the flexibility of the brushes reduces the response bandwidth, given by the natural frequency (8.9). Therefore, the motion due to regime I dominates the one due to regime II.

(iii) *Inclination of the brushes.* By Assumption 8.4, the inclination of the brushes, α, is never equal to $\pi/2$. In the limit case in which $\alpha = \pi/2$, in fact, the dynamic model (8.8) for regime I predicts zero net motion of the robot. When the brushes become straight ($\alpha \to \pi/2$), in fact, the brushbot starts operating mainly in regime II.

A factor that influences the brushbot motion is the position of multiple sets of brushes and actuators, which will be explicitly considered in the next sections. The presence of multiple brushes introduces constraints which are not taken into account in the model of regimes I and II. In fact, the superposition of the effects that different sets of brushes have due to their different orientations can result in drastically different behaviors depending on the regime in which the robot operates. Consider a brushbot configuration where three sets of brushes are oriented radially equally spaced along the circumference of the brushbot. This leads to the practical impossibility of motion of such a brushbot operating in regime

I, whereas can be exploited to achieve *holonomic* motion when operating in regime II.

In Sections 8.2 and 8.3, we show how to leverage the effects described above together with the models developed in this section in order to design and control fully-actuated and differential-drive brushbots. Moreover, in Section 8.2, we report experimental results to show the validity of the proposed models in predicting the motion of brushbots.

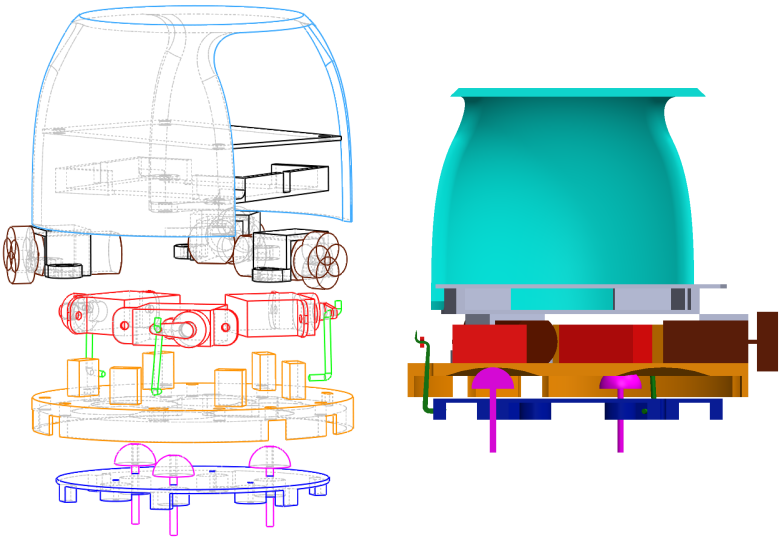
8.2 Design and Control of Brushbots

In this section, we present the design and control of a *fully-actuated* brushbot which can operate in regime I and II and can be used to validate the theoretical model developed in Section 8.1. The design shown in Fig. 8.9 is fully-actuated insofar as we can control both the velocity of the motors and the inclination of the metallic rods, which play the role of the brushes. The actuation of the speed of the motors is obtained through a standard regulation of the voltage supplied to the motors, whereas the inclination of the brushes is realized by means of a three-degree-of-freedom Stewart platform [194]. The controllable degrees of freedom of the platform are roll and pitch angles, encoded by the xy-components, A and B, of the vector normal to the platform, and the vertical position of the center of the platform, C. The inverse kinematics required to obtain the angular velocity of the servo motors, $\omega_{s,i}$, as a function of the desired angular and linear velocity of the Stewart platform are given by:

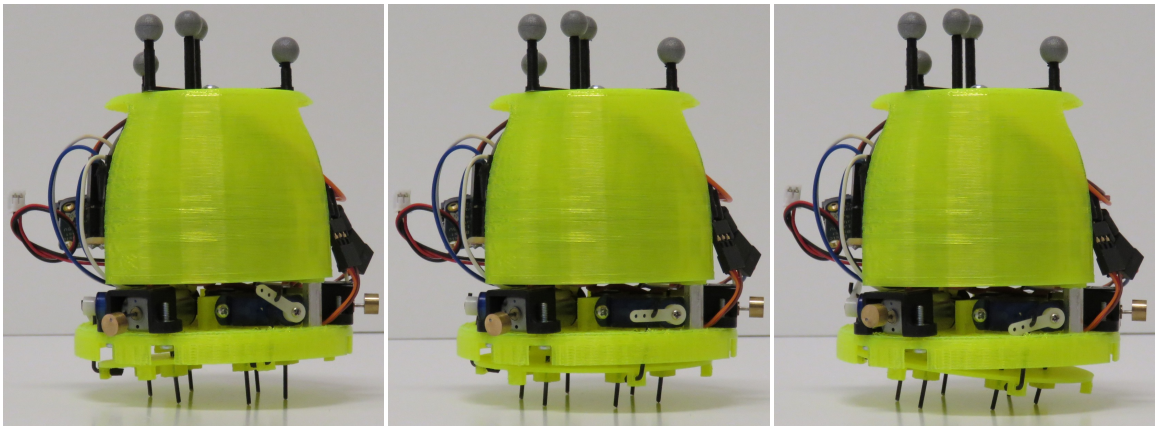
$$\omega_{s,i} = \frac{1}{d}\left(-x_i\dot{A} - y_i\dot{B} - \dot{C}\right), \tag{8.18}$$

where $[x_i, y_i, z_i]^T$, $i = 1, 2, 3$ are the positions in space of three points of the Stewart platform of which the third component, z_i, can be actuated through the servo motors according to the relation $\dot{z}_i = \omega_{s,i}d$, d being the length of the servo motor cranks.

As mentioned above, the design of the brushbot presented in this section is able to switch between operating regime I and II. In view of what has been discussed in point (iii) in Section 8.1.3, the switch from regime I to regime II is achieved by constraining the metallic



(a) (b)



(c) (d) (e)

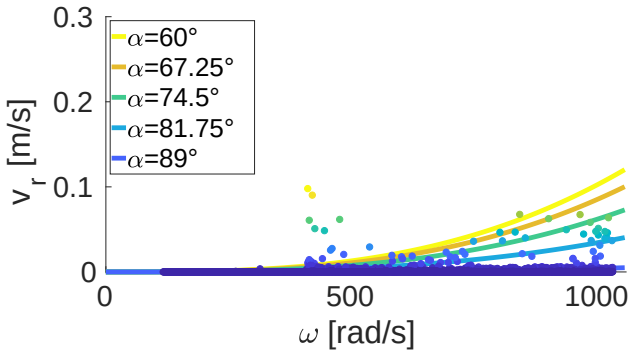
Figure 8.9: The presented fully-actuated brushbot. In Fig. 8.9a, the exploded view of the CAD model shows (from top to bottom): outer shell (light blue), PCB and battery support (black), vibration motors (brown), servo motors (red), 3dof-Stewart platform links (green), main body (yellow), brushes (purple), 3dof-Stewart platform (blue). Fig. 8.9b shows a section view of the actuation of the brushes which, connected through prismatic joints to the Stewart platform, can be oriented at different angles. Figures 8.9c to 8.9e showcase the actuation mechanism on a 3D printed prototype of the robot.

rods to remain vertical and by individually actuating the three vibration motors (shown in Fig. 8.9a). This concept is illustrated in Fig. 8.9b, which shows a section view of the brushbot: by pulling the Stewart platform up, the servo motors push the top hemispherical tip of the metallic rods against the main body. This prevents them from inclining. The actuation of the vibration motors, placed diametrically opposite with respect to each of the metallic rods, realizes the motion of the brushbot as described in Fig. 8.7 in three different sagittal planes of the robot.

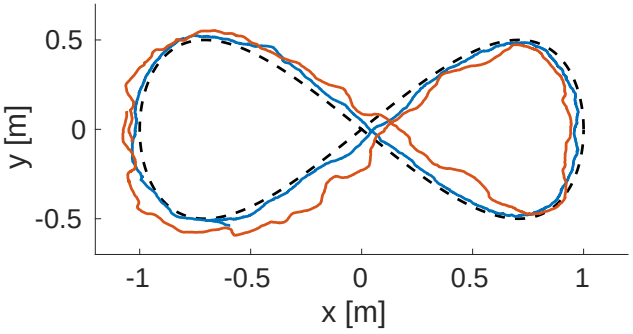
A series of experiments have been conducted in order to validate the derived dynamic model and to test a trajectory-tracking controller. The brushbot has been driven with different angles α and vibration motor speeds ω. The pose of the robot has been measured using an infrared-camera-based motion-capture system, for which the brushbot has been equipped with an identifying marker consisting of infrared-reflective balls (visible at the top of Figures 8.9c to 8.9e). Fig. 8.10a shows the results of this series of experiments: each dot represents a collected data point, whose colors encodes the inclination angle of the brushes, while the curves are the predictions of the model in (8.5). The physical characteristics of the brushbot required to predict its velocity have been calculated based on known material properties or obtained from the components datasheets, and they are the following: $l = 0.01$ m, $mr = 10^{-4}$ kg m, $E = 2.1\ 10^{11}$ N/m^2, and $I = 1/4\pi 1.2^4$ mm^4. The plot shows that the theoretical analysis described in Section 8.1 allowed us to develop a model for the brushbot which is able to accurately predict its motion.

In order to test the trajectory tracking performances of the designed brushbot, the following point-tracking controller has been devised:

$$\begin{cases} \omega = k_1 \|p_{\text{goal}} - p\| \\ \begin{bmatrix} A \\ B \end{bmatrix} = k_2 R^T(\psi)(p_{\text{goal}} - p) \\ C = 0, \end{cases} \tag{8.19}$$



(a)



(b)

Figure 8.10: Results of experiments conducted with the fully-actuated brushbot presented in this section. In Fig. 8.10a, the predicted VS measured robot velocities are shown. The measurement data are depicted as points whose color is function of the inclination angle α (following the legend), whereas the curves show the dependence of the robot velocity on the vibration motor velocities obtained for different inclination angle of the brushes. Fig. 8.10b shows the results of trajectory tracking experiments: the reference trajectory (black dashed line) has been given as input to the point-tracking controller (8.19). Two different curve parameterizations have been tested, characterized by lower and higher speed (3.5 and 7 cm/s), and the tracking results are depicted as a blue and a red curve, respectively.

where $k_1, k_2 > 0$, $p_{\text{goal}} \in \mathbb{R}^2$ is the point to track, $p \in \mathbb{R}^2$ is the position of the robot in the plane, ψ its orientation, $R^T(\psi) \in SO(2)$ is the rotation matrix which transforms vectors from the global reference system, in which p and ψ are measured, to the robot local reference system (where $\psi = 0$). In the tracking experiment shown in Fig. 8.10b, a point moving on the trajectory to track (black dashed line) is used as p_{goal} in (8.19) to obtain ω and A, B, C. The latter are transformed into servo motors inputs according to (8.18) and, together with ω, sent to the robot. The blue and red curves in Fig. 8.10b are the trajectories followed by the brushbot while tracking the reference trajectory with low (3.5 cm/s) and high (7 cm/s) speed, respectively. Tracking performance are very good at lower speeds, but they start to deteriorate as the speed of the robot increases. This is due to the fact that the derived model is not valid anymore and the robot starts transitioning from regime I to regime II.

8.3 Brushbots in Swarm Robotics

Leveraging the knowledge gained in the analysis of the brush dynamics, as well as macroscopic effects resulting from the presence of multiple sets of brushes, this section presents the design of a simple and robust brushbot. The time to build the brushbot that is presented in this section is, in fact, less than three hours, which include 3D printing, soldering and preparation of the brushes. The unit cost is kept below 30$, which can be significantly reduced if the number of robots to produce increases. The design of simple, easy and fast-to-build, robust brushbot makes it very appealing and suitable for swarm robotics applications, which deals with the coordination and interactions of a large number of robots.

Fig. 8.11 shows the schematic design of the brushbot presented in this section: it is a *differential-drive-like* brushbot, which consists of two sets of brushes mounted parallel to each other on two opposite sides of a rigid platform. Two motors (shown in Fig. 8.12) are mounted on top of each of the brushes. This design embodies the interplay between regime I and II described in Section 8.1 in a different way compared to the design in Section 8.2.

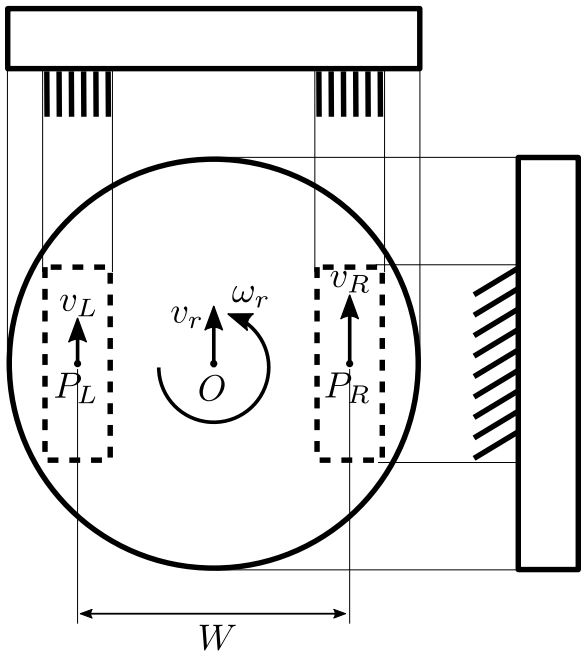
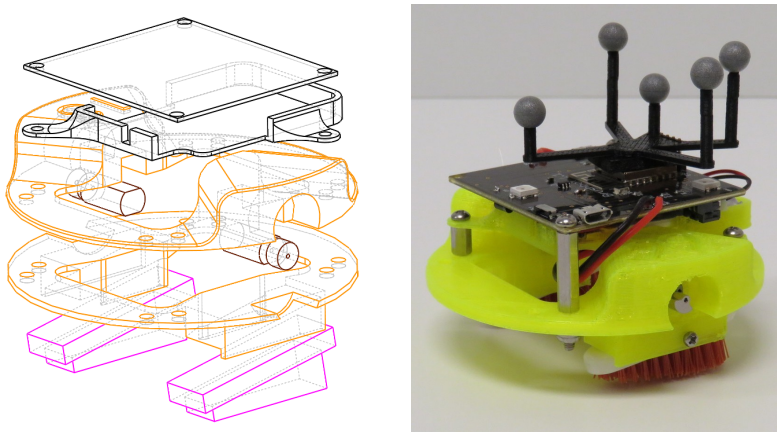


Figure 8.11: Differential-drive-like brushbot: two sets of brushes are mounted on the opposite sides of the robot body. Desired linear and angular velocities of the robot body can be achieved by varying the speed of vibration motors mounted on top of each set of brushes.



(a) (b)

Figure 8.12: Differential-drive-like brushbot. In Fig. 8.12a, the exploded view of the CAD model shows, from top to bottom: PCB and battery support (black), top body (orange), vibration motors (brown), bottom body (orange), brushes (purple). Fig. 8.12b shows a 3D printed prototype of the brushbot: infrared-reflective balls are mounted on top for tracking its pose.

The motion of the differential-drive brushbot can be described as follows:

- actuating the left motor produces a velocity given by (8.5), indicated as v_L in Fig. 8.11, at the left set of brushes (as described by regime I)
- at the same time, due to the actuation of the left motor, the robot pivots about the right set of brushes, which induces a net angular velocity, ω_r, of the robot (as predicted by regime II)

The rigid body dynamics of the robot are then:

$$v_{L,R} = v_r - \omega_r \times (P_{L,R} - O) \tag{8.20}$$

From (8.20), the expressions of linear and angular velocities of the differential-drive-like brushbot can be obtained:

$$v_r = \frac{v_L + v_R}{2}, \quad \omega_r = \frac{v_R - v_L}{W}, \tag{8.21}$$

where, with abuse of notation, all symbols have been used to denote the signed magnitudes of the vector quantities used in (8.20), their directions being given in Fig. 8.11. The motor speeds $\omega_{L,R}$ to realize the linear speeds $v_{L,R}$ used in (8.21) can be calculated using (8.5).

Modeling the brushbot as a unicycle, one can use controllers such as the one developed in [138] to implement complex swarm-robotics algorithms (see, e.g., [129]). The technique consists in obtaining a linear system equivalent to the unicycle by considering a point p positioned at distance d in front of the center of the robot, denoted by the coordinates x and y in Fig. 8.13. The unicycle dynamics have been used already in (4.30) in Chapter 4. In the following, we briefly summarize them, adopting the notation used to describe the dynamics

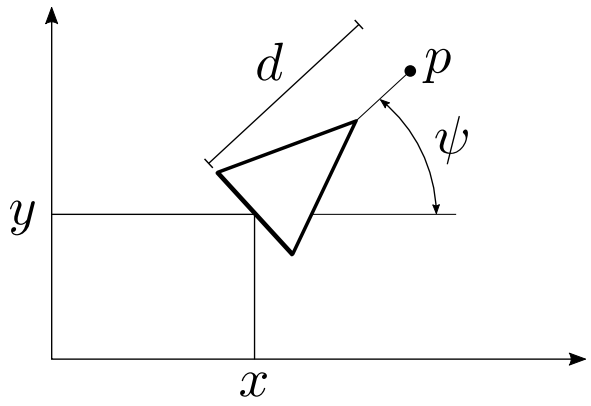


Figure 8.13: Unicycle robot, depicted as a triangle, used to model the differential drive brushbots. The state of the robot is given by its position, represented by the coordinates x and y, and orientation, denoted by ψ. The point p at a distance d in front of the unicycle is used to linearize the system in order to design simple motion control laws.

of the brushbots:

$$\begin{cases} \dot{x} = v\cos\psi \\ \dot{y} = v\sin\psi \\ \dot{\psi} = \omega, \end{cases} \tag{8.22}$$

whereas the dynamics of the point p moving in front of a unicycle can be written as:

$$\dot{p} = \underbrace{\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}}_{=:R(\psi)} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & d \end{bmatrix}}_{=:D(d)} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{8.23}$$

Therefore, once a controller for p is designed, the inputs v and ω to the unicycle can be easily computed as

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = D^{-1}(d)R^T(\psi)\dot{p}, \tag{8.24}$$

and the values of v_L and v_R can be compactly expressed as

$$\begin{bmatrix} v_L \\ v_R \end{bmatrix} = D_W^{-1}D^{-1}(d)R^T(\psi)\dot{p}, \tag{8.25}$$

where

$$D_W = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2W} & \frac{1}{2W} \end{bmatrix}. \tag{8.26}$$

Notice, however, that based on (8.5), it is not possible to reverse the direction of the speed on the ground generated by the combined action of vibration motors and brushes. Therefore, the velocities v_L and v_R corresponding to v and ω calculated from (8.24) need to be positive. The naive approach consisting of zeroing the values of v_L and v_R whenever they are negative might lead to practical impossibility of moving under arbitrary desired $\dot{p}$. To circumvent this issue, the idea is that of adaptively changing the value of d according to the differential equation

$$\dot{d} = u_d, \tag{8.27}$$

where u_d is an additional control input we can give to the system. With this modification, the linearized dynamics (8.23) become:

$$\dot{p} = \underbrace{R(\psi)D(d)D_W}_{=:B_1(\psi,d)} \begin{bmatrix} v_L \\ v_R \end{bmatrix} + \underbrace{\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}}_{=:B_2(\psi)} u_d = \underbrace{\begin{bmatrix} B_1(\psi,d) & B_2(\psi) \end{bmatrix}}_{=:B} \underbrace{\begin{bmatrix} v_L \\ v_R \\ u_d \end{bmatrix}}_{=:u_p} \tag{8.28}$$

Although d is changing, we want to keep it, for as much as we can, through u_d, to a desired value $\bar{d}$. Moreover, we would like to keep it positive and less than $\bar{d}$. The stated conditions lend themselves to be encoded using the following control Lyapunov function V_d and control Barrier function h_d:

$$V_d(d) = (d - \bar{d})^2 \tag{8.29}$$

$$h_d(d) = d(\bar{d} - d). \tag{8.30}$$

At this point, we can define the following optimization problem incorporating the CLF and

CBF constraints in a holistic fashion (similarly to what has been done in [46]):

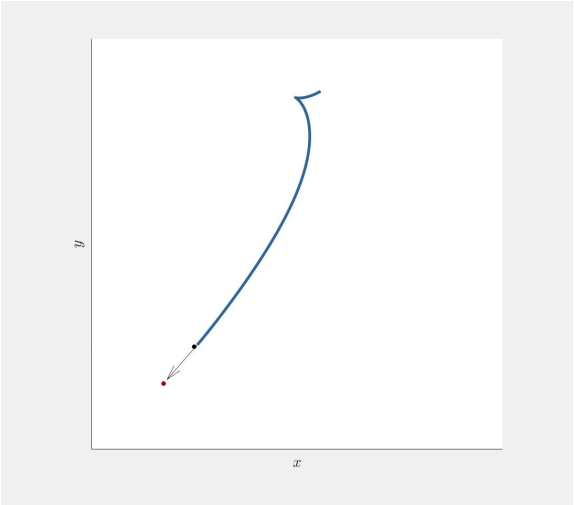
$$\begin{aligned} \underset{u_p,\delta}{\text{minimize}} \ & \left\| Bu_p - \hat{\dot{p}} \right\|^2 + \delta^2 \\ \text{subject to } & \frac{\partial V_d}{\partial d} u_d \leq -V_d(d) + \delta \\ & \frac{\partial h_d}{\partial d} u_d \geq -h_d(d) \\ & v_L \geq 0 \\ & v_R \geq 0 \end{aligned} \tag{8.31}$$

where $\hat{\dot{p}}$ is the desired velocity of the point p. The optimization program (8.31) is a convex quadratic program and, as such, can be solved very efficiently in order to evaluate the values of the control inputs v_L and v_R. δ is a slack variable which allows us to prioritize the safety constraint (enforced using h_d) over the stability constraint (enforced through V_d), by relaxing the latter.

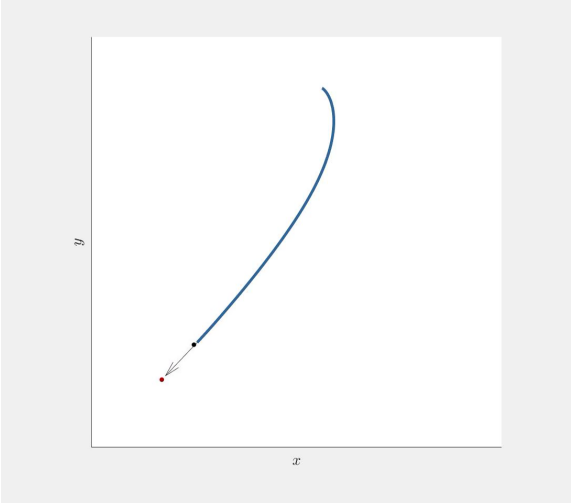
The results of the application of the described controllers are shown in Fig. 8.14, where the controller evaluated as in (8.24) (top figure) is compared with the controller obtained by zeroing negative values of v_L and v_R (bottom left) and the controller $v_L^\star$ and $v_R^\star$, solutions of the optimization problem (8.31). In the latter case, the values of v_L and v_R are never negative, but rather the value of l is changed in order to allow for the positivity constraint on v_L and v_R to be enforced.

As an example of an algorithm deployed on a real swarm of brushbots, in the following, we consider the coverage control algorithm developed in [32]. Fig. 8.15 shows 26 differential-drive-like brushbots on the Robotarium [130] running the coverage-control algorithm in order to evenly spread out over the shown rectangular domain. The boundaries of the Voronoi cells of the brushbots are depicted as grey lines.

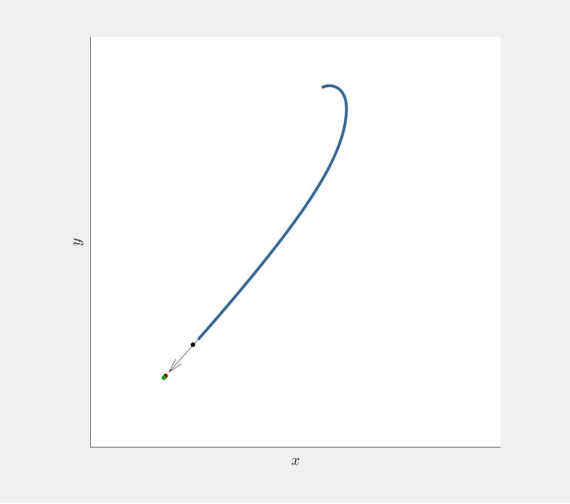
Observation 8.8. *As pointed out above, the advantages related to design simplicity and ease of assembly of the brushbots presented in this section, lead to robustness properties which are desirable for swarm robotics applications. In particular, the fact that the vibra-*



(a)



(b)



(c)

Figure 8.14: Trajectories obtained by running a proportional controller for the point p in Fig. 8.13 to reach the black dot. The orientation of the robot is denoted by the black arrow, while the position of the point p is depicted by the red dot. Notice how, in Fig. 8.14a, the trajectory makes the robot move backwards—resulting in negative values of v_L and v_R—before moving forward towards the black dot. Figure 8.14b shows the trajectory obtained by zeroing any negative value of v_L and v_R resulting from (8.24). Although this approach works in the presented case, it is not guaranteed to work in all situations. Finally, Fig. 8.14c is obtained by letting the robot execute the inputs v_L and v_R obtained by solving the QP (8.31).

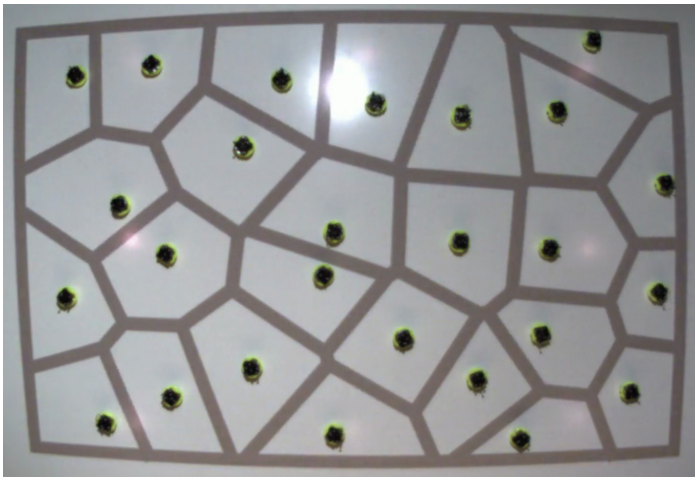


Figure 8.15: A swarm of 26 differential-drive-like brushbots (like the one shown in Fig. 8.12) performing coverage control [32]. The boundaries of the Voronoi cells corresponding to each robot are shown in grey.

tion motors do not have to be directly coupled with the brushes lets us design the robots in such a way that all the moving parts are contained in the convex hull of the robot main body (as can be seen in Fig. 8.12). This allows brushbots to tolerate collisions, even of significant magnitude, with other robots and obstacles present in the environment. In our related work [105], we show how this advantage can be used to programmatically achieve higher-level swarming behaviors, such as clustering and phase-separation.

8.4 Nanobrushbots

As mentioned at the beginning of the chapter, owing to its design simplicity, this class of vibration-driven robots lends itself to be miniaturized. Relevant applications in which small scale robots can be employed are ultra-low-energy long-term monitoring where vibration of the environment can be exploited to locomote. These include, for instance, traffic monitoring on suspended bridges or health monitoring of moving mechanical components. In fact, as the size of the brushbots reduce, it is not possible anymore to endow the robots with sensing, communication and computational modules. Nevertheless, the same mechanical design can be employed to leverage the vibration of the environment in which the micro-

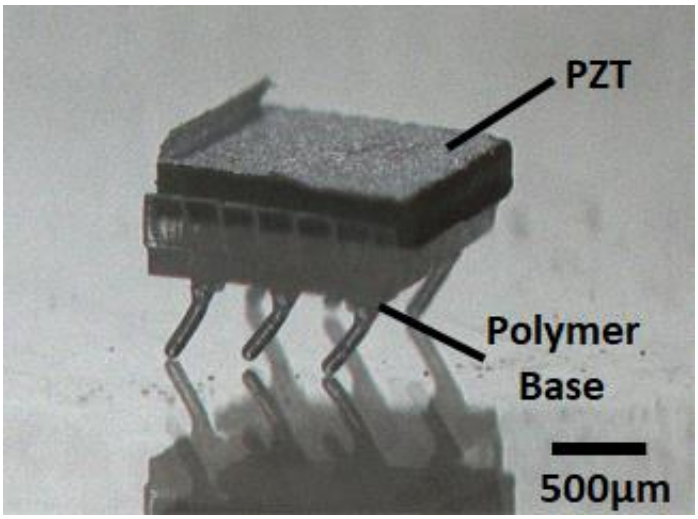


Figure 8.16: A 5 mg micro-brushbot fabricated by two-photon lithography [195].

brushbots are deployed in order to locomote.

A first attempt at miniaturizing the brushbot has been in [195] (see Fig. 8.16). Starting from this platform, we are interested in obtaining complex behavior without enduing the robots with computation, communication, sensing or active locomotion modules. In [68], we analyzed the effects of collisions among (macro-)brushbots on the formation of clusters and, in general, regions characterized by different densities, and how vibrations affect this phenomenon. In the context of structural health monitoring, these phenomena can be leveraged to localize structural imperfections which impede the motion of the micro-brushbots, allowing an external observer, to localize microscopic features of the structure. In the same way, *swarms* of micro-brushbots deployed on a suspended bridge may cluster when specific patterns of vibration—corresponding to specific traffic patterns—arise.

To validate the theories developed in [68], we manufactured 500 micro-brushbots and we deployed them on a circular vibrating platforms (see Fig. 8.16). The formation of clusters has been observed, and the characterization of its dynamics as a function of the vibration input has been carried out. The quantitative results show agreement with GPU-enabled simulations, shown in Fig. 8.18.

(a)

(b)

Figure 8.17: Cluster formation in a swarm of 500 micro-brushbots.

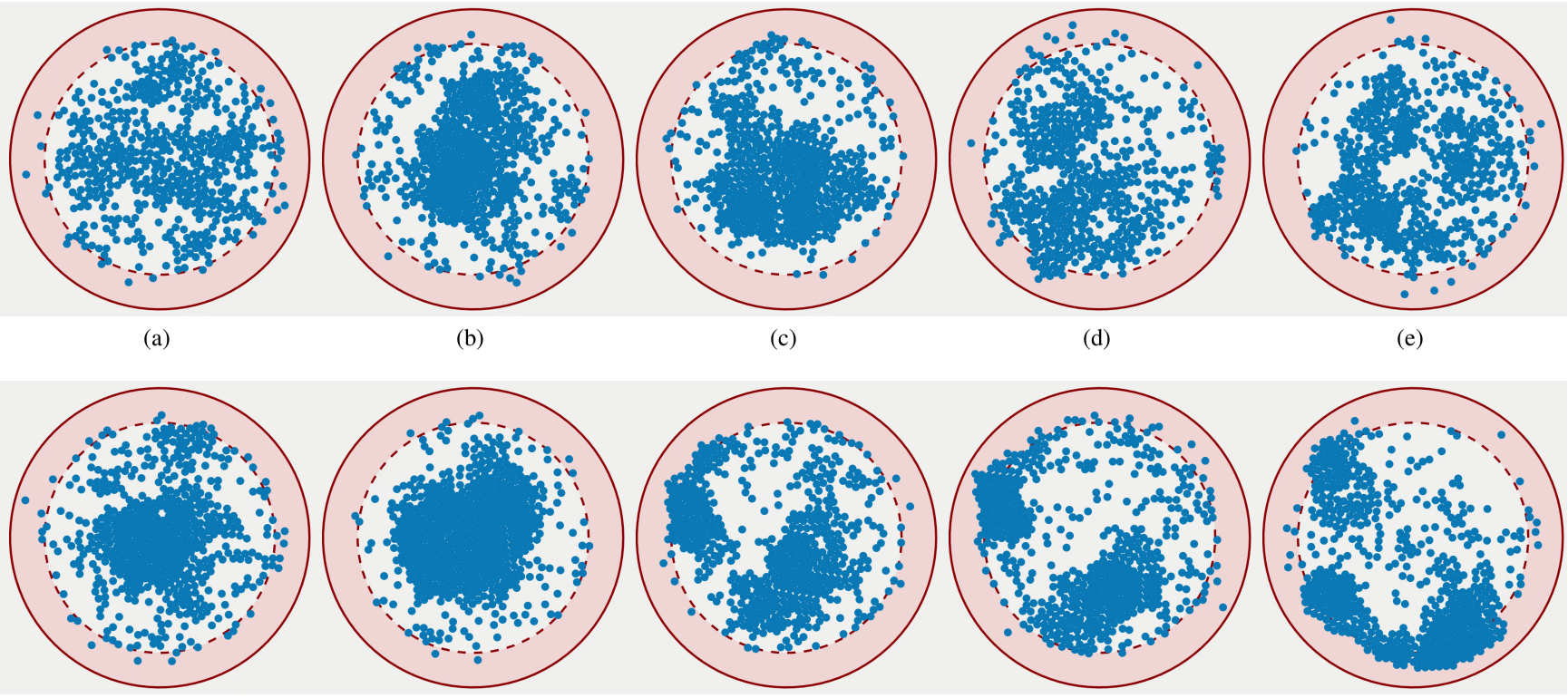


Figure 8.18: Dynamics of cluster formation and dissolution in swarms of 500 active particles, obtained using a GPU-enabled molecular dynamic simulator [68].

8.5 Conclusions

In this chapter, we presented a theoretical and experimental study of the brushbots, a class of vibration-driven robots. The use of brushes for locomotion has been investigated from a theoretical point of view, leading to the development of improved dynamic models of the brushes. Moreover, a series of experiments have been used to validate the derived theoretical models and to characterize their range of applicability. Furthermore, the design of two robotic platform is presented: a fully-actuated and a differential-drive-like brushbot. In particular, a swarm of 26 differential-drive-like brushbots has been used to showcase swarm-robotics applications in order to validate the modeling, design and control of this kind of robots. Finally, a miniaturized version of the brushbot and its applications in ultra-low-energy long-term monitoring scenarios is discussed.

CHAPTER 9
CONCLUSIONS AND FUTURE DIRECTIONS

As robots move more and more from curated laboratories and industrial settings to less structured, more dynamic and varied scenarios, new challenges arise. This is true all the more so when robots have to deployed and remain operational over truly long time horizons. Besides the obvious energetic problem, caused by the impossibility of carrying an infinite amount of energy resources, the fragility of overly optimized mechanical designs and control technologies leads to inevitable failure.

This thesis studies *long-duration robot autonomy*, a discipline that concerns itself with the deployment of robotic systems over sustained amounts of time. And it does so by analyzing it from two different, yet complementary, perspectives: control algorithms and robot design. The two parts in which this thesis is subdivided are devoted to these two aspects of long-duration robot autonomy, respectively.

In Part I, Chapter 3 defines the *persistification of robotic tasks* and describes an optimization-based control technique which, based on robot, environment and energy models, allows robotic systems to remain operational and persistently execute the tasks for which they have been deployed. The idea behind the persistification of robotic tasks is that of constraining the execution of tasks by the energy resources available at any time instant by the robots. In Chapter 4, this constraint-centric view is further developed to define the *constraint-driven control paradigm*. In these settings, robotic tasks are themselves expressed as constraints—together with the energy constraints defined by the persistification of robotic tasks—within a minimum-energy optimization-based control framework.

The constraint-driven control paradigm has been demonstrated to be effective in a number of different applications involving a variety of robotic platforms, ranging from multi-robot systems and robotic manipulators. Building up on this paradigm, Chapter 5 is devoted

to the development of an *energy-aware task allocation* framework for multi-robot systems. Owing to demonstrated adaptability and resilience properties, the proposed formulation is amenable for tasks that take place over long periods of time. The proposed task allocation algorithm is based on the idea that allocations of tasks among different robots can be effectively realized by allowing different robots to differently prioritize a stack of tasks they are required to execute. This way, both the allocation and the control input required to executed the allocated task are generated by means of an optimization-based control framework, where survivability constraints—which encode the task persistification—can be considered holistically with the allocation of tasks.

In Chapter 6, we considered another aspect of energy-awareness in long-duration robot autonomy, and developed a *communication-constrained distributed estimation algorithm* to be employed by a multi-robot system deployed in an environment with the objective of estimating an environmental phenomenon (such as temperature, light intensity, and occupancy map of the environment). The motivation for studying distributed estimation processes with communication constraints is that, after mobility, in this kind of applications, communication can be a non-negligible energy sink. Therefore, an algorithm to limit the amount of energy required by communication modules of autonomous mobile sensors is devised.

Part II of this thesis is dedicated to the study of robot design principles amenable for long-duration autonomy. In particular, the *SlothBot* and the *brushbots* are presented. The former is a slow-paced solar-powered wire-traversing robot envisioned for long-term environmental monitoring applications. In particular, in Chapter 7, two mechanical designs are proposed for such a platform, which are capable of moving on a mesh of wires by fail-safely switching between different branches of wires, and which move on a single wire, respectively. Besides the mechanical structure, techniques to control these platforms to move on a mesh of wires during environmental monitoring applications are presented. Chapter 8 concludes Part II and the main content of this thesis with the study of the interconnected

mechanics, dynamics, and control of the brushbots, a class of vibration-driven robots which can leverage vibrational energy of the environment in which they are deployed in order to locomote and operate. Validation of the developed structural models are presented, and the deployment of a swarm of brushbots is showcased both at macro- and a micro-scale.

To conclude, despite the fact that, in this thesis, the design of control algorithms and of mechanical structures of robots have been discussed in two distinct parts, we firmly believe that these should rather be considered as *two intimately intertwined processes of the creation and operation of robotic platforms*. This idea characterizes research areas such as *morphological programming*, where the mechanical design of a robot determines and implements the desired control actions. The opposite, where control determines the structure of a robot, has only started to be explored. *Co-design* principles, for instance, dictate the tight coordination of mechanics and control during design phases. However, their interplay during the operation of robotic systems is not fully understood yet. It is embracing the described philosophy that future research will be carried out in order to leverage the advantages of the synergy between mechanical intelligence and intelligent control for the development of robots truly capable of being self-sustained and survive under most (not necessarily all) conditions faced during their lives.

Appendices

APPENDIX A

PROOFS

Proof of Theorem 3.9. Given the properties of class $\mathcal{K}$ functions [196] and using the fact that $h_1(x)$ has relative degree ρ, $\dot{h}_\rho(x)$ is given by the following expression:

$$\dot{h}_\rho = L_f^\rho h_1(x) + L_g L_f^{\rho-1} h_1(x) u + \sum_{i=1}^{\rho-1} \sum_{\mathscr{C} \in \binom{\rho-1}{i}} \prod_{j \in \mathscr{C}} \frac{\partial \alpha_j}{\partial h_j} L_f^{\rho-i} h_1(x). \tag{A.1}$$

Consequently, the choice of $u \in K_\rho(x)$ renders the set $\mathcal{C}_\rho = \{x \in \mathbb{R}^n : \ h_\rho(x) \geq 0\}$ forward invariant. By recursively applying (3.22) $\rho - 1$ times, $\mathcal{C}_1$ is proved to be forward invariant. □

Proof of Lemma 3.12, following the proof of Theorem 1 in [44]). If h is a time-varying CBF on $\mathcal{D} \times \mathbb{R}_+$ and $u \in K(x,t)$, from (3.25) and (3.26) we can derive the following differential inequality:

$$\dot{h}(x,t) \geq -\alpha(h(x,t)). \tag{A.2}$$

Now, consider the following boundary condition problem:

$$\begin{cases} \dot{\zeta} = -\alpha(\zeta) \\ \zeta(t_0) = h(x(t_0), t_0) > 0, \end{cases} \tag{A.3}$$

whose solution is given by $\zeta(t) = \beta(\zeta(t_0), t - t_0)$, β being a class $\mathcal{KL}$ function (Lemma 4.4 in [147]). From (A.2), making use of the Comparison Lemma (Lemma 3.4 in [147]), we have that:

$$h(x(t), t) \geq \beta(\zeta(t_0), t - t_0), \ \ \forall t \geq t_0. \tag{A.4}$$

Hence, if $h(x(t_0), t_0) > 0$ and therefore $x(t_0) \in \mathcal{C}(t)$, using the properties of class $\mathcal{KL}$

functions, (A.4) ensures that $h(x(t),t) > 0\ \forall t \geq t_0$ and so $x(t) \in \mathcal{C}(t)\ \forall t \in [t_0, t_0 + \Delta t_{\max}(x_0)]$. Thus, $\mathcal{C}(t)$ is forward invariant. □

Proof of Theorem 3.16. Given the properties of class $\mathcal{K}$ functions and using the fact that $h_1(x) = -L_f V(x)$ has relative degree $\rho - 1$, the following expression of $\dot{h}_\rho$ can be derived:

$$\dot{h}_\rho(x) = -L_f^\rho V(x) - L_g L_f^{\rho-1} V(x) u + \sum_{i=1}^{\rho-2} \sum_{\mathscr{C} \in \binom{\rho-2}{i}} \prod_{j \in \mathscr{C}} \frac{\partial \alpha_j}{\partial h_j} (-L_f^{\rho-1-i} V(x)). \tag{A.5}$$

The choice of $u \in K'_\rho(x)$ will render the set $\mathcal{C}_\rho = \{x \in \mathbb{R}^n : \ h_\rho(x) \geq 0\}$ forward invariant. Similarly to what has been done in Theorem 3.9, by the recursive application of (3.28), $\mathcal{C}_1$ is proven to be forward invariant. Then, by LaSalle's Theorem (Theorem 4.4 in [147]), as $\{x^*\}$ is the largest invariant set in $\partial\mathcal{C}_1$, one has that $x(t) \to x^*$ as $t \to \infty$. □

Proof of Theorem 3.19. By Theorem 3.18, the controller (3.38) results in the forward invariance, i.e., safety, of the set $\mathcal{S}$. Therefore, we only need to confirm that, if the controller (3.38) is safe, then u will track the control signal $k(x,t)$. To this end, consider the following time-varying Lyapunov function:

$$V(u,x,t) = \frac{1}{2}\|u - k(x,t)\|^2. \tag{A.6}$$

Its time derivative evaluates to:

$$\begin{aligned}\dot{V} &= \frac{\partial V}{\partial u}\dot{u} + \frac{\partial V}{\partial k}\dot{k} \\ &= \frac{\partial V}{\partial u}\phi_k(x,u,t) + \frac{\partial V}{\partial k}L_f k(x,t) + \frac{\partial V}{\partial k}\frac{\partial k}{\partial t} \\ &= (u-k(x,t))^T\left(\phi_k(x,u,t) - L_f k(x,t) - \frac{\partial k}{\partial t}\right) \\ &= (u-k(x,t))^T\left(\frac{\alpha}{2}\left(k(x,t)-u\right)\right) \\ &= -\frac{\alpha}{2}\|u-k(x,t)\|^2 \\ &= -\alpha V(u,x,t)\end{aligned} \tag{A.7}$$

where we used the assumption that the controller $\phi_k(x,u,t)$ is safe, that implies $v^* = 0$. Thus, $V \to 0$ or, equivalently, $u \to k(x,t)$, i.e., the input u will track the control signal $k(x,t)$. □

Proof of Proposition 3.26. Follows from the application of Theorem 3.9, Lemma 3.12 and Theorem 3.16. □

Proof of Theorem 4.1. Similar to Theorem 4.3 in [131]. □

Proof of Theorem 4.2, based on [116]). See [46] for the forward invariance. Let

$$V(x) = \begin{cases} -h(x) & x \in \mathbb{R}^n \setminus \mathcal{S} \\ 0 & x \in \mathcal{S} \end{cases} \tag{A.8}$$

be a control Lyapunov candidate function. Thus, $V(x) > 0$ for $x \in \mathbb{R}^n \setminus \mathcal{S}$ and $V(x) = 0$ for $x \in \mathcal{S}$. Moreover,

$$\dot{V} = \frac{\partial V}{\partial x}\dot{x} = L_f V(x) + L_g V(x)u = \begin{cases} -L_f h(x) - L_g h(x)u & x \in \mathbb{R}^n \setminus \mathcal{S} \\ 0 & x \in \mathcal{S}. \end{cases} \tag{A.9}$$

Furthermore, since h is continuously differentiable, V is continuously differentiable as

well. Then, by hypothesis, $\dot{V} = -L_f h(x) - L_g h(x) u \leq \alpha(h(x)) < 0$ for $x \in \mathbb{R}^n \setminus \mathcal{S}$ and $\dot{V} = 0$ for $x \in \mathcal{S}$. By Theorem 4.2, $\mathcal{S}$ is forward invariant. Moreover, $\mathcal{S}$ is closed, since it is the inverse image of the closed set $[0, \infty) \subseteq \mathbb{R}$ under the continuous map h. Therefore, by Theorem 2.8 in [197], the system (4.1) is uniformly globally asymptotically stable with respect to the set $\mathcal{S}$. Thus, there exists a class $\mathcal{KL}$ function β [196] such that, given any initial state x_0, the solution $x(t)$ satisfies $d(x(t), \mathcal{S}) \leq \beta(d(x_0, \mathcal{S}), t),\ \forall t \geq 0$, where $d(y, \mathcal{S}) \triangleq \inf_{z \in \mathcal{S}} \|y - z\|$. Hence, as $t \to \infty$, $x(t) \to \in \mathcal{S}$. □

Proof of Proposition 4.4. The KKT conditions for the problem in (4.15) are

$$\begin{cases} -\dfrac{\partial h}{\partial x} u^* - \alpha(h(x)) - \delta^* \leq 0 \\ \lambda^* \geq 0 \\ \lambda^* \left(-\dfrac{\partial h}{\partial x} u^* - \alpha(h(x)) - \delta^* \right) = 0 \\ \begin{bmatrix} 2u^* \\ 2\delta^* \end{bmatrix} + \lambda^* \begin{bmatrix} -\dfrac{\partial h}{\partial x}^T \\ -1 \end{bmatrix} = 0, \end{cases} \tag{A.10}$$

where u^*, δ^* and λ^* are primal and dual optimal points [121]. First of all, we note that, if $\lambda^* = 0$, then $u^* = 0$ by the fourth equation in (A.10). Therefore, from the first equation in (A.10), $-\alpha(h(x)) \leq 0$. This is equivalent to $-\alpha(-J(x)) \leq 0$ and, since $J(x) \geq 0$, this implies that $J(x) = 0$. In case $\lambda^* > 0$, from the third and fourth equation in (A.10), one has $\lambda^* = -2\alpha(h(x)) \left(1 + \|\frac{\partial h}{\partial x}\|^2\right)^{-1}$, and therefore, $u^* = -\alpha(h(x)) \frac{\partial h}{\partial x}^T \left(1 + \|\frac{\partial h}{\partial x}\|^2\right)^{-1}$. Since $J(x) \geq 0\ \forall x \in \mathbb{R}^n$ and J is continuously differentiable, one can show that $J(\bar{x}) = 0 \Rightarrow \left.\frac{\partial J}{\partial x}\right|_{x=\bar{x}} = 0$. Thus, we can unify the two cases, $\lambda^* = 0$ and $\lambda^* > 0$, and write the expression of the optimal u as follows:

$$u^* = \frac{\alpha(-J(x)) \frac{\partial J}{\partial x}^T}{1 + \|\frac{\partial J}{\partial x}\|^2}. \tag{A.11}$$

With this expression of the input u, the evolution in time of the cost J is given by

$$\dot{J} = \frac{\partial J}{\partial x}\dot{x} = \frac{\partial J}{\partial x}u^* = \frac{\alpha(-J(x))\|\frac{\partial J}{\partial x}\|^2}{1+\|\frac{\partial J}{\partial x}\|^2}. \tag{A.12}$$

So,

$$\frac{\partial J}{\partial x} \neq 0 \Rightarrow \dot{J} < 0 \qquad \text{and} \qquad \frac{\partial J}{\partial x} = 0 \Rightarrow \dot{J} = 0. \tag{A.13}$$

Hence, as $t \to \infty$, $x(t) \to x^*$, such that $\frac{\partial J}{\partial x}(x^*) = 0$. □

Proof of Corollary 4.16. Proceeding similarly to the proof of Proposition 4.4, the solution to (4.16) evaluates to

$$u^* = \frac{\alpha(-J(x))\frac{\partial J}{\partial x}^T}{\|\frac{\partial J}{\partial x}\|^2}, \tag{A.14}$$

and, therefore, $\dot{J} = \alpha(-J(x))$. Thus, $J \to 0$ as $t \to \infty$ [147]. Hence, as $t \to \infty$, $x(t) \to x^*$ such that $\frac{\partial J}{\partial x}(x^*) = 0$, and so $J(x^*) = 0$. □

Proof of Proposition 4.6. Similarly to what has been done in Propositon 4.4, it can be shown that

$$\dot{J} = \frac{-c(J(x))^\gamma\|\frac{\partial J}{\partial x}\|^2}{1+\|\frac{\partial J}{\partial x}\|^2}. \tag{A.15}$$

Thus, by Theorem 4.1, we conclude that

$$\frac{\partial J}{\partial x} \neq 0 \Rightarrow J \to 0 \text{ in finite time}, \tag{A.16}$$

and

$$\frac{\partial J}{\partial x} = 0 \Rightarrow \dot{J} = 0. \tag{A.17}$$

Hence, $x \to x^*$, with $\frac{\partial J}{\partial x}(x^*) = 0$, in finite time. Indeed, as shown in [146], $h(x)$ such that $\dot{h} \geq -c(h(x))^\gamma$ is a finite-time convergence control barrier function for the system characterized by single integrator dynamics, $\dot{x} = u$. □

Proof of Proposition 4.7. Proposition 4.4 ensures that, by imposing the global constraint

$-\frac{\partial J}{\partial x}u \geq -\alpha(-J(x))$, constructed using the whole state vector x, the cost J is decreasing towards a stationary point. We want to show that, by imposing only local constraints (i. e., such that robot i only needs information about its neighbors), the multi-robot system is able to enforce the global constraint and, hence, to minimize the cost J in a decentralized fashion.

We proceed by starting to sum up the constraints for each robot, obtaining:

$$\begin{aligned}\sum_{i=1}^{N}\left(-\frac{\partial J_i}{\partial x_i}u_i\right) &\geq \sum_{i=1}^{N}(-\alpha(-J_i(x)) - \delta_i) \\ \geq -\alpha\left(-\sum_{i=1}^{N}J_i(x)\right) - \delta &\geq -\alpha\left(-J(x)\right) - \delta,\end{aligned} \tag{A.18}$$

where we used the superadditivity property of α, and we set $\delta = \sum_{i=1}^{N}\delta_i$. Moreover, since the graph $\mathcal{G}$, which encodes the neighboring relations between the robots, is undirected, we have that $\frac{\partial J_i}{\partial x_i} = \frac{1}{2}\frac{\partial J}{\partial x_i}$. Thus,

$$\frac{\partial J}{\partial x}u \geq -2\alpha\left(-J(x)\right) - 2\delta = -\alpha'\left(-J(x)\right) - \delta', \tag{A.19}$$

where $\frac{\partial J}{\partial x} = \left[\frac{\partial J}{\partial x_1}, \ldots, \frac{\partial J}{\partial x_N}\right]$, $u = \left[u_1^T, \ldots, u_N^T\right]^T$, and α' an extended class $\mathcal{K}$ function. Hence, by Proposition 4.4, x will converge to a stationary point of J.

Finally, we note that a class $\mathcal{K}$ function $\alpha(x) = cx^\gamma$, defined for $x < 0$ is convex, and hence superadditive, for $x < 0$. Applying Proposition 4.6, the statement holds. □

Proof of Proposition 5.8. Solving (5.19) at time k yields $u^{(k)}$, $\delta^{(k)}$, and $\alpha^{(k)}$. At time $k+1$, by Proposition 3 in [10] where $\alpha = \alpha^{(k)}$ and $J_m(x) = -h_m(x)$, if $\alpha^{(k+1)} = \alpha^{(k)}$ and $\delta^{(k+1)} = \delta^{(k)}$, then $\|u^{(k+1)}\| < \|u^{(k)}\|$ is obtained using (5.33). Let

$$V(\alpha, u, \delta) = \sum_{i=1}^{n_r}\left(C\|\Pi_i\alpha_{-,i}\|^2 + \|u_i\|^2 + l\|\delta_i\|_{S_i}^2\right), \tag{A.20}$$

be a candidate Lyapunov function for the multi-robot system controlled via the solutions of

the optimization problem (5.19) (see scheme in Fig. 5.3). Notice that V is equal to the cost (5.19a) and it is positive definite since S_i is positive definite for all i by assumption. Then, one has:

$$\begin{aligned} V^{(k+1)} &= V(\alpha^{(k+1)}, u^{(k+1)}, \delta^{(k+1)}) \\ &\leq V(\alpha^{(k)}, u^{(k+1)}, \delta^{(k)}) \\ &< V(\alpha^{(k)}, u^{(k)}, \delta^{(k)}) = V^{(k)}. \end{aligned} \tag{A.21}$$

Therefore, $V^{(k)} \to 0$ as $k \to \infty$. Thus, $u^{(k)} \to 0$ as $k \to \infty$, and $x_i^{(k)} \to x_{m,i}^\star$ for some m, by the driftless assumption on the robot model (5.31). □

Proof of Proposition 5.9. For notational convenience, we let the $\bar{\alpha} = \begin{bmatrix} \alpha_{-,1}^T & \dots & \alpha_{-,n_r}^T \end{bmatrix}^T \in \{0,1\}^{n_t n_r}$ be the vector composed of the stacked columns of α, and

$$\bar{\Phi} = \mathbb{1}_{n_r} \otimes \Phi \tag{A.22}$$

$$\bar{\Theta} = \mathbb{1}_{n_r} \otimes \Theta \tag{A.23}$$

$$\bar{\Psi} = \mathbb{1}_{n_r} \otimes \Psi, \tag{A.24}$$

$\otimes$ denoting the Kronecker product. From (5.20) and with the notation introduced above, one has that

$$\bar{\Phi}\bar{\alpha} \geq 0, \tag{A.25}$$

where the symbol $\geq$ is always intended component-wise. Then, as $\delta \in \mathbb{R}_{\geq 0}$ (see discussions in [7] and [10]), the constraints (5.19b) and (5.19c) in (5.19) can be re-written as follows:

$$\delta^{(k)T} L_f h(x^{(k)}) + \delta^T L_g h(x^{(k)}) u^{(k)} \geq -\delta^{(k)T} \gamma(h(x^{(k)})) - \delta^{(k)T} \delta^{(k)} \tag{A.26}$$

$$\bar{\alpha}^T \bar{\Phi}^T \bar{\Theta} \bar{\delta}^{(k)} + \bar{\alpha}^T \bar{\Phi}^T \bar{\Phi} \bar{\alpha}^{(k)} \leq \bar{\alpha}^T \bar{\Phi}^T \bar{\Psi}. \tag{A.27}$$

Similarly, the constraints (5.19d) to (5.19g), can be re-written as

$$\bar{\alpha}^{(k)T} A_\alpha^T A_\alpha \bar{\alpha}^{(k)} \leq \bar{\alpha}^{(k)T} A_\alpha^T b_\alpha \tag{A.28}$$

$$\delta^{(k)T} A_\delta^T A_\delta \delta^{(k)} \leq \delta^{(k)T} A_\delta^T b_\delta. \tag{A.29}$$

Then, define the following candidate Lyapunov function:

$$V(x) = \gamma(h(x))^T \gamma(h(x)), \tag{A.30}$$

where $h(x) = [h_1(x), \ldots, h_{n_t}(x)]^T$ and $\gamma(h(x))$ is intended as a component-wise application of the extended class $\mathcal{K}_\infty$ function to the vector $h(x)$. We want the following condition on its time derivative to be satisfied at every time step k

$$\begin{aligned} \dot{V}(x^{(k)}, u^{(k)}) &= 2\gamma(h(x^{(k)}))^T \frac{\mathrm{d}\gamma}{\mathrm{d}h} \frac{\mathrm{d}h}{\mathrm{d}x} f(x^{(k)}) + 2\gamma(h(x^{(k)}))^T \frac{\mathrm{d}\gamma}{\mathrm{d}h} \frac{\mathrm{d}h}{\mathrm{d}x} g(x^{(k)}) u^{(k)} \\ &\leq -cV(x^{(k)}), \end{aligned} \tag{A.31}$$

with $c \in \mathbb{R}_{>0}$.

Defining $\varphi^{(k)} = [\gamma(h(x^{(k)})), u^{(k)}, \delta^{(k)}, \bar{\alpha}^{(k)}, 1]^T$, the inequalities (A.31), (A.26), (A.27), (A.28), (A.29) can be compactly written as follows, respectively:

$$\varphi^{(k)T} B_0^{(k)} \varphi^{(k)} \leq 0 \tag{A.32}$$

$$\varphi^{(k)T} B_1^{(k)} \varphi^{(k)} \leq 0 \tag{A.33}$$

$$\varphi^{(k)T} B_2^{(k)} \varphi^{(k)} \leq 0 \tag{A.34}$$

$$\varphi^{(k)T} B_3^{(k)} \varphi^{(k)} \leq 0 \tag{A.35}$$

$$\varphi^{(k)T} B_4^{(k)} \varphi^{(k)} \leq 0, \tag{A.36}$$

where B_0, B_1, B_2, B_3, and B_4 are defined in (5.35).

Thus, applying the S-procedure [121], the linear matrix inequality (5.34) in the vari-

ables τ_1, τ_2, τ_3, τ_4 is obtained. If a solution to (5.34) exists for all k, then (A.31) is satisfied for all k, and therefore $V(x^{(k)}) \to 0$. Consequently $x^{(k)}$ converges as $k \to \infty$, and so do the sequences $\{u^{(k)}\}_{k\in\mathbb{N}}$, $\{\delta^{(k)}\}_{k\in\mathbb{N}}$, and $\{\alpha^{(k)}\}_{k\in\mathbb{N}}$, solution of (5.19), parameterized by $x^{(k)}$. □

Proof of Theorem 7.1. Let us start proving that (7.12) ⇒ (7.13), with which we mean that a solution to (7.12) is also a solution of (7.13).
Let $C(p_i) = \prod_{j=1}^{N_\text{w}} (a_j^T p_i + b_j)$. This way we can describe the wire-traversing constraints as follows:

$$p_i \in \mathcal{G} \quad \Leftrightarrow \quad C(p_i) = 0,\ p_i \in X. \tag{A.37}$$

Writing the Lagrangian for the constrained minimization problem (7.12), one obtains:

$$L(p_1, \ldots, p_N, \lambda) = J(p_1, \ldots, p_N) + \sum_{i=1}^{N} \lambda_i C(p_i), \tag{A.38}$$

where $\lambda = [\lambda_1, \ldots, \lambda_N]^T$ is the Lagrange multiplier. Let $p_1^\star, \ldots, p_N^\star$ be a local minimizer of (7.12). The following necessary condition has to be satisfied ([121]):

$$\begin{aligned} \frac{\partial L}{\partial p_i}(p_i^\star) = \frac{\partial J}{\partial p_i}(p_i^\star) + \lambda_i \sum_{k=1}^{N_\text{w}} a_k^T \prod_{\substack{j=1 \\ j\neq k}}^{N_\text{w}} (a_j^T p_i^\star + b_j) = 0 \\ \forall i = \{1, \ldots, N\}. \end{aligned} \tag{A.39}$$

Assume that robot i is on wire $\bar{k}$: as a result $a_{\bar{k}}^T p_i^\star + b_{\bar{k}} = 0$. So, (A.39) reduces to:

$$\begin{aligned} \frac{\partial L}{\partial p_i}(p_i^\star) = \frac{\partial J}{\partial p_i}(p_i^\star) + \lambda_i a_{\bar{k}}^T \prod_{\substack{j=1 \\ j\neq \bar{k}}}^{N_\text{w}} (a_j^T p_i^\star + b_j) = 0 \\ \forall i = \{1, \ldots, N\}. \end{aligned} \tag{A.40}$$

From [129] we know that $\frac{\partial J}{\partial p_i}(p_i^\star) \parallel (\rho_i - p_i^\star)$, where $\parallel$ is the parallel symbol. Since

$\lambda_i \prod_{\substack{j=1 \\ j\neq\bar{k}}}^{N_\mathrm{w}} (a_j^T p_i^\star + b_j) \in \mathbb{R}$ is a scalar, we have that $(\rho_i - p_i^\star) \parallel a_{\bar{k}}$. So, $(\rho_i - p_i^\star)$ is orthogonal to the wire $\bar{k}$ and, therefore, $p_i^\star$ minimizes the distance from ρ_i. From [129], we know that $\rho_i,\ i = 1, \ldots, N$ are solutions of (7.7). Hence, $p_i^\star$ is a local minimizer of (7.13).

We now prove that (7.13) $\Rightarrow$ (7.12), i.e. a local minimizer of (7.13) is also a local minimizer of (7.12). The constraints on (7.13) are equivalent to $p_i^\mathrm{U} = \rho_i \ \forall i \in \{1, \ldots, N\}$, as shown in [32]. Substituting this expression of p_i^U in (7.13), one obtains the following unconstrained minimization problem:

$$\min_{p_i^\mathrm{C} \in \mathcal{G}} \left\| p_i^\mathrm{C} - \rho_i \right\|, \tag{A.41}$$

whose solution $p_i^{\mathrm{C}\star}$ is the closest point to ρ_i that is on the wires defined by $\mathcal{G}$. This means that $\left(p_i^{\mathrm{C}\star} - \rho_i\right) \parallel a_k$ for some $k \in \{1, \ldots, N_\mathrm{w}\}$. Since also $\frac{\partial J}{\partial p_i}\left(p_i^{\mathrm{C}\star}\right) \parallel \left(p_i^{\mathrm{C}\star} - \rho_i\right)$, $\exists \lambda_i \in \mathbb{R} \ \Big|\ \frac{\partial L}{\partial p_i}\left(p_i^{\mathrm{C}\star}\right) = 0$. Hence, a solution of (7.13) is also a local minimizer of (7.12). $\square$

Proof of Theorem 7.10. By the Definition 7.3 of conformal map, f^{-1} exists and it is continuous as it is the map f itself. As the operator $\Re(\cdot)$ is continuous and the composition of continuous functions is continuous, one has that m_{kj} is continuous for the vertical strip of the complex plane defined by $w_1 < \Re\left(f^{-1}(x)\right) < w_2$. For $\Re\left(f^{-1}(x)\right) \leq w_1$ and $\Re\left(f^{-1}(x)\right) \geq w_2$, m_{kj} is constant and so continuous. For $\Re\left(f^{-1}(x)\right) = w_1$, $f\left(\Re\left(f^{-1}(x)\right)\right) = f\left(w_1\right) = p_{kj}^{(1)}$ by Definition 7.9 of f_{kj}. Hence m_{kj} is continuous on the vertical line of the complex plane defined by $\Re\left(f^{-1}(x)\right) = w_1$. A similar argument holds for when $\Re\left(f^{-1}(x)\right) = w_2$. Hence, the mapping $\widetilde{M}$ is continuous over each triangular domain.

Now the continuity of m_{kj} across adjacent domains T_{kj_1} and T_{kj_2} or $T_{k_1 j_n}$ and $T_{k_2 j_m}$ is left (see Fig. A.1). Let us define $l_{kj_{12}}$ to be the common segment of the two adjacent regions $T_{kj_1}, T_{kj_2} \subsetneq P_k$ (see Fig. A.1a). For $x \in l_{kj_{12}}$ one has that $\Re\left(f_{kj_1}^{-1}(x)\right) \geq w_2$ and $\Re\left(f_{kj_2}^{-1}(x)\right) \leq w_1$. Therefore, in the former case x is mapped to $p_{kj_1}^{(2)}$, whilst in the latter case x is mapped to $p_{kj_2}^{(1)}$. The two points coincide, hence $\widetilde{M}$ is continuous on $l_{kj_{12}}$.

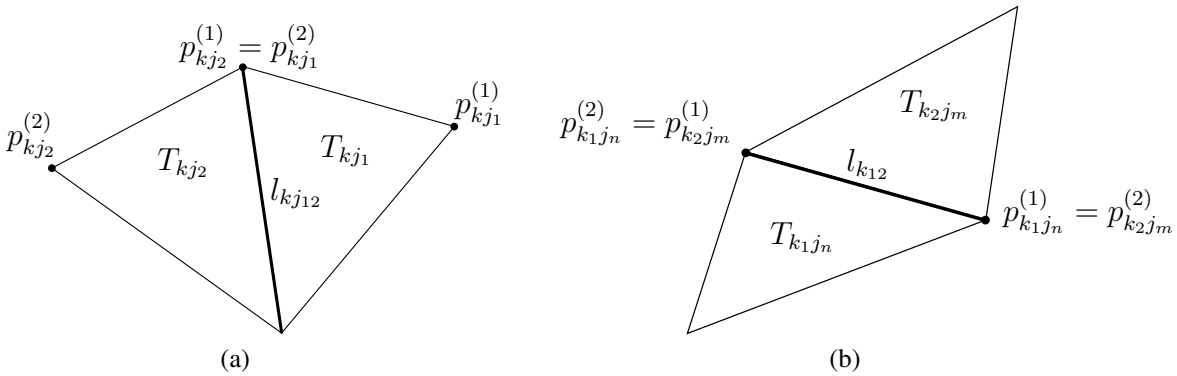


Figure A.1: Adjacent triangular regions over whose common edges the continuity of the function $\widetilde{M}$ has to be shown

Let us now define $l_{k_{12}}$ the common segment of the two adjacent regions $T_{k_1j_n} \subsetneq P_{k_1}$ and $T_{k_2j_m} \subsetneq P_{k_2}$, that is also the only common segment between the two polygons P_{k_1} and P_{k_2} (see Fig. A.1b). By Definition 7.9, for $x \in l_{k_{12}}$ one has $\Re\left(f^{-1}_{k_1j_n}(x)\right) = f^{-1}_{k_1j_n}(x)$ and $\Re\left(f^{-1}_{k_2j_m}(x)\right) = f^{-1}_{k_2j_m}(x)$. Since $w_1 < \Re\left(f^{-1}_{k_1j_n}(x)\right) < w_2$ and $w_1 < \Re\left(f^{-1}_{k_2j_m}(x)\right) < w_2$, we can write:

$$m_{k_1j_n}(x) = f_{k_1j_n}\left(\Re\left(f^{-1}_{k_1j_n}(x)\right)\right) = f_{k_1j_n}\left(f^{-1}_{k_1j_n}(x)\right) = x \tag{A.42}$$

$$m_{k_2j_m}(x) = f_{k_2j_m}\left(\Re\left(f^{-1}_{k_2j_m}(x)\right)\right) = f_{k_2j_m}\left(f^{-1}_{k_2j_m}(x)\right) = x. \tag{A.43}$$

So $m_{k_1j_n}(x) = m_{k_2j_m}(x)\ \forall x \in l_{k_{12}}$. Hence, $\widetilde{M}$ is continuous on $l_{k_{12}}$. □

Proof of Proposition 7.22. Expanding the L^2 norm in (7.60), one has:

$$\begin{aligned} &\|\mathscr{F}(\theta) - \mathscr{F}(\phi)\|^2_{L^2(\mathcal{I}^N)} \\ &= \int_{\mathcal{I}^N} |\mathscr{F}(\theta) - \mathscr{F}(\phi)|^2 \\ &= \int_{\mathcal{I}^N} \left| \sum_{i \in \mathcal{N}} \int_{\mathcal{V}_i(P)} |p_i - q|^2 \underbrace{(\theta(q) - \phi(q))}_{\circledast} \, \mathrm{d}q \right|^2. \end{aligned} \tag{A.44}$$

As the term $\circledast$ in (A.44) is positive, the minimum of $\|\mathcal{F}(\theta) - \mathcal{F}(\phi)\|^2_{L^2(\mathcal{I}^N)}$ is achieved when the difference $\theta(q) - \phi(q)$ is minimized. As θ has to be concave and $\theta > \phi$, by

Definition 7.21, θ that minimizes (A.44) is the *concave envelope* of ϕ. $\square$

APPENDIX B

HARDWARE SPECIFICATIONS

Table B.1: Hardware specification of the multi-body design of the SlothBot

Component	**Purpose**	**Specifications**
ESP32	Computing	Up to 240 MHz clock speed, 520 kB RAM, IEEE 802.11 b/g/n Wi-Fi, Bluetooth v4.2 BLE, 16 GPIO pins, 4 SPI, 2 I^2S, 2 I^2C, 3 UART, 18 12-bit ADC, 2 8-bit DAC, 10 touch sensors
DC motors	Locomotion	0.8826 Nm maximum torque, 1000:1 reduction ratio, 6 V operating voltage, 1.6 A stall current
Motor controller	DC motor speed control	2.7-10.8 V input voltage range, 1.2 A maximum current per motor
Servo motors	Wire switching	0.1598 Nm maximum torque, 5 V operating voltage
Battery	System power supply	1000 mAh, 2-cell LiPo, 25C discharge rate

Table B.2: Cost breakdown of the multi-body design of the SlothBot

Component	**Quantity**	**Cost per unit**	**Details**
DC motors	2	$22.95	-
Servo motors	5	$3.75	-
Electronics	-	$26.85	ESP32, motor controllers, voltage regulators
Various	-	$9.95	Battery, 3D printed chassis
	$101.45		

Table B.3: Hardware specification of the single-body design of the SlothBot

Component	**Purpose**	**Specifications**
Rasperry PI Zero W	Computing and data streaming	Up to 1 GHz clock speed, 512 MB RAM, IEEE 802.11 b/g/n Wi-Fi, Bluetooth v4.1 BLE, Mini HDMI and USB On-The-Go ports, Micro USB power, HAT-compatible 40-pin header
Teensy 3.2	Interface with sensors and actuators	Up to 96 MHz clock speed, 64 kB RAM, 2 kB EEPROM, 34 GPIO pins, 21 13-bit ADC, 1 12-bit DAC, 12 touch sensing inputs, 12 PWM outputs, USB, 3 serial ports, 1 SPI, 2 I^2C
Solar panel	Solar energy harvesting	6 V, 9 W
Solar charger MPPT circuit	Power balance and battery charging	3.7-4.2 V LiPo batteries, 1 A maximum charge rate, temperature monitoring, low-battery detection
Voltage regulator	Power regulation	For DC motors: 12 V output voltage, 5 A maximum input current, 2.9 V minimum operating voltage, 8 mA maximum quiescent current; For all other electronic components: 5 V output voltage, 1.2 A maximum input current, 0.5 V minimum operating voltage, 3 mA maximum quiescent current.
Power sensor	Battery and circuit power monitoring	0-36 V bus voltage range, 15 A maximum input current, I^2C interface with programmable addresses
DC motors	Locomotion	2.2555 Nm maximum torque, 378:1 reduction ratio, 12 V operating voltage, 14 RPM and 100 mA (no-load performance), 1.1 A stall current, 48 CPR quadrature encoder
Motor controller	DC motor speed control	4.5-28 V input voltage range, 5 A peak output current per channel, 2.6 A continuous output current, 20 kHz maximum PWM frequency
External sensors	Distance and environmental data collection	Measured quantities: distance (ultrasonic technology), temperature, humidity, pressure, altitude, full-spectrum light, visible light, infrared light, illuminance, eCO2, TVOC; 5 V maximum input voltage, I^2C interface
External LEDs	Visual interface	2 individually programmable RGB LEDs, 5 V maximum input voltage
Battery	System power supply	10000 mAh, 1-cell LiPo

Table B.4: Cost breakdown of the single-body design of the SlothBot

Component	Quantity	Cost per unit	Details
Raspberry PI Zero W	1	$17.00	-
Teensy 3.2	1	$19.95	-
Solar panel	1	$78.95	-
Solar charger MPPT circuit	1	$17.50	-
Voltage regulator (DC motors)	1	$13.95	-
Voltage regulator (all other electronic components)	2	$4.95	-
Power sensor	2	$9.95	-
DC motors	2	$34.95	-
Motor controllers	2	$8.49	-
External sensors	-	$73.38	2 ultrasonic sensors, 1 temperature and pressure sensor, 1 luminosity sensor, 1 air quality sensor
External LEDs	2	$1.99	-
Battery	1	$23.99	-
Various	-	$500.00	Printed circuit board, wheels, components for waterproofing, acrylic components, fasteners
	$865.38		

Table B.5: Hardware specification of the differential-drive design of the brushbot

Component	**Purpose**	**Specifications**
ESP8266	Computing	Up to 160 MHz clock speed, 32 kB RAM, IEEE 802.11 b/g/n Wi-Fi, 16 GPIO pins, SPI, I^2C, 10-bit ADC
Eccentric rotating mass vibration motors	Locomotion	3 V operating voltage, 70 mA operating current, 7500 RPM
Motor controller	DC motor speed control	2 full bridges, 2-7 V input voltage range, 1.6 A peak current, PH/EN and PWM control modes
Charging circuit	Wireless battery charging	For batteries with any number of cells up to 30 V, programmable charging current
Battery	Power supply	1000 mAh, 1-cell LiPo

Table B.6: Cost breakdown of the differential-drive design of the brushbot

Component	Quantity	Cost per unit	Details
Eccentric rotating mass vibration motors	2	$4.75	-
Various	-	$30.00 [a]	Printed circuit board, 3D printed chassis, battery, brushes, fasteners, wireless charging coils
	$39.50		

[a] Price significantly scales with quantity.

REFERENCES

[1] L. E. Parker, "Multiple mobile robot systems," in *Springer Handbook of Robotics*, Springer, 2008, pp. 921–941.

[2] E. Stump and N. Michael, "Multi-robot persistent surveillance planning as a vehicle routing problem," in *International Conference on Automation Science and Engineering*, IEEE, 2011, pp. 569–575.

[3] A. English, P. Ross, D. Ball, and P. Corke, "Vision based guidance for robot navigation in agriculture," in *International Conference on Robotics and Automation*, IEEE, 2014, pp. 1693–1698.

[4] M. Egerstedt, J. N. Pauli, G. Notomista, and S. Hutchinson, "Robot ecology: Constraint-based control design for long duration autonomy," *Annual Reviews in Control*, vol. 46, pp. 1–7, 2018.

[5] M. E. Csete and J. C. Doyle, "Reverse engineering of biological complexity," *Science*, vol. 295, no. 5560, pp. 1664–1669, 2002.

[6] R. E. Ricklefs, *The economy of nature*. Macmillan, 2008.

[7] G. Notomista and M. Egerstedt, "Constraint-driven coordinated control of multi-robot systems," in *American Control Conference*, IEEE, 2019, pp. 1990–1996.

[8] M. Santos, S. Mayya, G. Notomista, and M. Egerstedt, "Decentralized minimum-energy coverage control for time-varying density functions," in *International Symposium on Multi-Robot and Multi-Agent Systems*, IEEE, 2019, pp. 155–161.

[9] G. Notomista, S. Mayya, M. Selvaggio, M. Santos, and C. Secchi, "A set-theoretic approach to multi-task execution and prioritization," in *International Conference on Robotics and Automation*, 2020.

[10] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *European Control Conference*, Jun. 2019, pp. 2071–2076.

[11] Y. Emam, S. Mayya, G. Notomista, A. Bohannon, and M. Egerstedt, "Adaptive task allocation for heterogeneous multi-robot teams withevolving and unknown robot capabilities," in *International Conference on Robotics and Automation*, 2020.

[12] G. Notomista, S. Mayya, Y. Emam, C. Kroninger, A. Bohannon, S. Hutchinson, and M. Egerstedt, "A resilient and energy-aware task allocation framework for heterogeneous multi-robot systems," *Transactions on Robotics*, 2020, (under review).

[13] M. Ohnishi, G. Notomista, M. Sugiyama, and M. Egerstedt, "Constraint learning for control tasks with limited duration barrier functions," *Automatica*, 2020, (under review).

[14] G. Notomista, Y. Emam, and M. Egerstedt, "The SlothBot: A novel design for a wire-traversing robot," *Robotics and Automation Letters*, vol. 4, no. 2, pp. 1993–1998, 2019.

[15] G. Notomista, S. Mayya, A. Mazumdar, S. Hutchinson, and M. Egerstedt, "A study of a class of vibration-driven robots: Modeling, analysis, control and design of the brushbot," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2019, pp. 5101–5106.

[16] K. Toussaint, N. Pouliot, and S. Montambault, "Transmission line maintenance robots capable of crossing obstacles: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 26, no. 5, pp. 477–499, 2009.

[17] N. Pouliot and S. Montambault, "Geometric design of the linescout, a teleoperated robot for power line inspection and maintenance," in *International Conference on Robotics and Automation*, IEEE, 2008, pp. 3970–3977.

[18] J. Sawada, K. Kusumoto, Y. Maikawa, T. Munakata, and Y. Ishikawa, "A mobile robot for inspection of power transmission lines," *Transactions on Power Delivery*, vol. 6, no. 1, pp. 309–315, 1991.

[19] M. Nayyerloo, S Yeganehparast, A. Barati, and M Foumani, "Mechanical implementation and simulation of monolab, a mobile robot for inspection of power transmission lines," *International Journal of Advanced Robotic Systems*, vol. 4, no. 3, pp. 381–386, 2007.

[20] A. J. Phillips, J. M. Major, and G. R. Bartlett, *Line inspection robot and system*, US Patent 8,666,553, Mar. 2014.

[21] M. Rahimi, R. Pon, W. J. Kaiser, G. S. Sukhatme, D. Estrin, and M. Srivastava, "Adaptive sampling for environmental robotics," in *International Conference on Robotics and Automation*, IEEE, vol. 4, 2004, pp. 3537–3544.

[22] J. Billingsley, A. Visala, and M. Dunn, "Robotics in agriculture and forestry," in *Springer Handbook of Robotics*, Springer, 2008, pp. 1065–1077.

[23] F. Becker, S. Boerner, V. Lysenko, I. Zeidis, and K. Zimmermann, "On the mechanics of bristle-bots-modeling, simulation and experiments," in *International Symposium on Robotics*, VDE, 2014, pp. 1–6.

[24] G. Cicconofri and A. DeSimone, "Motility of a model bristle-bot: A theoretical analysis," *International Journal of Non-Linear Mechanics*, vol. 76, pp. 233–239, 2015.

[25] A. Ollero, S. Lacroix, L. Merino, J. Gancet, J. Wiklund, V. Remuß, I. V. Perez, L. G. Gutiérrez, D. X. Viegas, M. A. G. Benitez, *et al.*, "Multiple eyes in the skies: Architecture and perception issues in the comets unmanned air vehicles project," *Robotics & Automation Magazine*, vol. 12, no. 2, pp. 46–57, 2005.

[26] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni, "Multi-AUV control and adaptive sampling in Monterey bay," *Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 935–948, 2006.

[27] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang, "Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey bay," *Journal of Field Robotics*, vol. 27, no. 6, pp. 718–740, 2010.

[28] R. Graham and J. Cortés, "Adaptive information collection by robotic sensor networks for spatial estimation," *Transactions on Automatic Control*, vol. 57, no. 6, pp. 1404–1419, 2012.

[29] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and Autonomous Systems*, vol. 8, no. 1, pp. 47–63, 1991.

[30] R. O'Flaherty and M. Egerstedt, "Optimal exploration in unknown environments," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2015, pp. 5796–5801.

[31] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *International Conference on Robotics and Automation*, IEEE, vol. 1, 2000, pp. 476–481.

[32] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[33] S. G. Lee, Y. Diaz-Mercado, and M. Egerstedt, "Multirobot control using time-varying density functions," *Transactions on Robotics*, vol. 31, no. 2, pp. 489–493, 2015.

[34] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[35] M. Morris and S. Tosunoglu, "Survey of rechargeable batteries for robotic applications," in *Florida Conference on Recent Advances in Robotics*, 2012.

[36] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *International Conference on Robotics and Automation*, IEEE, 2015, pp. 1093–1099.

[37] B. Bethke, J. How, and J. Vian, "Multi-UAV persistent surveillance with communication constraints and health management," in *Guidance, Navigation, and Control Conference*, AIAA, 2009, p. 5654.

[38] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2011, pp. 3667–3672.

[39] N. Kamra and N. Ayanian, "A mixed integer programming model for timed deliveries in multirobot systems," in *International Conference on Automation Science and Engineering*, IEEE, 2015, pp. 612–617.

[40] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, 2015.

[41] L. Liu and N. Michael, "Energy-aware aerial vehicle deployment via bipartite graph matching," in *International Conference on Unmanned Aircraft Systems*, IEEE, 2014, pp. 189–194.

[42] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[43] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[44] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Conference on Decision and Control*, IEEE, 2014, pp. 6271–6278.

[45] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference*, Jun. 2019, pp. 3420–3431.

[46] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *Transactions on Automatic Control*, 2016.

[47] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference*, IEEE, 2016, pp. 322–328.

[48] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *American Control Conference*, IEEE, 2016, pp. 2252–2258.

[49] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *Transactions on Robotics*, vol. 35, no. 5, pp. 1186–1205, 2019.

[50] G. Notomista, X. Cai, J. Yamauchi, and M. Egerstedt, "Passivity-based decentralized control of multi-robot systems with delays using control barrier functions," in *International Symposium on Multi-Robot and Multi-Agent Systems*, IEEE, 2019, pp. 231–237.

[51] G. Notomista, M. Wang, M. Schwager, and M. Egerstedt, "Enhancing game-theoretic autonomous car racing using control barrier functions," in *International Conference on Robotics and Automation*, 2020.

[52] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *International Conference on Robotics and Automation*, IEEE, 2017, pp. 1699–1706.

[53] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *International Conference on Robotics and Automation*, IEEE, 2012, pp. 3293–3298.

[54] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead," *Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012.

[55] R. C. Arkin and G. Vachtsevanos, "Techniques for robot survivability," in *International Symposium on Robotics and Manufacturing*, 1990, pp. 383–388.

[56] S. Mishra, S. Rodriguez, M. Morales, and N. M. Amato, "Battery-constrained coverage," in *International Conference on Automation Science and Engineering*, IEEE, 2016, pp. 695–700.

[57] S. Martin and P. Corke, "Long-term exploration & tours for energy constrained robots with online proprioceptive traversability estimation," in *International Conference on Robotics and Automation*, IEEE, 2014, pp. 5778–5785.

[58] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[59] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[60] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[61] L. E. Parker, "Heterogeneous multi-robot cooperation," Massachusetts Institute of Technology, Cambridge, Artificial Intelligence Lab, Tech. Rep., 1994.

[62] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," in *International Conference on Robotics and Automation*, IEEE, 2005, pp. 1145–1150.

[63] F. Tang and L. E. Parker, "A complete methodology for generating multi-robot task solutions using ASyMTRe-D and market-based task allocation," in *International Conference on Robotics and Automation*, 2007, pp. 3351–3358.

[64] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Autonomous Robots*, vol. 44, no. 3, pp. 547–584, 2020.

[65] M. Irfan and A. Farooq, "Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities," in *International Conference on Intelligent Systems Engineering*, IEEE, 2016, pp. 210–215.

[66] T. W. Mather and M. A. Hsieh, "Macroscopic modeling of stochastic deployment policies with time delays for robot ensembles," *International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[67] S. Berman, Á. Halász, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[68] S. Mayya, S. Wilson, and M. Egerstedt, "Closed-loop task allocation in robot swarms using inter-robot encounters," *Swarm Intelligence*, vol. 13, no. 2, pp. 115–143, 2019.

[69] W. Abbas and M. Egerstedt, "Characterizing heterogeneity in cooperative networks from a resource distribution view-point," *Communications in Information and Systems*, vol. 14, 2014.

[70] T. Balch, "Hierarchic social entropy: An information theoretic measure of robot group diversity," *Autonomous robots*, vol. 8, no. 3, pp. 209–238, 2000.

[71] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[72] H. Ravichandar, K. Shaw, and S. Chernova, "STRATA: A unified framework for task assignments in large teams of heterogeneous agents," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 38, p. 38, 2020.

[73] L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa, "Distributed coordination in heterogeneous multi-robot systems," *Autonomous Robots*, vol. 15, no. 2, pp. 155–168, 2003.

[74] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[75] N. Palmieri, X.-S. Yang, F. De Rango, and A. F. Santamaria, "Self-adaptive decision-making mechanisms to balance the execution of multiple tasks for a multi-robots team," *Neurocomputing*, vol. 306, pp. 17–36, 2018.

[76] S Fatima and M. Wooldridge, "Adaptive task resources allocation in multi-agent systems," in *International Conference on Autonomous Agents*, ACM, 2001, pp. 537–544, ISBN: 158113326X.

[77] N. Iijima, A. Sugiyama, M. Hayano, and T. Sugawara, "Adaptive task allocation based on social utility and individual preference in distributed environments," *Procedia Computer Science*, vol. 112, pp. 91–98, 2017.

[78] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient flocking for mobile robot teams," *Robotics and Automation letters*, vol. 2, no. 2, pp. 1039–1046, 2017.

[79] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by reconfiguration: Exploiting heterogeneity in robot teams," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2019, pp. 6518–6525.

[80] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[81] J. Q. Shi, R. Murray-Smith, D. M. Titterington, and B. A. Pearlmutter, "Filtered gaussian processes for learning with large data-sets," in *Switching and Learning in Feedback Systems*, Springer, 2005, pp. 128–139.

[82] S. Das, S. Roy, and R. Sambasivan, "Fast gaussian process regression for big data," *Big Data Research*, vol. 14, pp. 12–26, 2018.

[83] A. G. Wilson, C. Dann, and H. Nickisch, "Thoughts on massively scalable gaussian processes," *arXiv preprint arXiv:1511.01870*, 2015.

[84] M. Tavassolipour, S. A. Motahari, and M. T. M. Shalmani, "Learning of gaussian processes in distributed and communication limited systems," *Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[85] D. Gu and H. Hu, "Spatial gaussian process regression with mobile sensor networks," *Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1279–1290, 2012.

[86] J. Chen, K. H. Low, Y. Yao, and P. Jaillet, "Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems," *Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 901–921, 2015.

[87] M. Nayyerloo, X. Chen, W. Wang, and J. G. Chase, "Cable-climbing robots for power transmission lines inspection," in *Mobile Robots*, X. Chen, Y. Chen, and J. Chase, Eds., Rijeka: IntechOpen, 2009, ch. 4.

[88] S.-i. Aoshima, T. Tsujimura, and T. Yabuta, "A wire mobile robot with multi-unit structure," in *International Workshop on Intelligent Robots and Systems*, IEEE/RSJ, 1989, pp. 414–421.

[89] N. Ranasinghe, J. Everist, and W.-M. Shen, "Modular robot climbers," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2007.

[90] K. H. Cho, Y. H. Jin, H. M. Kim, H. Moon, J. C. Koo, and H. R. Choi, "Caterpillar-based cable climbing robot for inspection of suspension bridge hanger rope," in *International Conference on Automation Science and Engineering*, IEEE, 2013, pp. 1059–1062.

[91] N. Morozovsky and T. Bewley, "SkySweeper: A low DOF, dynamic high wire robot," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2013, pp. 2339–2344.

[92] A. Pagnano, M. Höpf, and R. Teti, "A roadmap for automated power line inspection. maintenance and repair," *Procedia Cirp*, vol. 12, pp. 234–239, 2013.

[93] P. Debenest, M. Guarnieri, K. Takita, E. F. Fukushima, S. Hirose, K. Tamura, A. Kimura, H. Kubokawa, N. Iwama, and F. Shiga, "Expliner - Robot for inspection of transmission lines," in *International Conference on Robotics and Automation*, May 2008, pp. 3978–3984.

[94] X.-D. Yu, L.-Y. Duan, and Q. Tian, "Highway traffic information extraction from Skycam MPEG video," in *International Conference on Intelligent Transportation Systems*, IEEE, 2002, pp. 37–42.

[95] R. R. Thompson and M. S. Blackstone, *Three-dimensional moving camera assembly with an informational cover housing*, US Patent 6,873,355, Mar. 2005.

[96] B. L. Jordan, M. A. Batalin, and W. J. Kaiser, "NIMS RD: A rapidly deployable cable based robot," in *International Conference on Robotics and Automation*, IEEE, 2007, pp. 144–150.

[97] M. F. Campos, G. A. Pereira, S. R. Vale, A. Q. Bracarense, G. A. Pinheiro, and M. P. Oliveira, "A robot for installation and removal of aircraft warning spheres on aerial power transmission lines," *Transactions on Power Delivery*, vol. 18, no. 4, pp. 1581–1582, 2003.

[98] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[99] S. Carpin, "Distributed coverage while not being covered," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2012, pp. 842–848.

[100] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *International Conference on Robotics and Automation*, IEEE, 2010, pp. 4982–4989.

[101] J.-M. Breguet and R. Clavel, "Stick and slip actuators: Design, control, performances and applications," in *International Symposium on Micromechatronics and Human Science*, IEEE, 1998, pp. 89–95.

[102] P. Vartholomeos and E. Papadopoulos, "Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators," in *International Conference on Robotics and Automation*, IEEE, 2006, pp. 649–654.

[103] L Giomi, N Hawley-Weld, and L Mahadevan, "Swarming, swirling and stasis in sequestered bristle-bots," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 469, no. 2151, p. 20 120 637, 2013.

[104] J. Klingner, A. Kanakia, N. Farrow, D. Reishus, and N. Correll, "A stick-slip omnidirectional powertrain for low-cost swarm robotics: Mechanism, calibration, and control," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2014, pp. 846–851.

[105] S. Mayya, G. Notomista, D. Shell, S. Hutchinson, and M. Egerstedt, "Non-uniform robot densities in vibration driven swarms using phase separation theory," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2019, pp. 4106–4112.

[106] G. Notomista, S. F. Ruf, and M. Egerstedt, "Persistification of robotic tasks using control barrier functions," *Robotics and Automation Letters*, vol. 3, no. 2, pp. 758–763, 2018.

[107] S. Piller, M. Perrin, and A. Jossen, "Methods for state-of-charge determination and their applications," *Journal of Power Sources*, vol. 96, no. 1, pp. 113–120, 2001.

[108] U. Krewer, F. Röder, E. Harinath, R. D. Braatz, B. Bedürftig, and R. Findeisen, "Dynamic models of Li-Ion batteries for diagnosis and operation: A review and perspective," *Journal of The Electrochemical Society*, vol. 165, no. 16, A3656, 2018.

[109] W.-Y. Chang, "The state of charge estimating methods for battery: A review," *International Scholarly Research Notices*, vol. 2013, 2013.

[110] M. Murnane and A. Ghazel, "A closer look at state of charge (SOC) and state of health (SOH) estimation techniques for batteries," *Analog Devices*, vol. 2, pp. 426–436, 2017.

[111] L. Zhao, M. Lin, and Y. Chen, "Least-squares based coulomb counting method and its application for state-of-charge (SOC) estimation in electric vehicles," *International Journal of Energy Research*, vol. 40, no. 10, pp. 1389–1399, 2016.

[112] H. A. Toliyat and G. B. Kliman, *Handbook of electric motors*. CRC press, 2018, vol. 120.

[113] L. Gao, S. Liu, and R. A. Dougal, "Dynamic lithium-ion battery model for system simulation," *Transactions on Components and Packaging Technologies*, vol. 25, no. 3, pp. 495–505, 2002.

[114] C. Daniel and J. O. Besenhard, *Handbook of battery materials*. John Wiley & Sons, 2012.

[115] L. W. Tu, "Bump functions and partitions of unity," *An Introduction to Manifolds*, pp. 127–134, 2008.

[116] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[117] Y. Huang, S. Z. Yong, and Y. Chen, "Guaranteed vehicle safety control using control-dependent barrier functions," in *American Control Conference*, IEEE, 2019, pp. 983–988.

[118] A. D. Ames, G. Notomista, Y. Wardi, and M. Egerstedt, "Integral control barrier functions for dynamically defined control laws," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 887–892, 2020.

[119] Y. Wardi, C. Seatzu, J. Cortes, M. Egerstedt, S. Shivam, and I. Buckley, "Tracking control by the newton-raphson method with output prediction and controller speedup," *arXiv preprint arXiv:1910.00693*, 2019.

[120] J. Garche and A. Jossen, "Battery management systems (BMS) for increasing battery life time," in *Telecommunications Energy Special Conference*, IEEE, 2000, pp. 81–88.

[121] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[122] A. D. Ames and M. Powell, "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs," in *Control of Cyber-Physical Systems*, Springer, 2013, pp. 219–240.

[123] Y. Girdhar and G. Dudek, "Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring," *Autonomous Robots*, vol. 40, no. 7, pp. 1267–1278, 2016.

[124] J. G. Bellingham and K. Rajan, "Robotics in remote and hostile environments," *Science*, vol. 318, no. 5853, pp. 1098–1102, 2007.

[125] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[126] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *American Control Conference*, IEEE, 2013, pp. 4196–4201.

[127] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[128] A. Okabe and A. Suzuki, "Locational optimization problems solved through Voronoi diagrams," *European Journal of Operational Research*, vol. 98, no. 3, pp. 445–456, 1997.

[129] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[130] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The Robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[131] S. P. Bhat and D. S. Bernstein, "Finite-time stability of continuous autonomous systems," *SIAM Journal on Control and Optimization*, vol. 38, no. 3, pp. 751–766, 2000.

[132] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization - I: Algorithm," *Transactions on Automatic Control*, vol. 58, no. 9, pp. 2162–2175, 2013.

[133] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[134] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[135] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH computer graphics*, ACM, vol. 21, 1987, pp. 25–34.

[136] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.

[137] S. Lloyd, "Least squares quantization in pcm," *Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[138] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential ϵ-tracking and ϵ-stabilization of first-order nonholonomic SE(2) vehicles," in *American Control Conference*, IEEE, vol. 6, 2002, pp. 4690–4695.

[139] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[140] G. Notomista. (2018). Notomista, Egerstedt - Constraint Driven Coordinated Control of Multi Robot Systems, Youtube.

[141] B. P. Gerkey and M. J. Matarić, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *International Conference on Robotics and Automation*, IEEE, vol. 3, 2003, pp. 3862–3868.

[142] G. Notomista and M. Egerstedt, "Persistification of robotic tasks," *Transactions on Control Systems Technology*, 2020.

[143] J. Bridle and A. van Rensburg, "Discovering the limits of ecological resilience," *Science*, vol. 367, no. 6478, pp. 626–627, 2020.

[144] P. S. Gonçalves, P. D. Torres, C. O. Alves, 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*, 2009.

[145] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, Cambridge University Press, vol. 51, 1955, pp. 406–413.

[146] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2018, pp. 3723–3729.

[147] H. K. Khalil, "Nonlinear systems, 3rd," *New Jewsey, Prentice Hall*, vol. 9, 2002.

[148] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. Siam, 1994, vol. 15.

[149] V. Yakubovich, "S-procedure in nonlinear control theory," *Vestnick, Leningrad University. Mathematics*, vol. 4, pp. 73–93, 1997.

[150] J. Lee and S. Leyffer, *Mixed integer nonlinear programming*. Springer Science & Business Media, 2011, vol. 154.

[151] F. Granot and J. Skorin-Kapov, "Some proximity and sensitivity results in quadratic integer programming," *Mathematical Programming*, vol. 47, no. 1-3, pp. 259–268, 1990.

[152] A. V. Fiacco and Y. Ishizuka, "Sensitivity and stability analysis for nonlinear programming," *Annals of Operations Research*, vol. 27, no. 1, pp. 215–235, 1990.

[153] M. Grant, S. Boyd, and Y. Ye, *CVX: Matlab software for disciplined convex programming*, 2009.

[154] G. Optimization, *Gurobi optimizer reference manual*, 2015.

[155] G. Notomista and M. Egerstedt, "Communication constrained distributed spatial field estimation using mobile sensor networks," in *IFAC World Congress*, 2020.

[156] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[157] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[158] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press, 2006, vol. 1.

[159] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams, "Approximation methods for gaussian process regression," *Large-scale kernel machines*, pp. 203–224, 2007.

[160] Z. Wu, "Compactly supported positive definite radial functions," *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 283–292, 1995.

[161] R. Furrer, M. G. Genton, and D. Nychka, "Covariance tapering for interpolation of large spatial datasets," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 502–523, 2006.

[162] T. Gneiting, "Compactly supported correlation functions," *Journal of Multivariate Analysis*, vol. 83, no. 2, pp. 493–508, 2002.

[163] A. A. Jamshidi and M. J. Kirby, "Examples of compactly supported functions for radial basis approximations," in *International Conference on Machine Learning Models Technologies and Applications*, 2006, pp. 155–160.

[164] B. Hamers, J. A. Suykens, and B. De Moor, "Compactly supported RBF kernels for sparsifying the gram matrix in LS-SVM regression models," in *International Conference on Artificial Neural Networks*, Springer, 2002, pp. 720–726.

[165] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 67–93, 2001.

[166] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 299–312, 2001.

[167] R. Ababou, A. C. Bagtzoglou, and E. F. Wood, "On the condition number of covariance matrices in kriging, estimation, and simulation of random fields," *Mathematical Geology*, vol. 26, no. 1, pp. 99–133, 1994.

[168] C. Heil, *A basis theory primer*. Birkhäuser, 2010.

[169] B. Gärtner and S. Schönherr, "Smallest enclosing ellipses: Fast and exact," 1997.

[170] E. D. Nering and A. W. Tucker, *Linear Programs & Related Problems: A Volume in the Computer Science and Scientific Computing Series*. Elsevier, 1992.

[171] G. Notomista and M. Egerstedt, "Coverage control for wire-traversing robots," in *International Conference on Robotics and Automation*, IEEE, 2018, pp. 1–6.

[172] G. Notomista, M. Santos, S. Hutchinson, and M. Egerstedt, "Sensor coverage control using robots constrained to a curve," in *International Conference on Robotics and Automation*, IEEE, 2019, pp. 3252–3258.

[173] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.

[174] R. C. Arkin, *Behavior-based robotics*. MIT Press, 1998.

[175] M. Egerstedt, "Behavior based robotics using hybrid automata," in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2000, pp. 103–116.

[176] W. Rudin, *Real and complex analysis*. Tata McGraw-Hill Education, 1987.

[177] B. Riemann, "Grundlagen für eine allgemeine theorie der functionen einer veränderlichen complexen grösse," PhD thesis, EA Huth, 1851.

[178] T. A. Driscoll and L. N. Trefethen, *Schwarz-Christoffel mapping*. Cambridge University Press, 2002, vol. 8.

[179] J. Kieffer, "Uniqueness of locally optimal quantizer for log-concave density and convex error weighting function," *Transactions on Information Theory*, vol. 29, no. 1, pp. 42–47, 1983.

[180] H. Mine and M. Fukushima, "A minimization method for the sum of a convex function and a continuously differentiable function," *Journal of Optimization Theory and Applications*, vol. 33, no. 1, pp. 9–23, 1981.

[181] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[182] A. Ramezani, S.-J. Chung, and S. Hutchinson, "A biomimetic robotic platform to study flight specializations of bats," *Science Robotics*, vol. 2, no. 3, Art–No, 2017.

[183] J. Nakanishi, T. Fukuda, and D. E. Koditschek, "A brachiating robot controller," *Transactions on Robotics and Automation*, vol. 16, no. 2, pp. 109–123, 2000.

[184] T. Landgraf, "Robobee: A biomimetic honeybee robot for the analysis of the dance communication system," PhD thesis, 2013.

[185] J.-P. Afman, M. Mote, and E. Feron, "Motion rectification for an homeostasis-enabling wheel," *arXiv preprint arXiv:1705.04399*, 2017.

[186] D. E. Koditschek and M. Buehler, "Analysis of a simplified hopping robot," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 587–605, 1991.

[187] M. Azad and R. Featherstone, "Balancing control algorithm for a 3D under-actuated robot," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2014, pp. 3233–3238.

[188] A. A. Transeth, K. Y. Pettersen, and P. Liljebäck, "A survey on snake robot modeling and locomotion," *Robotica*, vol. 27, no. 7, pp. 999–1015, 2009.

[189] A. Mohammadi and M. W. Spong, "Path following control of swimming magnetic helical microrobots subject to step-out frequencies," in *Conference on Control Technology and Applications*, IEEE, 2018, pp. 60–66.

[190] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.

[191] Y.-T. Kim and D.-E. Kim, "Novel propelling mechanisms based on frictional interaction for endoscope robot," *Tribology Transactions*, vol. 53, no. 2, pp. 203–211, 2010.

[192] V. Radhakrishnan, "Locomotion: Dealing with friction," *Proceedings of the National Academy of Sciences*, vol. 95, no. 10, pp. 5448–5455, 1998.

[193] S. Timoshenko, *History of strength of materials: with a brief account of the history of theory of elasticity and theory of structures*. Courier Corporation, 1983.

[194] D. Stewart, "A platform with six degrees of freedom," *Proceedings of the institution of mechanical engineers*, vol. 180, no. 1, pp. 371–386, 1965.

[195] D. Kim, Z. Hao, J. Ueda, and A. Ansari, "A 5 mg micro-bristle-bot fabricated by two-photon lithography," *Journal of Micromechanics and Microengineering*, vol. 29, no. 10, p. 105 006, 2019.

[196] C. M. Kellett, "A compendium of comparison function results," *Mathematics of Control, Signals, and Systems*, vol. 26, no. 3, pp. 339–374, 2014.

[197] Y. Lin, E. D. Sontag, and Y. Wang, "A smooth converse lyapunov theorem for robust stability," *SIAM Journal on Control and Optimization*, vol. 34, no. 1, pp. 124–160, 1996.

VITA

Gennaro Notomista received a B.S. in Mechanical Engineering from the Università degli Studi di Napoli "Federico II" in 2012, a M.Eng. in Automotive Engineering from the Technische Hochschule Ingolstadt in 2015, a M.S. in Mechanical Engineering from Università degli Studi di Napoli "Federico II" in 2016, and a M.S. in Mathematics from the Georgia Institute of Technology in 2019. He has been awarded with a Fulbright Scholarship in 2016. He is the recipient of the Alumni Small Grant 2020 sponsored by the Embassy of the United States of America to Italy.