Distributed Control and Organization of Communicating Mobile Robots: Design, Simulation, and Experimentation

Von der Fakultät Konstruktions-, Produktions- und
Fahrzeugtechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von
Henrik Ebel
aus Kempten (Allgäu)

Hauptberichter:	Prof. Dr.-Ing. Prof. E.h. Peter Eberhard
Mitberichter:	Prof. Dr.-Ing. Qirong Tang Prof. Dr. sc. Sebastian Trimpe

Tag der mündlichen Prüfung: 14. Juli 2021

Institut für Technische und Numerische Mechanik
der Universität Stuttgart

2021

Preface

This dissertation was written over the course of my work as a member of the research staff at the Institute of Engineering and Computational Mechanics (ITM) at the University of Stuttgart. Indeed, anywhere else, this dissertation could not have become what it is. Omnipresent collegiality, helpfulness, the always-open office doors, and signature coffee breaks have made my time at the institute so worth living and enriching and have formed the foundation for a marvelous working atmosphere. Since the institute's characteristic atmosphere seems to have persisted throughout many generations of doctoral researchers, it is also a testament to the qualities of the institute's leadership, which directs and cultivates the institute, its atmosphere, and its staff members to make it such a unique place to be for a doctoral researcher. Of course, this atmosphere builds upon the characters of all the wonderful colleagues and friends that have worked at the ITM during the years I have been there so far.

I would like to express my deepest gratitude to Prof. Peter Eberhard for believing in my abilities early on, already giving me great advice when I was an undergraduate and graduate student, providing me with the opportunity to work at the ITM, and guiding me proficiently through my dissertation work and scientific life. I am very grateful for the freedom I was granted in research and for all the advice I got whenever needed. It fills me with joy that I am given the opportunity to continue my work at the ITM. Moreover, without the help and advice of Dr.-Ing. Pascal Ziegler, Peter Schöler, and my colleague Wei Luo, the creation of all the hardware for the experimental research involved in the dissertation would have been nigh impossible. Furthermore, I thank Prof. Jörg Fehr for being a great motivator and pointing me in a lot of right directions. I thank Prof. Michael Hanss for all the discussions about the peculiarities of many languages and for pulling the strings behind quite some of the institute's social activities.

With Prof. Qirong Tang and Prof. Sebastian Trimpe, I am happy to have had such interested and professional co-examiners and I thank them for all the effort and scrutiny invested. In addition, I extend my gratitude to Prof. Qirong Tang for having been my host during a two-month research visit to his laboratory in Shanghai in 2018. Many of the qualities of this dissertation are rooted in insights from my time in Shanghai. Moreover, I would like to thank Prof. Aki Mikkola and Dr. Marko Matikainen for being great supervisors and hosts during a four-month research visit to Lappeenranta University of Technology as a master's student in the winter of 2015/16. Not only will my stay in Finland be a lifelong-memorable experience due to the kindness and openness of the research staff there but it also demonstrated to me how much joy scientific work can bring and, thus, it helped to set me onto the path leading to this thesis and beyond.

I thank my colleagues and friends at the ITM for being awesome advice-givers and great company. Furthermore, I had the pleasure to supervise well over 20 students and countless student assistants, who helped tremendously in teaching and research, and I would like to thank each and every one of them for all the insights that we gained together. Special thanks go to my friends from the Simulation Technology study program, who already made my undergraduate and graduate studies so much more enjoyable and continue to be great company. It was almost ten years ago that we ventured together into a journey that, ultimately, lead most of us into the world of scientific research. And, as often in life, some of the strongest ties are forged on common journeys. Last but not least, I would like to express my deepest gratitude to my parents and brothers for their unconditional support throughout my studies and, simply, for being great parents and siblings throughout all my life.

Stuttgart, July 2021 *Henrik Ebel*

Contents

Zusammenfassung

Die kommunikationsbasierte Kooperation mehrerer robotischer Systeme hat das Potenzial, den Horizont dessen, was mit robotischer Automation erreichbar ist, nachhaltig zu erweitern. Daher rücken dynamisch anpassbare robotische Netzwerke zunehmend in den Fokus der Forschung, die in den letzten Jahrzehnten zuvorderst die Verbesserung einzelner Roboter vorangetrieben hat. Hierbei verbindet sich mit der in diesem Sinne verteilten Robotik die Hoffnung, die erreichbare Flexibilität, Robustheit und Leistungsfähigkeit gegenüber zentralisierten Ansätzen entscheidend positiv zu beeinflussen. Trotz dieser charakteristischen, erwarteten Vorteile sind robotische Netzwerke jedoch noch vorwiegend Gegenstand von Forschung und Forschungsvisionen und noch nicht in die Breite der Anwendung gelangt. Dies mag auch damit im Zusammenhang stehen, dass die erwarteten Vorteile zum Teil durch eine gesteigerte Systemkomplexität aufgewogen werden, schließlich ist die Entwicklung eines zuverlässigen und dynamisch anpassbaren verteilten Systems durchaus herausfordernd. Dabei ist bereits die herkömmliche Robotik durch ihren interdisziplinären Charakter ein anspruchsvolles Betätigungsfeld mit mannigfaltigen Problemstellungen. Infolgedessen ist die verteilte Robotik in einem Stadium, in dem sich Fortschritte durch zielgerichtete Forschung anhand klar definierter Modellprobleme erzielen lassen. Aus diesem Grund stellt sich diese Dissertation den Herausforderungen der verteilten Robotik anhand einer kooperativen Transportaufgabe. Dabei sollen omnidirektionale mobile Roboter polygonale Objekte komplett selbstständig und nur durch Schubkräfte transportieren. Diese Aufgabe ist hervorragend als Modellproblem geeignet, da sie alle für das Forschungsfeld charakteristischen Herausforderungen mit sich bringt und gleichzeitig durch ihren anschaulichen Charakter eine intuitive Beurteilung der Funktionstüchtigkeit ersonnener Regel- und Organisationsverfahren ermöglicht. Die Dissertationsschrift behandelt alle Aspekte der Aufgabe auf umfassende Art und Weise. So werden neben den Organisations- und Regelverfahren auch die zugrundeliegende verteilte Programmarchitektur sowie sogar die eingesetzten, extra für die Forschung in der verteilten Robotik entwickelten mobilen Roboter entworfen und untersucht. Die im Rahmen von Simulationen und Experimenten erzielten Ergebnisse zeugen hierbei von einer beinahe beispiellosen Einsatzflexibilität des ersonnenen Transportansatzes. Insbesondere werden alle maßgeblichen, versprochenen Vorteile der verteilten Robotik praktisch umgesetzt, wobei der dynamischen Anpassung des robotischen Netzwerks eine besondere Bedeutung zukommt. Die vielseitige Anwendbarkeit des vorgestellten Ansatzes geht hierbei vor allem auf den Einsatz optimierungsbasierter Ansätze für die wichtigsten Unterprobleme zurück. Die Transportaufgabe wird aufgeteilt in eine Formationsregelungsaufgabe sowie eine Organisationsaufgabe. Letztere besteht darin, für den Objekttransport geeignete Formationen zu bestimmen. Dies ermöglicht die Nutzung verteilter modellprädiktiver Regelung für die Formationsregelung und die Lösung der Organisationsaufgabe mithilfe von verteilter Optimierung. Darüber hinaus werden in der Dissertation auch die selbstständige Aushandlung einer Aufgabenverteilung

zwischen den Robotern sowie die lokale und globale Navigation behandelt. Die erzielten Erkenntnisse erstrecken sich über die unmittelbaren Erfordernisse des Modellproblems hinaus. Ein Teilaspekt, der sich auch in anderen Forschungsvorhaben in der verteilten Robotik als nützlich erweisen könnte, ist die entwickelte verteilte Programmarchitektur, die insbesondere den Übergang von Simulationen zu Experimenten stark vereinfacht. Ebenso sind die entwickelten mobilen Roboter auch für andere Aufgaben und die Behandlung anderer Forschungsfragen geeignet. Überdies ist die Formationsregelung von allgemeinerer Nützlichkeit. Daher wird der vorgeschlagene modellprädiktive Ansatz gesondert behandelt und analysiert. Hierbei bestätigt der Ansatz seine aufgrund theoretischer Überlegungen erwarteten Vorteile auch in Experimenten und im Vergleich zu einem gebräuchlicheren Ansatz, obwohl letzterer so modifiziert wurde, dass auch dieser Beschränkungen der Stellgröße berücksichtigen kann. Nicht zuletzt ist eine vorgeschlagene, verteilte Fassung eines Partikelschwarmoptimierers auf Basis der erweiterten Lagrange-Funktion potenziell weit über die Robotik hinaus von Nutzen. In der Transportaufgabe leistet der Algorithmus gute Dienste bei der Bestimmung von zum Transport geeigneten Formationen.

Abstract

Leveraging the communication-based cooperation of multiple robotic systems has the potential to significantly further the state of the art of what is achievable with robotic automation. Therefore, beyond solely improving the capabilities of individual robotic agents, reconfigurable robotic networks have come to the attention of research and industry. However, despite the potential to increase flexibility, robustness, and performance, robotic networks are not yet in widespread application, with many research challenges remaining. After all, developing a reliable, reconfigurable distributed system is very difficult, adding to the manifold, interdisciplinary challenges posed by robotics in general. Hence, to better understand and subsequently overcome these challenges, distributed robotics is still in a state where it can benefit significantly from research that tackles well-defined benchmark problems. Consequently, this thesis faces the challenges of distributed robotics at the example of a cooperative transportation task. In the task, omnidirectional mobile robots cooperate to maneuver polygonal objects purely by pushing forces and in a completely self-reliant manner. The task is found to be a formidable benchmark problem since it raises all major challenges of the field while still being easily graspable, intuitively making evident the qualities of the control and organization schemes employed. The thesis discusses all aspects of the task in an encompassing manner, not only including the design of the employed control and organization methods, but also the software architecture and even the custom robotic hardware employed. Results from simulations and real-world hardware experiments show that the proposed scheme is of unprecedented versatility, putting into practice all major promises of distributed robotics, including plug-and-play control for online reconfigurations of the robotic network. This is achieved by relying, at heart, on optimization-based schemes. The task is decomposed into a formation control task and the organizational task of inferring formations useful for manipulation, allowing the usage of distributed model predictive control for formation control and of distributed optimization for organization. Further challenges dealt with include self-reliant task allocation as well as local and global navigation. However, the contributions of the thesis extend beyond the immediate needs of the benchmark problem. A component that may prove helpful in other research endeavors in the field includes the devised distributed software architecture, which greatly facilitates the transition from simulations to experiments. Similarly, the custom mobile robot and different proposed setups of the formation controller are also suited to other tasks and projects. Due to formation control's universal appeal, the proposed approach based on distributed predictive control is analyzed separately from the transportation task. In experiments, the predictive control-based approach confirms its theory-rooted advantages in comparison to a more traditional approach, despite the latter being modified to also respect input constraints. Finally, a proposed distributed version of an augmented Lagrangian particle swarm optimization algorithm, which is used to devise formations in the thesis, may even prove useful far beyond robotics.

Chapter 1

Introduction

Robotic automation has been a major contributing factor to the labor productivity growth across various industries in certain industrialized countries, even seemingly wielding an influence on economy-wide productivity measures [GraetzMichaels18, KromannEtAl19]. While initial development was largely focused on robotics in factories and similar controlled industrial environments, robots have now started to pervade into other areas, including agriculture, transportation, medicine, health care, and households [Hägele16, HazarikaDixit18]. Nowadays, in some areas, the social acceptance of working together with robots or even of substituting human by robotic work may be as much of a challenge as the engineering problems involved [SavelaTurjaOksanen18]. However, the mere fact that such societal questions are raised is a testament to the vast technological and methodological progress that has made the widespread automation of work tasks through robotic systems not only conceivable but a reality. A driver of progress has been the improvement of individual robotic agents' capabilities, with improvements in the fields of control engineering, sensing and vision, software and information technology, the semiconductor industry, and many more having worked in tandem to realize what has previously been science fiction [HazarikaDixit18].

Apart from improving the individual robot, another fruitful area for efficiency gains may be the cooperation of multiple robots to automate work tasks. A robotic workforce may be used more efficiently if larger tasks could be handled by a larger, cooperating group of robots, whereas the large group could potentially split up to handle multiple smaller tasks simultaneously in smaller groups. Similarly, robustness may be increased since a group of robots could reorganize and continue work if one of the robots breaks down due to a technical failure. Furthermore, strength can lie in numbers, i.e., many robots may be able to achieve tasks inconceivable for an individual robot operating on its own. Nowadays, especially communication-based cooperation is believed to be a promising area to seek robotic productivity gains [GrauEtAl17, WenHeZhu18]. In another take on the area, difficulties raised by communication have motivated re-

search on robotic automation approaches relying on communication-free cooperation, see, e.g., [YamadaSaito01, ChenEtAl15, DehghaniMenhaj16]. Whereas such communication-free schemes may always have merit in hostile environments or disaster scenarios, continued progress in communication technology has lowered the burden of communication. Therefore, the expected advent of low-latency, high-bandwidth, and highly reliable wireless data transfer [SachsEtAl18] is projected to have a significant and lasting impact on the field of robotics [HolfeldEtAl16], potentially reshaping it fundamentally [KehoeEtAl15]. However, this is still very much an emerging field, with a reconfigurable, fault-tolerant, distributed robotic network solving practical, productive tasks being more of a vision than a widespread reality. It may be conjectured that this is due to the broad and interdisciplinary array of challenges and difficulties involved. Not only is it the case that new challenges are introduced, such as communication and a distributed software architecture bringing up issues related to concurrency, but also that certain challenges from robotics, such as control, need to be rethought for the distributed, communication-based setting. Thus, mostly separate from applied robotics research, control theory has strived to develop a theoretical foundation to support reliable, distributed control in robotics and beyond, see, e.g., the overviews given by [BulloCortésMartínez09, MesbahiEgerstedt10, BaiArcakWen11, ChristofidesEtAl13, NegenbornMaestre14]. Still, being focused on theoretical foundations, the newly devised methods are usually not immediately evaluated by means of actual, realistic realizations that explicitly include wireless communication and a distributed software architecture. This may be because overcoming the technical and engineering challenges involved is problematic enough to constitute and motivate its very own research fields.

On the one hand, this includes numerical methods and frameworks to formulate and solve distributed optimization problems as they appear in optimization-based control [BurkVölzGraichen21, FarinaEtAl20]. Progress in this area can free people working in pure control theory, striving to try their methods in realistic examples, and more application-oriented robotics scientists from self-developing distributed numerical algorithms and communication from the ground up. Since major contributions in this area are very recent, it seems that time is ripe to transfer more of the theoretical progress made to problems in robotics.

On the other hand, robotics research has long since tried its hand at solving practical tasks and benchmark problems in cooperative distributed robotics, long before the more recent progress in formal distributed control theory. Tasks in mobile robotics are widely studied in particular, including more abstract tasks such as motion coordination and formation control as well as more concrete ones such as multi-robot transportation, manipulation, cooperative target search, and even robot soccer [AraiPagelloParker02]. In any case, the research field is still in a state where meaningful benchmark problems, which incorporate the key challenges of distributed robotics, are instrumental in developing and evaluating new strategies for automated robotic behavior. In that regard, it is worth remembering

that the defining purpose of a robot is to automate work, which is also reflected in the very etymology of the word itself. And, in the classical, probably most common definition, robotic work refers to mechanical work, e.g., performed by robots manipulating physical objects [HazarikaDixit18]. This implies that robotics, in its classical sense, is inseparable from mechanics. Consequently, it may be argued that a well-designed benchmark scenario should also be appealing from a mechanical perspective. This thesis distills this line of thinking into the goal to develop an encompassing, communication-based control and organization scheme at the example of and tested by means of a mechanically meaningful benchmark problem while relying, at heart, on methods rooted in control theory. In particular, it is the goal to live up to the promises of distributed robotics by letting different numbers of robots self-reliantly adapt to different scenarios, allowing robots to leave and join the robotic network in the midst of the task solution. With regard to highly time-critical decisions, the robots shall cooperate as equals, without there being necessary an external, centralized decision-making instance. To accomplish this, all challenges involved in the field shall be dealt with in an encompassing manner, as made necessary by the benchmark problem's requirements. This includes the robotic hardware employed, the distributed software architecture including communication, as well as simulation, and, of course, the control and organization methods governing the behavior of the robots. Furthermore, it is the goal to design the distributed software architecture in a very modular manner, with well-defined interfaces between all sub-methods employed. The resulting control and organization scheme shall be scenario-agnostic, in the sense that it can accommodate self-reliantly to different scenarios without having to succumb to human-lead parameter tuning. In particular, this thesis studies how the optimization-based formulation and solution of the major involved sub-problems can help to achieve the desired versatility of the scheme.

However, all this raises the question of which task actually composes a worthwhile benchmark problem. It should be a mechanical task that is still intuitively clear to make apparent the potential virtues of distributed robotics without already the description of the benchmark problem absorbing an unnecessary amount of attention on the way toward the truly encompassing treatment intended. At the same time, the problem should be easily accessible to the employment of different numbers of robots to develop and test the realization of distributed robotics' promises with regard to flexibility. In the same vein, it should be possible to intuitively vary the problem to test the robotic network's ability to accommodate to different scenarios. When chosen this way, the problem automatically raises all the aforementioned major challenges of the research field. As a reaction to these requirements, this thesis opts for a transportation task in which omnidirectional mobile robots cooperatively transport an object. The robots are not rigidly attached to the transported object in any way and, therefore, can only push but not pull the object. The task is advantageous in the sense that it is a truly cooperative task since an accurate manipulation usually not only benefits from but actually requires the simultaneous,

coordinated usage of multiple mobile robots. Furthermore, the task can easily be varied by transporting objects of different shapes along a variety of paths. Also, it is intuitively clear that robots can easily join and leave the transportation process. Figure 1.1 gives a glimpse of typical transportation scenarios to be considered. The figure already gives an outlook to some of the results obtained with what is proposed in the thesis, with the ultimate goal of the subsequent chapters being to explain how these results come to be.

It should be noted that it is not only the author of this thesis who finds cooperative transportation to be an adequate benchmark problem for distributed robotics. Indeed, in the form of different variations, the task has been widely studied throughout the previous decades, both because of being able to intuitively show the virtues of distributed robotics and because it may have practical applications in logistics [AraiPagelloParker02]. In that regard, approaches appearing in literature can be categorized into prehensile and non-prehensile transportation strategies. In prehensile approaches, the robots grasp the object to transport it. Examples for approaches of this kind can, e.g., be found in [GroßMondadaDorigo06, MiyataEtAl02, WangSchwager16, FarivarnejadWilsonBerman16]. Therefore, the scheme proposed in this thesis belongs to the non-prehensile category, with there being a unilateral contact between each robot and the object. Other non-prehensile schemes are, e.g., proposed in [YamadaSaito01, ChenEtAl15, NeumannKitts16, DaiEtAl16]. Although such discussions are relegated toward the end of the thesis, it may already be noted that this thesis' scheme is quite unprecedented in its versatility when compared to preexisting approaches in the literature. Most schemes merely cater to a very limited set of transportation scenarios. Beyond the references provided so far, an introductory, separate literature overview is deliberately avoided. Instead, relevant literature is cited throughout the thesis whenever appropriate.

To arrive at its goals, the thesis is structured as follows. Chapters 2 and 3 introduce the fundamentals necessary for the remaining chapters. Therefore, Chapter 2 deals with aspects from modeling as necessary for the design of the transportation scheme and for simulation. On the one hand, it is discussed how mechanical systems can be modeled as multibody systems. On the other hand, it is discussed how information relevant to robotic decision-making can be modeled. In that regard, introduced concepts that will prove useful in different applications throughout the thesis are graphs and convex polytopes. Furthermore, it is briefly discussed how these two can be employed for path planning and for building a mathematical model of the workspace the robots operate in. Subsequently, Chapter 3 introduces the optimization and control prerequisites necessary for the cooperative transportation scheme. Hence, building upon some introductory remarks on optimization, in the form of model predictive control, a very popular optimization-based control scheme is introduced. Afterward, different theory-driven approaches that allow extending the application of model predictive control to the distributed domain are discussed. The section is concluded with an introduction to a more conventional,

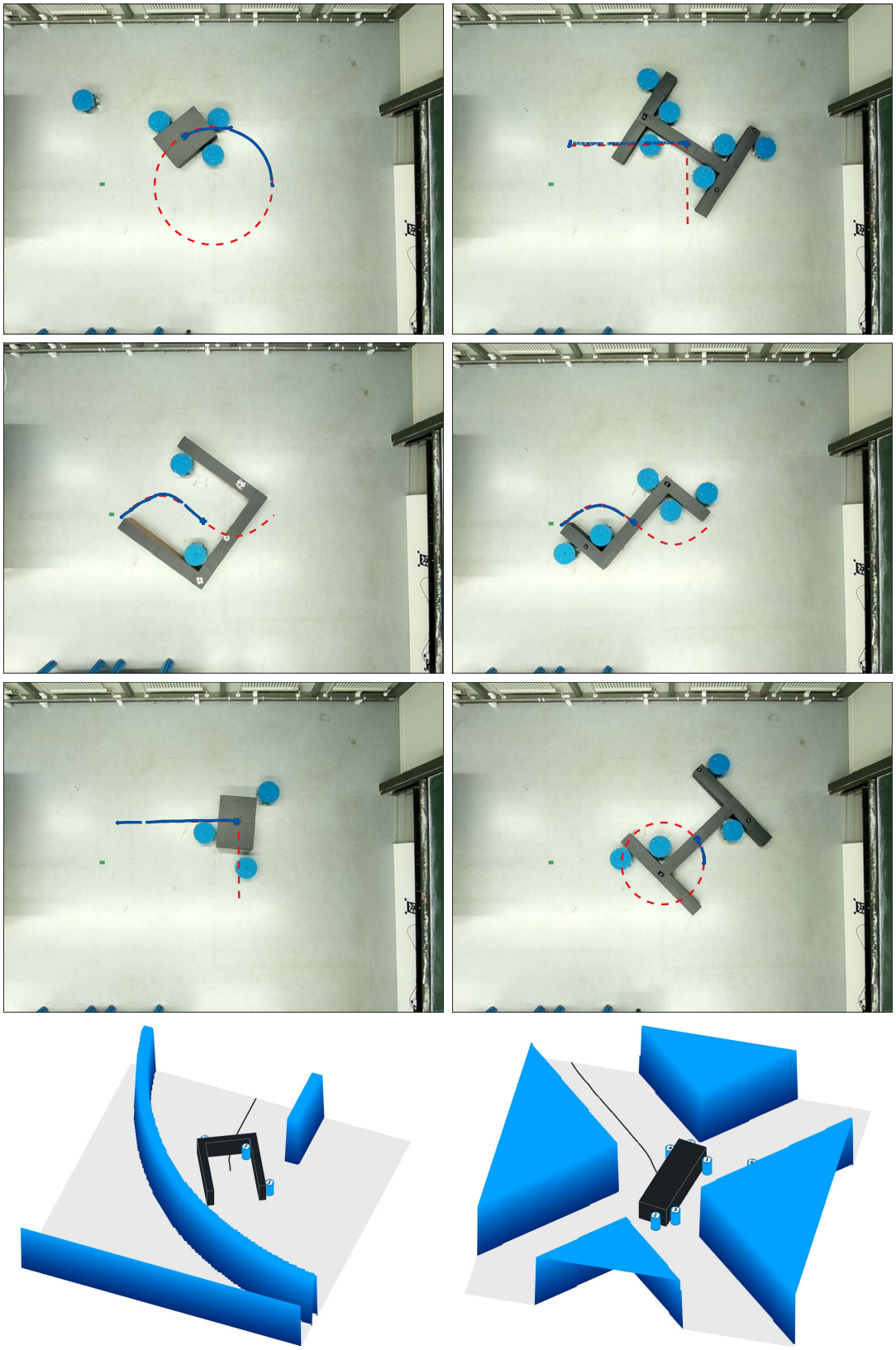


Figure 1.1: Impressions from experiments and simulations on cooperative transportation

graph-algebraic approach to distributed control, which is useful for comparison purposes. Building upon these foundations, Chapter 4 introduces, in rather general terms and based on defining features of distributed cooperative behavior, the conceptual framework and distributed system architecture later applied to the benchmark problem. Moreover, a custom robot design, devised for this thesis's requirements, is introduced, including some insights from building a mechanical model of the robot. It is then the purpose of Chapter 5 to develop the methods that allow solving concrete tasks within the framework from the previous chapter, with a focus on the needs of the transportation problem. Distributed optimization and distributed model predictive control become the key techniques to be used within the approach. Furthermore, local and global navigation schemes are devised to allow a safe navigation of the robots around the object as well as the planning of collision-free paths to transport the object through obstacle-ridden environments. While some of the methods developed are analyzed and put to the test right away, it is Chapter 6 that lets all methods work in tandem for cooperative transportation. At first, the devised transportation scheme is analyzed based on simulations, focusing on specific aspects in a well-controlled environment without unknown disturbances interfering. Subsequently, experiments using the custom-built robotic hardware show whether the findings extend to the physical domain. In addition, by means of large-scale simulations, the transportation through obstacle-ridden environments is investigated. Finally, the thesis's findings and contributions are summarized in the concluding chapter, providing directions into which subsequent research endeavors may venture.

Chapter 2

Fundamentals from Modeling

Automating real-world tasks by employing robotic agents needs hardware that is appropriate to handle the task. The relationship between the approach to the task and the type of hardware employed is reciprocal. If the hardware design is fixed, its limitations confine the set of possible task solution schemes. In contrast, a clear understanding of the task's physical requirements is essential for picking, refining, or designing a robot. As an immediate consequence, the dynamic behavior of the robots employed needs to be well understood. An accurate mechanical or mechatronical model is, therefore, essential. The model needs to be simple enough to allow the timely calculation of long dynamic simulations, but accurate enough to capture the aspects most critical for the setting. Identifying the latter is one of the critical challenges to engineers working in the field. However, merely being able to adequately describe the behavior of a mechatronical system is insufficient for robotics. For a system to be considered 'robotic', it needs to operate in an automated manner, giving rise to the decisive roles of automatic control and of algorithmics. Many control methods directly rely on a mathematical model of the system and, even for those that do not, control design can benefit significantly from a simulation-driven workflow. Yet, devising algorithms that automate robotic behavior requires an entirely different notion of modeling. Whereas dynamic mechanical models usually consist of differential or differential-algebraic equations, algorithms for automated behavior need a mathematical description of the robots' environment, the task, and the information necessary for the task solution. Hence, it is necessary to look at how the relevant information can be modeled, and how algorithms work with the information to decide on the robots' actions. Therefore, this chapter does not only introduce the required aspects of mechanical modeling, but also considers the modeling of information as necessary for the challenges found in this thesis. Starting with mechanics, the framework of multibody system dynamics is introduced briefly. Some notation introduced therein will prove useful for the description of motion throughout the whole thesis. The subsequent section is devoted to the modeling of information, considering graphs, paths, as well as polygons and convex polytopes as foundational building blocks for

robotic task solution. Building upon these, a brief look at the mathematical representation of the robots' workspace concludes the chapter.

2.1 Mechanical Modeling – Multibody Systems

Building a dynamic model for mechanical systems can be divided into two subaspects, kinematics and kinetics. Whereas kinetics studies the relationship between forces and motion, kinematics deals with the pure description of motion. It introduces the necessary formalisms, considers the description of motion in different kinds of coordinates and reference frames, but it is oblivious to the cause of motion. As such, it is useful beyond classical mechanics and can be considered its own field [BottemaRoth90] with applications such as robotic motion planning, computer vision, and computer graphics. In distributed robotics, each robot and potentially each of its sensors and actuators perceives or acts with regard to its own sense of direction, motivating the usage of various reference frames moving with the robots or with parts of them. However, tasks to be solved by the robots and the fundamental axioms of kinetics are usually formulated with respect to an inertial frame of reference. This makes transformations between different reference frames essential and highlights the necessity of a clear notational convention to indicate the reference frame a quantity is given in. To that end, assume that a quantity of interest $\boldsymbol{z}(t) \in \mathbb{R}^3$ shall be analyzed, with $t \geq 0$ denoting time. For instance, this quantity can be the position of a specific point of a robot, such as its center of mass, or a velocity or acceleration. It shall be given in the standard basis of a Cartesian coordinate system of an inertial frame of reference $\mathcal{K}_\mathrm{I}$. The notation ${}^{\alpha}\boldsymbol{z}(t)$ shall denote the same quantity expressed in the basis of the moving reference frame $\mathcal{K}_\alpha(t)$, with the mutually orthogonal basis vectors $\boldsymbol{e}_1^\alpha(t)$, $\boldsymbol{e}_2^\alpha(t)$, $\boldsymbol{e}_3^\alpha(t) \in \mathbb{R}^3$ of unit length. Generally, the absence of an identifier in the upper-left superscript shall mean that the corresponding quantity is given in the basis of $\mathcal{K}_\mathrm{I}$. The notation ${}^{\breve{\alpha}}\boldsymbol{z}(t)$ shall denote the quantity not only represented in the basis of the frame $\mathcal{K}_\alpha(t)$, but relative to its moving origin $\boldsymbol{o}_\alpha(t) \in \mathbb{R}^3$, e.g., ${}^{\breve{\alpha}}\boldsymbol{z}(t) = {}^{\alpha}\boldsymbol{z}(t) - {}^{\alpha}\boldsymbol{o}_\alpha(t)$ for a quantity describing a position. All considered coordinate systems shall be orthonormal and right-handed. Therefore, there exists an orthogonal rotation matrix ${}^{\mathrm{I}}\boldsymbol{S}_\alpha(t)$ that transforms the coordinates of a vector given in the basis of $\mathcal{K}_\alpha(t)$ to a coordinate vector given in the basis of $\mathcal{K}_\mathrm{I}$, i.e., $\boldsymbol{z} = {}^{\mathrm{I}}\boldsymbol{S}_\alpha(t)\,{}^{\alpha}\boldsymbol{z}$. Subsequently, for brevity, the time-dependency may be omitted notation-wise if it does not cause ambiguities, e.g., $\mathcal{K}_\alpha = \mathcal{K}_\alpha(t)$. Often, the analysis of planar motion suffices. In such instances, the first two coordinates will be used to describe the motion, disregarding the third coordinate.

This notation is useful to formulate mechanical models for a wide class of technical systems. Usually, robots and many other technical systems consist of a set of individual, coupled construction elements. An approach to derive an idealized model for systems of this kind is the framework of multibody systems [SchiehlenEberhard14, Woernle16].

Therein, the system is regarded as consisting of bodies that make up the whole mass of the system, whereas the idealized elements coupling the bodies are assumed to be massless. If deformations of the bodies are negligible, they can be modeled as rigid bodies. Then, the system's elasticity is concentrated in the coupling elements. If the system has a tree-like structure without non-holonomic constraints, it is often possible to derive an ordinary differential equation describing the dynamics of the system. Focusing on the latter case, for a system consisting of n_b rigid bodies, that are coupled so that the system has n_f degrees of freedom, the kinematics can be expressed in terms of a minimal set of n_f independent coordinates, so-called generalized coordinates $\boldsymbol{q} \in \mathbb{R}^{n_\mathrm{f}}$. They uniquely define the position of every point of the system. In the first step, each body $i \in \{1, \dots, n_\mathrm{b}\}$ is described kinematically by the position of its center of mass $\boldsymbol{r}_i(\boldsymbol{q})$ and the rotation matrix ${}^\mathrm{I}\boldsymbol{S}_{\mathrm{B}_i}(\boldsymbol{q})$. Therein, $\mathcal{K}_{\mathrm{B}_i}$ is a body-fixed frame with its origin located in $\boldsymbol{r}_i(\boldsymbol{q})$. The angular velocity vector $\boldsymbol{\omega}_i$ can be inferred from the skew-symmetric matrix $\tilde{\boldsymbol{\omega}}_i = {}^\mathrm{I}\dot{\boldsymbol{S}}_{\mathrm{B}_i}\,{}^\mathrm{I}\boldsymbol{S}_{\mathrm{B}_i}^\mathsf{T}$ by means of the relationship $\boldsymbol{\omega}_i^\mathsf{T} = \begin{bmatrix}(\tilde{\boldsymbol{\omega}}_i)_{3,2} & (\tilde{\boldsymbol{\omega}}_i)_{1,3} & (\tilde{\boldsymbol{\omega}}_i)_{2,1}\end{bmatrix}$. This allows to express the translational acceleration $\boldsymbol{a}_i$ and the angular acceleration $\boldsymbol{\alpha}_i$ of each body in the form

$$\boldsymbol{a}_i = \boldsymbol{J}_{\mathrm{T},i}(\boldsymbol{q}, t)\,\ddot{\boldsymbol{q}} + \bar{\boldsymbol{a}}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, t), \tag{2.1}$$

$$\boldsymbol{\alpha}_i = \boldsymbol{J}_{\mathrm{R},i}(\boldsymbol{q}, t)\,\ddot{\boldsymbol{q}} + \bar{\boldsymbol{\alpha}}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \tag{2.2}$$

with the Jacobian matrices $\boldsymbol{J}_{\mathrm{T},i}$, $\boldsymbol{J}_{\mathrm{R},i}$ and the local accelerations $\bar{\boldsymbol{a}}_i$, $\bar{\boldsymbol{\alpha}}_i$. These expressions can be used in the Newton and Euler equations to formulate a kinetic description [SchiehlenEberhard14]. Isolating each body of mass m_i, this results in

$$m_i\,\boldsymbol{a}_i = \boldsymbol{f}_i^\mathrm{a} + \boldsymbol{f}_i^\mathrm{r}, \tag{2.3}$$

$$\boldsymbol{J}_i\,\boldsymbol{\alpha}_i + \tilde{\boldsymbol{\omega}}_i\,\boldsymbol{J}_i\,\boldsymbol{\omega}_i = \boldsymbol{l}_i^\mathrm{a} + \boldsymbol{l}_i^\mathrm{r}. \tag{2.4}$$

Therein, the moment of inertia matrix $\boldsymbol{J}_i$ is given with respect to the corresponding body's center of mass. The vectors $\boldsymbol{f}_i^\mathrm{r}$, $\boldsymbol{l}_i^\mathrm{r}$ comprise the reaction forces and moments acting on the ith body, and $\boldsymbol{f}_i^\mathrm{a}$, $\boldsymbol{l}_i^\mathrm{a}$ denote the applied forces and moments, respectively. In the following, it is assumed that the applied forces and moments do not depend on the reactions and that they may only depend on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and time t. Equations (2.3), (2.4) still contain the unknown reactions, although, in many applications, one is solely interested in the rigid body motions. The equations of motion, devoid of the reactions, can be established with the d'Alembert-Lagrange principle [SchiehlenEberhard14]. For the considered multibody system and with the virtual displacements $\delta\boldsymbol{r}_i = \boldsymbol{J}_{\mathrm{T},i}\,\delta\boldsymbol{q}$ and virtual rotations $\delta\boldsymbol{s}_i = \boldsymbol{J}_{\mathrm{R},i}\,\delta\boldsymbol{q}$, this takes the form

$$\sum_{i=1}^{n_\mathrm{b}} \left((\boldsymbol{f}_i^\mathrm{a} - m_i\,\boldsymbol{a}_i)^\mathsf{T}\,\delta\boldsymbol{r}_i + (\boldsymbol{l}_i^\mathrm{a} - \boldsymbol{J}_i\,\boldsymbol{\alpha}_i - \tilde{\boldsymbol{\omega}}_i\,\boldsymbol{J}_i\,\boldsymbol{\omega}_i)^\mathsf{T}\,\delta\boldsymbol{s}_i \right) = 0 \quad \forall \boldsymbol{q} \tag{2.5}$$

$$\implies \delta\boldsymbol{q}^\mathsf{T} \sum_{i=1}^{n_\mathrm{b}} \left[\boldsymbol{J}_{\mathrm{T},i}^\mathsf{T}\,(m_i\,\boldsymbol{a}_i - \boldsymbol{f}_i^\mathrm{a}) + \boldsymbol{J}_{\mathrm{R},i}^\mathsf{T}\,(\boldsymbol{J}_i\,\boldsymbol{\alpha}_i + \tilde{\boldsymbol{\omega}}_i\,\boldsymbol{J}_i\,\boldsymbol{\omega}_i - \boldsymbol{l}_i^\mathrm{a}) \right] = 0 \quad \forall \boldsymbol{q}. \tag{2.6}$$

Using the fact that the virtual generalized displacements δq_j, $j \in \{1, \dots, n_\mathrm{f}\}$, are mutually independent, the equations of motion are obtained and can be written in the form

$$\boldsymbol{M}(\boldsymbol{q}, t)\, \ddot{\boldsymbol{q}} + \boldsymbol{k}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \boldsymbol{f}_\mathrm{ga}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t), \tag{2.7}$$

representing a nonlinear ordinary differential equation for $\boldsymbol{q}(t)$. Therein, $\boldsymbol{M} \in \mathbb{R}^{n_\mathrm{f} \times n_\mathrm{f}}$ is the symmetric and positive definite mass matrix, $\boldsymbol{k} \in \mathbb{R}^{n_\mathrm{f}}$ denotes the vector of the generalized gyroscopic forces, and $\boldsymbol{f}_\mathrm{ga} \in \mathbb{R}^{n_\mathrm{f}}$ is the vector of the generalized applied forces.

The multibody dynamics fundamentals described above suffice for the applications considered in this thesis. A more extensive treatment of multibody systems can be found in the books [SchiehlenEberhard14] and [Woernle16]. Furthermore, the formalism described here can be extended nicely to incorporate elastic bodies subject to small deformations [SchwertassekWallrapp99]. For the large-deformation case, different formalisms are prudent [Shabana97, Shabana20]. These could, e.g., be worthwhile to simulate robots handling soft objects that undergo large deformations.

For model-based control, it is often critical to employ a model posing as little computational demand as possible. Therefore, one often tends to neglect even effects that can significantly influence the model's character or computational complexity, such as friction or exact contact descriptions, hoping that feedback control can compensate for the modeling errors made. Hence, in many applications, for the design of nominal, model-based controllers, an even more lightweight linearized dynamics is employed. Often, the control input $\boldsymbol{u} \in \mathbb{R}^{n_u}$, that is governed by the controller, consists of a subset of the applied forces and moments. Extracting those from the vector of generalized forces, it is possible to write the dynamics (2.7) in the form

$$\boldsymbol{M}(\boldsymbol{q}, t)\, \ddot{\boldsymbol{q}} + \boldsymbol{k}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \bar{\boldsymbol{f}}_\mathrm{ga}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \bar{\boldsymbol{B}}(\boldsymbol{q}, t)\, \boldsymbol{u}. \tag{2.8}$$

Linearizing around the operating point defined by $\boldsymbol{q}_\mathrm{o}$, $\dot{\boldsymbol{q}}_\mathrm{o}$, $\ddot{\boldsymbol{q}}_\mathrm{o}$, $\boldsymbol{u}_\mathrm{o}$ with $\Delta\boldsymbol{q} \coloneqq \boldsymbol{q} - \boldsymbol{q}_\mathrm{o}$, $\Delta\boldsymbol{u} \coloneqq \boldsymbol{u} - \boldsymbol{u}_\mathrm{o}$, the linearized equations of motion

$$\boldsymbol{M}_\ell(t)\ \Delta\ddot{\boldsymbol{q}} + \boldsymbol{P}_\ell(t)\, \Delta\dot{\boldsymbol{q}} + \boldsymbol{Q}_\ell(t)\, \Delta\boldsymbol{q} = \boldsymbol{h}(t) + \boldsymbol{B}_\ell(t)\ \Delta\boldsymbol{u} \tag{2.9}$$

with

$$\boldsymbol{M}_\ell(t) = \boldsymbol{M}(\boldsymbol{q}_\mathrm{o}, t), \tag{2.10}$$

$$\boldsymbol{P}_\ell(t) = \frac{\partial \boldsymbol{k}}{\partial \dot{\boldsymbol{q}}}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t) - \frac{\partial \bar{\boldsymbol{f}}_\mathrm{ga}}{\partial \dot{\boldsymbol{q}}}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t), \tag{2.11}$$

$$\boldsymbol{Q}_\ell(t) = \frac{\partial \boldsymbol{k}}{\partial \boldsymbol{q}}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t) - \frac{\partial \bar{\boldsymbol{f}}_\mathrm{ga}}{\partial \boldsymbol{q}}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t) - \frac{\partial \bar{\boldsymbol{B}}}{\partial \boldsymbol{q}}(\boldsymbol{q}_\mathrm{o}, t)\, \boldsymbol{u}_\mathrm{o} + \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{q}}(\boldsymbol{q}_\mathrm{o}, t)\, \ddot{\boldsymbol{q}}_\mathrm{o}, \tag{2.12}$$

$$\boldsymbol{h}(t) = \boldsymbol{f}_\mathrm{ga}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t) + \bar{\boldsymbol{B}}(\boldsymbol{q}_\mathrm{o}, t)\, \boldsymbol{u}_\mathrm{o} - \boldsymbol{M}(\boldsymbol{q}_\mathrm{o}, t)\, \ddot{\boldsymbol{q}}_\mathrm{o} - \boldsymbol{k}(\boldsymbol{q}_\mathrm{o}, \dot{\boldsymbol{q}}_\mathrm{o}, t), \text{ and} \tag{2.13}$$

$$\boldsymbol{B}_\ell(t) = \bar{\boldsymbol{B}}(\boldsymbol{q}_\mathrm{o}, t) \tag{2.14}$$

is obtained. Usually, $\boldsymbol{q}_\mathrm{o}$ and $\boldsymbol{u}_\mathrm{o}$ are chosen so that they are a steady state-input pair, meaning that $\boldsymbol{h}(t) = \boldsymbol{0}$. Control engineering, system analysis, and model-based control design usually rely on a state-space representation of the examined system. For instance, for the linear dynamics (2.9), with the state $\boldsymbol{x} \coloneqq \begin{bmatrix}\Delta\boldsymbol{q}^\mathsf{T} & \Delta\dot{\boldsymbol{q}}^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and in the case of $\boldsymbol{h}(t) = \boldsymbol{0}$, this yields a state-space representation of the form

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{M}_\ell^{-1}\boldsymbol{Q}_\ell & -\boldsymbol{M}_\ell^{-1}\boldsymbol{P}_\ell\end{bmatrix}}_{=:\boldsymbol{A}} \boldsymbol{x} + \underbrace{\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{M}_\ell^{-1}\boldsymbol{B}_\ell\end{bmatrix}}_{=:\boldsymbol{B}} \Delta\boldsymbol{u}. \tag{2.15}$$

Therein, $\boldsymbol{I}$ is the identity matrix. In addition, an output $\boldsymbol{y} \in \mathbb{R}^{n_y}$ may be defined as a function of the state and input, describing, e.g., measurements available to observers or controllers. In the linear case, the output takes the form

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\Delta\boldsymbol{u}. \tag{2.16}$$

2.2 Information Modeling

Robotic systems are employed to automate tasks. Except for the most simple, static tasks, this requires considering and processing information gathered at system runtime to automatically deduce the robotic behavior appropriate in the current situation. Usually, the gathered information needs to be reprocessed to obtain a representation that is adequate for decision-making algorithms to operate on. In a field as diverse as robotics, a disquisition on information modeling can only be incomplete, with this section merely focusing on the aspects relevant to the model example of cooperative object transportation. Still, the foundations described are useful for a wide variety of tasks posed, among them tasks that include navigation or the handling of objects, with most concepts being useful far beyond robotics. The first subsection introduces graphs, which find application, e.g., in navigation and distributed control. The second subsection deals with the definition of paths, which may be deduced from a navigation graph. The third subsection introduces polygons and convex polytopes, which can be useful to describe objects to be handled, the workspace environment, or constraints in optimal control and optimization. Finally, it is described how environments, and the navigation therein, can be dealt with algorithmically.

2.2.1 Graphs

A graph is an abstract mathematical concept and can be defined as a tuple of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ [GodsilRoyle01, HartNilssonRaphael68]. Yet, illustrations of graphs can be found in many places of daily life, e.g., in maps of subway systems. Thus, in some important applications, the nodes represent points of interest on a map, e.g., locations

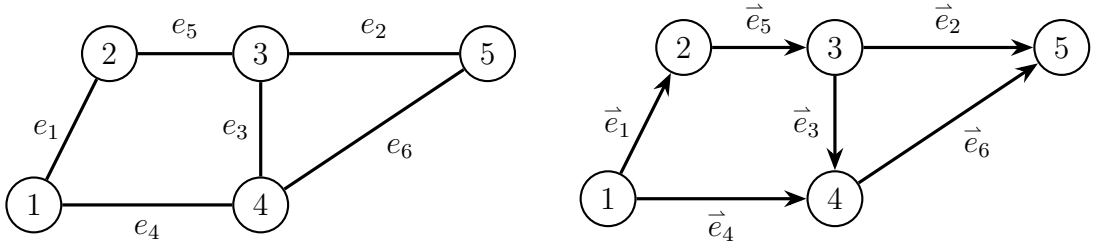


Figure 2.1: Illustration of an undirected and of a directed graph, with the set of nodes $\mathcal{V} = \{1, 2, \ldots, 5\}$ and the sets of edges $\mathcal{E} = \{e_1, e_2, \ldots, e_6\}$, and $\vec{\mathcal{E}} = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_6\}$, respectively

to be inspected by a robot, or subway stations in a city. An edge signals that one can directly move between the two points of interest connected by the edge, without passing through another node. In a different interpretation, each node represents a robot, and the edges embody the possibility of communication between the robots. In literature, graph nodes are also called vertices [GodsilRoyle01]. In this thesis, vertices will appear in the description of polytopes, so the denomination as nodes helps to prevent equivocality. In some applications, communication might be unidirectional for technical reasons, resembling a one-way street in a road network, whereas, in other applications, each communication channel may be bidirectional by nature. Therefore, it makes sense to distinguish between directed and undirected graphs and edges. For a directed graph $\vec{\mathcal{G}} = \left(\mathcal{V}, \vec{\mathcal{E}}\right)$, the set of edges $\vec{\mathcal{E}} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of ordered pairs of nodes, defining a sense of direction by their order. For $v_i, v_j \in \mathcal{V}$, the edge $(v_i, v_j) \in \vec{\mathcal{E}}$ leads from v_i to v_j. This kind of edge is called a directed edge or arc. In contrast, the set of edges $\mathcal{E}$ of an undirected, or simple, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains unordered pairs of nodes. In this thesis, only graphs without self-loops are considered, i.e., it is assumed that for all $(v_i, v_j) \in \mathcal{E}$ it holds that $v_i \neq v_j$. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ may be written as a directed graph $\vec{\mathcal{G}} = \left(\mathcal{V}, \vec{\mathcal{E}}\right)$ by replacing each undirected edge with two directed edges so that $\{v_1, v_2\} \in \mathcal{E} \iff (v_1, v_2) \in \vec{\mathcal{E}} \wedge (v_2, v_1) \in \vec{\mathcal{E}}$ for all pairs of nodes $v_1, v_2 \in \mathcal{V}$. In a directed graph, a successor v_j of node v_i is a node for which there exists an edge $(v_i, v_j) \in \vec{\mathcal{E}}$, meaning that it can be reached from v_i by following a directed edge that leads from v_i to v_j. Usually, graphs are illustrated by depicting nodes as circles, directed edges as arrows, and undirected edges as lines, see Figure 2.1. Such an illustration need not encode any spatial information about the nodes since a graph only encodes the relationship between the nodes. Hence, there are many possible illustrations of a graph. However, for some applications, e.g., path planning and navigation, it can be instructive to draw the nodes at their physical locations.

Each node v_i, v_j of an edge $\vec{e} = (v_i, v_j)$ or $e = \{v_i, v_j\}$ is said to be incident with the edge. Two nodes are adjacent if they are incident with the same edge. Therefore, adjacency describes a relationship between two nodes, whereas incidence describes a relationship between a node and an edge. An undirected graph is complete if all nodes are adjacent.

For graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ or $\vec{\mathcal{G}} = \left(\mathcal{V}, \vec{\mathcal{E}}\right)$, a (graph-theoretic) path is a sequence $v_1, \ldots, v_n$ of nodes $v_i \in \mathcal{V}$ with $\{v_j, v_{j+1}\} \in \mathcal{E}\ \forall j \in \{1, \ldots, n-1\}$ or $(v_j, v_{j+1}) \in \vec{\mathcal{E}}\ \forall j \in \{1, \ldots, n-1\}$, respectively. An undirected graph is connected if any two nodes are connected by a path. For some applications, such as finding shortest paths, it is useful to introduce so-called edge weights that may, for instance, encode the Euclidean distance between the physical representations of adjacent nodes. Hence, in a weighted directed graph, there exists a mapping $\check{w}\colon \vec{\mathcal{E}} \to \mathbb{R}$ assigning a weight to each edge.

While illustrations of graphs as in Figure 2.1 are intuitive and can already be very useful when devising algorithms, it is also possible to represent graphs in the form of matrices. These matrices' algebraic properties relate to properties of the graph, allowing graph interpretations of well-known algebraic results. An encompassing algebraic treatment of graphs can be found in [GodsilRoyle01], with the following delineations being limited to this thesis' needs. A matrix that will later on prove important for control design is the incidence matrix $\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}} \in \mathbb{R}^{n_\mathcal{V} \times n_\mathcal{E}}$ of a directed graph $\vec{\mathcal{G}} = \left(\mathcal{V}, \vec{\mathcal{E}}\right)$ with $n_\mathcal{V} \coloneqq |\mathcal{V}|$ nodes, $n_\mathcal{E} \coloneqq \left|\vec{\mathcal{E}}\right|$ edges, the node set $\mathcal{V} = \{v_1, \ldots, v_{n_\mathcal{V}}\}$, and the edge set $\vec{\mathcal{E}} = \{\vec{e}_1, \ldots, \vec{e}_{n_\mathcal{E}}\}$. Its entries are given by

$$\left(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}\right)_{i,j} = \begin{cases} 1 & \text{if } \exists\, \hat{v} \in \mathcal{V}\colon\ (\hat{v}, v_i) = \vec{e}_j, \\ -1 & \text{if } \exists\, \hat{v} \in \mathcal{V}\colon\ (v_i, \hat{v}) = \vec{e}_j, \\ 0 & \text{else.} \end{cases} \tag{2.17}$$

For instance, for the directed graph from Figure 2.1, this yields

$$\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}} = \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.18}$$

Each of its columns corresponds to an edge and each of its rows to a node, with the entry 1 indicating that the edge ends at the node, and a -1 indicating that it starts at the node. Therefore, each column of the matrix has exactly two entries that sum to zero. Hence, with the vector of ones $\mathbf{1}_{n_\mathcal{V}} \in \mathbb{R}^{n_\mathcal{V}}$, it holds that $\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}^\mathsf{T} \mathbf{1}_{n_\mathcal{V}} = \mathbf{0}$. Whereas the incidence matrix encodes the relationship between edges and nodes, the so-called adjacency matrix encodes the relationship between nodes. Subsequently, only the adjacency matrix $\boldsymbol{\mathcal{A}}_{\vec{\mathcal{G}}}$ of a weighted directed graph $\vec{\mathcal{G}}$ is relevant. Its entries are defined as

$$\left(\boldsymbol{\mathcal{A}}_{\vec{\mathcal{G}}}\right)_{i,j} = \begin{cases} \check{w}(v_i, v_j) & \text{if } (v_i, v_j) \in \vec{\mathcal{E}}, \\ \infty & \text{else.} \end{cases} \tag{2.19}$$

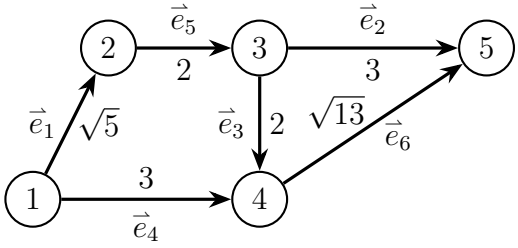
Sometimes it makes sense to work with directed versions of undirected graphs. In this vein, an oriented graph [GodsilRoyle01] is a graph that is obtained from an undirected graph

by replacing each undirected edge with an arbitrarily directed one. An orientation of a graph shall be the mapping that maps an undirected graph to an oriented graph. Indeed, the directed graph from Figure 2.1 can be seen as an oriented graph since there exists an orientation that maps the undirected graph in the figure to the directed one. While there may be many possible orientations of a graph, some key properties of oriented graphs do not depend on the specific orientation. For instance, if the underlying undirected graph is connected, the rank of the incidence matrix of any corresponding oriented graph is given by $\operatorname{rank}\left(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}\right) = n_{\mathcal{V}} - 1$ [GodsilRoyle01]. As will become clear in Section 3.3, there exists a physical interpretation of this fact in formation control, relating to the intuitive notion that a given relative positioning of a group of robots defines the robot positions up to an undefined, common translation.

Maybe the most intuitive application of graphs in robotics and beyond is navigation. The problem of finding a suitable path through an environment may be reduced to a map in the form of a weighted graph – similar to how a human driver would plan a road trip by finding and memorizing roads (edges) that connect certain cities (nodes) in a road network. Often, one is interested in a shortest path, meaning that, among all possible paths leading from the start to the goal, the sum of the chosen edges' weights is minimal. The edge weights can, but need not, be distances. They might also refer to expected travel time or energy consumption along the edge. Due to its usefulness in robotic navigation and to provide an algorithmic example on how to operate on graphs, the remainder of this subsection is devoted to the solution of the shortest path problem. Subsequently, it is assumed that the graph is directed, connected, and that all edge weights are positive, i.e., $\breve{w} : \vec{\mathcal{E}} \to \mathbb{R}_+$. By virtue of the physical meaning of the graph, these assumptions are not very restrictive. There exist some very well-known algorithms to calculate shortest paths in this kind of graph, e.g., Dijkstra's algorithm [Dijkstra59] and the A* algorithm [HartNilssonRaphael68], which both guarantee that the shortest path is found. Both of these algorithms are so-called greedy algorithms. Beginning at the start node, when deciding which node to consider next, they greedily pick the node that seems to be an optimal choice with regard to a cost function. The algorithms memorize the information necessary to reconstruct the shortest currently known path to each node previously considered. The length of that path is also recorded. The cost function is designed to reduce the number of nodes that have to be considered to find the shortest path to the goal. In Dijkstra's algorithm, a node's current cost is equal to the length of the shortest previously discovered path from the start to the node in question. The edge weights needed to calculate this cost can be read directly from the adjacency matrix. The A* algorithm adds a heuristic cost term that is designed to lower-bound the length of the shortest path from the candidate node to the goal node. Indeed, under a mild assumption on this additional heuristic cost term [HartNilssonRaphael68], it can be proven that the A* algorithm is in some sense optimal with regard to the number of nodes considered to find a shortest path. In navigation, with the edge weights denoting Euclidean distances between

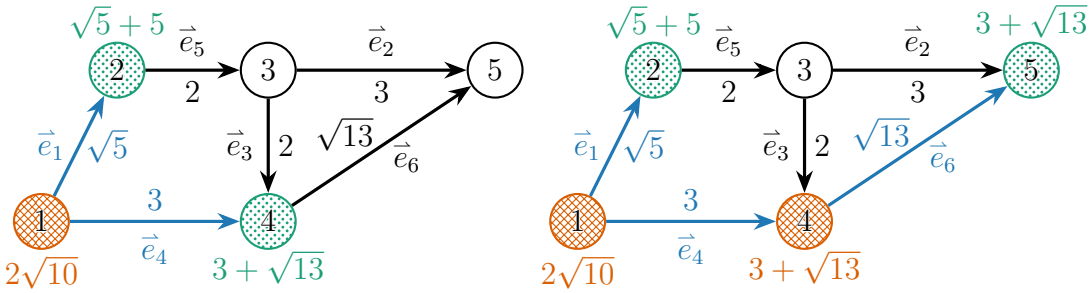
the locations represented by the nodes, this assumption is, e.g., fulfilled if the heuristic cost is the Euclidean distance between the candidate node and the goal node. Then, no other algorithm operating on the same set of information can consistently consider fewer nodes than the A* algorithm to reliably find a shortest path [HartNilssonRaphael68]. For this reason, it is still widely applied and cited in all kinds of works that need to solve the shortest path problem – and for the same reason, it is employed in this thesis in all instances where shortest paths are of use. Example 1 gives an illustratory example on how the algorithm operates.

Example 1: The Shortest Path Problem and the A* Algorithm



Without delving into implementational details, the A* algorithm can be readily understood by tracing its steps in a small example graph. Consider the graph displayed on the right-hand side, which is a weighted version of the directed graph from Figure 2.1. The edge weights are written alongside the edges. They are proportional to the Euclidean distances between the node centers as printed on paper, e.g., $\breve{w}(\vec{e}_1) = \sqrt{5}$. The goal is to find the shortest path from node 1 to node 5. The Euclidean distance from a candidate node to the goal node is used as the heuristic cost term. The algorithm maintains a set of so-called open nodes that is initialized to contain only the start node, which is node 1 in the example. Similarly, a set of closed nodes, which is empty initially, contains nodes that will not be considered anymore. Each iteration of the algorithm starts by considering an open node with lowest cost, preferably the goal node if it qualifies. In the example's first iteration, node 1 is considered. If the candidate node is the goal node, the algorithm terminates by moving it to the set of closed nodes. If not, the algorithm adds all non-closed successor nodes of the candidate node to the set of open nodes and calculates all successor nodes' cost function values. Furthermore, it memorizes the individual edges that lead to each successor if the shortest path to the successor via the candidate node is shorter than any corresponding path discovered previously. The iteration concludes by moving the candidate node from the set of open nodes to the set of closed nodes. Thus, after the first iteration, nodes 2 and 4 are in the set of open nodes, with costs $\sqrt{5}+5$ and $3+\sqrt{13}$, respectively. Since node 4 has lower cost, it is the next candidate node. It only has one successor node, which is the goal node. The algorithm adds it to the set of open nodes and notes that edge $\vec{e}_6$ is used to reach it. After the second iteration, nodes 2 and 5 are in the set of open nodes, but the goal node 5 has the lowest cost, and hence it is the new candidate node. Consequently, the algorithm terminates. The shortest path can be reconstructed by backtracking from node to node via the memorized edges, yielding the shortest path $\vec{e}_4, \vec{e}_6$. The

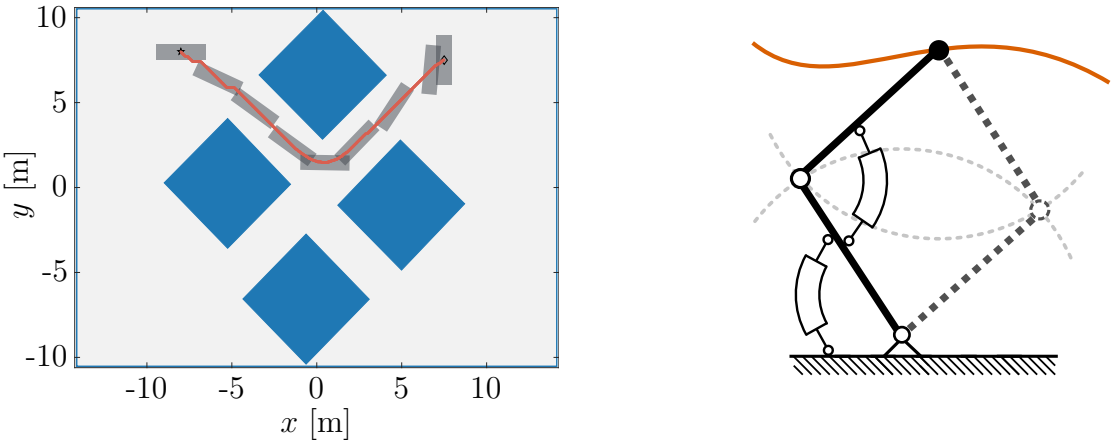
drawings below illustrate the situation at the ends of the first two iterations of the A* algorithm. Closed nodes are hatched in orange, open nodes are dotted in green, and memorized edges are drawn in blue. The cost values of nodes for which they have been calculated are printed in the node's color next to the respective node.



2.2.2 Paths, Trajectories, and Motion Planning

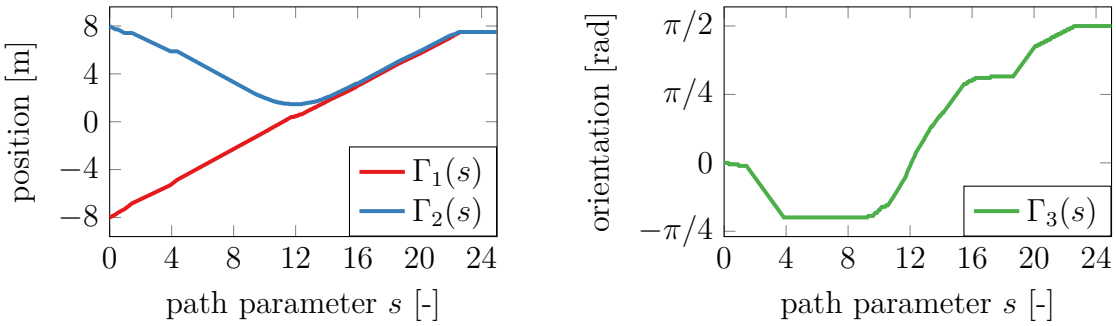
In robotics, the term path usually does not refer to a graph-theoretic path as introduced in the preceding subsection. Instead of a sequence of distinct nodes, it refers to a continuous function $\boldsymbol{\Gamma}\colon \mathcal{I} \subseteq \mathbb{R} \to \mathcal{Y} \subseteq \mathbb{R}^d$ that maps a path parameter from an interval $\mathcal{I}$ to a state or output of a robotic system. When thinking of a robot modeled as in Section 2.1, a point on the path may, e.g., refer to a specific value of the generalized coordinate vector, thereby defining the desired configuration or pose of the robot. Therefore, in robotics literature, the set of all admissible values of the generalized coordinates is called the configuration space $\mathcal{C}_\mathrm{S}$ [LynchPark17]. In some applications, there can be multiple possible configurations, i.e., values of the generalized coordinate vector, that result in the same path point. This can be the case for the end effector of a robotic manipulator. Two illustrative examples can be found in Example 2.

Example 2: Paths



The two images above show paths in two different applications in the field of robotics. The left image shows a scenario that may appear in cooperative object

transportation in which a rectangular object shall be transported through an environment that contains obstacles, which are depicted in blue. A possible path for the object is depicted in orange, with the object orientations along the path illustrated by the object shape drawn in translucent gray. The path leads from the upper left to the upper right. The image on the right-hand side illustrates a path, drawn in orange, to be followed by a point of a robotic manipulator consisting of two articulated robotic arms depicted as thick straight lines. The path in the transportation case incorporates both position coordinates as well as a coordinate related to the orientation of the object. For every point along the path, there is exactly one fitting configuration of the object. In contrast, there are up to two configurations for every path point in the example on the right-hand side, as indicated by the dashed configuration of the robotic manipulator. This kind of illustration does not give any information on the parameterization of the path. A possible parameterization of the object path is shown in the plots below. The path is described by the function $\boldsymbol{\Gamma}\colon [0,\infty) \to \mathbb{R}^3,\ s \mapsto \begin{bmatrix}\Gamma_1(s) & \Gamma_2(s) & \Gamma_3(s)\end{bmatrix}^\mathsf{T}$. The first two coordinates Γ_1, Γ_2 refer to the x- and y-coordinates of the object's position, respectively, whereas the third coordinate Γ_3 refers to the object's orientation by giving its rotation angle relative to the object's initial position.



Insofar, it has not been discussed how the parameterization of the path should be chosen. Depending on the use case, specific choices might be practical. For instance, for a path in the plane or three-dimensional space, an arc-length parameterization can be intuitive since it introduces a natural notion of distance into the parameterization. This is more intricate if the path coordinates contain spatial and rotational quantities since there is no common notion of distance for the two. By itself, a path is a purely geometric object and does not include any notion of time. Henceforth, it does not describe a motion per se. However, it may be parameterized in time, in which case it is usually called a trajectory.

Subsequently, assume that a path shall be planned in a workspace $\mathcal{W} \subseteq \mathbb{R}^n$, with $n = 2$ or $n = 3$, that contains obstacles represented by the closed set $\mathcal{O} \subset \mathcal{W}$. The geometry of the robot in the configuration defined by the generalized coordinate $\boldsymbol{q}$ shall be given by $\mathcal{T}(\boldsymbol{q}) \subseteq \mathbb{R}^n$. The robotic system's output that shall track the path shall be given

by $\boldsymbol{y}(\boldsymbol{q}) \in \mathcal{Y}\ \forall\, \boldsymbol{q} \in \mathcal{C}_\mathrm{S}$. With these quantities, the task of path planning can be defined as finding a path $\boldsymbol{\Gamma} : \mathcal{I} = [a,\, b] \subseteq \mathbb{R} \to \mathcal{Y}$ from some starting point $\bar{\boldsymbol{s}} =: \boldsymbol{\Gamma}(a) \in \mathcal{Y}$ to a goal point $\bar{\boldsymbol{g}} =: \boldsymbol{\Gamma}(b) \in \mathcal{Y}$ so that there exists a continuous function $\boldsymbol{f} : [a,\, b] \to \mathcal{C}_\mathrm{S}$ with $\boldsymbol{\Gamma}(\lambda) = \boldsymbol{y}(\boldsymbol{f}(\lambda))$, $\mathcal{T}\,(\boldsymbol{f}(\lambda)) \cap \mathcal{O} = \emptyset\ \forall\ \lambda \in [a,\, b]$. The latter requirement ensures that, geometrically, the path can be followed by the robot without collisions with the obstacles. Therefore, the environment can be seen as a constraint to path planning. However, this definition of path planning does not consider the dynamics of the robot or object that shall follow the path. For instance, due to inertia and limitations of the actuation, it might be impossible to find control inputs that let the robot follow the path with finite, non-vanishing velocity. Therefore, one may require that there exists a trajectory following the path that can actually be executed. This means that the trajectory is a solution of the differential equation describing the robot dynamics for suitable, admissible control inputs. For this reason, this requirement on path planning is sometimes referred to as a differential constraint [KavrakiLaValle16]. In this interpretation, it is not a purely geometric task anymore, and it may be subsumed under the more general term of motion planning.

Due to the vastly different dynamics that may be considered, motion planning is a very manifold task. Even the underlying, purely geometric problem can be highly complex, with general, exact solutions sometimes being very arduous to obtain, potentially to the point of intractability [KavrakiLaValle16, Sharir89]. Hence, a lot of research focuses on specialized schemes that target specific types of environments or configuration spaces [KavrakiLaValle16]. In the case of general motion planning, the investigation is often restricted to specific dynamics or concentrates on the local, reactive avoidance of obstacles [MinguezLamirauxLaumond16]. Therefore, this section purposefully refrains from looking at specific path, trajectory, or motion planning algorithms since they should usually be tailored to the specific problem at hand.

2.2.3 Convex Polytopes and Convexity

Convexity of sets and functions is highly relevant to optimization and optimization-based control. Moreover, specific kinds of convex sets are highly useful in geometric modeling. A set is convex if the closed line-segment connecting any two points of the set is in the set. For a convex set $\mathcal{M} \subseteq \mathbb{R}^d$, a function $f \colon \mathcal{M} \to \mathbb{R}$ is said to be convex if

$$f(\lambda\, \boldsymbol{a} + (1-\lambda)\, \boldsymbol{b}) \leq \lambda\, f(\boldsymbol{a}) + (1-\lambda)\, f(\boldsymbol{b}) \quad \forall\ \boldsymbol{a}, \boldsymbol{b} \in \mathcal{M},\ \lambda \in [0,1]\,, \tag{2.20}$$

which means that the line segment between the points $\begin{bmatrix}\boldsymbol{a}^\mathsf{T} & f(\boldsymbol{a})\end{bmatrix}^\mathsf{T}$ and $\begin{bmatrix}\boldsymbol{b}^\mathsf{T} & f(\boldsymbol{b})\end{bmatrix}^\mathsf{T}$ lies above the function graph for all $\boldsymbol{a}$, $\boldsymbol{b}$ in the domain of f [BoydVandenberghe04]. It is strictly convex if

$$f(\lambda\, \boldsymbol{a} + (1-\lambda)\, \boldsymbol{b}) < \lambda\, f(\boldsymbol{a}) + (1-\lambda)\, f(\boldsymbol{b}) \quad \forall\ \boldsymbol{a}, \boldsymbol{b} \in \mathcal{M},\ \boldsymbol{a} \neq \boldsymbol{b},\ \lambda \in (0,1)\,. \tag{2.21}$$

An illustration of these definitions is provided in Figure 2.2.

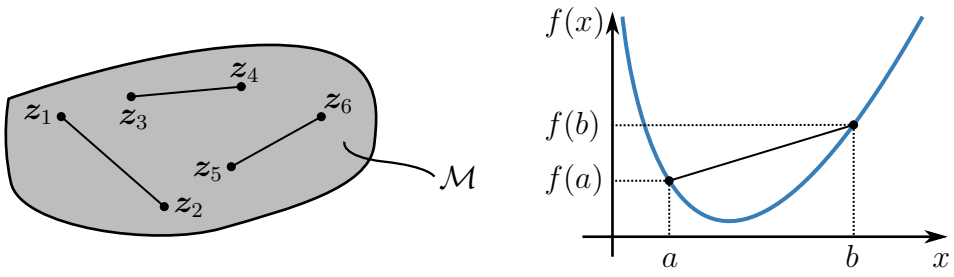


Figure 2.2: Illustration of a convex set $\mathcal{M}$ and of a (strictly) convex function $f : \mathbb{R} \to \mathbb{R}$

Convex polytopes, as a special kind of convex sets, deserve a separate examination since they can be represented by a finite set of linear inequalities, making them amenable to algorithmic treatment. Nevertheless, they are general enough to represent or approximate a useful variety of sets. This portends their usefulness in optimization, computer graphics, and simulations. Even certain non-convex sets can be represented as the union of convex polytopes. Unlike convexity, the formal definitions and nomenclature of polytopes and related concepts are inconsistent and contradicting across different authors and research fields [GrünbaumShephard69, O'Rourke98, BoydVandenberghe04]. In this thesis, a convex polytope $\mathcal{P}$ is defined as the intersection of finitely many closed half-spaces. In d dimensions, this can be written as

$$\mathcal{P} = \bigcap_{i=1}^{n_\mathrm{h}} \{\boldsymbol{z} \in \mathbb{R}^d \mid \boldsymbol{c}_i^\mathsf{T} \boldsymbol{z} \leq d_i\} = \{\boldsymbol{z} \in \mathbb{R}^d \mid \boldsymbol{C}\boldsymbol{z} \leq \boldsymbol{d}\} \tag{2.22}$$

with $0 < n_\mathrm{h} < \infty$, $\boldsymbol{c}_i \in \mathbb{R}^d$, and the rows of the matrix $\boldsymbol{C} \in \mathbb{R}^{n_\mathrm{h} \times d}$ being $\boldsymbol{c}_i^\mathsf{T}$, $i = 1, \ldots, n_\mathrm{h}$. Here and in the following, inequalities in vector equations denote element-wise inequalities. If non-empty and of dimension $d-1$, the intersection of a hyperplane $\{\boldsymbol{z} \in \mathbb{R}^d \mid \boldsymbol{c}_i^\mathsf{T} \boldsymbol{z} = d_i\}$ with the boundary of the polytope gives a facet of the polytope. Individual points obtained by intersecting facets are called vertices. A polytopic set or polytope shall denote a set that can be represented as the union of a finite number of convex polytopes. Hence, it may be non-convex, but, if it is bounded, its boundary is composed of finitely many planar, bounded, convex facets. A bounded two-dimensional polytopic set is called a polygon.

For bounded convex polytopes, there exist two representations that are very useful in different circumstances. The representation used in the definition above is referred to as the half-space representation. However, it is also possible to represent a bounded convex polytope as the so-called convex hull of a finite set of points. The convex hull of the set $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_\mathrm{c}}\} \subset \mathbb{R}^d$ is defined as the convex combination of all the points in the set, i.e.,

$$\mathrm{convhull}\left(\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_\mathrm{c}}\}\right) = \left\{ \sum_{i=1}^{n_\mathrm{c}} \lambda_i \boldsymbol{v}_i \;\middle|\; \sum_{i=1}^{n_\mathrm{c}} \lambda_i = 1, \quad \lambda_i \geq 0 \;\; \forall\, i \in \{1, \ldots, n_\mathrm{c}\} \right\}. \tag{2.23}$$

This kind of representation of a bounded convex polytope is called the vertex representation. This is due to the fact that the set of points to be convexly combined must at least contain

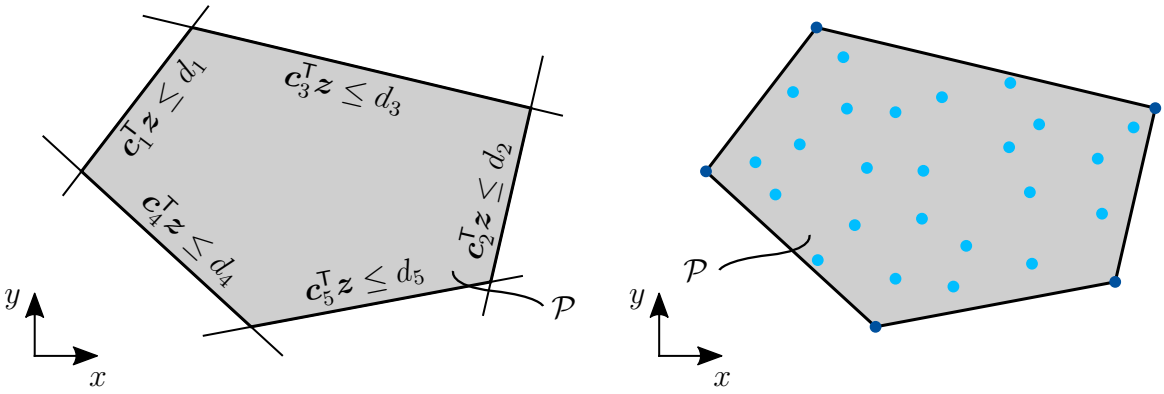


Figure 2.3: Illustration of the vertex and half-space representations of a bounded convex polytope $\mathcal{P}$. The polytope is the convex hull of all points depicted in blue, although the dark blue points, which are the vertices of $\mathcal{P}$, suffice to describe $\mathcal{P}$ as their convex hull.

the vertices of the polytope. An illustration of the vertex and half-space representations of a bounded convex polytope is given in Figure 2.3.

If a polytope represents an object to be handled by robots or obstacles in the workspace, checking whether a point is in the polytope is a very useful elementary operation that may be executed many times to detect collisions in path planning and simulation. Therefore, the efficiency of such an operation is paramount. For a convex polytope in half-space representation as in Equation (2.22), this can be done by checking whether $\boldsymbol{C}\boldsymbol{z} \leq \boldsymbol{d}$ holds for the point $\boldsymbol{z}$ that shall be probed. Being amenable to vectorized evaluation, this can be executed efficiently on many modern processing units. However, objects, obstacles, and the like can be non-convex and, therefore, not representable in half-space representation. This warrants to look at strategies to subdivide non-convex polytopes into a set of convex ones. Due to the nature of many applications, this task's treatment is often restricted to two or three dimensions. For the applications in this thesis, two dimensions suffice and subsequent disquisitions are limited to that case. In two dimensions, a non-convex polygon can be represented as the union of a finite number of triangles, which is called a triangulation of the set. A very well-known triangulation that can be calculated efficiently is the Delaunay triangulation of a set of points. It can directly triangulate a convex polygon when supplied with the polygon's vertices. In the case of a non-convex polygon, the triangulation covers the polygon's convex hull. The Delaunay triangulation maximizes the smallest angle of the triangles among all possible triangulations [BergEtAl08], which can benefit numeric algorithms operating on the triangulation [Sibson78]. The triangulation can be interpreted as a geometric realization of a graph with the triangle vertices as graph nodes and the straight-line segments as edges. To adequately triangulate non-convex polygons, it would be useful if all facets of the polygon would correspond to an edge in this graph. Then, each triangle's interior would either be a subset of the polygon or it would lie outside the polygon. Hence, a subset of the triangles would be a triangulation of the non-convex

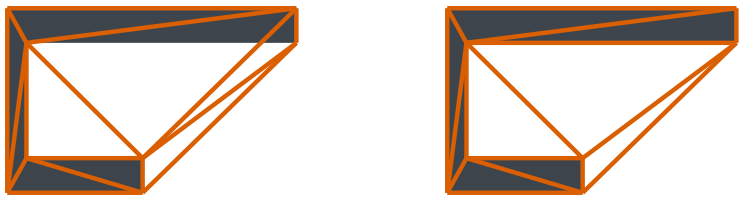


Figure 2.4: The Delaunay triangulation of the dark-gray object's vertices is given on the left-hand side. A constrained Delaunay triangulation, containing all the edges of the object, is given on the right-hand side.

polygon. However, in general, not every facet has a corresponding graph edge. But there exist techniques that allow to enforce that specific edges are part of the graph. The result need not have the favorable numerical properties of a Delaunay triangulation anymore, but there exist calculation techniques that try to retain the properties as far as possible [Chew89, Sloan93]. In literature, this kind of triangulation is referred to as a constrained Delaunay triangulation. Figure 2.4 illustrates the Delaunay triangulation and its constrained counterpart for an object shape as a group of robots could transport it.

Computational geometry, for instance in the form of triangulation, as well as convex polytopes can each be considered their specialized fields of research [O'Rourke98, Grünbaum03]. Hence, relying on them in robotics algorithms allows to benefit from the manifold results in these areas. Later, convex polytopes will resurface in optimization-based control and organization to represent constraint sets. While Figure 2.4 gives some insight into representing objects handled by robots, modeling the robots' workspace warrants further discussion.

2.2.4 Workspace Modeling

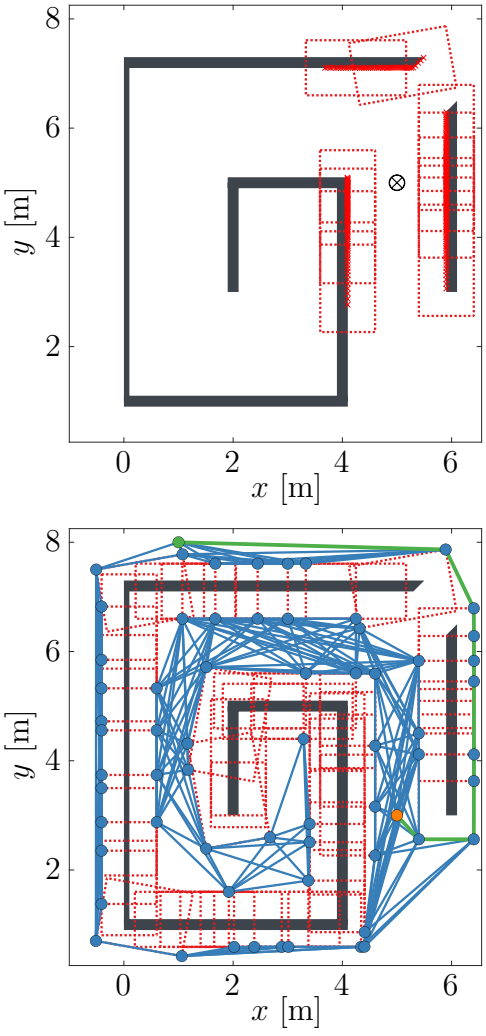
The concept of the workspace has already appeared during path and motion planning in Section 2.2.2, with the aim to devise paths or trajectories that evade obstacles in the workspace environment. The successful accomplishment of this task necessitates a mathematical model of the workspace's geometry or of the obstacles therein. Such a mathematical representation is also necessary for simulation purposes, e.g., for collision detection. Similar to the task of motion planning discussed above, there is an abundance of approaches resulting from a variety of application-specific considerations. For instance, these may include the kind of robot and obstacles considered, and whether the workspace model needs to be built in real time based on an evolving set of information or whether it can be built in a preprocessing step. In particular, integrating additional sensor information, such as point clouds from light detection and ranging sensors, can pose a significant challenge for some approaches but can be straightforward for others. One possible approach is to represent the obstacles in the workspace as polytopic sets [O'Rourke98].

Alternatively, the workspace may be discretized into a set of volumetric cells, with each cell encoding information on whether the corresponding volume is considered to be occupied by obstacles [HornungEtAl13]. Nevertheless, with regard to the algorithmic processing of the workspace, some concepts can be found that can be useful irrespective of how exactly the workspace is modeled. Many approaches try to construct a connectivity graph [Sharir89] based on the information encoded in the model of the workspace [KavrakiLaValle16, O'Rourke98]. Nodes in the connectivity graph represent finitely many positions in the workspace in which the robot can be situated without collisions. Edges between nodes signify that the robot can move freely between the nodes incident with the edge. There are many possible design policies for such a graph. For instance, it can be designed to only contain nodes that are part of shortest paths around the obstacles. Alternatively, nodes may be placed as far away as possible from obstacles to obtain paths with maximum clearance. It may be highly non-trivial to find appropriate collision-free positions in the workspace and to determine between which of those it is possible to move without collisions [KavrakiLaValle16, O'Rourke98]. To solve this, a concept common to many approaches is the construction of a map in which the obstacles in the environment are enlarged by a certain amount so that the remaining free space corresponds to collision-free positions. For instance, if the robot has a circular footprint, the obstacles are enlarged by at least the robot radius. A simple, practical example in that regard is illustrated in Excursus 1. In a more general setting, beyond the immediate needs of this thesis, the Minkowski addition is a formal concept for the addition of sets and therefore useful in obtaining and describing enlarged representations of obstacles in an exact fashion [O'Rourke98]. This is true in particular when the obstacles and the robot are represented as convex polytopes since the Minkowski addition of convex polytopes gives a convex polytope [Schneider93]. While the theory and challenges described so far are of relevance to much of robotics, some of the theoretical foundations most characteristic to cooperative distributed robotics have not yet been touched upon. In particular, these include how to organize and control the robots in a distributed fashion so that each robot makes decisions that serve the common goal.

Excursus 1: Connectivity Graph of a Polygonal Workspace Model

This excursus considers a mobile robot that moves in the plane and that is equipped with a light detection and ranging sensor, giving measurements of the distances to surrounding surfaces. The goal is to obtain a connectivity graph that can be used to calculate a shortest path from a starting to a goal position. Here, the first step is to construct a polygonal model of the obstacles based on a set of distance measurements. Following [EbelSharafian ArdakaniEberhard17b], obstacles may be represented by sets of rectangles as a special case of a convex polygon. To that end, the set of measured points is augmented by adding copies of the measured points further away in radial direction, assuming some minimal obstacle

thickness. The resulting set is partitioned into multiple subsets, e.g., if the difference between adjacent measurements' distance values exceeds a certain threshold, indicating that they might belong to two different obstacles. For each subset, a rectangle of minimal area containing all points of the subset is calculated. Enlarged versions of these rectangles are calculated to account for the finite robot radius. A possible result is illustrated on the right in the upper picture. The obstacles are drawn in dark gray, simulated sensor measurements are indicated by red crosses along the obstacle edges, with the resulting enlarged rectangles depicted in red. The encircled cross marks the sensor position. In the example, having included three further sets of measurements, a connectivity graph, designed to calculate shortest paths, is constructed. All rectangle vertices not contained in any other enlarged rectangle are added as nodes to the graph. Then, all possible edges between the nodes are added to the graph as long as they do not intersect any enlarged rectangle's interior. This is based on the observation that shortest paths around polytopes lead through their vertices. The resulting connectivity graph is illustrated in the lower image with the graph drawn in blue and the start and goal nodes in orange and green, respectively. The shortest path between them is highlighted with green lines. Evidently, without a priori information on the number of obstacles and on how they are measured, the resulting map's complexity is not upper-bounded. In contrast, for a map consisting of a fixed amount of volumetric cells, the complexity is upper-bounded, making it easier to guarantee real-time operation.



Chapter 3

Fundamentals from Distributed Control and Organization

Much of the perceived quality of a robotic system performing physical tasks is defined by how accurately it can execute the desired motions or exert the right levels of forces and, hence, by the automatic control methods employed to solve these tasks. Distributed robotics poses an additional challenge since the motions of multiple cooperating robots need to be coordinated. Beyond dynamic control, the effectiveness of a distributed robotic system can heavily depend on organizational considerations, e.g., how many robots to employ, or which robot to employ for which subtask, which is more intricate if these considerations have to be decided upon in a distributed fashion. Usually, compared to dynamic control tasks, organizational matters can be dealt with on a slower time-scale, often relying on a static or quasi-static problem description, without considering any kind of dynamic model. Fortunately, solution approaches to both areas share a common foundation, which is optimization. As it will become apparent in the problems studied in this thesis, formulating robotic tasks in terms of optimization problems can be very intuitive and introduces a natural separation between formulating and solving the problem.

The chapter starts by introducing the concepts from optimization most essential for the thesis. Furthermore, a concrete optimization method is introduced in the form of particle swarm optimization, which can be useful to solve very general organizational optimization problems with little assumptions on their structure and properties. The subsequent section introduces an optimization-based control approach popular in theory and application, which is model predictive control (MPC). Scientific development in the area has enabled the solution of the underlying problems in a distributed fashion. Therefore, subsequently, some perspective on the research field of distributed MPC (DMPC) is given, providing notions and categories to distinguish the various kinds of schemes that have been proposed in the theoretically oriented literature. As yet, DMPC seems to be more of a theory-driven research field, with applications in distributed robotics often employing other kinds of

control schemes. Therefore, Section 3.3 gives a brief introduction to a popular approach to distributed control based on results from algebraic graph theory, which lays the foundations for application-oriented comparisons between the two approaches, allowing to design an approach appropriate for this thesis' model task of cooperative object transportation in Chapters 4 and 5.

3.1 Optimization

A general formulation for a constrained optimization problem can be given in the form

$$\underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimize}} \quad c(\boldsymbol{z}) \tag{3.1}$$

$$\text{subject to} \quad h_m(\boldsymbol{z}) \leq 0,\ m \in \{1,\dots,n_\text{i}\}, \tag{3.2}$$

$$g_j(\boldsymbol{z}) = 0,\ j \in \{1,\dots,n_\text{e}\}. \tag{3.3}$$

Therein, $\boldsymbol{z}$ is the optimization variable, $c\colon \mathbb{R}^n \to \mathbb{R}$ is the cost function, and $h_m\colon \mathbb{R}^n \to \mathbb{R}$, $g_j\colon \mathbb{R}^n \to \mathbb{R}$ are the inequality and equality constraint functions, respectively. In total, there are $n_\text{i} \geq 0$ inequality and $n_\text{e} \geq 0$ equality constraints. The feasible set $\mathcal{F} = \{\boldsymbol{z} \in \mathbb{R}^n \mid h_m(\boldsymbol{z}) \leq 0,\ m \in \{1,\dots,n_\text{i}\},\ g_j(\boldsymbol{z}) = 0,\ j \in \{1,\dots,n_\text{e}\}\}$ is the set of values of the optimization variable satisfying the constraints. The problem is said to be infeasible if $\mathcal{F} = \emptyset$, whereas the case of unconstrained optimization is recovered if $\mathcal{F} = \mathbb{R}^n$. Often, all functions appearing in the optimization problem are smooth, opening up the problem to a wide field of theory and numerical solution algorithms that have been developed over the years, see, e.g., [NocedalWright06] for an encompassing treatise on the topic. Subsequently, quantities corresponding to optimal solutions of optimization problems will be marked by a star, with $\boldsymbol{z}^\star$ being an optimal solution and $c^\star = c(\boldsymbol{z}^\star)$ being the corresponding cost. Analytically, an equality constraint $\hat{g}(\boldsymbol{z}) = 0$ can be expressed equivalently by two inequality constraints $\hat{g}(\boldsymbol{z}) \leq 0$, $-\hat{g}(\boldsymbol{z}) \leq 0$, although numerical optimization algorithms tend to treat them differently [NocedalWright06]. If, for a $\hat{\boldsymbol{z}} \in \mathcal{F}$, an inequality constraint holds with equality, i.e., $h_m(\hat{\boldsymbol{z}}) = 0$, the constraint is said to be active at that point. An equality constraint is always active [BoydVandenberghe04]. The active set at a point $\hat{\boldsymbol{z}}$ is the set of constraints that are active at $\hat{\boldsymbol{z}}$, with an optimal active set being the active set at a solution of the optimization problem [NocedalWright06].

In general, little can be said about the solutions of problem (3.1)-(3.3); most importantly, even if the problem is feasible, it is not clear whether the problem has one or multiple globally optimal solutions. Furthermore, it may have local optima that are not global, with a point $\hat{\boldsymbol{z}}$ being locally optimal if it has lowest cost of all nearby feasible points, i.e., $\exists\, r > 0\colon c(\hat{\boldsymbol{z}}) \leq c(\boldsymbol{z})\ \forall \boldsymbol{z} \in \mathcal{F}\colon \|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2 \leq r$. Very useful first-order optimality conditions are the so-called Karush-Kuhn-Tucker (KKT) conditions [NocedalWright06, BoydVandenberghe04].

To formulate them, the Lagrangian function

$$L(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = c(\boldsymbol{z}) + \sum_{m=1}^{n_\mathrm{i}} \lambda_m \, h_m(\boldsymbol{z}) + \sum_{j=1}^{n_\mathrm{e}} \mu_j \, g_j(\boldsymbol{z}) \tag{3.4}$$

is a useful quantity [BoydVandenberghe04], with $\boldsymbol{\lambda} \in \mathbb{R}^{n_\mathrm{i}}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_\mathrm{e}}$ being the vectors of the so-called Lagrange multipliers associated with the inequality and equality constraints, respectively. Following [NocedalWright06], if the cost and constraint functions are continuously differentiable and if the gradients of the constraints in the active set at a locally optimal point $\hat{\boldsymbol{z}}$ are linearly independent, then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{\mu}}$ that fulfill the KKT conditions

$$\nabla_{\boldsymbol{z}} L\left(\hat{\boldsymbol{z}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}\right) = \boldsymbol{0}, \tag{3.5}$$

$$h_m(\hat{\boldsymbol{z}}) \leq 0 \quad \forall m \in \{1, \ldots, n_\mathrm{i}\}, \tag{3.6}$$

$$g_j(\hat{\boldsymbol{z}}) = 0 \quad \forall j \in \{1, \ldots, n_\mathrm{e}\}, \tag{3.7}$$

$$\hat{\lambda}_m \geq 0 \quad \forall m \in \{1, \ldots, n_\mathrm{i}\}, \tag{3.8}$$

$$\hat{\lambda}_m \, h_m(\hat{\boldsymbol{z}}) = 0 \quad \forall m \in \{1, \ldots, n_\mathrm{i}\}. \tag{3.9}$$

To determine whether there can be multiple local minima, convexity is decisive. The problem (3.1)-(3.3) is a convex optimization problem if the cost function c and the feasible set $\mathcal{F}$ are convex. In that case, every locally optimal solution is also globally optimal [NocedalWright06]. Furthermore, if the cost function is not only convex but strictly convex, there exists a unique globally optimal solution. Optimization problems can be written in different, equivalent ways, meaning that their solutions are identical, although cost and constraint functions are not. Convex optimization problems are easier to handle and interpret if the equality constraint functions are required to be affine, i.e., $g_j(\boldsymbol{z}) = \boldsymbol{a}_j^\mathsf{T} \boldsymbol{z} + b_j$. With this requirement and the inequality constraint functions h_m being convex, it becomes geometrically clear that the feasible set is convex since sublevel sets of convex functions are convex. Hence, convex optimization problems and convex programming are often directly defined this way [BoydVandenberghe04, NocedalWright06].

The solution effort can benefit significantly if the optimization problem has further structure. Of particular importance for the control of linear systems are so-called quadratic programs and, in particular, their convex variants. A quadratic program is an optimization problem with quadratic cost and affine constraint functions. A typical standard form, as it can be supplied to many specialized solvers, is of the form

$$\underset{\boldsymbol{z} \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \boldsymbol{z}^\mathsf{T} \boldsymbol{H} \boldsymbol{z} + \boldsymbol{f}^\mathsf{T} \boldsymbol{z} \tag{3.10}$$

$$\text{subject to} \quad \boldsymbol{C} \boldsymbol{z} \leq \boldsymbol{d}, \tag{3.11}$$

$$\boldsymbol{A} \boldsymbol{z} = \boldsymbol{b} \tag{3.12}$$

with $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ being symmetric, $\boldsymbol{C} \in \mathbb{R}^{n_\mathrm{i} \times n}$, $\boldsymbol{A} \in \mathbb{R}^{n_\mathrm{e} \times n}$, and the inequality in Equation (3.11) to be evaluated in an element-wise fashion. The feasible set is convex since it is a

convex polytope, which becomes evident when expressing each equality as two inequalities, giving the polytope in half-space representation, cf. Equation (2.22). The cost function is convex if $\boldsymbol{H}$ is positive semi-definite, i.e., $\boldsymbol{H} \geq 0$, and strictly convex if it is positive definite, i.e., $\boldsymbol{H} > 0$. For a convex quadratic program, it can be shown that the KKT conditions (3.5)-(3.9) are not only necessary but also sufficient [BoydVandenberghe04]. In some applications, it is necessary to solve multiple, related convex quadratic programs of the form (3.10)-(3.12), with certain quantities in the optimization problem depending on a vector of parameters $\boldsymbol{p} \in \mathbb{R}^{n_\mathrm{p}}$. For instance, this is the case for linear model predictive control with a quadratic cost function, as it will appear in the subsequent section. In this case, neglecting constant terms that do not change the optimal solution $\boldsymbol{z}^\star = \boldsymbol{z}^\star(\boldsymbol{p})$ of the problem, the optimization problem can be brought to the form

$$\underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\,\boldsymbol{z}^\mathsf{T}\boldsymbol{H}\boldsymbol{z} + \boldsymbol{p}^\mathsf{T}\boldsymbol{F}\boldsymbol{z} \tag{3.13}$$

$$\text{subject to} \quad \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{W} + \boldsymbol{E}\boldsymbol{p}, \tag{3.14}$$

which is referred to as a multi-parametric quadratic program in the literature [BemporadEtAl02]. Not only has the solution as a function of the parameters a special structure, but it is even possible to solve a problem of this form explicitly [TøndelJohansenBemporad03, FaíscaDuaPistikopoulos07]. The practical significance of this circumstance will become apparent later at the example of linear MPC.

While problems of the form (3.13)-(3.14) can be very useful for control purposes, general, organizational tasks to be solved for successful cooperative robotic behavior can be of a very different character. Formulating them as an optimization problem may lead to a non-convex problem, with potentially multiple local and global optima. The cost and constraint functions may be non-smooth or even discontinuous. In these cases, typical gradient-based optimization algorithms are not suitable, even if gradient information was available, since they converge to a local minimum determined by the starting point of the optimization. This is especially crucial to robotics since the behavior should be fully automated. Therefore, there is no human supervision that could provide insightful initial guesses, restarting and reparameterizing the optimization process until a satisfactory result has been achieved. Henceforth, global optimization strategies need to be used, which is facilitated by the fact that typical organizational tasks are not as time-critical as closed-loop control, which usually requires a guaranteed sampling time. An overview of global optimization techniques can be found in [Rao09], with the approaches often being based on heuristic ideas with, of course, there being no guarantee to find a global optimum.

For the further course of the thesis, it suffices to briefly introduce one approach to global optimization, which is to be used later. It is based on the particle swarm optimization (PSO) algorithm first introduced in [KennedyEberhart95], which is striking due to its simplicity. The algorithm is stochastic, works iteratively, and does not consider any constraints. It works with a set of candidate solutions of the optimization problem, which can be thought

of as a population of particles mimicking social behavior. Subsequently, it is assumed that a cost function $c\colon \mathbb{R}^n \to \mathbb{R}$ shall be minimized, and that there are n_{ps} particles, with their positions at iteration $k \in \mathbb{N}_0$ being denoted as $\tilde{\boldsymbol{z}}_i^{[k]} \in \mathbb{R}^n$, $i \in \{1, \ldots, n_{\mathrm{ps}}\}$. Following [ShiEberhart98], starting from arbitrary initial values $\tilde{\boldsymbol{z}}_i^{[0]}$, $\Delta\tilde{\boldsymbol{z}}_i^{[0]}$ for each particle, the particles' positions are updated according to the iteration rule

$$\Delta\tilde{\boldsymbol{z}}_i^{[k]} = w_1^{\mathrm{p}}\,\Delta\tilde{\boldsymbol{z}}_i^{[k-1]} + w_2^{\mathrm{p}}\,r_{1,i}^{[k]}\,(\tilde{\boldsymbol{z}}_i^{\mathrm{best},[k]} - \tilde{\boldsymbol{z}}_i^{[k]}) + w_3^{\mathrm{p}}\,r_{2,i}^{[k]}\,(\tilde{\boldsymbol{z}}_{\mathrm{swarm}}^{\mathrm{best},[k]} - \tilde{\boldsymbol{z}}_i^{[k]}), \tag{3.15}$$

$$\tilde{\boldsymbol{z}}_i^{[k+1]} = \tilde{\boldsymbol{z}}_i^{[k]} + \Delta\tilde{\boldsymbol{z}}_i^{[k]}. \tag{3.16}$$

Therein, $\tilde{\boldsymbol{z}}_i^{\mathrm{best},[k]}$ and $\tilde{\boldsymbol{z}}_{\mathrm{swarm}}^{\mathrm{best},[k]}$ are the best positions, i.e., with lowest cost, attained by particle i and by the whole swarm so far, respectively. The stochastic nature of PSO becomes manifest through the fact that $r_{1,i}^{[k]}$ and $r_{2,i}^{[k]}$ are each drawn from a uniform distribution on the interval $[0, 1]$. The three terms in Equation (3.15) are designed to mimic inertia, cognitive, and social behavior, respectively [ShiEberhart98]. The algorithm can be tuned to a specific problem through the positive parameters w_1^{p}, w_2^{p}, and w_3^{p}. It is convergent if the condition $\frac{1}{2}\,(w_2^{\mathrm{p}} + w_3^{\mathrm{p}}) - 1 < w_1^{\mathrm{p}} < 1$ holds [van den BerghEngelbrecht06]. The parameterization of the optimization algorithm can be made easier by scaling the optimization variable so that the range of reasonable values of the optimization variable mimics a hypercube as closely as possible. In this context, it can make sense to saturate $\Delta\tilde{\boldsymbol{z}}_i^{[k]}$ to not exceed the edge length of the hypercube best representing the range of the optimization variable. Another popular adaptation of the algorithm is to add a purely stochastic term to the right-hand side of Equation (3.15) to encourage further exploration even if the problem has converged [SedlaczekEberhard06]. Such a term is usually referred to as craziness in the literature. Furthermore, it can make sense to use a different parameterization for the update of the best particle's position than for the rest of the particles. Being a reimplementation of the algorithm implemented by [SedlaczekEberhard06], these extensions are also implemented and used in the PSO algorithm employed in this thesis. Beyond these, there exist many more extensions in the literature that shall not be covered here [PoliKennedyBlackwell07]. A more crucial shortcoming of the algorithm is that it does not explicitly consider constraints. Fortunately, with the augmented Lagrangian method [NocedalWright06], a strategy to solve constrained optimization problems by solving a sequence of unconstrained problems with a modified cost function has been shown to work well with PSO [SedlaczekEberhard06]. The modified cost function, called the augmented Lagrangian function, is based on the Lagrangian function but adds quadratic penalty terms, which are scaled by scalar penalty factors. The penalty is designed to ensure that a strict local minimum of the original constrained optimization problem is also a strict local minimum of the augmented Lagrangian function if the optimal Lagrange multipliers and large enough penalty factors are inserted [NocedalWright06]. Since neither the optimal Lagrange multipliers nor an appropriate choice of the penalty factors are known a priori, the factors and estimates of the multipliers are modified iteratively based on the constraint violations observed from the solution of the unconstrained problem, which uses the aug-

mented Lagrangian function as the cost function. Following [SedlaczekEberhard06], the method is henceforward referred to as augmented Lagrangian particle swarm optimization. Practical details will become apparent in Chapter 5, where this type of algorithm is put to work in a distributed fashion. However, before that, it is time to examine how to leverage optimization for control purposes.

3.2 Model Predictive Control

While this thesis makes use of predictive control in its distributed form, it is instructive first to take a look at MPC in its centralized variant to obtain an understanding of its fundamental principles of operation. This is especially true since the optimization problems solved in a distributed MPC controller are structurally very similar to those from centralized MPC, with DMPC theory being an extension of centralized MPC theory. Although the discussion of distributed MPC later in the thesis will focus on linear systems, it makes sense to introduce some of the theoretical basics of MPC for nonlinear systems since it absorbs the linear case without the fundamentals being more arduous to explain than for the linear case. The discussion will also make clear the main advantages of MPC.

3.2.1 The Concept of Model Predictive Control

It is the basic idea of model predictive control to optimize in each time step a cost function over a finite prediction horizon, using a model of the control system's dynamics to predict its behavior over the horizon [RawlingsMayneDiehl17, Maciejowski01]. A central conceptual advantage is that constraints on the control inputs and states can be considered explicitly in the optimization problem. Therefore, if necessary for optimal performance, the controller will utilize the system's capabilities to the fullest extent, potentially and reliably operating at the boundaries of the admissible state and input sets. Having obtained an optimal sequence of inputs as the optimization problem's solution, only the first part of that sequence is applied to the system and the optimization problem is solved anew in the next time step. Therefore, the prediction horizon recedes in time with each time step that passes, which is why the approach is also called receding horizon control [MayneMichalska90, PrimbsNevistić00]. The control input is usually kept constant during a single sampling interval, which is also the case in this thesis. MPC schemes can be formulated for continuous-time as well as for discrete-time systems, with this thesis focusing on discrete-time schemes. Naturally, discrete-time schemes can be applied to continuous-time systems if the model is discretized beforehand, which can be done in an exact fashion for linear systems. The predictive, receding-horizon nature of MPC necessitates some additional notational conventions, which, subsequently, will be introduced for the discrete-time case only. The notation $(\cdot \,|\, t)$ shall denote trajectories predicted at time

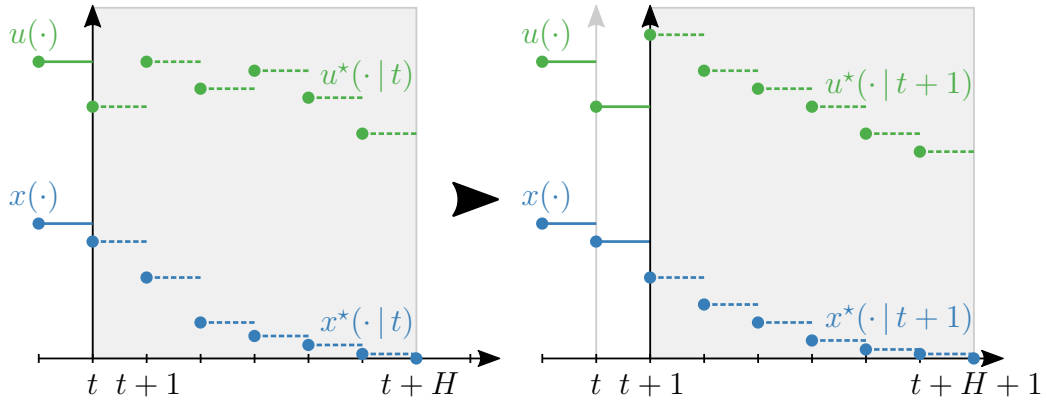


Figure 3.1: Illustration of the receding horizon strategy for a discrete-time system. The current prediction horizon is shaded in gray.

step t over the prediction horizon of length $H \in \mathbb{N}$, $H > 1$. For instance, $\boldsymbol{x}(k\,|\,t) \in \mathbb{R}^{n_x}$, $k \in \{t, \dots, t+H\}$, denotes the state sequence predicted based on the observed or measured real system state $\boldsymbol{x}(t)$ and the candidate input sequence $\boldsymbol{u}(k\,|\,t)$, $k \in \{t, \dots, t+H-1\}$. The optimal input sequence solving the MPC optimization problem at time step t shall be denoted as $\boldsymbol{u}^\star(k\,|\,t)$, $k \in \{t, \dots, t+H-1\}$, of which $\boldsymbol{u}(t) = \boldsymbol{u}^\star(t\,|\,t)$ is applied to the system. Similarly, $\boldsymbol{x}^\star(k\,|\,t)$, $k \in \{t, \dots, t+H\}$, shall denote the corresponding optimal state sequence which is predicted using the optimal input sequence. Employing this notation, the receding horizon strategy is illustrated in Figure 3.1.

MPC has been used extensively in industrial applications early on, although mostly in the process industries [QinBadgwell03], which is probably because systems in this area tend to have slower dynamics, allowing the timely calculation of the control inputs even without today's refined algorithms and computation power. Nowadays, the underlying theory is well-developed. Specialized optimization approaches and high-performance implementations allow today the successful, practical control of systems with fast dynamics also in the nonlinear case [HouskaFerreauDiehl11a, KäpernickGraichen14, AnderssonEtAl19]. A generic, nonlinear discrete-time MPC optimization problem for asymptotically stabilizing a steady state may be given in the form

$$\underset{\boldsymbol{u}(\cdot\,|\,t)}{\text{minimize}} \quad \left(J(\boldsymbol{x}(t), \boldsymbol{u}(\cdot\,|\,t)) = \sum_{k=t}^{t+H-1} \ell(\boldsymbol{x}(k\,|\,t),\, \boldsymbol{u}(k\,|\,t)) + J_\mathrm{f}(\boldsymbol{x}(t+H\,|\,t)) \right) \tag{3.17}$$

$$\text{subject to} \quad \boldsymbol{x}(k+1\,|\,t) = \boldsymbol{f}(\boldsymbol{x}(k\,|\,t), \boldsymbol{u}(k\,|\,t))\,, \tag{3.18}$$

$$\boldsymbol{u}(k\,|\,t) \in \mathcal{U}, \tag{3.19}$$

$$\boldsymbol{x}(k\,|\,t) \in \mathcal{X}, \quad k \in \{t, \dots, t+H-1\}, \tag{3.20}$$

$$\boldsymbol{x}(t+H\,|\,t) \in \mathcal{X}_\mathrm{f}, \tag{3.21}$$

$$\boldsymbol{x}(t\,|\,t) = \boldsymbol{x}(t). \tag{3.22}$$

Therein, the nonlinear, discrete-time dynamic model is given by (3.18). Due to the

constraints (3.19)-(3.21), the inputs, states, and the terminal state within the horizon are constrained to the closed sets $\mathcal{U}$, $\mathcal{X}$, and the closed terminal set $\mathcal{X}_\mathrm{f} \subset \mathcal{X}$, respectively. In the cost function (3.17), the term $\ell(\boldsymbol{x}(k \mid t), \boldsymbol{u}(k \mid t))$, that is summed up over the horizon, is called the stage cost, whereas $J_\mathrm{f}(\boldsymbol{x}(t+H \mid t))$ is called the terminal cost since it penalizes the terminal state. To have the chance of obtaining a functioning controller, e.g., for stabilizing a steady state, some fundamental design properties must be met. In general, the steady state to be stabilized is assumed to be contained in the constraint sets and the stage cost $\ell \colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is usually continuous, positive definite with respect to the steady state, and radially unbounded, i.e., increasing in an unbounded fashion as the control error approaches infinity. Mostly, the stage cost is designed to have a single, global minimum, located in the steady state, which is also assumed subsequently. Since, in general, the MPC control law is not explicitly known, but merely implicitly given as the solution of an optimization problem, there is usually no closed-form expression available that describes the closed loop. Therefore, proving the nominal stability of a closed-loop MPC scheme can be more complicated than for other state-feedback controllers that are available as a closed-form expression. In theory-driven approaches, the terminal cost J_f and the terminal set $\mathcal{X}_\mathrm{f}$ are chosen in specific, interdependent ways to help guarantee closed-loop stability [MayneEtAl00, ChenAllgöwer98]. This usually means that the remainder of the optimization problem can be tuned relatively freely to the needs of the application, including, e.g., weighting factors in the stage cost as well as the extents of the constraint sets $\mathcal{X}$ and $\mathcal{U}$. Hence, with an MPC optimization problem designed in this manner, there is a certain level of decoupling between ascertaining formal properties like stability and tuning the performance, which is in stark contrast to many simpler, classic control approaches like PID control. In addition, weights in the cost function and constraints on the inputs and states are very intuitive in their meaning, so that it is often rather clear which tuning parameter to modify in order to meet a specific control goal. Furthermore, the approach works naturally with multiple-input multiple-output (MIMO) systems. In contrast, classic schemes like PID control are usually restricted to single-input single-output (SISO) systems. Therefore, real-world implementations of classic control methods for MIMO systems often rely on the application of multiple, separate controllers in heuristic, complicated control loop designs that are hard to maintain. Hence, apart from its conceptual advantages concerning constraint handling, the motivation to use an MPC controller may often enough not be seen in performance improvements, but rather in its practical advantages with regard to tuning and a simplified control structure.

Without delving into too much technical detail, there are multiple techniques to guarantee the nominal asymptotic stability of the closed loop for the problem above. These are all based on Lyapunov's stability theorem [Khalil02], which, essentially, consists of finding a function that is decreasing along all trajectories of the closed loop. Such a so-called Lyapunov function can be seen as a generalization of the concept of energy from mechanics. If the sum of potential and kinetic energy is decreasing along all trajectories of a mechanical

system – as it is the case for an autonomous, damped mechanical system – it will eventually come to a standstill, i.e., the system will have converged to a steady state. It is also clear that, in this case, the system on its own cannot reach states of higher energy, which corresponds to the notion of stability. For a time-invariant, continuous-time system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ with $\boldsymbol{f}\colon \mathcal{D} \subset \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{x}}$, $\boldsymbol{f}(\boldsymbol{0}) = \boldsymbol{0}$, and $\boldsymbol{x} = \boldsymbol{0} \in \mathcal{D}$ being the steady-state to be inspected, a function $V\colon \mathcal{D} \to \mathbb{R}$ is a Lyapunov function if it is continuously differentiable and if

$$V(\boldsymbol{0}) = 0, \qquad V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathcal{D} \setminus \{0\}, \tag{3.23}$$

$$\dot{V}(\boldsymbol{x}) \coloneqq \frac{\partial V}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \leq 0 \quad \forall \boldsymbol{x} \in \mathcal{D}. \tag{3.24}$$

The latter means that the derivative of V along the solution trajectories of the system is non-positive on $\mathcal{D}$. If such a Lyapunov function exists, $\boldsymbol{x} = \boldsymbol{0}$ is a stable equilibrium, with it being asymptotically stable if $\dot{V}(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{0}\}$, see [Khalil02]. A system of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ may appear as the closed loop of a system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ controlled by a state-feedback law $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$, so the theory is useful both for the stability analysis of autonomous systems as well as for the design of feedback control. As such, it will also reappear in Section 3.3, which introduces graph-algebraic control. Proofs of discrete-time MPC schemes, however, usually rely on discrete-time versions of these relations. Although introductions to Lyapunov theory for continuous-time systems are more common [Khalil02, KalmanBertram60a], a concise treatment of the discrete-time case is available in [KalmanBertram60b]. In brief, for a discrete-time system $\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t))$, $t \in \mathbb{N}_0$, $\boldsymbol{f}\colon \mathcal{D} \subset \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{x}}$ with $\boldsymbol{0} \in \mathcal{D}$ and $\boldsymbol{f}(\boldsymbol{0}) = \boldsymbol{0}$, a function $V\colon \mathcal{D} \to \mathbb{R}$ is a Lyapunov function if

$$V(\boldsymbol{0}) = 0, \qquad V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathcal{D} \setminus \{0\}, \tag{3.25}$$

$$\Delta V(\boldsymbol{x}) \coloneqq V(\boldsymbol{f}(\boldsymbol{x})) - V(\boldsymbol{x}) \leq 0 \quad \forall \boldsymbol{x} \in \mathcal{D}. \tag{3.26}$$

Once again, the existence of such a function implies stability of $\boldsymbol{x} = \boldsymbol{0}$, with asymptotic stability being obtained if V is continuous and $\Delta V(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{0}\}$. In both the continuous-time and discrete-time cases, asymptotic stability holds globally if V is additionally radially unbounded, i.e., if $V(\boldsymbol{x}) \to \infty$ as $\|\boldsymbol{x}\| \to \infty$ for some norm $\|\cdot\|$.

In the case of MPC, the foundational idea is to use the minimized cost function $J^\star(\boldsymbol{x}(t)) \coloneqq J(\boldsymbol{x}(t), \boldsymbol{u}^\star(\cdot \,|\, t))$ as a Lyapunov function candidate. In general, $J^\star(\boldsymbol{x}(t))$ is not directly available as a function of $\boldsymbol{x}(t)$, because the optimal solution of the MPC problem is usually not known explicitly. Nevertheless, following [MayneEtAl00], by wisely choosing $\mathcal{X}_\mathrm{f}$ and J_f, it may be shown that

$$J^\star(\boldsymbol{x}(t+1)) = J^\star(\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}^\star(t \,|\, t))) \leq J^\star(\boldsymbol{x}(t)) - \ell(\boldsymbol{x}(t), \boldsymbol{u}^\star(t \,|\, t)). \tag{3.27}$$

Due to the usual design of $\ell(\cdot, \cdot)$, this implies that, indeed, the Lyapunov function candidate is decreasing outside of the steady state that shall be stabilized asymptotically. The key

to show that Equation (3.27) holds is to construct a feasible candidate solution from the optimal solution of the previous time step. By optimality of the unknown optimal solution of the new time step, this candidate solution provides an upper bound on the optimal cost of the new time step, which can be reformulated and upper-bounded further to lead to the right-hand side of Equation (3.27). Theoretical literature, see, e.g., [MayneEtAl00], proposes several ways to facilitate the construction of a candidate solution and upper-bounding the corresponding cost quantities to arrive at Equation (3.27). Depending on the approach taken, it can be useful to design the terminal set $\mathcal{X}_\mathrm{f}$ as a positively invariant set with respect to the loop closed by a stabilizing terminal controller that satisfies the input constraints within $\mathcal{X}_\mathrm{f}$. In this case, apart from minor technical assumptions, showing asymptotic closed-loop stability comes down to checking that a joint condition on the terminal cost, the stage cost, and the terminal controller holds within the terminal set. This condition implies that the terminal cost is a local Lyapunov function within the terminal set for the loop closed by the terminal controller [MayneEtAl00]. Due to the applications appearing later on, the subsequent delineations will focus mostly on the linear case. Therefore, the discussion of concrete, stabilizing design choices is deferred to the linear case discussed below. In any case, it is important to ensure that the optimization problem is recursively feasible. This means that if the optimization problem is feasible initially, it remains feasible in subsequent time steps assuming that the system to be controlled behaves like the nominal system.

While guarantees on the nominal behavior are certainly a desirable feature of a controller, it is important to note that they do not guarantee the functioning of the controller in a real-world system. Hence, even with formal, nominal guarantees, extensive testing of the real-world closed-loop system must be performed to ensure its safety. This is due to many factors. Firstly, no model is perfect, and therefore, a real-world system will never behave exactly like its nominal model. Secondly, systems in the real world are subject to exogenous disturbances. Not only can they deteriorate performance, but they can even lead to infeasibility of the problem if a constraint, differently than previously planned, cannot be satisfied anymore due to a disturbance. Thirdly, optimization algorithms may not be perfect; their solutions are usually only accurate up to a finite residuum, and, in limited calculation time, they might not find an optimal solution. There exist techniques to mitigate some of these issues, e.g., suboptimal MPC [ScokaertMayneRawlings99] and robust MPC techniques that can deal with disturbances with known bounds, see, e.g., [MayneSeronRakovi\'c05, Rakovi\'c12, LimonEtAl10]. However, robust MPC techniques tend to make the optimization problem more complicated and hence harder to solve accurately and robustly in limited time. In fact, in many practical applications, control designers refrain from using additional constraints to obtain guarantees on nominal closed-loop properties because their design procedures can become arduous and they complicate the solution of the resulting optimization problem [HouskaFerreauDiehl11b, BächleHentzeltGraichen13]. This holds true especially if the optimization problem devoid of additional stabilizing

constraints does not have any state constraints whatsoever, since input constraints may be dealt with more efficiently. Still, working as close as possible to these formal techniques, and, in particular, understanding how they establish the guarantees, can help greatly to design MPC controllers that work well in real-world applications. Furthermore, there are formal proof techniques to show the stability of the MPC closed loop without any additional stabilizing constraints, with the decisive factor being to choose the horizon long enough [BocciaGrüneWorthmann14, LimonEtAl06]. This is also a possible theoretical explanation why schemes without such constraints often function very well in practice, even without having proven their formal guarantees.

In the important linear case, the cost function is usually chosen to be quadratic and the constraint sets are assumed to be convex and polytopic. For the simple task of asymptotically stabilizing a steady state at $\boldsymbol{x} = \boldsymbol{u} = \boldsymbol{0}$, the optimization problem may be given in the form

$$\begin{aligned}
\underset{\boldsymbol{u}(\cdot\,|\,t)}{\text{minimize}} \quad & \sum_{k=t}^{t+H-1} \|\boldsymbol{x}(k\,|\,t)\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}(k\,|\,t)\|_{\boldsymbol{R}}^2 + \|\boldsymbol{x}(t+H\,|\,t)\|_{\boldsymbol{P}}^2 && (3.28)\\
\text{subject to} \quad & \boldsymbol{x}(k+1\,|\,t) = \boldsymbol{A}\boldsymbol{x}(k\,|\,t) + \boldsymbol{B}\boldsymbol{u}(k\,|\,t), && (3.29)\\
& \boldsymbol{C}_u\,\boldsymbol{u}(k\,|\,t) \leq \boldsymbol{d}_u, && (3.30)\\
& \boldsymbol{C}_x\,\boldsymbol{x}(k\,|\,t) \leq \boldsymbol{d}_x, \quad k \in \{t, \dots, t+H-1\}, && (3.31)\\
& \boldsymbol{C}_{\mathrm{f}}\,\boldsymbol{x}(t+H\,|\,t) \leq \boldsymbol{d}_{\mathrm{f}}, && (3.32)\\
& \boldsymbol{x}(t\,|\,t) = \boldsymbol{x}(t). && (3.33)
\end{aligned}$$

In the cost function, squared weighted norms are employed, which are defined as $\|\boldsymbol{v}\|_{\boldsymbol{V}}^2 := \boldsymbol{v}^{\mathsf{T}}\boldsymbol{V}\boldsymbol{v}$ for a vector $\boldsymbol{v}$ and a symmetric, positive semi-definite matrix $\boldsymbol{V}$ of fitting dimensions. The symmetric weighting matrices $\boldsymbol{Q}$, $\boldsymbol{R}$, and, if non-zero, $\boldsymbol{P}$ are selected to be positive definite. Furthermore, it is assumed subsequently that $(\boldsymbol{A}, \boldsymbol{B})$ is controllable.

For a modification of the above problem with an infinite horizon $H = \infty$ and without any constraints or terminal cost, there exists an explicit solution $\boldsymbol{u}(t) = \bar{\boldsymbol{K}}\boldsymbol{x}$ with the gain matrix taking the form $\bar{\boldsymbol{K}} = -(\boldsymbol{R}+\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{S}\boldsymbol{A}$. This controller is called the (infinite-horizon) linear quadratic regulator (LQR), which can be obtained from the positive-definite solution $\boldsymbol{S}$ of the discrete-time algebraic Riccati equation (DARE) [DoratoLevis71, Laub79]. Henceforth, linear MPC can be seen as a finite-horizon approximation of the constrained LQR problem [ScokaertRawlings98] and therefore as a reaction to the hardship of obtaining a constrained LQR in closed form. For the MPC problem, it can be shown that the sufficient asymptotic stability requirements described above can be met by choosing the terminal cost weight $\boldsymbol{P} = \boldsymbol{S}$ as the solution of the DARE, and the terminal set as the maximal positively invariant set defined by

$$\mathcal{X}_{\mathrm{f}} = \left\{ \boldsymbol{x} \in \mathbb{R}^{n_{\mathrm{x}}} \mid \boldsymbol{C}_x\boldsymbol{x} \leq \boldsymbol{d}_x, \quad \boldsymbol{C}_u\bar{\boldsymbol{K}}\boldsymbol{x} \leq \boldsymbol{d}_u, \quad (\boldsymbol{A}+\boldsymbol{B}\bar{\boldsymbol{K}})\,\boldsymbol{x} \in \mathcal{X}_{\mathrm{f}} \right\}. \tag{3.34}$$

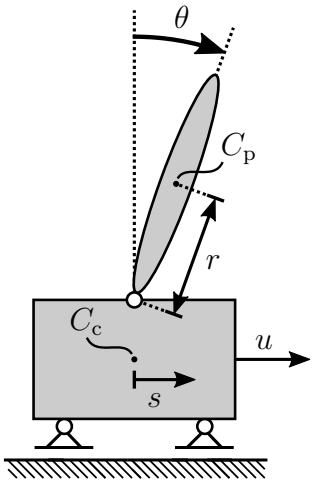
This allows to use the LQR as a terminal controller to obtain candidate inputs within the

terminal set. Then, within the terminal set, the terminal cost is a local Lyapunov function for the loop closed by the LQR [MayneEtAl00]. Under specific circumstances, the above choice of the terminal cost gives a direct relationship of the finite-horizon MPC problem to its infinite-horizon counterpart. Indeed, it can be shown that the infinite-horizon, unconstrained LQR cost of regulating any state $\bar{\boldsymbol{x}}$ into the origin is $\bar{\boldsymbol{x}}^\mathsf{T}\boldsymbol{P}\bar{\boldsymbol{x}}$ [KwakernaakSivan72], which is precisely the terminal cost of the MPC problem for $\bar{\boldsymbol{x}} = \boldsymbol{x}(t+H\,|\,t)$. Hence, if the horizon is chosen long enough so that, without actually enforcing the terminal constraint, the optimal terminal state lies within the terminal set anyway, the optimal cost of the MPC problem is exactly equal to the optimal cost of the corresponding infinite horizon problem [ScokaertRawlings98].

An algorithm to calculate the invariant set from Equation (3.34) may be found in [GilbertTan91]. However, in some circumstances, as they will appear later in the thesis, it may not be desirable to perform the potentially extensive calculation of the invariant terminal set and terminal cost. In that case, a simpler, valid choice of terminal set and terminal controller can be useful, namely $\mathcal{X}_\mathrm{f} = \{\mathbf{0}\}$ and the zero input as the terminal controller. Subsequently, this choice of terminal constraint is referred to as a zero terminal constraint, or, more general in case of non-zero setpoints, as a terminal equality constraint. Naturally, the controller will only function starting from those states from which the state $\boldsymbol{x} = \mathbf{0}$ can actually be reached within the prediction horizon. Compared to a controller with a larger terminal set, this may limit the applicability of the controller. However, depending on the desired operating region of the controller, this may not always pose a problem. Example 3 gives a practical example for a linear MPC controller designed with stabilizing design ingredients in the form of a terminal set and terminal cost, showing that a terminal set may result in many additional state constraints already for a system of moderate size.

Example 3: Linear MPC for a Pendulum on a Cart

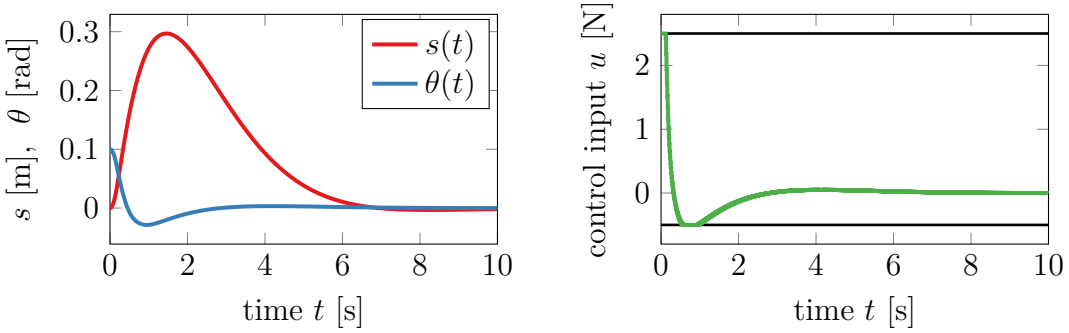
To give practical insight on the design process of an MPC controller, this example considers an inverted pendulum on a cart as it is illustrated on the right-hand side. The system's planar motions are described using the generalized coordinates $\boldsymbol{q} = \begin{bmatrix} s & \theta \end{bmatrix}^\mathsf{T}$. The controller shall govern the force u so that the otherwise unstable equilibrium $\boldsymbol{q} = \dot{\boldsymbol{q}} = \mathbf{0}$ is stabilized. By nature, the system's dynamics is nonlinear. Nevertheless, as will be seen, to merely stabilize the system for small deviations from the setpoint, a linear controller suffices. The cart, with the center of mass C_c, is of mass $m_\mathrm{c} = 0.5\,\mathrm{kg}$, whereas the pendulum is of mass $m_\mathrm{p} = 1\,\mathrm{kg}$ with $r =$



0.5 m and its relevant principal moment of inertia about its center of mass C_p being $J = 0.1\,\mathrm{kg\,m^2}$. Following Section 2.1, this leads to the equations of motion

$$\begin{bmatrix} m_\mathrm{c} + m_\mathrm{p} & m_\mathrm{p}\, r \cos(\theta) \\ m_\mathrm{p}\, r \cos(\theta) & J + m_\mathrm{p}\, r^2 \end{bmatrix} \begin{bmatrix} \ddot{s} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} -m_\mathrm{p}\, r\, \dot{\theta}^2 \sin(\theta) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ m_\mathrm{p}\, g\, r \sin(\theta) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u,$$

which follows the template given by Equation (2.8). To obtain a model that can be used within the linear MPC controller, the equations of motion are linearized around the operating point $\boldsymbol{q}_\mathrm{o} = \dot{\boldsymbol{q}}_\mathrm{o} = \boldsymbol{0}$ and brought to state-space form, see Equations (2.9)-(2.15). Afterwards, it is discretized using a zero-order hold with the sampling time $t_\mathrm{s} = 0.01\,\mathrm{s}$. The MPC problem is formulated according to Equations (3.28)-(3.33) with $\mathcal{X} = \mathbb{R}^4$, $\mathcal{U} = \{u \in \mathbb{R} \mid -0.5\,\mathrm{N} \leq u \leq 2.5\,\mathrm{N}\}$, $H = 35$, $\boldsymbol{Q} = \boldsymbol{I}$, and $R = 1$. The terminal cost weight $\boldsymbol{P}$ is chosen as the solution of the DARE corresponding to the infinite-horizon LQR, with the terminal set as per Equation (3.34) and calculated using the algorithm from [GilbertTan91]. The resulting convex, polytopic terminal set consists of 720 individual constraints. The plots below illustrate the controller's behavior in a simulation of the nonlinear system, with $\theta(0) = 0.1\,\mathrm{rad}$, $s(0) = \dot{s}(0) = \dot{\theta}(0) = 0$. Indeed, the controller manages to stabilize the system, reliably operating the system at the input bounds when necessary. It is worth noting that the corresponding controller with the simpler zero terminal constraint is not initially feasible for the chosen initial condition. It merely starts to be feasible at initial angles of around $\theta(0) \approx 6 \cdot 10^{-4}\,\mathrm{rad}$, highlighting that not all kinds of stabilizing design ingredients are practical for all kinds of systems.



One of the key disadvantages of MPC is that the controller is rather hard to grasp, since its underlying feedback law is not available in closed form. However, for linear MPC as introduced above, it is even possible to calculate an explicit solution of the optimization problem [BemporadEtAl02]. Recognizing that the system dynamics (3.29) can be recursively inserted into the cost function (3.28) and the state constraints (3.31), (3.32) allows to bring the problem to the form of a multi-parametric quadratic program as introduced in Equations (3.13)-(3.14). In the problem given above, the set of parameters, on which the optimization problem's solution depends, consists only of the measured

state $\boldsymbol{x}(t)$. In more general problems, additional parameters may appear, e.g., a reference value in the case of a tracking controller [LimonEtAl08]. The explicit solution can be obtained by exploiting the Karush-Kuhn-Tucker conditions (3.5)-(3.9). This leads to an affine control law for each valid combination of active constraints. Consequently, together with the number of possible combinations of active constraints, also the controller's complexity may grow exponentially with the number of constraints [BemporadEtAl02]. Henceforth, although the calculation effort to explicitly calculate the controller may be prohibitive and even become intractable for larger problems, these considerations allow to draw valuable conclusions on the solution's character and properties. Indeed, not only is the overall solution a continuous, piecewise-affine function of the parameters, with the optimal cost being piecewise quadratic, but it can also be shown that the set of parameters can be partitioned into convex polytopes with one affine control law being optimal in each polytope [BemporadEtAl02]. This solution structure can be used to help greatly with the repeated online solution of linear MPC problems [FerreauEtAl14, MitzeMönnigmann20]. Indeed, all quadratic programs occurring later in this thesis are solved with the open-source software qpOASES, which takes advantage of the problem structure to solve subsequent, related multi-parametric quadratic programs more swiftly [FerreauEtAl14]. Furthermore, as shown in [MönnigmannOttenJost15], the insight on the solution of linear MPC problems may even be useful to get a better grasp on nonlinear MPC control laws.

To conclude this brief overview of model predictive control fundamentals, it can only be stressed that the MPC research field is very active, with many specialized variants, extensions, and combinations thereof having been researched. Naturally, the area crucial to this thesis is distributed MPC, which will be treated in greater detail subsequently. Beyond that, apart from the already briefly mentioned robust MPC and tracking control variants, there also exist economic MPC schemes [EllisDurandChristofides14] aiming at using more general cost functions, e.g., to obtain an economic behavior of a system which need not consist of asymptotically stabilizing a fixed, known setpoint. Furthermore, a field that has become very active in recent years is learning-based MPC. There, the goal is to integrate models or sub-models that are inferred from data and not from first principles, see, e.g., [AswaniEtAl13, LimonCalliessMaciejowski17, HewingKabzanZeilinger19], with research striving to maintain guarantees on the closed-loop behavior [BerberichEtAl21]. Learning techniques may also be leveraged to learn an approximation of an MPC controller, ideally drastically reducing the calculation time while keeping the controller's favorable properties [HertneckEtAl18].

3.2.2 Distributed MPC

Distributed MPC can be seen as a reaction to two demands. Firstly, for some systems, such as a group of cooperating robots, the existing system structure naturally suggests that each subsystem makes its own decisions, albeit in coordination with the other systems. Closely

related to this is the hope that the absence of a central control unit, such as a leading robot, will increase the reliability, robustness, and, in particular, expandability of the system. Secondly, for general classes of systems, solving one central, large optimization problem may become computationally intractable within the system's sampling time requirements. Therefore, solving multiple, smaller subproblems on different processing units could make the model predictive control of large-scale systems meet challenging real-time requirements. However, different from the multi-robot case, a general system's innate structure may not suggest a natural decomposition into subsystems, making the decomposition a part of the control design process. Due to these different motivations and various finer-grained nuances of the control problem to be solved, there is a wide and growing range of different distributed MPC schemes [ChristofidesEtAl13, NegenbornMaestre14]. Hence, this section aims to give some key criteria to differentiate the plethora of theory-driven approaches, aiming to deduce which types of schemes are particularly useful for the cooperative task solution with otherwise dynamically independent mobile robots.

The technologically simplest way to decentralize the model predictive control of a large-scale system is to cut the system into multiple subsystems, neglecting any couplings between them, and formulating a controller for each subsystem that only operates on information from that single subsystem. One may then seek conditions on the neglected couplings and potentially appropriate modifications of the optimization problems to still be able to guarantee a satisfactory closed-loop behavior of the whole system. Naturally, this concept is not restricted to MPC, with work on it dating back many decades, see, e.g., [WangDavison73]. Many authors refer to this kind of control as decentralized control, also within the area of MPC [Scattolini09, ChristofidesEtAl13]. However, this term is not particularly useful to distinguish this kind of control from distributed MPC techniques that also do not rely on any centralized entity despite not neglecting couplings. To that end, the denomination as decoupled control seems more prudent. In decoupled MPC strategies, the couplings may be treated as disturbances on the individual systems, with robustness techniques being used to obtain closed-loop guarantees [MagniScattolini06, RaimondoMagniScattolini07]. This approach is more suited to weak dynamic couplings. Hence, highly cooperative tasks in distributed robotics, e.g., described by a coupled cost function, need distributed schemes that utilize shared or communicated information.

Indeed, it is important to distinguish where exactly couplings enter the optimization problem. When formulating a specific control task as an MPC problem, couplings may appear in the dynamics, constraints, and cost. Coupled constraints may appear if limited resources are used jointly by the subsystems, whereas a coupled cost can be used to encode a cooperative control goal even of dynamically decoupled systems. Usually, schemes can only deal with a subset of these types of couplings. For instance, the scheme from [MüllerRebleAllgöwer12] allows for couplings in state and input constraints as well as in the cost function, whereas the work from [RichardsHow07] considers couplings in the constraints only. However, both are limited to dynamically decoupled systems. In contrast,

the schemes from [StewartEtAl10, StewartWrightRawlings11] can deal with couplings in the dynamics, but generally not with coupled constraints, although [StewartEtAl10] contains an extension to treat coupled input constraints. The couplings in the cost function can be distinguished further. In the most general case, the overall cost function is inseparable with regard to the contributions of the individual subsystems, not presuming any special additive structure. However, if the cost function has a special structure, a wider range of schemes becomes applicable. For instance, the scheme in [StewartEtAl10] assumes that the overall cost is a weighted sum of subsystem cost functions, with the subsystem cost functions only being coupled implicitly through the dynamics. In [MüllerRebleAllgöwer12], the authors also use a separability property to simplify their proposed control scheme significantly. In general, schemes in which a controller takes into account the effects of its control actions on the cost functions of other controllers are called cooperative schemes. Depending on the coupling structure of the problem to be solved, one subsystem's control problem may not depend on information from every other subsystem. Subsystems whose information is indeed necessary for the solution of another subsystem's control problem are referred to as its neighbors.

Furthermore, different schemes often arrive at different solution strategies [LiuEtAl10]. In some schemes, e.g., [StewartEtAl10, StewartWrightRawlings11, FerramoscaEtAl13], communication and the solution of the distributed optimization problems can happen in parallel, usually in an iterative fashion. This means that the solution can be improved by multiple, synchronous iterations of communication and computation within one time step, at best recovering the performance of the centralized MPC problem as the number of iterations approaches infinity [StewartEtAl10]. Nevertheless, a functioning controller is already obtained with one iteration per time step. Contrasting these iterative schemes, there also exist sequential schemes [LiuEtAl10, MüllerRebleAllgöwer12] in which the subsystem controllers optimize one after another, sending information on the optimization's result to the neighboring systems immediately afterward and, for each robot, only once every time step. Naturally, while non-neighboring systems can still optimize in parallel, this approach is not as universally scalable as iterative schemes with their parallel computation and communication, although sequential optimization makes it easier to consider coupled constraints [MüllerRebleAllgöwer12].

From a technical perspective, the actual optimization problems solved by the distributed controllers are not fundamentally different from optimization problems solved for centralized MPC. The decisive difference lies in the fact that each subsystem only optimizes over its own control input, with the current control inputs and the corresponding predicted state trajectories of the other systems being unknown. Hence, they are approximated by means of feasible candidates that are built based on the communicated results from the previous time step and inserted as additional parameters into the optimization problem. Interestingly, iterative DMPC schemes can often be seen as an attempt to solve an MPC problem, that has been designed in a centralized fashion, with distributed optimization

methods designed to parallelize the solution of general optimization problems. For instance, the schemes from [StewartEtAl10, FerramoscaEtAl13] are akin to the iterative Jacobi algorithm [BertsekasTsitsiklis89]. Approaching distributed MPC from this alternative direction, it can be promising to seek the application of other methods from the field of distributed and parallel optimization [NotarstefanoNotarnicolaCamisa19]. However, some otherwise efficient approaches require a centralized entity coordinating the distributed solution of the subproblems [BraunEtAl16, BraunEtAl18].

Apart from showing the variety of the research field, the above disquisitions make it relatively clear that there cannot be one scheme to fit all applications. Henceforth, the most crucial step is to carefully formulate the optimization problem that encapsulates the control goals and to closely inspect the structure and the couplings between quantities from different subsystems. This already coarsely determines which schemes are a potential fit. Crucial to this thesis are the requirements of cooperative distributed robotics. A common requirement is that couplings in the cost function are a necessity, often without any separable structure. This stems from the fact that a priori, a group of cooperating robots is dynamically independent and actuated individually, with the couplings and cooperation arising primarily from the fact that the robots shall solve one common task, meaning that the robots try to minimize a joint cost function in a distributed manner. Furthermore, robotics usually necessitates short sampling times, especially compared to applications in process industries, which are a typical field for the application of DMPC in the literature [AlvaradoEtAl11]. Thus, although sequential schemes can be used for a moderate number of cooperating robots [EbelSharafian ArdakaniEberhard17a], iterative schemes seem beneficial, since, for those, all robots can optimize in parallel. Apart from that, all considerations from centralized MPC persist, e.g., whether the nominal dynamics is linear or nonlinear and whether more general control goals than setpoint stabilization can be considered, with economic distributed MPC having received ample attention in recent years [MüllerAllgöwer14, KöhlerMüllerAllgöwer18]. Of particular importance to robotics is the topic of tracking control, since it is rarely the goal in robotics to stabilize a never-changing setpoint. Instead, if robots perform work tasks, it usually means that the robots or end-effectors move to different points in space or follow a trajectory. Therefore, it is at least necessary that the setpoint to be stabilized can be changed at system runtime. However, if terminal constraints are used as stabilizing design ingredients, e.g., a terminal set around the setpoint, the optimization problem might become infeasible if the set around the new setpoint cannot be reached within the prediction horizon. This can be a problem for centralized MPC as well as for distributed MPC, with terminal constraints also being used in certain DMPC schemes, see, e.g., [StewartEtAl10, MüllerRebleAllgöwer12]. One strategy to deal with this is to introduce an artificial steady state as an additional optimization variable. Tracking the corresponding artificial reference while penalizing deviations of the artificial from the real reference allows to keep recursive feasibility for arbitrary changes of the reference, asymptotically stabilizing the reference value if it is feasible

and kept constant. Based on a scheme of this type for centralized MPC [LimonEtAl08], nominal guarantees for a distributed scheme have been proven in [FerramoscaEtAl13], employing a terminal cost and terminal set calculated for the full, centralized system in a preprocessing step. The scheme is of the iterative type and can deal with linear dynamics, coupled constraints, and a fully coupled cost function. Henceforth, it seems to be a good candidate for many use cases in distributed robotics, at least as long as linear dynamics can be used. To help with the latter, it can be possible and acceptable in some cases to linearize along reference trajectories or employ subordinate controllers for the feedback linearization of the subsystems [Schnelle18]. However, when looking for genuinely nonlinear DMPC schemes meeting the most typical requirements of distributed robotics, i.e., tracking control with support for coupled, cooperative cost functions, the situation seems to be bleaker, at least regarding schemes with nominal stability guarantees. For instance, the rather encompassing overview in [NegenbornMaestre14] does not list a single scheme meeting these requirements. Nevertheless, this need not mean that a heuristically designed, nonlinear DMPC scheme, without proven nominal guarantees, cannot work well in practice. Still, driven by the type of hardware employed, this thesis's main application will rely on a linear dynamic model in the distributed control layer, demonstrating the practical coordinative usefulness of linear distributed tracking control.

In the schemes referenced above, which mostly come from the theoretical MPC community, actual implementations usually focus on fixed, small numbers of systems, both when real hardware is involved [AlvaradoEtAl11] as well as in simulative scenarios [StewartEtAl10, StewartWrightRawlings11]. Similarly, the online reconfiguration of the controllers, with systems spontaneously joining or leaving, is usually not treated. This is probably due to the implementational complexity raised by not having merely one control loop but rather a variable number of synchronously running control loops for which also networked communication has to be implemented. Sometimes, although without loss of generality, even the theory is described for two systems only [StewartEtAl10, StewartWrightRawlings11] because even the notation can get arduous, which is a testament to the schemes' practical complexity. Hence, it will be very interesting to see within Chapters 5 and 6 how DMPC can be employed to cope with the practical challenges of a dynamic network of mobile robots. However, due to the technical complexity of DMPC, it seems prudent to have handy a control paradigm that poses less implementational and computational challenges, permitting comparisons between the schemes to see whether the application of DMPC brings about enough practical advantages to warrant dealing with its complexities. Such a control paradigm is introduced in the subsequent section.

3.3 Graph-Algebraic Control

The defining idea of the graph-algebraic approach to distributed control is to represent as a graph the information exchange structure between several systems that shall cooperate. The approach is usually motivated to arise from coordination problems of a group of systems, usually called agents, with typically simple dynamics, e.g., single integrator dynamics [MesbahiEgerstedt10]. Examples include a group of mobile robots agreeing on and moving to a common position, maintaining a formation, or agreeing on a common velocity, often inspired by natural phenomena like the flocking behavior of a swarm of birds [BulloCortésMartínez09, MesbahiEgerstedt10, BaiArcakWen11]. Many variants and extensions exist, e.g., assuming either bidirectional or unidirectional communication and extending the theory to more general system dynamics. As the name suggests, the area is built upon algebraic representations of graphs, e.g., the incidence matrix introduced in Section 2.2.1, with these representations and their properties helping with control design. The foundational task is to let a group of mobile robots, modeled as single integrators, meet at a point, which is referred to as the agreement problem [BaiArcakWen11]. This can be extended to formation control by letting the relative vectors between the robot positions not converge to zero but to desired, non-zero values. Hence, formation control can be seen as a shifted agreement problem. While the focus on the positioning of mobile robots seems limiting, in more abstract manners, agreement means that a group of systems agrees on and converges to a common state, whereas formation control means that the systems coordinate their states relative to one another.

To formulate agreement and formation control problems using a graph-algebraic paradigm, the incidence matrix is useful. Assuming bidirectional communication, the initial step is to represent the communication structure as an undirected graph and to define an arbitrary oriented graph from that. This is shown in the two illustrations on the left of Figure 3.2, with three robots participating, although robots 1 and 3 cannot directly communicate with one another. To obtain more compact matrices for, e.g., handling the two movement directions of robots in the plane, it is useful to define the short-hand

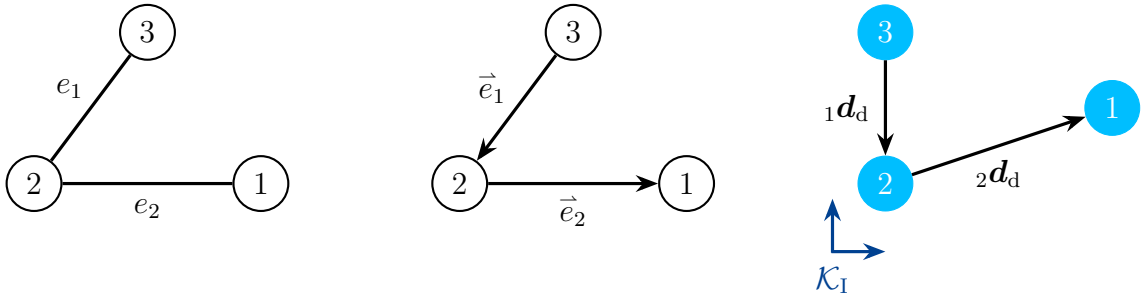


Figure 3.2: From left to right, illustrations of a bidirectional communication graph, a corresponding oriented graph, and a desired formation are shown, respectively.

notation $\bar{\boldsymbol{\mathcal{B}}} \coloneqq \boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}} \otimes \boldsymbol{I}_2$ with the identity $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$, $n \in \mathbb{N}$, and $\otimes$ denoting the Kronecker product. This can be directly generalized for any dimension, but, due to the premises of the thesis and for illustratory purposes, the subsequent expressions are given for two dimensions, sticking with the intuitively accessible notion of robots moving in the plane. It follows that the relative vectors between the robot positions along the edges of the oriented graph can be obtained from the robot positions ${}_i\boldsymbol{x} \in \mathbb{R}^2$ by means of $\boldsymbol{d} = \bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\boldsymbol{x}$ with $\boldsymbol{d} = \begin{bmatrix} {}_1\boldsymbol{d}^{\mathsf{T}} & \dots & {}_{n_\mathcal{E}}\boldsymbol{d}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$, $\boldsymbol{x} = \begin{bmatrix} {}_1\boldsymbol{x}^{\mathsf{T}} & \dots & {}_{n_\mathcal{V}}\boldsymbol{x}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. This can be readily verified for the example from Figure 3.2 since the transposed incidence matrix of the oriented graph therein is given by

$$\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}^{\mathsf{T}} = \begin{bmatrix} 0 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix}. \tag{3.35}$$

A specific, desired formation given by ${}_i\boldsymbol{d}_\mathrm{d} \overset{!}{=} {}_i\boldsymbol{d}$ is illustrated on the right-hand side of the figure. From that illustration, it is intuitively clear that the formation, merely defined by relative vectors, can be attained anywhere in the plane. Mathematically, this follows from the aforementioned fact that the rank of the incidence matrix of an orientation of a connected graph with $n_\mathcal{V}$ nodes is $n_\mathcal{V} - 1$ [GodsilRoyle01], and, therefore, it holds that $\operatorname{rank}\left(\bar{\boldsymbol{\mathcal{B}}}\right) = 2\operatorname{rank}\left(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}\right) = 2\,(n_\mathcal{V} - 1)$. Thus, the null space of $\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}$ is of dimension 2 and can be identified as $\ker\left(\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\right) = \{\mathbf{1}_{n_\mathcal{V}} \otimes \boldsymbol{n} \mid \boldsymbol{n} \in \mathbb{R}^2\}$ due to $\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}^{\mathsf{T}}\mathbf{1}_{n_\mathcal{V}} = \mathbf{0}$, formally giving the invariance to shifts of the whole formation by a displacement $\boldsymbol{n}$.

For robots modeled with the single-integrator dynamics ${}_i\dot{\boldsymbol{x}} = {}_i\boldsymbol{u}$, the agreement problem can be solved with the feedback

$$\boldsymbol{u} = -\bar{\boldsymbol{\mathcal{B}}}\boldsymbol{d} = -\bar{\boldsymbol{\mathcal{B}}}\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\boldsymbol{x} =: -(\boldsymbol{L} \otimes \boldsymbol{I}_2)\boldsymbol{x} \tag{3.36}$$

where $\boldsymbol{u} = \begin{bmatrix} {}_1\boldsymbol{u}^{\mathsf{T}} & \dots & {}_{n_\mathcal{V}}\boldsymbol{u}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. Therein, $\boldsymbol{L} = \boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}^{\mathsf{T}}$ is the so-called Laplacian matrix of the underlying undirected communication graph [MesbahiEgerstedt10]. It is independent of the orientation used to obtain $\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}$ [GodsilRoyle01] and it is, by definition, symmetric and positive semi-definite. Closed-loop convergence can be proven by using $V(\boldsymbol{x}) = \frac{1}{2}\,\boldsymbol{d}^{\mathsf{T}}\boldsymbol{d} = \frac{1}{2}\,\boldsymbol{x}^{\mathsf{T}}(\boldsymbol{L} \otimes \boldsymbol{I}_2)\,\boldsymbol{x}$ as a Lyapunov function candidate and employing LaSalle's invariance principle [Khalil02] to conclude that the system state approaches the set $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^{2\,n_\mathcal{V}} \mid (\boldsymbol{L} \otimes \boldsymbol{I}_2)\,\boldsymbol{x} = \mathbf{0}\}$, which is positively invariant with respect to the closed loop. For a connected graph, $\boldsymbol{L}$ has exactly one eigenvalue that is zero, with $\mathbf{1}_{n_\mathcal{V}}$ as a corresponding eigenvector [GodsilRoyle01]. Consequently, the closed loop approaches $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^{2\,n_\mathcal{V}} \mid {}_1x_i = {}_2x_i = \dots = {}_{n_\mathcal{V}}x_i,\ i \in \{1, 2\}\}$, as desired. Attaining and maintaining a formation with $\boldsymbol{d}_\mathrm{d} \neq \mathbf{0}$ can be treated as a shifted agreement problem with $\boldsymbol{u} = -\bar{\boldsymbol{\mathcal{B}}}(\boldsymbol{d} - \boldsymbol{d}_\mathrm{d})$.

From these disquisitions, it is evident that this approach to cooperative control is vastly different in its structure and way of thinking than DMPC. In DMPC, as long as the selected scheme meets the structural requirements of the system dynamics and the control task, the control goal and dynamics can be simply inserted into the optimization problem, whereas the graph-algebraic approach may require a new derivation. Although, motivated by the

hardware setup employed, this thesis later on focuses on very specific dynamics, Excursus 2 gives an instructive example of how the theory can be extended to linear second-order dynamics. For more general cases, [BaiArcakWen11] explains a rather systematic approach to design graph-algebraic controllers based on the concept of passivity [Khalil02].

Excursus 2: Agreement for Double Integrator Dynamics

When mobile robots are modeled by means of second-order dynamics of the form ${}_i\boldsymbol{M}\,{}_i\ddot{\boldsymbol{q}} = {}_i\boldsymbol{u}$ with ${}_i\boldsymbol{q} \in \mathbb{R}^2$ as the position of robot i and ${}_i\boldsymbol{M} = m_i\,\boldsymbol{I}_2 > 0$, the above derivations can be extended rather easily. Agreement can be reached with the feedback $\boldsymbol{u} = -\boldsymbol{D}\dot{\boldsymbol{q}} - \bar{\boldsymbol{\mathcal{B}}}\boldsymbol{d} = -\boldsymbol{D}\dot{\boldsymbol{q}} - (\boldsymbol{L} \otimes \boldsymbol{I}_2)\boldsymbol{q}$, which, by means of the diagonal matrix $\boldsymbol{D} > 0$, introduces artificial damping into the closed loop. Defining $\boldsymbol{q} \coloneqq \begin{bmatrix} {}_1\boldsymbol{q}^\mathsf{T} & \dots & {}_{n_\mathcal{V}}\boldsymbol{q}^\mathsf{T} \end{bmatrix}^\mathsf{T}$, $\boldsymbol{x} \coloneqq \begin{bmatrix} \boldsymbol{q}^\mathsf{T} & \dot{\boldsymbol{q}}^\mathsf{T} \end{bmatrix}^\mathsf{T}$, $\boldsymbol{M}$, and $\boldsymbol{u}$ such that $\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{u}$ and using the Lyapunov function candidate $V(\boldsymbol{x}) = \frac{1}{2}\,\dot{\boldsymbol{q}}^\mathsf{T}\boldsymbol{M}\dot{\boldsymbol{q}} + \frac{1}{2}\,\boldsymbol{d}^\mathsf{T}\boldsymbol{d}$ gives $\dot{V}(\boldsymbol{x}) = \dots = -\dot{\boldsymbol{q}}^\mathsf{T}\boldsymbol{D}\dot{\boldsymbol{q}}$. Hence, the closed loop approaches the positively invariant set

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathbb{R}^{4\,n_\mathcal{V}} \;\middle|\; \dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = \boldsymbol{0} \right\} = \left\{ \boldsymbol{x} \in \mathbb{R}^{4\,n_\mathcal{V}} \;\middle|\; \dot{\boldsymbol{q}} = \boldsymbol{0},\ (\boldsymbol{L} \otimes \boldsymbol{I}_2)\,\boldsymbol{q} = \boldsymbol{0} \right\} \tag{3.37}$$

by means of LaSalle's invariance principle [Khalil02], yielding

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathbb{R}^{4\,n_\mathcal{V}} \mid \dot{\boldsymbol{q}} = \boldsymbol{0},\ {}_1q_i = {}_2q_i = \dots = {}_{n_\mathcal{V}}q_i,\ i \in \{1, 2\} \right\} \tag{3.38}$$

by the same arguments as before. A simulation result for the agreement of three robots subject to the communication structure from Figure 3.2 is shown in Figure 3.3. The parameters are chosen to $m_1 = m_2 = m_3 = 1\,\mathrm{kg}$ and $\boldsymbol{D} = 2\,\boldsymbol{I}_6\,\mathrm{kg/s}$. The continuous control law is implemented in a discretized fashion using a zero-order hold with a sampling time of 0.05 s, assuming perfect communication between the systems. Figure 3.4 shows a simulation result for formation control using the same parameters but the modified control law $\boldsymbol{u} = -\boldsymbol{D}\dot{\boldsymbol{q}} - \bar{\boldsymbol{\mathcal{B}}}\,(\boldsymbol{d} - \boldsymbol{d}_\mathrm{d})$ with $\boldsymbol{d}_\mathrm{d} = \begin{bmatrix} {}_1\boldsymbol{d}_\mathrm{d}^\mathsf{T} & {}_2\boldsymbol{d}_\mathrm{d}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and ${}_1\boldsymbol{d}_\mathrm{d} = \begin{bmatrix} 0 & -2 \end{bmatrix}^\mathsf{T}\,\mathrm{m}$, ${}_2\boldsymbol{d}_\mathrm{d} = \begin{bmatrix} 3 & 1 \end{bmatrix}^\mathsf{T}\,\mathrm{m}$, which corresponds to the formation shape on the right-hand side of Figure 3.2.

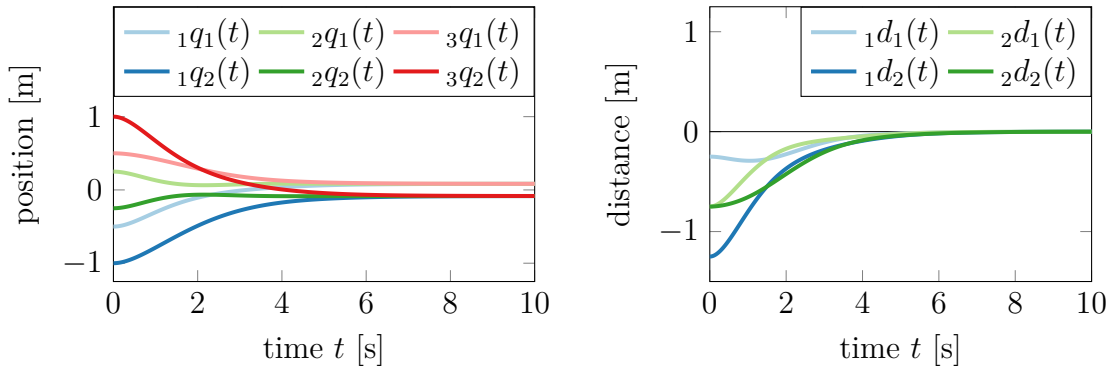


Figure 3.3: Simulation result of the agreement of double integrators

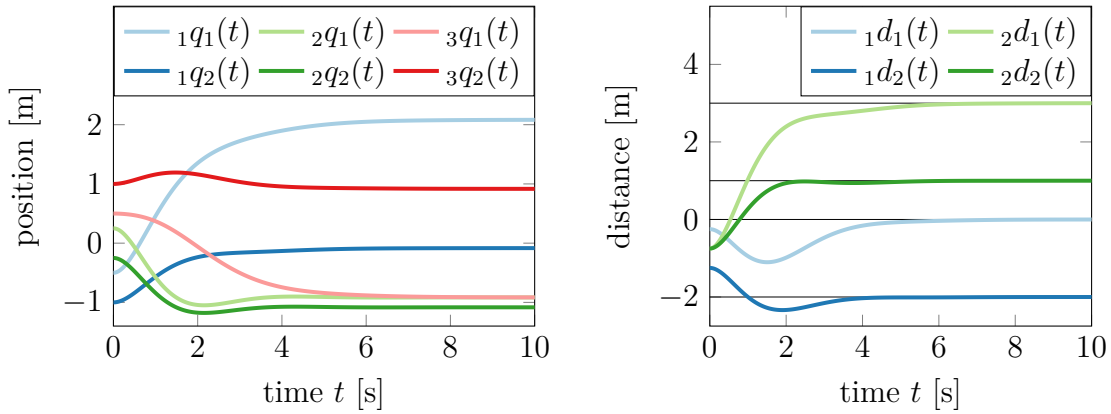


Figure 3.4: Simulation result of double integrators moving into formation

Chapter 4

Designing a Scheme for Cooperative Robotic Behavior

As hinted at by the fundamentals discussed, distributed robotics' central challenge is its reliance on fundamentals and methods from a wide variety of research directions. A particular difficulty is the design of a structure accommodating, arranging, and interconnecting all these methods so that practical tasks can be solved reliably by the robotic system. Following this line of thought, this chapter deduces the structure of a scheme for cooperative robotic task solution from requirements and features characteristic to distributed robotics. Hence, in the first section, these requirements are defined and discussed, with the proposed approach and system architecture being introduced in the subsequent section. The chapter concludes with the design of a mobile robot tailored to the proposed system architecture's operational requirements, preparing the ground for the design of the concrete control and organization schemes operating within the architecture.

4.1 Defining Features of Distributed Cooperative Behavior

The robustness and flexibility advantages associated with the field can only come to full fruition if it is possible that any of the entities in the robotic network can leave or enter the task at any time. Hence, time-critical decisions should at best be made in a thoroughly distributed manner, meaning that these decisions are not made by any centralized entity, e.g., by one of the robots acting as a leader. This is particularly important with regard to the dynamic control of the robots since that needs to happen within a predefined sampling time, so it cannot be expected that there is enough spare time to shift control tasks between robots in the network. In contrast, for slower-paced organizational tasks, like constructing a map of the environment and, e.g., determining the general movement direction of the

robots, it is often acceptable if one of the robots takes responsibility for solving the task. If the corresponding robot leaves the network, there is usually enough time for another robot to take over the responsibility. This is particularly true for tasks that can be safely interrupted for a while. For instance, if the responsibility for mapping the environment needs to be reassigned, the robots may be able to stop in a coordinated manner and stay stationary until mapping is resumed and it is safe to proceed. This also highlights why dynamic control must work at any time because only then can the robots pause the task solution in a well-defined, coordinated manner. Furthermore, for time-critical dynamic control, the computational effort should not increase substantially with the number of robots cooperating, giving a further reason against the usage of a centralized entity since the centralized problem usually grows in size at least linearly with increasing numbers of robots. This suggests using either computationally inexpensive control methods, like graph-algebraic control, or distributed MPC, where each robot optimizes only its own control input.

Furthermore, apart from fully realizing the flexibility of a distributed robotic system, with robots joining and leaving at any time, another self-imposed requirement of this thesis is that the resulting scheme should be scenario agnostic. This means that, within the defined scope of the task, the proposed scheme shall automatically accommodate different scenarios without any manual, scenario-based tuning of parameters. In particular, this implies that there must be a clear mathematical interface for defining the task and scenario, making the task amenable to automation and making it implicitly clear which kinds of scenarios can be handled. This once more highlights why the thesis' model task of cooperative transportation is a formidable benchmark case since it is accessible to the solution with different numbers of robots, allowing for a wide range of different, challenging scenarios due to different object shapes and environments to be considered. It poses challenges not only with regard to control but also with regard to organization, with the robots needing to automatically organize and reorganize around the object, e.g., when robots join or leave the task solution. Hence, although the general architecture proposed in this chapter is useful beyond this specific choice of task, cooperative object transportation will already serve as an intuitively accessible example from now on. Moreover, due to the complexity of a distributed control system, it is particularly crucial to design and test the scheme and its implementation in meaningful simulation scenarios. Since the implementation can be very challenging, particularly regarding communication and all the intricacies arising from it, it is of utmost importance for the development process that these simulations share as much of the program structure as possible with the hardware experiments. Therefore, also in simulations, the final program structure should be present, with each robot's control logic running separately, exchanging data via communication. Finally, to reduce the amount of communication necessary, the methods proposed in the thesis will strive to communicate one type of data at most once per time step of the digital real-time control.

4.2 Approach and System Architecture

Meeting the requirements discussed above, this thesis proposes a very modular system architecture. Not only does the control logic of every robot run in separate programs already in simulation, but even control and organization tasks are split into different, communicating programs for each robot. Since all these programs communicate via network protocols, they can run anywhere on a network, e.g., all on the same or on a group of simulation computers, or on onboard processing units of physical mobile robots. Furthermore, the architectural splitting of overarching organizational tasks from closed-loop control tasks results in programs that can run concurrently, facilitating their operation on different time scales. The latter can be necessary since the execution of an organizational planning step may take considerably longer than the sampling time of the closed-loop control. Indeed, the cycle time of planning tasks may even potentially vary significantly over time, depending on the difficulty of the current situation. Subsequently, the entirety of the logic controlling the time-critical, dynamic behavior of one dynamical system, e.g., of one mobile robot, is called a control agent. A program entity governing organizational matters shall be denoted as an organization agent. Similarly, a robotic agent shall denote the control and organization agents associated with one robot.

The proposed design paradigm is vastly different from the classic paradigm encountered in non-distributed robotics. The procedures executed by a robot operating on its own can be categorized into sensing, planning, and control [Sharir89], which are also usually executed in this order and then repeated. For a robotic agent cooperating with others in a distributed fashion, sensing is supplemented by communication since not all information is gathered by onboard sensors but also communicated by the other robotic agents. Similarly, individual planning is supplemented by organization, which can be defined as those planning tasks that need to be coordinated between the robots, leading to a semantic program structure as illustrated in Figure 4.1. Therein, parts that rely on communicated information are shaded in green. However, it would be short-sighted to perceive communication and organization as mere networked extensions of sensing and individual planning since the

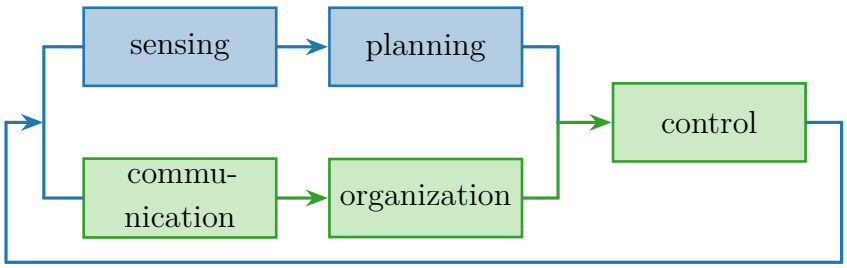


Figure 4.1: Semantic program structure of a robotic agent cooperating with other robotic agents to solve tasks in a distributed fashion

resources used by the former two are not entirely under the authority of the robotic agent itself. In particular, it is unclear when exactly and in which order information is received from the other robotic agents. Hence, depending on the criticalness of the information, the robotic agents either need to wait until all necessary information has been exchanged or the designed methods need to be robust enough so that they can run asynchronously, i.e., continuing to operate when certain information is still missing. Due to the arguments given above, even within one robotic agent, there may be concurrent processes running different tasks. The fact that it is not a priori clear when updated information becomes available from a concurrently running process can even raise technical issues such as the need to ascertain that data structures are not read while they are overwritten with incoming information. Otherwise, algorithms may work with inconsistent data, which can lead to undefined behavior. While this thesis does not intend to discuss these more technical sides of such an architecture, it is worth noting that related issues in concurrent computing have troubled computer science, far from robotics applications, for decades [Dijkstra65, BeschastnikhEtAl16, HuangZhang16].

Following this way of thinking, if inspected in more detail, the actual structure proposed to solve practical tasks is considerably more involved than in the schematic from Figure 4.1. For simulation purposes, where also a simulator has to be integrated, the software architecture takes the shape illustrated in Figure 4.2. Each light-gray box indicates a program running separately from the others, exchanging data with the others only via communication. The larger, dark-gray boxes with index tabs organize these programs into semantic groups, indicating their purpose. Each colored, outward-pointing arrow-like symbol indicates that the corresponding program is publishing data on a communication channel. Similarly, the corresponding opposing symbol indicates that the program is subscribed to a communication channel. Each color corresponds to one channel. While different channel configurations are conceivable, the given example is suitable to illustrate which program needs to exchange data with which other program. In practice, it can make sense for each robot to use its own set of channels so that, e.g., the control inputs of different robots are published on different channels. In Figure 4.2, without loss of generality, these are subsumed to one channel for each type of data for a more compact representation. Each robotic agent consists of a control agent and one or several organization agents. The control agent comprises all time-critical tasks and should be executed at a short, constant sampling rate. Apart from dynamic control, time-critical tasks usually also include some form of local mapping and individual navigation so that, e.g., collisions with obstacles can be avoided. In contrast, slower-paced global planning is part of an organization agent, of which there may be multiple if different organizational tasks shall be solved in a decoupled manner. In many situations, it can be desirable to manipulate the ongoing experiment in real time, e.g., to let robots spontaneously join or leave the task solution at the user's behest. This can be done via the event agent, which reads user inputs and publishes event messages, which are received by all agents whom it may concern.

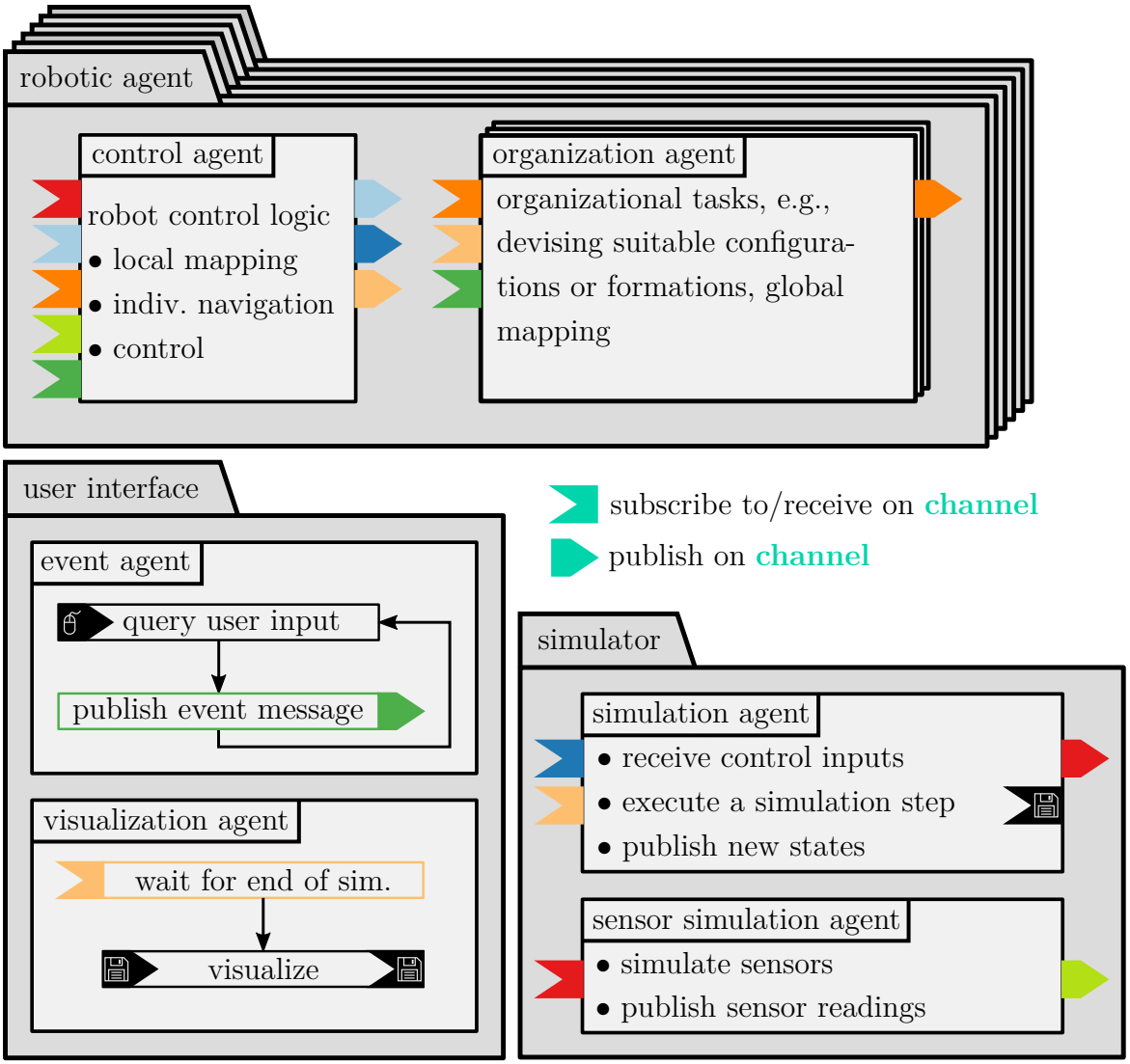


Figure 4.2: Proposed software architecture for the cooperative task solution with a group of simulated robots

Since the tasks are distributed over the network, also all information generated in the course of a simulation is distributed. However, to be able to work with and learn from the results of a simulation, it is important to gather and record all critical information in one place. By nature, the most critical information is collected and generated by the simulation agent. Hence, it also stores the result data, as indicated by the symbol with the floppy-disk pictogram. The robot agents may send additional debug data to the simulation agent to aggregate data relevant to the development process in one place. For instance, this data may be read by a visualization agent after the simulation has concluded, automatically creating and storing visualizations, e.g., animated sequences depicting the behavior of the robots over time. During usage, it is most practical to execute the visualization agent and the simulation agent on the same machine, reading and writing from and to the same memory.

Structure-wise, if experiments involving real-world hardware shall be performed, it merely suffices to replace the simulator with robotic hardware. Then, the control inputs, corresponding to the dark-blue channel, are received and executed by the robotic hardware. Conversely, onboard sensors or an external tracking system publish the state information and sensor data needed by the robotic agent. Prospective experiments involving real-world hardware give a reason why it can make sense to split the simulator into two concurrently running program types, namely the simulation and sensor simulation agents, as it is done in Figure 4.2. The simulation agent takes care of the time integration of all differential equations modeling the overall system consisting of the robots and their workspace. For instance, this includes the model equations of the individual robots and the dynamics of contacts between robots and objects in the workspace, e.g., the object to be transported in a cooperative transportation scenario. For mapping, navigation, and obstacle avoidance, however, robots also need some form of sensors to perceive the environment, e.g., light detection and ranging sensors that deliver distance measurements to surfaces surrounding the robot. These need to be simulated, too, which is the responsibility of the sensor simulation agent. Since these kinds of sensors can be comparatively expensive and are often not in the focus of research on dynamic control, it can be desirable to keep simulating them even when using real-world hardware, alleviating the need to equip many robots with such a sensor. Then, the real-world robot merely replaces the simulation agent and not the whole simulator. It is even conceivable with this structure that some robots use hardware sensors only, while others rely on simulated sensors.

A significant challenge posed by a structure as distributed as that of Figure 4.2 is how to realize the communication without any centralized entity. For instance, the robot operating system (ROS), an open-source middleware that has become very popular in robotics research and applications, relies on a single master node in the network, facilitating the peer-to-peer communication [QuigleyEtAl09]. While there exist extensions [TiderkoHoellerRöhling16], the fundamental design principle is still at odds with distributed robotics. Therefore, the applications in this thesis do not rely on ROS for communication, but make use of the

open-source lightweight communications and marshalling (LCM) library, which functions in a truly distributed fashion [HuangOlsonMoore10] and fits well to the publish-subscribe pattern described above. Furthermore, it relies on UDP multicast to communicate the data in the form of individual messages from the publishers to the subscribers, which corresponds well to typical communication topologies in distributed robotics. A message sent by one robotic agent can be of interest to potentially many of the other robotic agents. Hence, if unicast communication were employed, potentially even via TCP, communication links between the sending agent and each receiving agent would need to be established and the message would need to be sent individually to each recipient. Similar reasons, together with doubts about the real-time capabilities of the original ROS, have recently also sparked the development of ROS2, a new, profoundly different version of the classic robot operating system [ErősEtAl19, PuckEtAl20]. Thus, research projects which are not only looking for a communication library may also consider the usage of ROS2 in the future, since the robotic operating system can be seen as a whole open-source ecosystem giving access to a wide range of tools and algorithm implementations useful to robotics. However, for this thesis, a pure communication library suffices. Having now devised the general schematic of a software architecture tailored to distributed robotics, it seems advisable to look at robotic hardware that can serve as a powerful demonstrator for research on distributed robotics.

4.3 A Custom Mobile Robot for Cooperative Tasks

While distributed robotics predominantly seems to be a challenge with regard to control and organization algorithms, and, therefore, with regard to software, it can also greatly benefit from hardware appropriate for the task. In particular, research on different problems and the evaluation of different solution approaches may require a swift reconfiguration of hardware and software. Therefore, it is beneficial to have full authority over all aspects of the robotic hardware and the onboard software. This line of thought and research experience with different robotic hardware [EbelSharafian ArdakaniEberhard17a, EbelEtAl21] have lead to the decision to design a custom mobile robot for the purposes of this thesis and beyond [EbelWahrenLuo20]. The goal of this section is to introduce and intently analyze this custom robot design. To that end, the first subsection starts out with a more detailed inspection of the hardware requirements of a robot for research on distributed control and organization, with the resulting robotic hardware being explained subsequently. The second subsection deals with building a mechanical model of the robot. The model is designed in a way to provide a first insight into whether it meets the dynamic requirements formulated at the outset. Furthermore, in later chapters, the model will prove useful to evaluate the performance of control and organization schemes in meaningful simulation scenarios.

4.3.1 Hardware Design

Concerning the desired mechanical properties of the robot, maneuverability is the main focus. At best, the robot should be able to move omnidirectionally, meaning that the robot can move into any direction at any time if inertia effects are neglected. Hence, the movement capabilities should restrict the possible fields of application as little as possible so that the designed robot may also be useful to applications beyond those envisioned in this thesis. Furthermore, this allows an analysis of distributed control and organization methods that is not clouded by kinematic particularities of the robots employed. However, while the omnidirectional movement capabilities make it easier to use the robot in various applications, the derivations below will show that the description of the robot's kinematics becomes more intricate than for a more classic, non-omnidirectional robot.

Moreover, the designed robot should be easy to build and maintain, with parts either being readily available or, if need be, easy to manufacture, which is of particular importance for research on distributed robotics since a greater number of robots needs to be built and maintained to always be in a fully functional state. The robot should not merely be a platform for single-purpose research but also a tool useful in other research projects. Therefore, it should be easily extensible, both with regard to hardware as well as with regard to software, meaning that the onboard processing unit should be easy to program for. Since distributed robotics requires the exchange of data between cooperating robots, it should also have built-in wireless communication capabilities and still possess sufficient real-time control capabilities to run fast, low-level motor controllers, evaluating the motor encoders' angular velocity signals for feedback purposes and generating the pulse-width modulated motor supply voltages. The Beaglebone Blue [BeagleBoard.org Foundation] is a single-board computer meeting all of these requirements, both concerning wireless communication and real-time computation, making it the onboard computer of choice for the robot designed for this thesis. The board allows the direct connection of up to four DC motors and corresponding motor encoders. Thus, only one computation device is necessary to meet the fundamental requirements, keeping the robot design simple by using fewer individual parts. Apart from wireless network communication, it offers enough options to directly connect an additional computation device by wire, e.g., via serial communication. Hence, if algorithms with computational requirements exceeding the onboard capabilities shall be run in real time on the robot, an additional high-performance single-board computer can be added to the robot, handling computation-intensive tasks while the Beaglebone still deals with low-level motor control.

To obtain the desired omnidirectional movement capabilities, the robot uses four so-called Mecanum wheels [Ilon75], which is a popular setup for omnidirectional mobile robots [WampflerSaleckerWittenburg89, ZimmermannEtAl16, ZeidisZimmermann19]. An independent suspension for each wheel ensures sufficient contact of the wheels to the ground. Figure 4.3 shows different perspectives of a Mecanum wheel as it is used for

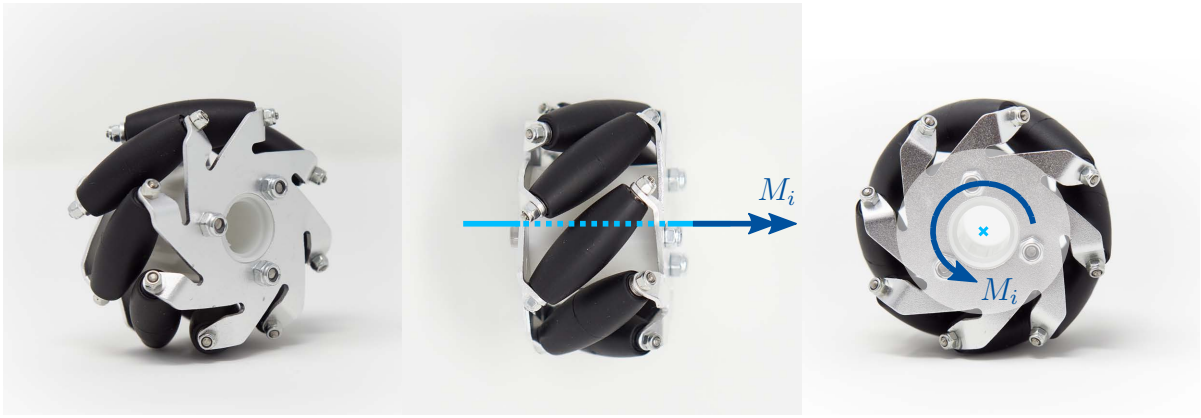


Figure 4.3: Different perspectives of a Mecanum wheel as used on the custom mobile robot

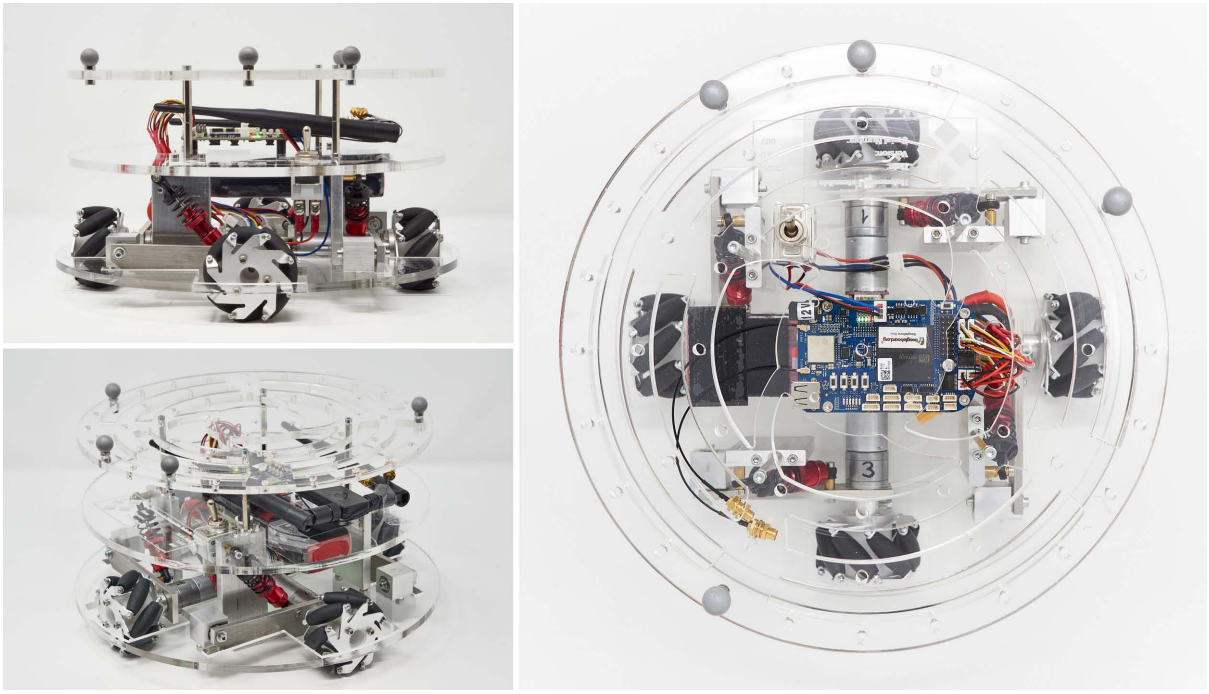


Figure 4.4: Photographs of the custom mobile robot

the robot. To propel the robot, a motor is used to exert a moment about the axis of the wheel as it would be done for a regular wheel, see moment M_i in the figure. The wheel's main body is rigid, with eight rollers mounted along the circumference of the wheel. The rollers can rotate freely about the axis they are mounted on, which is inclined by an angle of 45° to the rotation axis of the wheel's main body. The assembled robot, equipped with four of these wheels, can be seen in Figure 4.4. It has a circular footprint of radius $r_{\mathrm{R}} = 0.145\,\mathrm{m}$. The robot consists of three layers on top of one another, with the lowest layer consisting of the suspension, wheels, and motors, the second layer housing

the onboard computer, and the top layer being a marker plate useful to track the pose of the robot with an external tracking system. Due to the wheels employed, the robot's kinematics and thereby the mechanical modeling of the robot are especially interesting and will be treated subsequently.

4.3.2 Mechanical Model

While it is the goal that the custom robot is very maneuverable, with outer control loops not having to take into account specific particularities like non-holonomic kinematic constraints, a more detailed look does reveal some peculiarities. As in every modeling task, some idealizations and simplifications are prudent. The mechanical model shall be useful to study and represent the general movement capabilities. It hence seems particularly sensible to neglect effects that may heavily depend on parameters that are a priori unknown and may only be identified by experimentation with a finished hardware robot. Therefore, first and foremost, it is assumed that the wheels roll without slipping and that the robot moves in the plane. All parts of the robot are assumed to be rigid. Furthermore, apart from the reaction forces arising from rolling without slipping, friction is neglected. It is assumed that, at any time, there is only one contact point of each wheel with the floor, with it always being at the same position relative to the wheel's center. The wheels' width is considered to be small so that moments proportional to the wheel width can be neglected. Similarly, the diameter of the rollers on the circumference of the wheels is considered to be negligibly small. Prior works studying the dynamics of robots employing the same type of wheels, although in a different layout, arrive at very similar assumptions [ZimmermannEtAl16, ZeidisZimmermann19].

To get started with the kinematic description of the robot, Figure 4.5 gives a stylized drawing of those parts of the robot crucial for the kinematics. As the figure shows, the wheels, which are symbolized by white rectangles with diagonal lines, are mounted in a rotation-symmetric way. The diagonal lines indicate the rollers' directions at the underside of the wheel, with R_1 to R_4 being the wheels' centers of mass. It is assumed that the wheels' contact points with the ground are always exactly below the wheels' centers. The wheel arrangement is rather untypical for robots with Mecanum wheels. In setups appearing in the literature, Mecanum wheels are usually mounted like wheels on a traditional car, with the robot taking a rectangular shape [DickersonLapin91, ZeidisZimmermann19]. However, a round robot with a rotation-symmetric wheel arrangement seems to suit better the omnidirectional movement characteristics, which is why the depicted layout is chosen in this thesis.

To describe the kinematics, some additional notation is necessary. When considered independently, each wheel can roll freely into the direction orthogonal to the axis of the roller that is in contact with the ground. The velocity components of the wheel centers

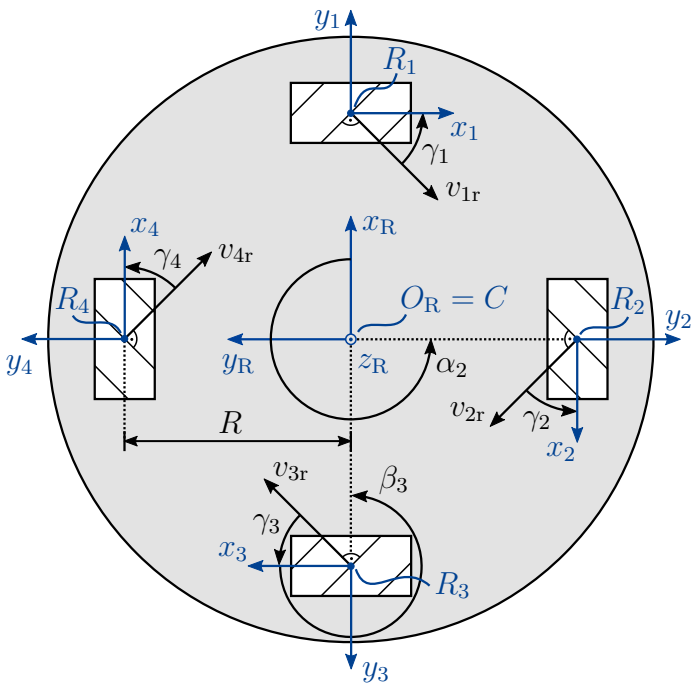


Figure 4.5: Schematic illustration of the custom mobile robot depicting the quantities necessary for the description of the robot's kinematics

along these directions are denoted as v_{ir}, $i \in \{1, \ldots, 4\}$, with positive directions as depicted in Figure 4.5. The origin O_R of the body-fixed frame of reference $\mathcal{K}_R$ is located in the robot's center of mass C, with the orthogonal projections of the wheels' centers of mass R_i onto the x_R-y_R-plane having the distance R to the origin O_R. The angle measured in the mathematically positive direction from the x_R-axis to the vector pointing from O_R to R_i is denoted as α_i. Similarly, β_i denotes the angle measured from the x_i-axis to the direction of the x_R-axis, whereas γ_i is measured from the vector v_{ir} to the x_i-axis. The wheels' angular velocities about the y_i-axis are denoted as $\dot{\varphi}_i$, $i \in \{1, \ldots, 4\}$. The robot's angular velocity about the z_R-axis is denoted as $\dot{\varphi}_R$, whereas ${}^{R}\boldsymbol{v}_C = \begin{bmatrix} {}^{R}v_{Cx} & {}^{R}v_{Cy} \end{bmatrix}^{\mathsf{T}}$ is the robot's absolute velocity in the x_R-y_R-plane represented in the basis of the body-fixed frame. The absolute velocity components of the wheels' centers in the directions of the x_i- and y_i-axes are denoted by v_{R_i,x_i} and v_{R_i,y_i}, respectively. The wheels are rolling without slipping, meaning that the velocity component of wheel i's contact point with the ground into the direction perpendicular to v_{ir} is zero. With the wheels having a radius of r, this gives the kinematic relationship

$$\begin{bmatrix} v_{R_i,x_i} - r\,\dot{\varphi}_i \\ v_{R_i,y_i} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \sin(\gamma_i) \\ \cos(\gamma_i) \end{bmatrix} = 0 \iff \begin{bmatrix} v_{R_i,x_i} \\ v_{R_i,y_i} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \sin(\gamma_i) \\ \cos(\gamma_i) \end{bmatrix} = \begin{bmatrix} r\,\dot{\varphi}_i \\ 0 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \sin(\gamma_i) \\ \cos(\gamma_i) \end{bmatrix}. \tag{4.1}$$

Furthermore, since the robot chassis is assumed to be rigid, each wheel center's planar absolute velocity ${}^{R}\boldsymbol{v}_{R_i}$ represented in the basis of the body-fixed frame can be expressed

in the form

$$ {}^{\mathrm{R}}\boldsymbol{v}_{R_i} = \underbrace{\begin{bmatrix} 1 & 0 & -R\sin(\alpha_i) \\ 0 & 1 & R\cos(\alpha_i) \end{bmatrix}}_{=:\boldsymbol{C}_i} \begin{bmatrix} {}^{\mathrm{R}}v_{\mathrm{C}x} \\ {}^{\mathrm{R}}v_{\mathrm{C}y} \\ \dot{\varphi}_{\mathrm{R}} \end{bmatrix}. \tag{4.2} $$

Transforming to the wheel coordinate system yields

$$ \begin{bmatrix} v_{R_i,x_i} \\ v_{R_i,y_i} \end{bmatrix} = \begin{bmatrix} \cos(\beta_i) & -\sin(\beta_i) \\ \sin(\beta_i) & \cos(\beta_i) \end{bmatrix} {}^{\mathrm{R}}\boldsymbol{v}_{R_i}. \tag{4.3} $$

Finally, inserting Equation (4.2) into Equation (4.3) and the result into Equation (4.1) allows to solve for $\dot{\varphi}_i$, which results in

$$ \dot{\varphi}_i = \frac{1}{r} \begin{bmatrix} \cos(\beta_i) + \sin(\beta_i)\frac{\cos(\gamma_i)}{\sin(\gamma_i)} & -\sin(\beta_i) + \cos(\beta_i)\frac{\cos(\gamma_i)}{\sin(\gamma_i)} \end{bmatrix} \boldsymbol{C}_i \begin{bmatrix} {}^{\mathrm{R}}v_{\mathrm{C}x} \\ {}^{\mathrm{R}}v_{\mathrm{C}y} \\ \dot{\varphi}_{\mathrm{R}} \end{bmatrix}. \tag{4.4} $$

By inserting different values for the angles α_i, β_i, and γ_i with $\gamma_i \neq k\pi$, $k \in \mathbb{Z}$, this kinematic relationship holds for a wide variety of wheel configurations. Using the values corresponding to the setup shown in Figure 4.5 yields

$$ \dot{\varphi}_1 = \frac{1}{r}\left({}^{\mathrm{R}}v_{\mathrm{C}x} - {}^{\mathrm{R}}v_{\mathrm{C}y} - R\,\dot{\varphi}_{\mathrm{R}}\right), \tag{4.5} $$

$$ \dot{\varphi}_2 = \frac{1}{r}\left(-{}^{\mathrm{R}}v_{\mathrm{C}x} - {}^{\mathrm{R}}v_{\mathrm{C}y} - R\,\dot{\varphi}_{\mathrm{R}}\right), \tag{4.6} $$

$$ \dot{\varphi}_3 = \frac{1}{r}\left(-{}^{\mathrm{R}}v_{\mathrm{C}x} + {}^{\mathrm{R}}v_{\mathrm{C}y} - R\,\dot{\varphi}_{\mathrm{R}}\right), \tag{4.7} $$

$$ \dot{\varphi}_4 = \frac{1}{r}\left({}^{\mathrm{R}}v_{\mathrm{C}x} + {}^{\mathrm{R}}v_{\mathrm{C}y} - R\,\dot{\varphi}_{\mathrm{R}}\right), \tag{4.8} $$

which can be expressed equivalently in the form

$$ {}^{\mathrm{R}}v_{\mathrm{C}x} = \frac{r}{2}\left(\dot{\varphi}_1 - \dot{\varphi}_2\right), \tag{4.9} $$

$$ {}^{\mathrm{R}}v_{\mathrm{C}y} = \frac{r}{2}\left(\dot{\varphi}_3 - \dot{\varphi}_2\right), \tag{4.10} $$

$$ \dot{\varphi}_{\mathrm{R}} = -\frac{r}{2R}\left(\dot{\varphi}_1 + \dot{\varphi}_3\right), \tag{4.11} $$

$$ \dot{\varphi}_4 = \dot{\varphi}_1 + \dot{\varphi}_3 - \dot{\varphi}_2. \tag{4.12} $$

Evidently, Equations (4.11) and (4.12) are integrable, and therefore, define holonomic constraints. Given suitable initial conditions, they allow, e.g., to express φ_4 and φ_{R} in terms of φ_1, φ_2, and φ_3. Hence, subsequently, the generalized velocities $\boldsymbol{s} \coloneqq \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 \end{bmatrix}^{\mathsf{T}}$ are used to describe the system dynamics, with the vector of generalized coordinates $\boldsymbol{q} \coloneqq \begin{bmatrix} x_{\mathrm{C}} & y_{\mathrm{C}} & \varphi_1 & \varphi_2 & \varphi_3 \end{bmatrix}^{\mathsf{T}}$ fully describing the current configuration of the robot.

Interestingly, the constraints (4.9) and (4.10) are non-integrable and hence non-holonomic, as it is also the case for the traditional wheel arrangement [ZeidisZimmermann19]. This

can be shown by means of a short calculation as follows. Denoting as $\boldsymbol{r}_\mathrm{C} = \begin{bmatrix} x_\mathrm{C} & y_\mathrm{C} \end{bmatrix}^\mathsf{T}$ the position of the robot's center of mass in the inertial frame of reference, it holds that

$$\begin{aligned} \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x}_\mathrm{C} \\ \dot{y}_\mathrm{C} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} &= \begin{bmatrix} \frac{r}{2}\cos(\varphi_\mathrm{R}) & \frac{r}{2}\left(\sin(\varphi_\mathrm{R}) - \cos(\varphi_\mathrm{R})\right) & -\frac{r}{2}\sin(\varphi_\mathrm{R}) \\ \frac{r}{2}\sin(\varphi_\mathrm{R}) & -\frac{r}{2}\left(\sin(\varphi_\mathrm{R}) + \cos(\varphi_\mathrm{R})\right) & \frac{r}{2}\cos(\varphi_\mathrm{R}) \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{s} \\ &=: \boldsymbol{G}\boldsymbol{s} =: \boldsymbol{g}_1\, s_1 + \boldsymbol{g}_2\, s_2 + \boldsymbol{g}_3\, s_3 \end{aligned} \tag{4.13}$$

with $\varphi_\mathrm{R} = \varphi_\mathrm{R}(\boldsymbol{q})$ being a function of the generalized coordinates $\boldsymbol{q}$, obtained by integrating Equation (4.11), and $\boldsymbol{g}_1$, $\boldsymbol{g}_2$, $\boldsymbol{g}_3$ being the columns of the matrix $\boldsymbol{G}$. Essentially, showing the non-holonomic character of the constraints comes down to calculating the so-called Lie brackets

$$[\boldsymbol{g}_i, \boldsymbol{g}_k] = \frac{\partial \boldsymbol{g}_k}{\partial \boldsymbol{q}}\, \boldsymbol{g}_i - \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{q}}\, \boldsymbol{g}_k \quad \forall\, i \neq k,\ i,k \in \{1,\ldots,3\}. \tag{4.14}$$

If non-zero, a Lie bracket corresponds to a motion consisting of the ordered concatenation of infinitesimal, non-commuting motions in the directions of $\boldsymbol{g}_i$, $\boldsymbol{g}_k$, $-\boldsymbol{g}_i$, $-\boldsymbol{g}_k$, see [Woernle16]. Hence, if the direction of the resulting net motion obtained through the Lie bracket is linearly independent from the columns in $\boldsymbol{G}$, the system can reach generalized coordinates that do not lie in the direction of its admissible velocities. Consequently, in this case, one of its kinematic constraints on velocity level is non-integrable and, therefore, non-holonomic. For instance, performing the calculation for the kinematic relationship from Equation (4.13) yields

$$\hat{\boldsymbol{g}}_1 \coloneqq [\boldsymbol{g}_1, \boldsymbol{g}_2] = \begin{bmatrix} -\frac{\sqrt{2}\,r^2}{4\,R}\sin\left(\varphi_\mathrm{R} + \frac{\pi}{4}\right) & \frac{\sqrt{2}\,r^2}{4\,R}\cos\left(\varphi_\mathrm{R} + \frac{\pi}{4}\right) & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}, \tag{4.15}$$

$$\hat{\boldsymbol{g}}_2 \coloneqq [\boldsymbol{g}_1, \boldsymbol{g}_3] = \begin{bmatrix} \frac{\sqrt{2}\,r^2}{4\,R}\cos\left(\varphi_\mathrm{R} + \frac{\pi}{4}\right) & \frac{\sqrt{2}\,r^2}{4\,R}\sin\left(\varphi_\mathrm{R} + \frac{\pi}{4}\right) & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}. \tag{4.16}$$

Indeed, whereas the matrix $\boldsymbol{G}$ is of rank 3, the matrix $\hat{\boldsymbol{G}} = \begin{bmatrix} \boldsymbol{g}_1 & \boldsymbol{g}_2 & \boldsymbol{g}_3 & \hat{\boldsymbol{g}}_1 & \hat{\boldsymbol{g}}_2 \end{bmatrix}$, which is composed of the columns of $\boldsymbol{G}$ together with $\hat{\boldsymbol{g}}_1$ and $\hat{\boldsymbol{g}}_2$, is of rank 5. Hence, it has full rank. Therefore, following the theory from [MurraySastry93], the robot dynamics is indeed subject to two non-holonomic constraints and the first-order system (4.13) is controllable.

The fact that non-holonomic constraints are present does have an influence on the choice of suitable approaches to derive a kinetic model of the robot. For instance, Lagrange's equations of the second kind can lead to a model delivering wrong results in general operating conditions [ZeidisZimmermann19]. Nevertheless, seeing the robot as a multi-body system, it is straightforwardly possible to calculate the Newton and Euler equations for each isolated body following Equations (2.3) and (2.4), expressing each kinematic quantity in terms of the generalized velocities $\boldsymbol{s}$, and eliminating the reaction forces and moments, for which it can be helpful to use the fact that they are orthogonal to the free movement directions. For the usage of the model in later chapters, it is necessary that

the simulated robot can exert forces on objects to be transported, i.e., it shall be able to push objects. To that end, subsequently, friction forces between the robot and objects are neglected and, hence, it is assumed that contact forces between the robot and objects act orthogonally on the circumference of the robot chassis. Notation-wise, the contact forces acting on the robot are subsumed in the vector $\boldsymbol{f}_{\mathrm{R}}^{\mathrm{a}} = \begin{bmatrix} f_{\mathrm{R}x} & f_{\mathrm{R}y} & 0 \end{bmatrix}^{\mathsf{T}}$, which may depend on the generalized coordinates and velocities. Expressing the latter in the body-fixed coordinate frame of the robot yields

$$\begin{aligned} {}^{\mathrm{R}}\boldsymbol{f}_{\mathrm{R}}^{\mathrm{a}} &= \begin{bmatrix} {}^{\mathrm{R}}f_{\mathrm{R}x} & {}^{\mathrm{R}}f_{\mathrm{R}y} & 0 \end{bmatrix}^{\mathsf{T}} \\ &= \begin{bmatrix} \cos(\varphi_{\mathrm{R}})\, f_{\mathrm{R}x} + \sin(\varphi_{\mathrm{R}})\, f_{\mathrm{R}y} & -\sin(\varphi_{\mathrm{R}})\, f_{\mathrm{R}x} + \cos(\varphi_{\mathrm{R}})\, f_{\mathrm{R}y} & 0 \end{bmatrix}^{\mathsf{T}}. \end{aligned} \tag{4.17}$$

Furthermore, the motor moments propelling the robot are denoted as M_i, $i \in \{1, \ldots, 4\}$, rotating wheel i about the y_i-axis in the mathematically positive direction, cf. Figure 4.5. With this, the equations of motion describing the evolution of the generalized velocities take the form

$$\begin{aligned} \ddot{\varphi}_1 = {} & c_1^{\mathrm{R}} (\dot{\varphi}_1 + \dot{\varphi}_3)(-\dot{\varphi}_1 + 2\dot{\varphi}_2 - \dot{\varphi}_3) + \left(c_2^{\mathrm{R}} + c_3^{\mathrm{R}}\right) M_1 + c_3^{\mathrm{R}} (M_2 + M_4) \\ & + \left(c_3^{\mathrm{R}} - c_2^{\mathrm{R}}\right) M_3 + c_4^{\mathrm{R}} \left({}^{\mathrm{R}}f_{\mathrm{R}x} - {}^{\mathrm{R}}f_{\mathrm{R}y}\right), \end{aligned} \tag{4.18}$$

$$\begin{aligned} \ddot{\varphi}_2 = {} & c_1^{\mathrm{R}} (\dot{\varphi}_1 + \dot{\varphi}_3)(\dot{\varphi}_3 - \dot{\varphi}_1) + \left(c_2^{\mathrm{R}} + c_3^{\mathrm{R}}\right) M_2 + c_3^{\mathrm{R}} (M_1 + M_3) \\ & + \left(c_3^{\mathrm{R}} - c_2^{\mathrm{R}}\right) M_4 - c_4^{\mathrm{R}} \left({}^{\mathrm{R}}f_{\mathrm{R}x} + {}^{\mathrm{R}}f_{\mathrm{R}y}\right), \end{aligned} \tag{4.19}$$

$$\begin{aligned} \ddot{\varphi}_3 = {} & c_1^{\mathrm{R}} (\dot{\varphi}_1 + \dot{\varphi}_3)(\dot{\varphi}_1 - 2\dot{\varphi}_2 + \dot{\varphi}_3) + \left(c_2^{\mathrm{R}} + c_3^{\mathrm{R}}\right) M_3 + c_3^{\mathrm{R}} (M_2 + M_4) \\ & + \left(c_3^{\mathrm{R}} - c_2^{\mathrm{R}}\right) M_1 + c_4^{\mathrm{R}} \left({}^{\mathrm{R}}f_{\mathrm{R}y} - {}^{\mathrm{R}}f_{\mathrm{R}x}\right) \end{aligned} \tag{4.20}$$

with positive constants c_1^{R}, c_2^{R}, c_3^{R}, and c_4^{R}, which depend on geometric and material properties of the robot. Apart from geometric quantities already introduced, they depend on the mass of the robot chassis m_{c}, the mass of an individual wheel m_{w}, the moment of inertia J_{c} of the chassis with respect to the z_{R}-axis, the principal moment of inertia J_{w1} of an individual wheel with respect to the y_i-axis of wheel i, and the moment of inertia J_{w2} of an individual wheel with respect to its other two principal axes. The four wheels are assumed to have the same geometric and material properties. The constants in the equations of motion evaluate to

$$c_1^{\mathrm{R}} = \frac{m_{\mathrm{t}}\, r^3}{16\, R \left(\frac{m_{\mathrm{t}} r^2}{4} + J_{\mathrm{w1}}\right)}, \qquad c_2^{\mathrm{R}} = \frac{1}{2\left(\frac{m_{\mathrm{t}}\, r^2}{4} + J_{\mathrm{w1}}\right)}, \tag{4.21}$$

$$c_3^{\mathrm{R}} = \frac{1}{4\left(\frac{J_{\mathrm{t}}\, r^2}{4\, R^2} + J_{\mathrm{w1}}\right)}, \qquad c_4^{\mathrm{R}} = \frac{c_2^{\mathrm{R}}\, r}{2} \tag{4.22}$$

with the complete robot's mass and relevant principal moment of inertia $m_{\mathrm{t}} = m_{\mathrm{c}} + 4\, m_{\mathrm{w}}$ and $J_{\mathrm{t}} = J_{\mathrm{c}} + 4\, J_{\mathrm{w2}} + 4\, m_{\mathrm{w}}\, R^2$, respectively.

To simulate the robot's motion, given appropriate initial conditions, Equation (4.11) needs to be integrated alongside with the Equations (4.18)-(4.20), at least in the case of a non-vanishing contact force $\boldsymbol{f}_{\mathrm{R}}^{\mathrm{a}} \neq \boldsymbol{0}$. Similarly, since the evolution of the position of the robot's

center of mass is usually a quantity of particular interest and also necessary to determine potential contact forces, it makes sense to additionally integrate the first two lines of Equation (4.13) alongside with the other equations. In any case, Equations (4.18)-(4.20) reveal that the dynamics of the wheels' angular velocities are heavily coupled. It will be interesting to see what this means for the desired high-level of maneuverability. To that end, the subsequent section presents a brief simulative analysis revealing the key dynamic properties of the chosen robot design with its unusual wheel setup.

4.3.3 Simulative Analysis

The striking property of the introduced robot's dynamics is that, indeed, for specific settings of the motor moments, it lets the robot perform a pure translatory motion into any direction, yielding the desired omnidirectional behavior. This is the required property most crucial for the object transportation benchmark case of this thesis since it means that the robot can exert pushing forces into any direction, independent of the robot's current orientation. Hence, the robot can contribute to the transportation of objects without reorienting itself even if its orientation is arbitrary. Pure translation can be achieved for motor moments that satisfy $M_3 = -M_1$ and $M_4 = -M_2$. The two plots on the left-hand side of Figure 4.6 depict a simulation result for this setting, with the constant motor moments $M_1 = -1 \cdot 10^{-3}\,\mathrm{N\,m}$ and $M_2 = -2 \cdot 10^{-3}\,\mathrm{N\,m}$ and the robot consequentially accelerating along a perfect straight line. For these and the following simulations, the relevant parameters are set to $m_\mathrm{c} = 2\,\mathrm{kg}$, $m_\mathrm{r} = 0.075\,\mathrm{kg}$, $R = 0.115\,\mathrm{m}$, $r = 0.03\,\mathrm{m}$, $J_\mathrm{c} = 0.021\,\mathrm{kg\,m^2}$, $J_\mathrm{w1} = 3.375 \cdot 10^{-5}\,\mathrm{kg\,m^2}$, and $J_\mathrm{w2} = 2.25 \cdot 10^{-5}\,\mathrm{kg\,m^2}$. In the examined simulations, the robot starts with its initial conditions set to zero, i.e., with $\boldsymbol{q}(0) = \mathbf{0}$ and $\varphi_\mathrm{R}(0) = 0$, with the chassis-fixed frame $\mathcal{K}_\mathrm{R}$ and the inertial frame of reference $\mathcal{K}_\mathrm{I}$ having the same orientation for $\varphi_\mathrm{R} = 0$. Similar to the pure translatory motion, but less crucial for this thesis, a pure rotatory motion about its center of mass can be executed if all motor moments are set to the same values. This can be seen on the right-hand side of Figure 4.6, which uses the settings $M_i = -1 \cdot 10^{-3}\,\mathrm{N\,m}$, $i \in \{1, \ldots, 4\}$.

However, for general motor moments, very arcane trajectories can be observed, with a visible oscillatory behavior, which is also observed for a traditional wheel layout [ZeidisZimmermann19]. An example can be seen in Figure 4.7 in which the constant motor moments $M_1 = -1 \cdot 10^{-3}\,\mathrm{N\,m}$, $M_2 = -2 \cdot 10^{-3}\,\mathrm{N\,m}$, $M_3 = -3\,M_1$, and $M_4 = -2\,M_2$ are applied. Hence, the very well-behaved impression of the dynamics gained from Figure 4.6 should not be taken for granted. This observation may raise concerns since, usually, motor moments are not set or controlled directly on real-world hardware robots. It is hard to measure or estimate the current motor moments in practice, whereas it is comparatively easy to measure or estimate the wheels' angular velocities. Therefore, it is usually the wheels' angular velocities which are controlled to reach desired values, which, by means of the kinematic relationships from Equations (4.5)-(4.8), correspond to the desired transla-

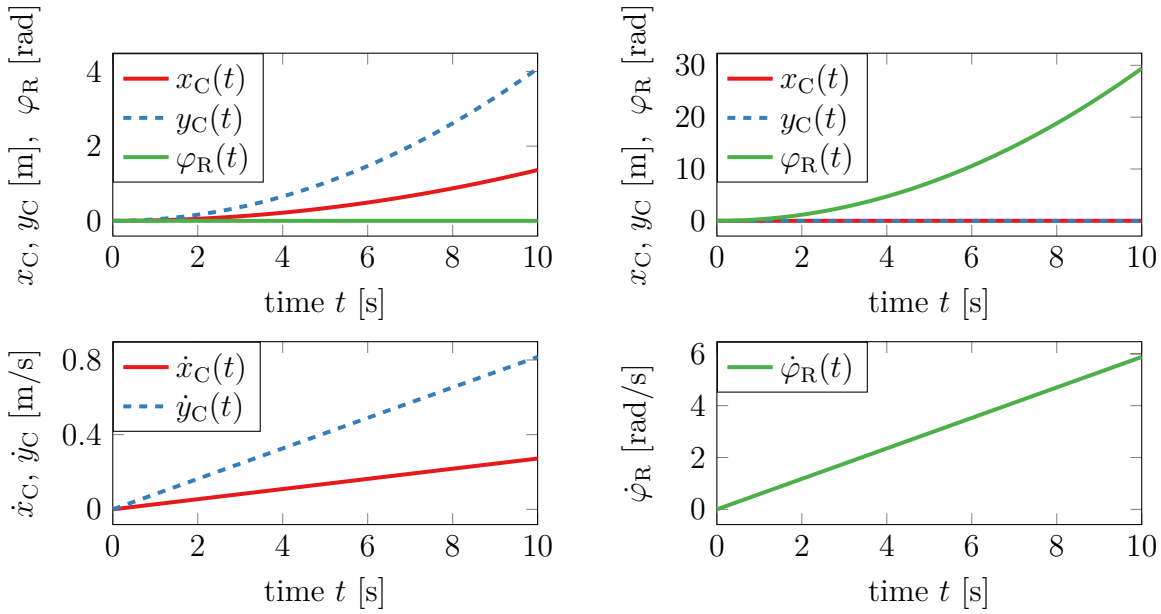


Figure 4.6: Simulation results of a pure translatory motion ($M_3 = -M_1$, $M_4 = -M_2$), depicted on the left-hand side, and of a pure rotatory motion ($M_1 = \cdots = M_4$), depicted on the right-hand side

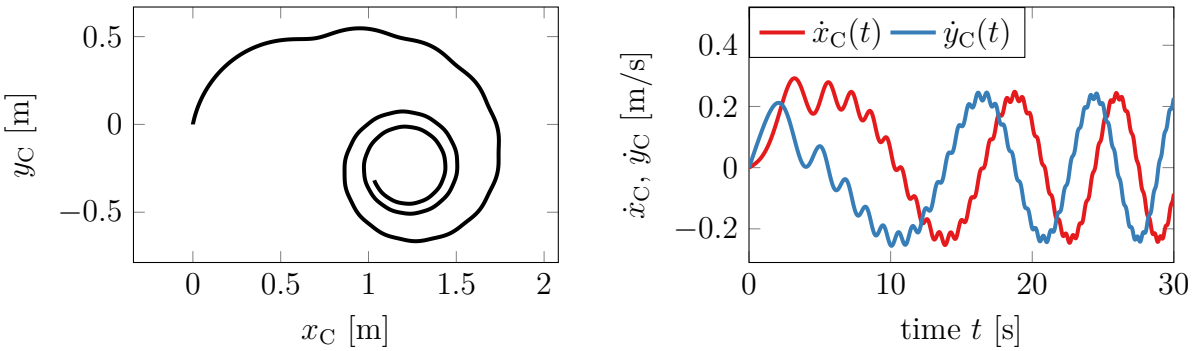


Figure 4.7: Simulation result for more arbitrary values of the motors moments, not corresponding to a pure translatory or pure rotatory motion

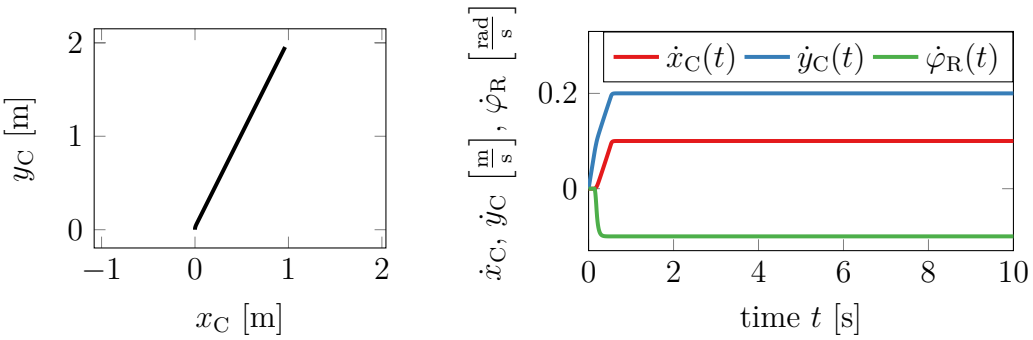


Figure 4.8: Simulation result with the angular velocities of the robot's wheels being controlled by four independent PI controllers to let the robot reach desired translational and angular velocities, designed to show the robot's directional stability

tional and angular velocities of the robot. Often, the low-level motor controllers governing the angular velocities use control approaches that are computationally lightweight to be executed at a high sampling rate on computationally limited but real-time capable microprocessors. Hence, it is a common approach to perceive each wheel as a single-input single-output system and to control its angular velocity, e.g., with a PID controller to reach the desired angular velocity. For this reason, a final simulation setup in this section considers a setup exactly as described, with four PI controllers, one for each wheel, and the desired angular velocities of the wheels being calculated using the corresponding kinematic relationships. As in a real-world robot, the motor moments are subject to constraints and are saturated at an absolute value of 0.01 N m, with the PI controllers using anti wind-up measures to prevent integration wind-up. The motor controllers are implemented in discrete time with a sampling time of 0.01 s and the proportional and integral gains are set to $k_p = 20 \cdot 10^{-3} \frac{\mathrm{N\,m\,s}}{\mathrm{rad}}$ and $k_i = 1 \cdot 10^{-3} \frac{\mathrm{N\,m}}{\mathrm{rad}}$, respectively. An exemplary, representative simulation result intended to show the robot's directional stability is displayed in Figure 4.8, with the desired velocities $\dot{x}_{C,d} = 0.1\,\mathrm{m/s}$, $\dot{y}_{C,d} = 0.2\,\mathrm{m/s}$, and $\dot{\varphi}_{R,d} = -0.1\,\mathrm{rad/s}$, respectively. As the results show, even when not directly setting the motor moments, this control structure lets the robot swiftly attain the desired translational and angular velocities, without a potential undesired, oscillatory behavior being visible. This also holds true if, in a more practical setting, the robot is tracking changing velocity setpoints under the influence of measurement noise, as the simulation result from Figure 4.9 shows. To obtain this result, the desired velocities are set to $\dot{x}_{C,d}(t) = a_1\,\omega_t \cos(\omega_t t)$, $\dot{y}_{C,d} = 2\,a_2\,\omega_t \cos(2\,\omega_t t)$, $\dot{\varphi}_{R,d} = 0\,\mathrm{rad/s}$ with the parameters contained therein set to $a_1 = 1.5\,\mathrm{m}$, $a_2 = 0.5\,\mathrm{m}$, and $\omega_t = 0.1\,\mathrm{rad/s}$. The simulated measurements of the four wheels' angular velocities are subjected to independent, identically distributed additive disturbances sampled from a zero-mean Gaussian distribution with a standard deviation of 0.2 rad/s. The plot shows the undisturbed simulated measurements, with the noise entering the control loop through the controllers, which use the noisy measurement signals. Even without any filtering of

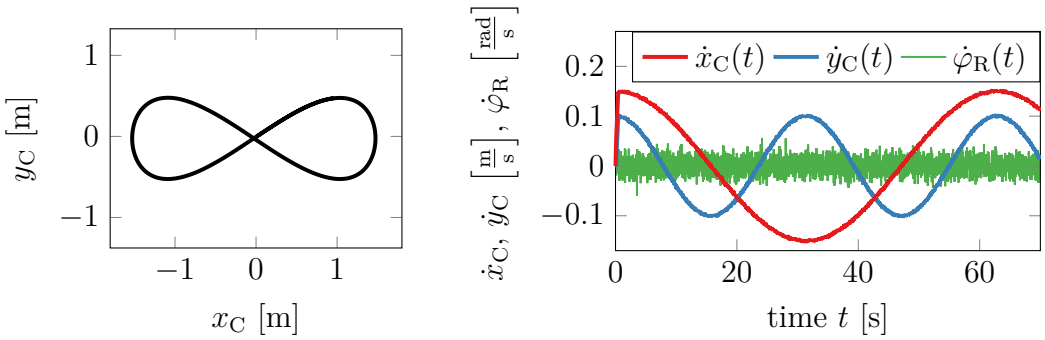


Figure 4.9: Simulation result with the angular velocities of the robot's wheels being controlled by four independent PI controllers to let the robot track a changing velocity setpoint, designed to result in a lemniscate-shaped trajectory

the measurement signals, the resulting lemniscate-shaped trajectory looks as desired, with only moderate noise being visible in the robot's translational velocities.

These very satisfactory results suggest that it is possible to use a cascade structure to control the robot's behavior. Since the low-level motor controllers act at a higher sampling rate and can swiftly and accurately control the robot's velocity within the relevant velocity range, outer, slower control loops can directly set desired velocities for the inner motor controllers, assuming that they are reached rapidly by the robot. This will motivate and allow the usage of simple single-integrator models in overarching, distributed control schemes that run at a lower sampling time than the motor controllers. In particular, the overarching controllers can therefore be based on a model that is not subject to any non-holonomic constraints.

Nevertheless, it is worth to recall that many effects have been neglected in the derivation of the mechanical model in Section 4.3.2, especially since most of these would be hard to parameterize accordingly, confounding the judgment of the fundamental movement capabilities of the robot resulting mostly from its kinematic setup. However, the hardware experiments shown in Chapters 5 and 6 will confirm that the real-world robot behaves as desired when acting within an overarching closed loop. Building a more detailed model and identifying corresponding parameters from experiments may still be a worthwhile subject of research. This is because, e.g., all effects related to friction are practically impossible to capture accurately in a model purely based on first principles, without using real-world measurement data. In that regard, the robot may serve as an interesting benchmark case for model identification and learning methods. A more detailed, data-enhanced model may then prove valuable in use cases where the robot is employed in an open-loop fashion, without using additional measurements that would allow to infer the pose of the robot. As a matter of fact, although beyond the scope of this thesis, the custom mobile robot has proved challenging and, thus, valuable for this purpose in

another research project [EschmannEbelEberhard21]. Similarly, it has already been used successfully in student projects that have employed the custom robot for purposes very different from those of this thesis [Pabst19, Wahren20], highlighting that, indeed, the design goals concerning the robot's versatility and ease of operation have been met. With the software structure and hardware design introduced and discussed in this chapter, the stage is now set to develop very concrete control and organization schemes to, in the end, successfully solve the challenging benchmark case of cooperative object transportation.

Chapter 5

Distributed Control and Organization for Cooperative Robotic Behavior

The methodological contributions of this thesis are driven by the challenges posed by employing a group of cooperating robots for a demanding transportation task. Nevertheless, most of the methods to be devised in this chapter can be of great use in a wider variety of tasks. Still, it is much easier to understand their purpose and significance at the example of cooperative transportation. Recalling the introduction of the thesis, this thesis opts for studying a non-prehensile transportation scenario in which robots push the object to be transported. Characteristic to the approach of this thesis, the task is interpreted as a special formation control task, allowing to leverage the full breadth of distributed control theory. In consequence, fitting to the architecture introduced in Chapter 4, the task solution is split into two main parts – an organization scheme, determining formations useful in the current situation, and a distributed control scheme governing the coordinated motion of the formation to translate and rotate the object as desired. The approach builds upon [EbelEberhard18, EbelEberhard19, EbelEtAl21]. The subtasks induced by the fundamental concept serve as a methodological road map for the research undertaken in this chapter and as an intuitive model example wherever more practical insight is immediately necessary to fully grasp the proposed methods. Hence, taking a look at the semantic structure of the task solution approach by means of Figure 5.1 gives a glimpse of what is to come in this chapter. Of course, being focused on semantics, the illustration does not reflect how the subtasks are actually executed. Instead, they are designed to fit into the distributed software architecture introduced in Chapter 4 and will be executed that way in the next chapter. Following Figure 5.1, the robots first need to agree on a configuration, or formation, around the object's edges that is useful to manipulate the object's pose as necessary. This step is referred to as formation synthesis from now on.

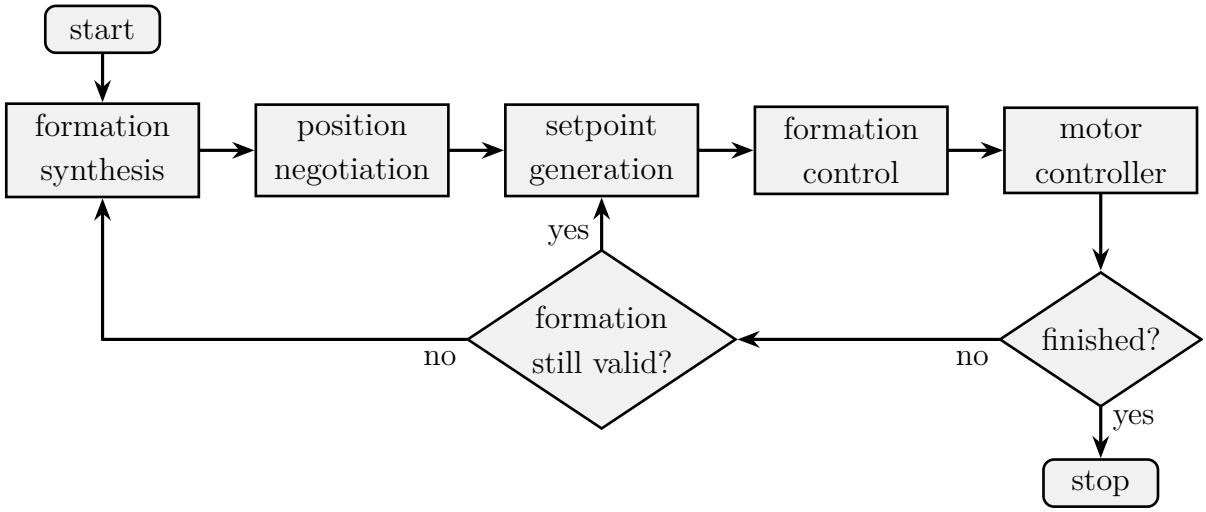


Figure 5.1: Semantic structure of the proposed solution approach to cooperative transportation, following [EbelEtAl21]

Later, the formation synthesis only needs to be re-executed when the current formation is not suitable anymore due to changing circumstances like, e.g., an altered transportation direction of the object. Hence, an open question that needs answers in this chapter is how to determine when exactly it is the case that a new formation needs to be synthesized. Having found a favorable configuration, the robots also need to negotiate self-reliantly which robot takes which position since there is no centralized entity telling each robot which position to pick, which would make it considerably easier to avoid conflicts. Based on an agreed-upon positioning of the robots around the object, and assuming that the robots know the path the object shall be transported along, suitable setpoints or reference values have to be calculated for the distributed formation controllers in order to make the object follow the desired path. Finally, the individual motor controllers of the robots close the loop.

Aiming to tackle all of these tasks, Section 5.1 starts the chapter with a detailed look at formation control, first and foremost proposing an unconventional setup based on distributed model predictive control. A setup based on more traditional graph-algebraic considerations is introduced afterwards, mimicking the DMPC controller's properties as closely as possible. This allows a simulative and experimental comparison between the two. Of course, the proposed controller setups are useful far beyond cooperative object transportation. The treatment of formation control in this chapter is closely based on the work published in [EbelEberhard21], which focuses solely on formation control without the context of its role in an overarching task, highlighting the relevance of the findings on their own account. Section 5.2 continues with organizational matters, in particular formation synthesis. Building upon and following [EbelEtAl21, EbelEberhard19, EbelEberhard18], the formation synthesis problem is formulated in terms of an optimization problem, with

the goal to obtain a scenario-agnostic scheme that can accommodate almost any polygonal object to be transported, including non-convex ones. As hinted at in the introduction, this greatly distinguishes the approach from prior approaches in the literature. The properties of the optimization-based problem formulation inspire a novel, distributed version of an augmented Lagrangian particle swarm optimizer, which, again, may be a tool universally useful, including other organizational tasks in distributed robotics and different fields of application. The section is concluded with the treatment of the position negotiation, which, by its nature, is a bit more specific in its applicability to the object transportation scenario, although it may be possible to draw some generalizing conclusions from it. Finally, Section 5.3 deals with navigation, i.e., the mapping of the workspace and the planning of paths through it, both for the individual navigation of each robot and for the global planning problem of finding a path for a general, non-convex polygonal object. The latter problem, therefore, even furnishes an extension of what is seen in Figure 5.1.

5.1 Formation Control

Formation control has already been briefly touched upon in the introduction to graph-algebraic control in Section 3.3, defining it as a group of mobile robots coordinating their relative positions to one another or, in a broader sense, as a group of systems coordinating their states relative to one another. While the methods described subsequently may generalize to the latter, it is the literal, robotic sense that is relevant to the thesis. Nevertheless, the literal definition still leaves much room for a wide range of formation control tasks and setups. An example is the question of whether the robots shall move as a whole while staging in the formation. If so, it needs to be distinguished whether they shall move the formation to a specific position, which is referred to as position control subsequently, or with a specific velocity, which is called velocity control from now on.

In the position-control case, the question arises which position within the formation shall be used for tracking. It can be practical to define a desired absolute position for one of the robots while the remaining robots merely follow this leading robot in formation. Alternatively, a so-called virtual leader may be used. It corresponds to a relative position within the formation that is not occupied by any physical robot. This choice also influences the communication structures that can be used. For instance, when using a physical leading robot, a chain-like communication graph, as it appeared earlier in Figure 3.2, can be employed.

As these considerations show, a treatise of formation control needs a precise problem definition and needs to focus on a specific class of problem setups if precise conclusions shall be drawn. In consequence, for the purposes of this thesis, the specifications of the formation control task must be distilled from the requirements of its usage in the transportation task. Controlling the formation's absolute position may be suitable for highly precise positioning

of the formation and, thereby, of the object transported alongside it. The velocity-control setup may, e.g., be useful for transportation over longer distances at prescribed velocities. Hence, both tasks will be considered subsequently. In the position-control case, it is especially important to allow the independent adjustment of the accuracies of controlling the robots' relative positioning and of the absolute positioning of the formation as a whole. This rules out the simple setup of a leading robot that is followed by the other robots using a sparse, e.g., chain-like communication structure. In fact, since there is no apparent reason why specific robots shall be treated differently in the transportation process, an undiscriminating communication structure seems appropriate. Therefore, in the position-control case, the formation controllers proposed in this thesis control the position ${}_{\mathrm{c}}\boldsymbol{x} \in \mathbb{R}^2$ of the formation's geometric center to the desired position ${}_{\mathrm{c}}\boldsymbol{x}_{\mathrm{d}}(t) \in \mathbb{R}^2$, with only planar motions being studied. At the same time, the robots shall maintain a given positioning relative to the formation center. In the velocity-control case, merely the robot positions relative to the formation center are controlled, while the whole formation shall move with the velocity ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}(t)$.

By definition, the geometric center is a linear combination of the robots' positions and, therefore, each formation controller needs the location of every other robot. Hence, the communication graph must be complete, with each robot exchanging data with every other robot. While this seems limiting at a first glance, the software architecture introduced in Chapter 4 lets all robots operate on the same communication network anyway, relying on multicast communication to exchange data, which is well-suited to these communication requirements. Furthermore, without changing the formation control methods themselves, it is possible to define multiple sub-teams of robots, with each formation controller only considering data from its sub-team, thereby limiting the burden posed for the communication infrastructure.

Apart from the requirements concerning communication, formation control also brings about some more subtle requirements regarding common reference frames. If the formation center's absolute position is controlled, the robots need a common inertial frame of reference $\mathcal{K}_{\mathrm{I}}$ in which the desired position is defined, whereas a non-inertial frame, moving with the formation, suffices if merely the robots' relative positioning is of interest. In any case, a common sense of direction, which is brought about automatically if the robots have a common reference frame they agree on, is necessary if the orientation of the formation shall be controlled. If objects of general shapes shall be transported efficiently through an environment, it is often necessary to rotate them along with the formation. For this reason, in the following, it is required that the robots share a common frame of reference. Finally, each robot shall have a unique identification number that it transmits along with all information published on the network, allowing the robots to associate and order incoming information reliably.

With the described requirements in mind, some notation needs to be introduced to

formulate the problem and the controllers in a mathematical manner. As it is the case later for the transportation process, robots may join and leave the formation. Hence, out of N robots present, only a time-variant subset, consisting of N_a robots and defined by the index set $\mathcal{R}_\mathrm{a} \coloneqq \left\{i_1, \ldots, i_{N_\mathrm{a}(t)}\right\} \subseteq \mathcal{R} \coloneqq \{1, \ldots, N\}$, is actively participating in the task solution at time $t \geq 0$. Therein, the indices are ordered in ascending order, i.e., $i_1 < i_2 < \cdots < i_{N_\mathrm{a}(t)}$. The position of robot $i \in \{1, \ldots, N\}$ at time t is denoted by ${}_i\boldsymbol{x}(t)$. Henceforth, the formation's geometric center is given by ${}_\mathrm{c}\boldsymbol{x}(t) = \sum_{i \in \mathcal{R}_\mathrm{a}(t)} {}_i\boldsymbol{x}(t)/N_\mathrm{a}(t)$. Subsequently, the notation introduced in Section 2.1 will prove handy to indicate in which reference frames and coordinates a quantity is given. As usual, quantities without any specific notational indication are given in the coordinates of and relative to the inertial frame of reference. The reference frame $\mathcal{K}_\mathrm{F}(t)$ shall move with the formation, with its origin moving with the formation's geometric center. The desired formation shape $\mathcal{F}_\mathrm{d}(t) \coloneqq \left\{ {}^{\check{\mathrm{F}}}_{i}\boldsymbol{x}_\mathrm{d}(t) \mid i \in \mathcal{R}_\mathrm{a}(t) \right\}$ is defined by the desired positions ${}^{\check{\mathrm{F}}}_{i}\boldsymbol{x}_\mathrm{d}(t)$ of the robots relative to the formation's geometric center in the coordinates of the moving frame $\mathcal{K}_\mathrm{F}(t)$. Hence, by rotating $\mathcal{K}_\mathrm{F}(t)$ relative to $\mathcal{K}_\mathrm{I}$, also the desired formation shape is rotated in the plane. By definition, it must hold that $\sum_{i \in \mathcal{R}_\mathrm{a}} {}^{\check{\mathrm{F}}}_{i}\boldsymbol{x}_\mathrm{d}(t) = \boldsymbol{0}$. As long as it does not give rise to ambiguities, the time-dependency of time-variant quantities may subsequently be omitted in the notation to the benefit of brevity. Motivated by the findings from Section 4.3 with regard to the movement capabilities of the considered robot design, for the synthesis of the distributed formation controller, the robots are modeled as single integrators, with robot i's dynamics being given by ${}_i\dot{\boldsymbol{x}} = {}_i\boldsymbol{u}$, with the control input ${}_i\boldsymbol{u} \in \mathbb{R}^2$ being the robot's velocity. The velocity values in the two directions shall not exceed u_max in absolute value. For simulation purposes, the model introduced in Section 4.3.2 is utilized, resulting in a cascaded control structure with the local PI controllers controlling the angular velocities of the wheels. As in Section 3.3, the stacked state and input vectors $\boldsymbol{x} \coloneqq \begin{bmatrix} {}_{i_1}\boldsymbol{x}^\mathsf{T} & \cdots & {}_{i_{N_\mathrm{a}}}\boldsymbol{x}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n_x}$ and $\boldsymbol{u} \coloneqq \begin{bmatrix} {}_{i_1}\boldsymbol{u}^\mathsf{T} & \cdots & {}_{i_{N_\mathrm{a}}}\boldsymbol{u}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n_u}$ are useful for controller derivation and analysis. For the chosen nominal dynamics, the dimensions of the vectors evaluate to $n_x = n_u = 2\,N_\mathrm{a}$. Furthermore, it will be useful to refer to sub-blocks of matrices. To that end, row i and column j of a matrix $\boldsymbol{V}$ shall be referred to by $\boldsymbol{V}_{i,:}$ and $\boldsymbol{V}_{:,j}$, respectively. Similarly, $\boldsymbol{V}_{i_1:i_2,:}$ with $i_1 \leq i_2$ shall be the matrix block including rows i_1 to i_2, whereas $\boldsymbol{V}_{:,j_1:j_2}$ with $j_1 \leq j_2$ shall denote the block including the columns j_1 to j_2. In the subsequent controller derivations, it is assumed that the controller of robot $\hat{\imath} \in \mathcal{R}_\mathrm{a}$ is derived. In the first step, the DMPC-based controller is looked at.

5.1.1 DMPC-Based Formation Controller

The proposed DMPC controller is formulated in a discrete-time setting, employing the control sampling time T_s, yielding the discrete-time dynamics $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + T_\mathrm{s}\boldsymbol{u}(k) =: \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, $k \in \mathbb{N}_0$. It is worth emphasizing that, in general, the DMPC-based approach is not limited to this dynamics since different linear dynamics can be inserted

straightforwardly, with care mostly being required with regard to the constraints. The focus on this dynamics is rather due to the overall control setup and type of robot considered. Since, due to the distributed setup, robot $\hat{\imath}$ only decides on its own control input ${}_{\hat{\imath}}\boldsymbol{u}$, the dynamics is reformulated to

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \underbrace{\boldsymbol{B}_{:,2\hat{\imath}-1:2\hat{\imath}}}_{=:\boldsymbol{B}_\mathrm{e}} {}_{\hat{\imath}}\boldsymbol{u}(k) + \underbrace{\begin{bmatrix}\boldsymbol{B}_{:,1:(2\hat{\imath}-2)} & \boldsymbol{B}_{:,2\hat{\imath}+1:(2N_\mathrm{a})}\end{bmatrix}}_{=:\boldsymbol{B}_\mathrm{p}} \underbrace{\begin{bmatrix} {}_{i_1}\boldsymbol{u}(k) \\ \vdots \\ {}_{\hat{\imath}-1}\boldsymbol{u}(k) \\ {}_{\hat{\imath}+1}\boldsymbol{u}(k) \\ \vdots \\ {}_{i_{N_\mathrm{a}}}\boldsymbol{u}(k) \end{bmatrix}}_{=:\boldsymbol{u}_\mathrm{p}(k)} \tag{5.1}$$

$$\Rightarrow \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}_\mathrm{e}\, {}_{\hat{\imath}}\boldsymbol{u}(k) + \boldsymbol{B}_\mathrm{p}\boldsymbol{u}_\mathrm{p}(k). \tag{5.2}$$

The first task in designing the DMPC optimization problem is to develop a suitable cost function, associating a higher cost with larger deviations from the formation shape. To this end, it is useful to describe the formation through an output $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \in \mathbb{R}^{n_y}$, with the output matrix $\boldsymbol{C}$ chosen differently depending on whether the position-control or velocity-control case is considered. In the latter case, the DMPC controller is merely occupied with attaining and maintaining the robots' relative positions, motivating to choose the output matrix to

$$\boldsymbol{C}_\mathrm{v} = \begin{bmatrix} -1/N_\mathrm{a}+1 & -1/N_\mathrm{a} & \cdots & -1/N_\mathrm{a} \\ -1/N_\mathrm{a} & -1/N_\mathrm{a}+1 & \cdots & -1/N_\mathrm{a} \\ \vdots & \ddots & \ddots & \ddots \\ -1/N_\mathrm{a} & \cdots & -1/N_\mathrm{a} & -1/N_\mathrm{a}+1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2N_\mathrm{a}\times 2N_\mathrm{a}}. \tag{5.3}$$

Then, each pair of lines of the output corresponds to a robot position relative to the formation center. Naturally, with $\mathrm{rank}(\boldsymbol{C}_\mathrm{v}) = 2\,(N_\mathrm{a}-1)$, there exist infinitely many states to reach one desired output since the formation center may be located anywhere in the plane as long as the relative positioning is maintained. This fits the earlier observations from the introduction of graph-algebraic control. If the formation center's position shall also be controlled, the output matrix is instead chosen to

$$\boldsymbol{C}_\mathrm{p} = \begin{bmatrix} 1/N_\mathrm{a} & 1/N_\mathrm{a} & \cdots & 1/N_\mathrm{a} \\ -1/N_\mathrm{a}+1 & -1/N_\mathrm{a} & \cdots & -1/N_\mathrm{a} \\ -1/N_\mathrm{a} & -1/N_\mathrm{a}+1 & \cdots & -1/N_\mathrm{a} \\ \vdots & \ddots & \ddots & \ddots \\ -1/N_\mathrm{a} & \cdots & -1/N_\mathrm{a} & -1/N_\mathrm{a}+1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2(N_\mathrm{a}+1)\times 2N_\mathrm{a}}. \tag{5.4}$$

Chosen this way, the formation center's prescribed position needs to be consistent with the robots' prescribed relative positions since the location of the formation center is entirely

determined by those. Alternatively, the lines corresponding to one of the robots' relative positions can be deleted from $\boldsymbol{C}_\mathrm{p}$ to obtain an invertible matrix. However, since this would mean that one of the robots is, without constitutive reason, treated differently than the others, the above choice seems more adequate.

Now, for a desired output $\boldsymbol{y}_\mathrm{d}$ that is kept constant for analysis purposes, formation control comes down to tracking a setpoint $\boldsymbol{y}_\mathrm{d}$ with the output $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ so that $\boldsymbol{y}(t)$ converges to $\boldsymbol{y}_\mathrm{d}$ as time t approaches infinity. If the output matrix $\boldsymbol{C}$ is set to $\boldsymbol{C}_\mathrm{v}$ for the velocity-control case, the desired output is defined as

$$\boldsymbol{y}_\mathrm{d} = \begin{bmatrix} {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_1}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_2}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & \cdots & {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_{N_\mathrm{a}}}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{5.5}$$

whereas it is set to

$$\boldsymbol{y}_\mathrm{d} = \begin{bmatrix} {}_{\mathrm{c}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_1}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_2}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & \cdots & {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}\,{}_{i_{N_\mathrm{a}}}^{\check{\mathrm{F}}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{5.6}$$

for the position-control case with the output matrix $\boldsymbol{C}_\mathrm{p}$. Naturally, if the desired formation shape changes, the desired output may change abruptly from one time instance to the next. The same happens if the formation center's desired position is changed from one time instance to the next, e.g., to let the formation follow a path in the position-control case. As delineated in Section 3.2.2's introduction to distributed MPC, this can be challenging for many traditional schemes relying on stabilizing terminal ingredients due to the setpoint change potentially making the optimization problem infeasible. The candidate scheme from [FerramoscaEtAl13], which was deemed promising in Section 3.2.2, provides a possible remedy in the form of an artificial setpoint. The artificial setpoint corresponds to a feasible steady state and input pair. When employing a terminal equality constraint, this allows to construct a feasible candidate solution by discarding the first input in the previous input sequence and appending the steady-state input at the end of the input sequence. Instead of directly adding the artificial steady state ${}_{i}\tilde{\boldsymbol{x}}_\mathrm{d}$ to the optimization problem, it can be convenient to derive a parameterization $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ of the system's steady states and the corresponding outputs by evaluating

$${}_{i}\tilde{\boldsymbol{x}}_\mathrm{d}(\boldsymbol{\theta}) = \left(\boldsymbol{M}_{\theta,\mathrm{f}}\right)_{1:n_x,:}\boldsymbol{\theta} =: \boldsymbol{M}_\theta\,\boldsymbol{\theta}, \tag{5.7}$$

$${}_{i}\tilde{\boldsymbol{y}}_\mathrm{d}(\boldsymbol{\theta}) \coloneqq \boldsymbol{C}\,{}_{i}\tilde{\boldsymbol{x}}_\mathrm{d}(\boldsymbol{\theta}) \tag{5.8}$$

with

$$\boldsymbol{M}_{\theta,\mathrm{f}} = \operatorname{null}\left(\begin{bmatrix} \boldsymbol{A} - \boldsymbol{I}_{n_x} & \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} & -\boldsymbol{I}_{n_y} \end{bmatrix}\right), \tag{5.9}$$

which can be obtained directly from the definition of a steady state and input pair [LimonEtAl08]. In Equation (5.9), $\operatorname{null}(\cdot)$ gives a matrix that contains in its columns a basis of the null space of its matrix argument. The steady-state input corresponding to the steady state can be obtained by instead selecting the matrix block $\left(\boldsymbol{M}_{\theta,\mathrm{f}}\right)_{n_x+1:n_x+n_u,:}$ in an expression otherwise similar to Equation (5.7). For the considered single-integrator

dynamics and in the absence of state constraints, all states are steady states with a steady-state input of zero. Hence, the steady states can be parameterized with the state variable itself, leading to the choices $n_\theta \coloneqq 2N_\mathrm{a}$, $\boldsymbol{M}_\theta \coloneqq \boldsymbol{I}_{2N_\mathrm{a}}$. With this, the formation control goal can be encapsulated in the cost function

$$J(\boldsymbol{x}(t), \boldsymbol{y}_\mathrm{d}, \boldsymbol{u}(\cdot \,|\, t), {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d}) = \sum_{k=t}^{t+H-1} \left(\|\boldsymbol{x}(k \,|\, t) - {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}(k \,|\, t)\|_{\boldsymbol{R}}^2 \right) + \|\boldsymbol{C}\, {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d} - \boldsymbol{y}_\mathrm{d}\|_{\boldsymbol{T}}^2 , \tag{5.10}$$

where the weighting matrices $\boldsymbol{R}$ and $\boldsymbol{T}$ are symmetric and positive definite and the weighting matrix of the state is defined as $\boldsymbol{Q} \coloneqq \boldsymbol{C}^\mathsf{T}\boldsymbol{\mathcal{D}}\boldsymbol{C} \geq 0$ with the diagonal weighting matrix $\boldsymbol{\mathcal{D}} > 0$. Using this cost function, the formation control problem can be solved through the DMPC optimization problem

$$\begin{aligned}
\underset{{}_{\hat{\imath}}\boldsymbol{u}(\cdot \,|\, t),\, \boldsymbol{\theta}}{\text{minimize}} \quad & J(\boldsymbol{x}(t), \boldsymbol{y}_\mathrm{d}, \boldsymbol{u}(\cdot \,|\, t), {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d}) && (5.11)\\
\text{subject to} \quad & \boldsymbol{x}(k+1 \,|\, t) = \boldsymbol{A}\boldsymbol{x}(k \,|\, t) + \boldsymbol{B}_\mathrm{e}\, {}_{\hat{\imath}}\boldsymbol{u}(k \,|\, t) + \boldsymbol{B}_\mathrm{p}\boldsymbol{u}_\mathrm{p}(k \,|\, t), && (5.12)\\
& \|{}_{\hat{\imath}}\boldsymbol{u}(k \,|\, t)\|_\infty \leq u_\mathrm{max}, && (5.13)\\
& \|\Delta\, {}_{\hat{\imath}}\boldsymbol{u}(k \,|\, t)\|_\infty \leq \Delta u_\mathrm{max}, \quad k \in \{t, \dots, t+H-1\}, && (5.14)\\
& \|{}_{\hat{\imath}}\boldsymbol{u}(t+H-1 \,|\, t)\|_\infty \leq \Delta u_\mathrm{max}, && (5.15)\\
& \boldsymbol{x}(t+H \,|\, t) = {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d}, && (5.16)\\
& {}_{\hat{\imath}}\tilde{\boldsymbol{x}}_\mathrm{d} = \boldsymbol{M}_\theta\boldsymbol{\theta}, && (5.17)\\
& \boldsymbol{x}(t \,|\, t) = \boldsymbol{x}(t). && (5.18)
\end{aligned}$$

Different from the setup in [FerramoscaEtAl13], the optimization problem also contains the constraints (5.14) limiting the control inputs' change

$$\Delta\, {}_{\hat{\imath}}\boldsymbol{u}(k \,|\, t) \coloneqq \begin{cases} {}_{\hat{\imath}}\boldsymbol{u}(k \,|\, t) - {}_{\hat{\imath}}\boldsymbol{u}(k-1 \,|\, t) & \text{if } k > t, \\ {}_{\hat{\imath}}\boldsymbol{u}(t \,|\, t) - {}_{\hat{\imath}}\boldsymbol{u}(t-1 \,|\, t-1) & \text{if } k = t \end{cases} \tag{5.19}$$

from one time instance to the next. This makes necessary the additional constraint (5.15) to ensure recursive feasibility. Without it, the candidate solution with the steady-state input of zero at the end can be inadmissible since it may be unreachable within the admissible change of input. Beyond the applications of this thesis, in the case of more general dynamics and steady states with corresponding non-zero steady-state inputs, the deviation of the terminal control input from the steady-state input would need to be constrained instead. The proposed optimization problem refrains from using a terminal set with an accompanying terminal cost since both would need to be recalculated for changing numbers of robots, which can become computationally difficult. Therefore, the terminal equality constraint (5.16) is used instead.

In the optimization problem, it would be desirable to optimize not only over robot $\hat{\imath}$'s own control input ${}_{\hat{\imath}}\boldsymbol{u}(\cdot \,|\, t)$ but over the whole input vector $\boldsymbol{u}(\cdot \,|\, t)$. However, this is impossible

Algorithm 1 Distributed MPC scheme

1: **input:** initial feasible sequences ${}_i\boldsymbol{u}_{\mathrm{cand}}(\cdot \,|\, 0) \quad \forall\, i \in \mathcal{R}_{\mathrm{a}}$

2: **at all time steps** $t \geq 1$**:**
all robots $i \in \mathcal{R}_{\mathrm{a}}$ set $\forall\, j \in \mathcal{R}_{\mathrm{a}} \setminus \{i\}$

$$ {}_j\boldsymbol{u}_{\mathrm{cand}}(t+k \,|\, t) = \begin{cases} {}_j\boldsymbol{u}_{\mathrm{a}}(t+k \,|\, t-1) & \text{for } k \in \{0, \dots, H-2\}, \\ \boldsymbol{0} & \text{for } k = H-1 \end{cases} $$

3: **at all time steps** $t \geq 0$**:**
in parallel, all robots $i \in \mathcal{R}_{\mathrm{a}}$
solve the optimization problem (5.11)-(5.18) with
${}_j\boldsymbol{u}(k \,|\, t) := {}_j\boldsymbol{u}_{\mathrm{cand}}(k \,|\, t),\ j \in \mathcal{R}_{\mathrm{a}} \setminus \{i\}$
$\Rightarrow {}_i\boldsymbol{u}^{\star}(\cdot \,|\, t),\ \boldsymbol{\theta}^{\star}(t)$
set and publish ${}_i\boldsymbol{u}_{\mathrm{a}}(\cdot \,|\, t) := (1/N_{\mathrm{a}})\ {}_i\boldsymbol{u}^{\star}(\cdot \,|\, t) + (1 - 1/N_{\mathrm{a}})\, {}_i\boldsymbol{u}_{\mathrm{cand}}(\cdot \,|\, t)$

4: in parallel, all systems $i \in \mathcal{R}_{\mathrm{a}}$ apply ${}_i\boldsymbol{u}_{\mathrm{a}}(t \,|\, t)$

5: set $t := t + 1$ and go to step 2

in the distributed setting since the other entries are determined by the other robots concurrently. Therefore, the idea is to use previously communicated information to construct a feasible candidate solution for the unknown entries, which correspond to the other robots $j \in \mathcal{R}_{\mathrm{a}} \setminus \{\hat{\imath}\}$. As usual, it needs to be taken care that this is done in a way that maintains the desired theoretical properties, including recursive feasibility [FerramoscaEtAl13]. This line of thought leads to the computation and communication scheme given in Algorithm 1. The applied control input ${}_{\hat{\imath}}\boldsymbol{u}_{\mathrm{a}}(t \,|\, t)$ results from a convex combination of the optimal input sequence and the candidate solution, which yields ${}_{\hat{\imath}}\boldsymbol{u}_{\mathrm{a}}(\cdot \,|\, t)$. In consequence, the candidate state trajectory resulting from the candidate input ${}_{\hat{\imath}}\boldsymbol{u}_{\mathrm{cand}}(\cdot \,|\, t)$ fulfills the terminal equality constraint for a convex combination of the individual optimal artificial steady states from the previous time step, ensuring recursive feasibility also of the terminal constraint. Being an iterative distributed MPC scheme, the algorithm would allow to optimize and communicate multiple times before progressing with step 4, to the end of iteratively improving the calculated control input. However, as motivated previously, to minimize the amount of communication per time step, this is not done in this thesis. After all, beneficially, already one iteration per time step delivers a feasible result.

A crucial consideration for distributed robotics is that it is not enough to formulate the DMPC optimization problem once, potentially in an offline, preprocessing step, and then to start the system and solve the preformulated problem repeatedly until the system operation stops. In contrast, it is constitutive to distributed robotics that the control network can be reconfigured dynamically, with robots joining and leaving at any time. A practical remedy can be to preformulate the problem for all numbers of robots that

may occur, saving them in memory, and reloading the appropriate optimization problem whenever necessary. Clearly, this does not yield the total flexibility sought for. Fortunately, it is possible to explicitly calculate the multi-parametric quadratic program representation of the optimization problem so that the matrices therein can be assembled algorithmically and very swiftly, allowing to reformulate the problem during real-time operation. Due to the additional parameters for the other robots' candidate solution, the additional optimization variable, and the constraints on the control input change, the structure of the resulting multi-parametric quadratic program is a bit different and more involved than for more classic MPC problems. Hence, a brief look at the problem is worthwhile despite its technical character. To that end, it is practical to collect in the vector $\boldsymbol{U}_\mathrm{e} = \begin{bmatrix} {}_i\boldsymbol{u}^\mathsf{T}(t\,|\,t) & \cdots & {}_i\boldsymbol{u}^\mathsf{T}(t+H-1\,|\,t) & \boldsymbol{\theta}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2H+n_\theta}$ all optimization variables. Similarly, the vector $\boldsymbol{U}_\mathrm{p} = \begin{bmatrix} \boldsymbol{u}_\mathrm{p}^\mathsf{T}(t\,|\,t) & \cdots & \boldsymbol{u}_\mathrm{p}^\mathsf{T}(t+H-1\,|\,t) & \boldsymbol{u}_{-1}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2(N_\mathrm{a}-1)H+2}$ contains all control inputs entering the problem as parameters. Therein, the short-hand notation $\boldsymbol{u}_{-1} \coloneqq {}_i\boldsymbol{u}(t-1\,|\,t-1) = {}_i\boldsymbol{u}(t-1)$ is used to denote the robot's previous control input, which is necessary for the constraints on $\Delta\,{}_i\boldsymbol{u}(\cdot\,|\,t)$. Overall, the optimization problem depends on the parameters $\boldsymbol{p} \coloneqq \begin{bmatrix} \boldsymbol{x}^\mathsf{T}(t) & \boldsymbol{y}_\mathrm{d}^\mathsf{T} & \boldsymbol{U}_\mathrm{p}^\mathsf{T} \end{bmatrix}^\mathsf{T}$. Recursively inserting the dynamics into the cost function and constraints while neglecting constant terms in the cost and reformulating everything neatly in matrix-vector form, the optimization problem (5.11)-(5.18) takes the equivalent form

$$\underset{\boldsymbol{U}_\mathrm{e}}{\text{minimize}} \quad \frac{1}{2}\,\boldsymbol{U}_\mathrm{e}^\mathsf{T}\boldsymbol{H}\boldsymbol{U}_\mathrm{e} + \boldsymbol{p}^\mathsf{T}\boldsymbol{F}\boldsymbol{U}_\mathrm{e} \tag{5.20}$$

$$\text{subject to} \quad \boldsymbol{M}_\mathrm{t1}\,\boldsymbol{U}_\mathrm{e} = \boldsymbol{M}_\mathrm{t2}\,\boldsymbol{p}, \tag{5.21}$$

$$\begin{bmatrix} -\Delta u_\mathrm{max} \\ \vdots \\ -\Delta u_\mathrm{max} \end{bmatrix} \leq \boldsymbol{M}_\mathrm{d}\,\boldsymbol{U}_\mathrm{e} \leq \begin{bmatrix} \Delta u_\mathrm{max} \\ \vdots \\ \Delta u_\mathrm{max} \end{bmatrix}, \tag{5.22}$$

$$-\Delta u_\mathrm{max}\,\boldsymbol{I}_2 + \boldsymbol{u}_{-1} \leq \boldsymbol{u}(t\,|\,t) \leq \Delta u_\mathrm{max}\,\boldsymbol{I}_2 + \boldsymbol{u}_{-1}, \tag{5.23}$$

$$-\Delta u_\mathrm{max}\,\boldsymbol{I}_2 \leq \boldsymbol{u}(t+H-1\,|\,t) \leq \Delta u_\mathrm{max}\,\boldsymbol{I}_2 \tag{5.24}$$

$$\begin{bmatrix} -u_\mathrm{max} \\ \vdots \\ -u_\mathrm{max} \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{I}_{2H} & \boldsymbol{0} \end{bmatrix} \boldsymbol{U}_\mathrm{e} \leq \begin{bmatrix} u_\mathrm{max} \\ \vdots \\ u_\mathrm{max} \end{bmatrix}. \tag{5.25}$$

Therein, all as yet undefined matrices are given explicitly in Appendix A.1, with the detailed expressions not being of import for the understanding of the subsequent disquisitions. The matrices are mostly constructed in a block-wise manner from the weighting and system matrices appearing in the optimization problem and can hence be efficiently constructed algorithmically, as desired. It is beneficial for some solvers to list separately the equality constraints (5.21), the inequality constraints (5.22), and the simple bounds (5.23)-(5.25). However, they can all be written as a set of inequality constraints to obtain the multi-parametric quadratic program in the form introduced in Equations (3.13), (3.14). Hence,

executing the DMPC formation controller comes down to supplying the optimization problem (5.20)-(5.25) to a quadratic program solver of choice within the solution strategy described by Algorithm 1. If the number of participating robots changes, the optimization problem is swiftly reconstructed and the candidate solutions are reset. Therefore, the robotic network can reconfigure itself fully automatically and in real time. When the output matrix $\boldsymbol{C}_\mathrm{p}$ is used in the cost function, the resulting applied control input ${}_i\boldsymbol{u}(t) \coloneqq {}_i\boldsymbol{u}_\mathrm{a}(t \,|\, t)$ lets the robots maintain the relative positioning while moving the formation center to the desired position. The choice of the diagonal weighting matrix $\boldsymbol{\mathcal{D}}$ allows to weigh differently the errors of the formation center and of the relative positioning of the robots. In contrast, if $\boldsymbol{C}_\mathrm{v}$ is used, the robots only maintain their relative positioning. A simple solution to extend the latter to the velocity-control case is to apply the input

$$ {}_i\boldsymbol{u}(t) \coloneqq {}_i\boldsymbol{u}_\mathrm{a}(t \,|\, t) + {}_\mathrm{c}\boldsymbol{v}_\mathrm{d}. \tag{5.26} $$

However, this naive approach can result in control inputs that exceed the input constraints. A more practical but also more involved way of circumventing this issue is discussed later within Section 5.1.3 since it also affects the graph-algebraic approach to formation control. Hence, the latter is discussed first.

5.1.2 Graph-Algebraic Formation Controller

While graph-algebraic control has already been introduced for the example of formation control in Section 3.3, the usual linear approach to graph-algebraic formation control does not consider input constraints at all. As will be shown below, using a nonlinear feedback law, the nominal control inputs obtained with the graph-algebraic approach can be made to fulfill the input constraints, too, which makes it more suitable for comparisons with the DMPC approach.

In the graph-algebraic case, the position- and velocity-control cases both follow as extensions of the simplest formation control case of robots merely attaining a desired relative positioning anywhere in the plane. Therefore, the latter goal is considered first. The first step in devising the graph-algebraic formation controller consists of determining the incidence matrix of an orientation of the communication graph. The considered control setup results in a complete communication graph, as exemplarily illustrated on the left-hand side of Figure 5.2 for four robots, each represented by one node. Since the resulting controller depends on the incidence matrix and since the controller, as for the DMPC case, needs to be synthesized automatically in the case of changing numbers of participating robots, an orientation of the communication graph and the accompanying incidence matrix need to be generated automatically. In the oriented graph displayed on the right-hand side of Figure 5.2, the chosen orientation and numbering of the edges facilitate the construction of the incidence matrix. This kind of oriented graph and numbering of the edges can be constructed by considering the nodes one after another, with an increasing number.

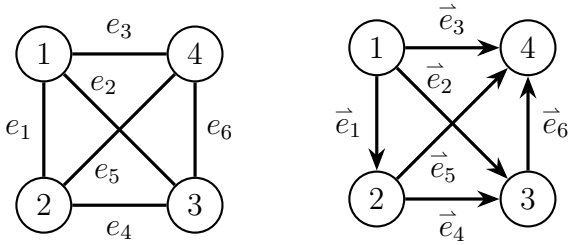


Figure 5.2: Illustration of an undirected, complete graph and of a corresponding oriented graph, with nodes $\mathcal{V} = \{1, 2, 3, 4\}$, and edges $\mathcal{E} = \{e_1, e_2, \ldots, e_6\}$ and $\vec{\mathcal{E}} = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_6\}$ [EbelEberhard21]

In each step, a directed edge is drawn to point from the current node $\hat{v}$ to every other node $v_i \neq \hat{v}$ for which there does not yet exist an edge pointing from v_i to $\hat{v}$. For N_a cooperating robots, this leads to a specific directed graph $\vec{\mathcal{G}}_{N_\mathrm{a}}$ with $n_\mathcal{E} = N_\mathrm{a}\,(N_\mathrm{a} - 1)/2$ edges. Its incidence matrix is of the form

$$\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}_{N_\mathrm{a}}} = \left[\begin{array}{c|c|c|c|c} -\mathbf{1}_{N_\mathrm{a}-1}^\mathsf{T} & \mathbf{0} & \mathbf{0} & & \\ \hline & -\mathbf{1}_{N_\mathrm{a}-2}^\mathsf{T} & & \ddots & \\ & & -\mathbf{1}_{N_\mathrm{a}-3}^\mathsf{T} & & \mathbf{0} \\ \boldsymbol{I}_{N_\mathrm{a}-1} & \boldsymbol{I}_{N_\mathrm{a}-2} & & \ddots & \\ \hline & & \boldsymbol{I}_{N_\mathrm{a}-3} & \ddots & -1 \\ \hline & & & & 1 \end{array}\right]. \tag{5.27}$$

Following what has been observed for the incidence matrix in Section 3.3, the matrix $\bar{\boldsymbol{\mathcal{B}}} = \boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}_{N_\mathrm{a}}} \otimes \boldsymbol{I}_2$ can be used to transcribe the robot positions $\boldsymbol{x}(t)$ to the stacked relative vectors between the robot positions by means of $\boldsymbol{d} = \bar{\boldsymbol{\mathcal{B}}}^\mathsf{T}\boldsymbol{x}$. However, in this thesis, the formation shape is not defined by the relative vectors between the robots but by the robots' desired positions ${}_i^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}$, $i \in \mathcal{R}_\mathrm{a}$, relative to the geometric formation center. Hence, the control goal of attaining the desired relative positioning can be given in the form

$$\lim_{t\to\infty} {}^{\check{\mathrm{F}}}\boldsymbol{x}(t) \stackrel{!}{=} {}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d} \tag{5.28}$$

with the stacked vector ${}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d} = \begin{bmatrix} {}_{i_1}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} & \cdots & {}_{i_{N_\mathrm{a}}}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ of the desired robot positions relative to the formation center. With regard to the influence of different frames and points of reference, it can be observed that

$$\boldsymbol{d} = \bar{\boldsymbol{\mathcal{B}}}^\mathsf{T}\boldsymbol{x} \iff {}^{\mathrm{F}}\boldsymbol{d} = \bar{\boldsymbol{\mathcal{B}}}^\mathsf{T}\,{}^{\mathrm{F}}\boldsymbol{x} \tag{5.29}$$

$$\iff {}^{\mathrm{F}}\boldsymbol{d} = \bar{\boldsymbol{\mathcal{B}}}^\mathsf{T}\,{}^{\check{\mathrm{F}}}\boldsymbol{x}. \tag{5.30}$$

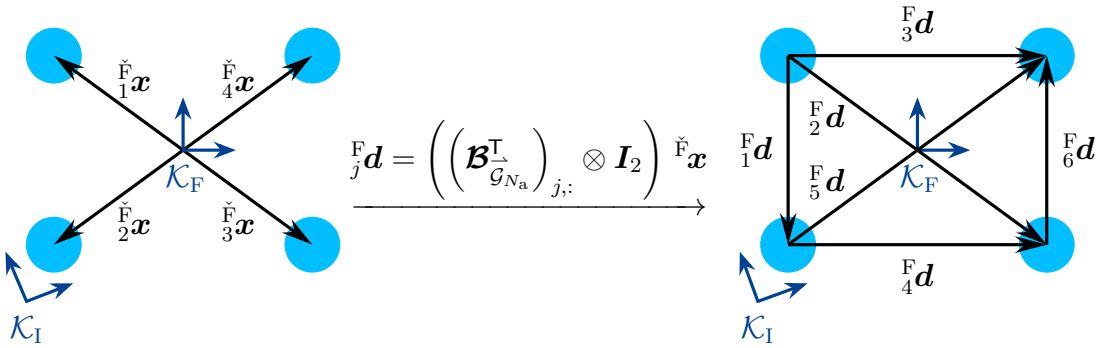


Figure 5.3: Illustration of an exemplary formation involving four robots [EbelEberhard21]

The equivalence (5.29) holds since each pair of lines of $\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}$ merely calculates differences between robot positions. Therefore, transforming the robot position vectors' coordinates before or after the left-multiplication with $\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}$ yields the same result. Similarly, equivalence (5.30) follows since the relative vectors between the positions are invariant to changes of the point of reference of the robot positions. Figure 5.3 illustrates the intuition behind Equation (5.30). Due to the properties of $\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}$ observed in Section 3.3, if the robots maintain the desired inter-robot positioning defined by $\boldsymbol{d}_\mathrm{d} = \bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\boldsymbol{x}_\mathrm{d}$, i.e., if $\boldsymbol{d}(t) = \boldsymbol{d}_\mathrm{d}$, it need not mean that $\boldsymbol{x}(t) = \boldsymbol{x}_\mathrm{d}$, but merely that $\boldsymbol{x}(t) = \boldsymbol{x}_\mathrm{d} + \boldsymbol{1}_{N_\mathrm{a}} \otimes \boldsymbol{n}$ for a translation $\boldsymbol{n} \in \mathbb{R}^2$. However, since the robot positions relative to the formation center are invariant to such a common translation of all robots, for the control goal (5.28), it holds that

$$\lim_{t\to\infty} {}^{\check{\mathrm{F}}}\boldsymbol{x}(t) \overset{!}{=} {}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d} \iff \lim_{t\to\infty} \boldsymbol{d}(t) \overset{!}{=} \boldsymbol{d}_\mathrm{d} \tag{5.31}$$

for every connected communication graph.

With the above, interpreting formation control as a shifted agreement problem, a shifted version of the classical agreement problem feedback (3.36) can be given by means of the equivalent expressions

$$\boldsymbol{u}_{\mathrm{rf,u}}(t) = -\bar{\boldsymbol{\mathcal{B}}}\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}(\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{d}) \tag{5.32}$$

$$\iff \quad \boldsymbol{u}_{\mathrm{rf,u}}(t) = -\bar{\boldsymbol{\mathcal{B}}}\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\left({}^{\mathrm{I}}\boldsymbol{S}_\mathrm{F}\left({}^{\check{\mathrm{F}}}\boldsymbol{x}(t) - {}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}\right)\right). \tag{5.33}$$

Singling out the feedback for robot $\hat{\imath}$ gives

$${}_{\hat{\imath}}\boldsymbol{u}_{\mathrm{rf,u}}(t) = -\left(\left(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}_{N_\mathrm{a}}}\right)_{\hat{\imath},:} \otimes \boldsymbol{I}_2\right) \bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}\left({}^{\mathrm{I}}\boldsymbol{S}_\mathrm{F}\left({}^{\check{\mathrm{F}}}\boldsymbol{x}(t) - {}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}\right)\right), \tag{5.34}$$

where it is possible to directly insert the formation shape as defined by ${}^{\check{\mathrm{F}}}\boldsymbol{x}_\mathrm{d}$. Inspired by this linear feedback, but with the aim to satisfy the input constraints, the nonlinear feedback

$${}_{\hat{\imath}}\boldsymbol{u}_{\mathrm{rf}}(t) = -\left(\left(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}_{N_\mathrm{a}}}\right)_{\hat{\imath},:} \otimes \boldsymbol{I}_2\right) \frac{u_{\max}}{n_{\mathcal{E},\hat{\imath}}} \tanh\Big((\boldsymbol{K} \otimes \boldsymbol{I}_2) \underbrace{\bar{\boldsymbol{\mathcal{B}}}^{\mathsf{T}}(\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{d})}_{=\boldsymbol{d}(t)-\boldsymbol{d}_\mathrm{d}}\Big) \tag{5.35}$$

is proposed. Therein, $n_{\mathcal{E},i}$ denotes the number of edges the node representing robot $i \in \mathcal{R}_\mathrm{a}$ is incident with. For the complete communication graph used for pure relative formation control, it evaluates to $n_{\mathcal{E},i} = N_\mathrm{a} - 1 =: \bar{n}_\mathcal{E}$ for all $i \in \mathcal{R}_\mathrm{a}$. Furthermore, the $\tanh(\cdot)$ function is applied in an element-wise manner and the diagonal, positive definite matrix $\boldsymbol{K} \in \mathbb{R}^{n_\mathcal{E} \times n_\mathcal{E}}$ can be used to weigh differently the errors corresponding to the different edges of the communication graph. Since it holds that $\sum_{j=1}^{n_\mathcal{E}} |(\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}_{N_\mathrm{a}}})_{\hat{\imath},j}| = n_{\mathcal{E},\hat{\imath}}$, this control input satisfies the input constraints by design, i.e., $\|{}_{\hat{\imath}}\boldsymbol{u}_\mathrm{rf}\|_\infty \leq u_\mathrm{max}$.

To show that the proposed feedback fulfills the control goal, it remains to be proven that, for the closed loop $\dot{\boldsymbol{x}} = \boldsymbol{u}_\mathrm{rf}$, it holds that $\lim_{t\to\infty} \boldsymbol{d}(t) \stackrel{!}{=} \boldsymbol{d}_\mathrm{d}$, which is equivalent to the original control goal due to (5.31). The Lyapunov function candidate

$$V(\boldsymbol{d}(\boldsymbol{x})) \coloneqq \sum_{j=1}^{n_\mathcal{E}} V_j({}_j\boldsymbol{d}(\boldsymbol{x})) := \sum_{j=1}^{n_\mathcal{E}} \frac{u_\mathrm{max}}{K_{jj}\, n_{\mathcal{E},j}} \log(\cosh(v_{1,j})\, \cosh(v_{2,j}))\,, \tag{5.36}$$

which employs the abbreviation $v_{i,j} = K_{jj}\,(({}_j\boldsymbol{d})_i - ({}_j\boldsymbol{d}_\mathrm{d})_i)$, $i \in \{1,2\}$, can be used to show this. By construction, the continuously differentiable candidate satisfies $V(\boldsymbol{d}) = \boldsymbol{0} \Leftrightarrow \boldsymbol{d} = \boldsymbol{d}_\mathrm{d}$, $V(\boldsymbol{d}) > 0\ \forall\, \boldsymbol{d} \neq \boldsymbol{d}_\mathrm{d}$, and $V(\boldsymbol{d}) \to \infty$ for $\|\boldsymbol{d} - \boldsymbol{d}_\mathrm{d}\| \to \infty$. Its derivative is obtained in the form

$$\begin{aligned}
\dot{V} &= \left(\frac{u_\mathrm{max}}{\bar{n}_\mathcal{E}} \tanh((\boldsymbol{K} \otimes \boldsymbol{I}_2)\,(\boldsymbol{d} - \boldsymbol{d}_\mathrm{d}))\right)^\mathsf{T} \dot{\boldsymbol{d}} && (5.37)\\
&= \left(\frac{u_\mathrm{max}}{\bar{n}_\mathcal{E}} \tanh\Big((\boldsymbol{K} \otimes \boldsymbol{I}_2)\,\bar{\boldsymbol{B}}^\mathsf{T}\,(\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{d})\Big)\right)^\mathsf{T} \bar{\boldsymbol{B}}^\mathsf{T} \dot{\boldsymbol{x}} && (5.38)\\
&\overset{(5.35)}{=} -\boldsymbol{u}_\mathrm{rf}^\mathsf{T} \dot{\boldsymbol{x}} && (5.39)\\
&= -\boldsymbol{u}_\mathrm{rf}^\mathsf{T} \boldsymbol{u}_\mathrm{rf} < 0 \quad \forall \left(\boldsymbol{u}_\mathrm{rf} \neq \boldsymbol{0} \overset{(5.35)}{\iff} \boldsymbol{d} \neq \boldsymbol{d}_\mathrm{d}\right). && (5.40)
\end{aligned}$$

Employing LaSalle's invariance principle [Khalil02] reveals that the closed loop approaches the positively invariant set of states $\mathcal{S}_\mathrm{rf} = \left\{\boldsymbol{x} \mid \dot{V}(\boldsymbol{x}) = 0 \iff \boldsymbol{u}_\mathrm{rf} = \boldsymbol{0} \iff \boldsymbol{d}(\boldsymbol{x}) \equiv \boldsymbol{d}_\mathrm{d}\right\}$, which completes the proof. However, the feedback $\boldsymbol{u}_\mathrm{rf}$ does not yet meet any of the two control goals since the formation is attained anywhere in the plane and neither moving with a desired velocity nor moving its center to a desired point. Nevertheless, the latter can actually be dealt with using a feedback of the form of $\boldsymbol{u}_\mathrm{rf}$, e.g., by introducing a virtual robot whose location is prescribed to be in the formation's geometric center and otherwise implementing $\boldsymbol{u}_\mathrm{rf}$ as if the additional, virtual robot would be a real one. Thus, the virtual robot also increments the value of $n_{\mathcal{E},\hat{\imath}}$ to $n_{\mathcal{E},\hat{\imath}} = N_\mathrm{a}$ in the feedback law. By modifying those entries of the error gain matrix $\boldsymbol{K}$ that belong to graph edges to which the virtual robot is incident, it is possible to weigh differently the error of the formation center and of the robots' positions relative to the formation center. Hence, at least on paper, the position-control feedback seems somewhat similar to the corresponding DMPC controller.

In the velocity-control case, if the formation shall move with the desired velocity ${}_\mathrm{c}\boldsymbol{v}_\mathrm{d}$ while the robots maintain their relative positioning, the feedback

$${}_{\hat{\imath}}\boldsymbol{u}_\mathrm{vf} = {}_{\hat{\imath}}\boldsymbol{u}_\mathrm{rf} + {}_\mathrm{c}\boldsymbol{v}_\mathrm{d} \tag{5.41}$$

can be employed, yielding the closed loop $\dot{\boldsymbol{x}} = \boldsymbol{u}_{\mathrm{vf}}$. The latter's behavior can be analyzed with the same Lyapunov function candidate as above, yielding the derivative

$$\dot{V} = \cdots = -\boldsymbol{u}_{\mathrm{rf}}^{\mathsf{T}} \dot{\boldsymbol{x}} = -\boldsymbol{u}_{\mathrm{rf}}^{\mathsf{T}} \left(\boldsymbol{u}_{\mathrm{rf}} + \mathbf{1}_{N_{\mathrm{a}}} \otimes {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}\right) \tag{5.42}$$

$$\overset{(5.35)}{=} -\boldsymbol{u}_{\mathrm{rf}}^{\mathsf{T}} \boldsymbol{u}_{\mathrm{rf}} - \left(\frac{u_{\max}}{\bar{n}_{\mathcal{E}}} \tanh\Big(\left(\boldsymbol{K} \otimes \boldsymbol{I}_2\right) \bar{\boldsymbol{B}}^{\mathsf{T}} \left(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{d}}\right)\Big)\right)^{\mathsf{T}} \underbrace{\bar{\boldsymbol{B}}^{\mathsf{T}} \left(\mathbf{1}_{N_{\mathrm{a}}} \otimes {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}\right)}_{=\mathbf{0}} \tag{5.43}$$

$$= -\boldsymbol{u}_{\mathrm{rf}}^{\mathsf{T}} \boldsymbol{u}_{\mathrm{rf}}. \tag{5.44}$$

Observing that the set

$$\mathcal{S}_{\mathrm{vf}} = \left\{\boldsymbol{x} \mid \dot{V}(\boldsymbol{x}) = 0 \iff \boldsymbol{u}_{\mathrm{rf}} = \mathbf{0} \iff \boldsymbol{d} = \boldsymbol{d}_{\mathrm{d}}\right\} \tag{5.45}$$

is positively invariant with respect to the loop closed by $\boldsymbol{u}_{\mathrm{vf}}$, LaSalle's invariance principle states that the closed-loop trajectories approach $\mathcal{S}_{\mathrm{vf}}$ as time approaches infinity. Since ${}_{i}\boldsymbol{u}_{\mathrm{vf}}\big|_{\boldsymbol{u}_{\mathrm{rf}}=\mathbf{0}} = {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ for all $i \in \mathcal{R}_{\mathrm{a}}$, in the limit as time approaches infinity, the closed loop behaves according to the dynamics $\dot{\boldsymbol{x}} = \mathbf{1}_{N_{\mathrm{a}}} \otimes {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$, as desired. Nevertheless, as it is the case for the velocity-control DMPC controller, without further measures, the feedback $\boldsymbol{u}_{\mathrm{vf}}$ will in general not respect the input constraints anymore, leading to some practical considerations for the velocity-control case and beyond.

5.1.3 Practical Considerations

The fact that the robots are modeled in the form of first-order systems complicates the velocity-control case. For a second-order model, e.g., in the DMPC case, it would be directly possible to include the deviation from a desired common velocity in the cost function. Still, were it not for the input constraints, the first-order setup would allow to directly add the formation's desired common velocity ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ to the inputs commanded by the controllers that maintain the pure relative positioning. Building upon this enticingly simple idea, it is possible to arrive at a control setup that completely circumvents the constraint issue in the case of the DMPC controller and extenuates it for the graph-algebraic controller.

To that end, for both control approaches, the feedback obtained at time step $t \geq 0$ for purely maintaining the robot's relative positioning is subsequently denoted as ${}_{i}\boldsymbol{u}_{\mathrm{rf}}(t)$. For this, it is worth noting that the graph-algebraic controllers, different than the DMPC controllers, have been formulated in continuous time. It is subsequently assumed that the graph-algebraic control inputs are applied in a sample-and-hold fashion, which is also how they are applied in the simulations and experiments later.

Instead of naively using the control input ${}_{i}\boldsymbol{u}_{\mathrm{vf}}(t) = {}_{i}\boldsymbol{u}_{\mathrm{rf}}(t) + {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$, it would be desirable to apply the input ${}_{i}\boldsymbol{u}_{\mathrm{vf}}(t) = {}_{i}\boldsymbol{u}_{\mathrm{rf}}(t) + {}_{\mathrm{c}}\tilde{\boldsymbol{v}}_{\mathrm{d}}(t)$ with

$${}_{\mathrm{c}}\tilde{\boldsymbol{v}}_{\mathrm{d}}(t) = \underset{\boldsymbol{v} \in \mathbb{R}^2}{\arg\min} \quad \left\|\boldsymbol{v} - {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}\right\|_1 \tag{5.46}$$

$$\text{subject to } \left\|{}_{i}\boldsymbol{u}_{\mathrm{rf}}(t) + \boldsymbol{v}\right\|_{\infty} < u_{\max} \quad \forall\, i \in \mathcal{R}_{\mathrm{a}}. \tag{5.47}$$

The resulting common velocity ${}_{\mathrm{c}}\tilde{\boldsymbol{v}}_{\mathrm{d}}(t)$ would come as close as possible to the desired value ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ while keeping all robots' applied control inputs admissible. Potentially the full set of inputs would be used to attain and maintain the formation, with a common motion happening only when none of the formation controllers saturates the input constraints. Hence, maintaining the formation would have priority over the common motion, which fits the very notion of formation control. With the robots optimizing concurrently and, importantly, exchanging data with one another only once every time step, for $i \neq \hat{\imath}$, the inputs ${}_{i}\boldsymbol{u}_{\mathrm{rf}}(t)$ are not available at time step t since they have not yet been communicated. A realizable approximation can be obtained by instead using the formation controllers' inputs from the previous time step, i.e.,

$$
\begin{aligned}
{}_{\mathrm{c}}\bar{\boldsymbol{v}}_{\mathrm{d}}(t) \;&=\; \underset{\boldsymbol{v}\in\mathbb{R}^2}{\arg\min} \quad \left\|\boldsymbol{v} - {}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}\right\|_1 && (5.48)\\
&\qquad \text{subject to } \left\|{}_{i}\boldsymbol{u}_{\mathrm{rf}}(t-1) + \boldsymbol{v}\right\|_\infty \leq u_{\max} \;\; \forall\, i \in \mathcal{R}_{\mathrm{a}} && (5.49)
\end{aligned}
$$

with ${}_{\mathrm{c}}\bar{\boldsymbol{v}}_{\mathrm{d}}(0) \coloneqq \boldsymbol{0}$. Since all robots rely on the same, previously communicated information to solve this optimization problem, they will also automatically arrive at an identical value for ${}_{\mathrm{c}}\bar{\boldsymbol{v}}_{\mathrm{d}}(t)$. While the resulting input ${}_{i}\boldsymbol{u}_{\mathrm{vf}}(t) = {}_{i}\boldsymbol{u}_{\mathrm{rf}}(t) + {}_{\mathrm{c}}\bar{\boldsymbol{v}}_{\mathrm{d}}(t)$ does not necessarily satisfy the input constraints, it is expected that the constraint violation

$$
\varepsilon_{\mathrm{vel}} \coloneqq \max\left\{\left\|{}_{i}\boldsymbol{u}_{\mathrm{rf}}(t) + {}_{\mathrm{c}}\bar{\boldsymbol{v}}_{\mathrm{d}}\right\|_\infty - u_{\max},\; 0\right\} \tag{5.50}
$$

is small. This is because the feedback laws are continuous in their parameters, i.e., in the current robot positions and in the desired formation shape. In particular, both are even Lipschitz continuous. While this is immediately apparent for the graph-algebraic feedback law (5.35), in the case of the DMPC controller, it follows from the properties of a multi-parametric quadratic program, where the feedback is always a piecewise affine, continuous function of the parameters. Therefore, if the parameter's change is small enough, the resulting change in the control input is small. In usual operation, with a suitably short sampling time, this is expected to hold since the robot positions will not change drastically from one time instant to the next so that the constraint violation $\varepsilon_{\mathrm{vel}}$ may be tolerable. Unfortunately, the desired formation shape, which is part of the parameters, can change drastically if the robots shall attain a different formation shape at system runtime. Since enabling adaptive formation shapes is a defining feature in a reconfigurable robotic network, a solution to the issue seems very desirable, especially if it does not induce superfluous conservativity into the closed loop. Beneficially, the possibility to constrain the change of the control input enables a solution for the DMPC controller by replacing $u_{\max}$ with $u_{\max} - \Delta u_{\max}$ both in the input constraints (5.13) of the DMPC optimization problem and in the constraint (5.49) of the optimization problem for the determination of the common formation velocity. This does introduce some conservativity since, e.g., for ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}} = \boldsymbol{0}$, the maximum norm of the applied control input will not exceed $u_{\max} - \Delta u_{\max}$. Nevertheless, for the probably more common case of $\left\|{}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}\right\|_\infty \geq \Delta u_{\max}$, the maximum admissible control

input may indeed be attained. Hence, this setup is used subsequently for the DMPC controller in the velocity-control case, whereas the graph-algebraic controller directly relies on the unmodified version of the optimization problem from Equations (5.48) and (5.49) to obtain the current common formation velocity.

5.1.4 Comparative Analysis

Despite aiming at the same control goals, the DMPC and graph algebraic setups are vastly different both with regard to their theoretical background and the complexity of the numerical calculation of the resulting control input. While some of the theoretical advantages of MPC and DMPC have already been discussed in an abstract manner in Section 3.2, the solution effort and complexity of the DMPC controller warrant an investigation whether these theoretical advantages are actually of practical benefit for formation control. Therefore, subsequently, the performance of the two control approaches is analyzed and compared in simulative and experimental scenarios. Indeed, great care needs to be taken when comparing control schemes that are so vastly different. They are tuned in different, incomparable ways, which can make it impossible to draw unbiased conclusions. Hence, the brief simulative study focuses on simple scenarios designed to single out individual properties of the controllers that are independent from the tuning of their parameters. By excluding effects that are hard to predict and quantify, such as unknown disturbances and time delays, these simulation scenarios therefore allow a quantitative analysis of the fundamental properties inherent to the control schemes themselves. Subsequent, more elaborate hardware experiments then show whether the proposed controllers can actually deal with the effects neglected in simulation.

Simulative Analysis

The simulation uses the program architecture proposed in Section 4.2. However, pure formation control does not contain any organizational tasks and hence, the robotic agent consists only of the control agent, which is solely occupied with the dynamic control of the robot. All software composing the robotic agent is written in C++, whereas simulation and visualization happen in Matlab. Using the communication-based, distributed architecture already in simulation, combining different programming languages and environments can be done without further complications. The robots are simulated with the model derived in Section 4.3.2, using the same geometric and material parameters and the same settings of the motor controllers as they have been used in Section 4.3.3. However, in this section, the motor moments are not saturated in any way in order to exclude one more effect that may cloud the judgment of the formation controllers themselves. In contrast, the commanded directional velocity is saturated at $u_{\max} = 0.2\,\mathrm{m/s}$. To easily distinguish between the controllers for the position- and velocity-control cases, the DMPC controller for position

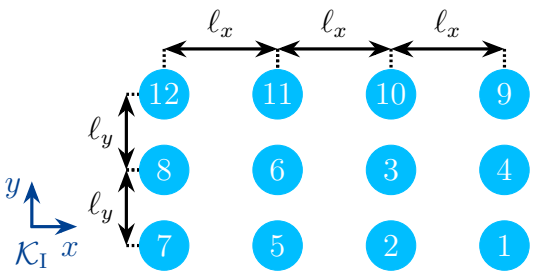


Figure 5.4: Formation with twelve robots as considered in the simulation scenarios

control is denoted as the DMPC$^\mathrm{p}$ controller, whereas the one for the velocity-control case is abbreviated as the DMPC$^\mathrm{v}$ controller. Similarly, the notations AGTC$^\mathrm{p}$ and AGTC$^\mathrm{v}$ are used to refer to the corresponding algebraic graph theory-based counterparts. Both variants of the DMPC controller employ the prediction horizon $H = 30$ and the weighting matrices $\boldsymbol{R} = 0.6\,\boldsymbol{I}$ and $\boldsymbol{T} = 1 \cdot 10^4\,\boldsymbol{\mathcal{D}}\boldsymbol{I}$ with the diagonal weighting matrix $\boldsymbol{\mathcal{D}}$ being set to

$$\boldsymbol{\mathcal{D}}^\mathrm{p} = \begin{bmatrix} 0.5\,\boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{5.51}$$

for the DMPC$^\mathrm{p}$ controller and to $\boldsymbol{\mathcal{D}}^\mathrm{v} = \boldsymbol{I}$ for the DMPC$^\mathrm{v}$ controller. The change of the control input is left unconstrained for the DMPC$^\mathrm{p}$ controller, making it more similar in its properties to the AGTC$^\mathrm{p}$ controller, but constrained by $\Delta u_\mathrm{max} \coloneqq 0.02\,\mathrm{m/s}$ in the velocity-control case. This allows to reap the benefits of the proposed setup for the determination of the common formation velocity. All controllers, including the graph-algebraic ones, are applied with the sampling time $T_\mathrm{s} = 0.05\,\mathrm{s}$. If not stated otherwise, the AGTC$^\mathrm{p}$ and AGTC$^\mathrm{v}$ controllers employ the error gain $\boldsymbol{K} = 3\,\boldsymbol{I}$.

The first simulation puts to the test the DMPC$^\mathrm{p}$ and AGTC$^\mathrm{p}$ controllers, with the intent to study the influence of the number of robots on the conservativity of the control schemes. The scenario is simulated for increasing numbers of robots $N_\mathrm{a} \in \{2, 4, 6, 8, 12\}$. The formation shape for twelve robots is depicted in Figure 5.4, with each circle denoting a robot and the encircled numbers corresponding to the robots' identification numbers. The inter-robot distances are set to $\ell_x = 1.0\,\mathrm{m}$ and $\ell_y = 0.75\,\mathrm{m}$. For formations involving fewer robots, the first N_a robot positions are picked from the formation shape depicted in the figure. The robots are assumed to be in stationary, perfect formation initially and, beginning at $t = 3\,\mathrm{s}$, shall move the formation's geometric center from its initial position ${}_\mathrm{c}\boldsymbol{x}(0) = \boldsymbol{0}\,\mathrm{m}$ to ${}_\mathrm{c}\boldsymbol{x}_\mathrm{d} = \begin{bmatrix} 3 & 0 \end{bmatrix}^\mathsf{T}\,\mathrm{m}$. Hence, due to the absence of exogenous disturbances, the robots' behavior is predominantly determined by the positional error of the geometric formation center since the error in the robots' relative positions stays imperceptible. One might assume that the controllers command the formation to drive with the maximum admissible velocity toward the setpoint, at least if the controllers are not tuned in a too conservative manner. However, as the results from Figure 5.5 show, at

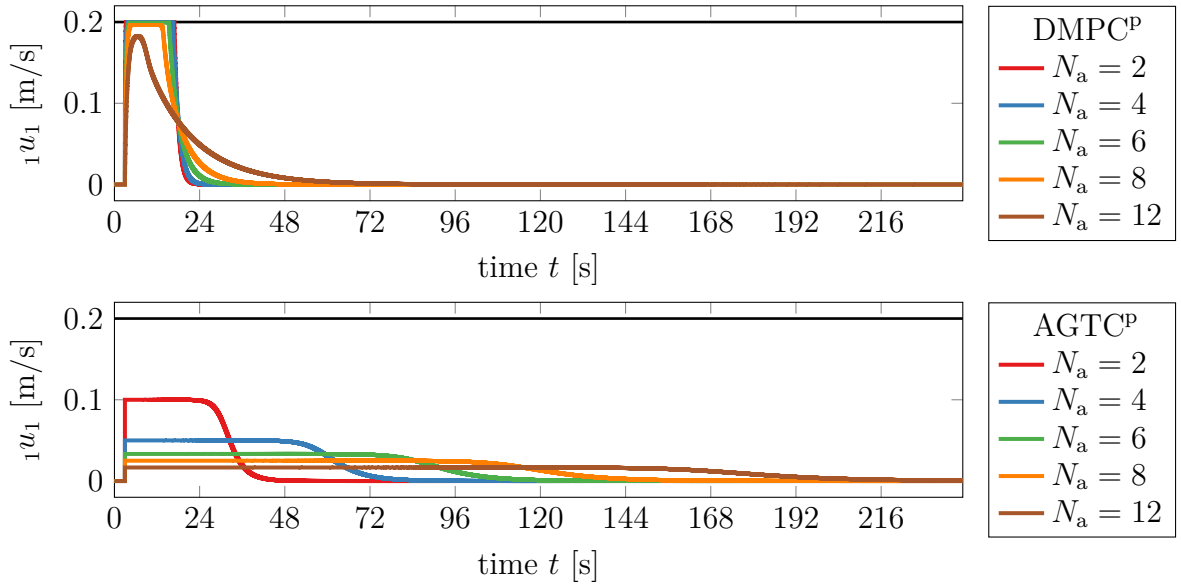


Figure 5.5: Comparison of selected control inputs resulting from the applications of the DMPC$^{\mathrm{p}}$ and AGTC$^{\mathrm{p}}$ schemes, with the DMPC$^{\mathrm{p}}$ results being shown in the upper plot

least for the AGTC$^{\mathrm{p}}$ controller, the situation is vastly different. Even for only two robots, the controller only commands a maximum directional velocity of $u_{\max}/2$, with it decreasing further for every robot added to the formation. While the figure only shows the relevant control input of robot 1, due to the setup of the task, the plots look indistinguishably similar for the other robots.

Taking this result as a motivation to reconsider the setup of the AGTC$^{\mathrm{p}}$ controller, it turns out that this is a fundamental limitation that is independent of the tuning of the controller. The communication graph used for the controller is complete, with an additional virtual robot representing the formation center. Hence, each node representing a robot is adjacent to every other robot and to the virtual robot, meaning that each robot node is incident with N_{a} edges, i.e., $n_{\mathcal{E},i} = N_{\mathrm{a}}$ for all $i \in \mathcal{R}_{\mathrm{a}}$. However, in this specific, synthetic simulation setup, for each robot, there is only a control error along one of these edges, namely the one corresponding to the formation center. Inspecting the feedback law (5.35) from this point of view reveals that, under these circumstances, the control input is upper-bounded by $u_{\max}/N_{\mathrm{a}}$ in its absolute value, which fits precisely to the maximum values attained in Figure 5.5. This may be extenuated by building the formation controller upon a more sparse but still connected communication graph since this kind of conservativity decreases as the graph's sparsity increases. However, this would run contrary to the goal of being able to directly assign different weights to the absolute positioning of the formation center and the relative control error, which would mean trading one disadvantage with another. Of course, this synthetic, rather untypical setup with errors only occurring along one edge per robot constitutes the worst case with regard to the scheme's conservativity. Nevertheless,

as the results for the DMPC$^{\mathrm{p}}$ controller show, it does not pose as much of a challenge to the optimization-based nature of DMPC with its explicit consideration of the constraints. Still, the conservativity also increases with increasing numbers of robots, which can be explained by the control task getting more difficult with larger numbers of robots, with the predicted errors summing up over more robots. Furthermore, in a bit more of a subtle manner, also the convex combination in step 3 of the DMPC scheme from Algorithm 1 introduces some degree of conservativity by giving greater weight to the candidate solution for larger numbers of robots. In any case, the performance of the DMPC$^{\mathrm{p}}$ scheme seems vastly superior in this simple, yet surprisingly challenging, test. To give some perspective, when measuring the time spans it takes for the formation to reach the goal up to 5 cm into the vicinity of the goal position, starting to measure from the setpoint change, it approximately takes between $T_2 = 15.55\,\mathrm{s}$ for two robots and $T_{12} = 55.15\,\mathrm{s}$ for twelve robots with the DMPC$^{\mathrm{p}}$ controller, but between $T_2 = 33.9\,\mathrm{s}$ and $T_{12} = 203.95\,\mathrm{s}$ with the AGTC$^{\mathrm{p}}$ controller.

In the second scenario, the velocity-control setups are put to the test. Four robots once more start in perfect formation, with the same four-robot formation being used as in the previous scenario. They shall move with a desired common velocity ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}} \coloneqq \begin{bmatrix} 0.18 & 0 \end{bmatrix}^{\mathsf{T}}\,\mathrm{m/s}$. However, starting at $t = 7\,\mathrm{s}$, the desired formation shape increases linearly in size, starting at the initial values $\ell_x = 1.0\,\mathrm{m}$, $\ell_y = 0.75\,\mathrm{m}$ for the inter-robot distances and reaching the values $\ell_x = 3.0\,\mathrm{m}$, $\ell_y = 3.25\,\mathrm{m}$ at $t = 9.5\,\mathrm{s}$. From that time onward, the desired formation shape's size is kept constant until it is switched back to the original size at $t = 27\,\mathrm{s}$, this time in a discontinuous manner. It suffices to look at the result data for a subset of the robots due to symmetry. As the results depicted in Figure 5.6 show, the input sequence commanded by the DMPC$^{\mathrm{v}}$ scheme respects the input constraint at all times, with specific inputs still coming very close to the input bounds. The AGTC$^{\mathrm{v}}$ scheme's results, however, confirm the theoretical suspicions as it mostly respects the input constraints but can, for short amounts of time, exceed them drastically. In one time instance in this scenario, the maximum admissible input is exceeded by more than 50 %.

Judging by the results of both schemes, however, the determination of the formation's common velocity through Equations (5.48) and (5.49) serves its purpose well. In those time instances in which the formation controllers are very active due to the formation shape changing its size, the common velocity is reduced as desired, giving more priority and input authority to correcting the relative positioning. This allows the formation controllers to reduce the relative control errors of the robots swiftly. Figure 5.7, which representatively depicts the relative control error ${}_1\Im^{\mathrm{rel}} \coloneqq \left\| {}_1\boldsymbol{x} - {}_{\mathrm{c}}\boldsymbol{x} - {}^{\mathrm{I}}\boldsymbol{S}_{\mathrm{F}}\, {}^{\check{\mathrm{F}}}_{1}\boldsymbol{x}_{\mathrm{d}} \right\|_2$ of robot 1, confirms this.

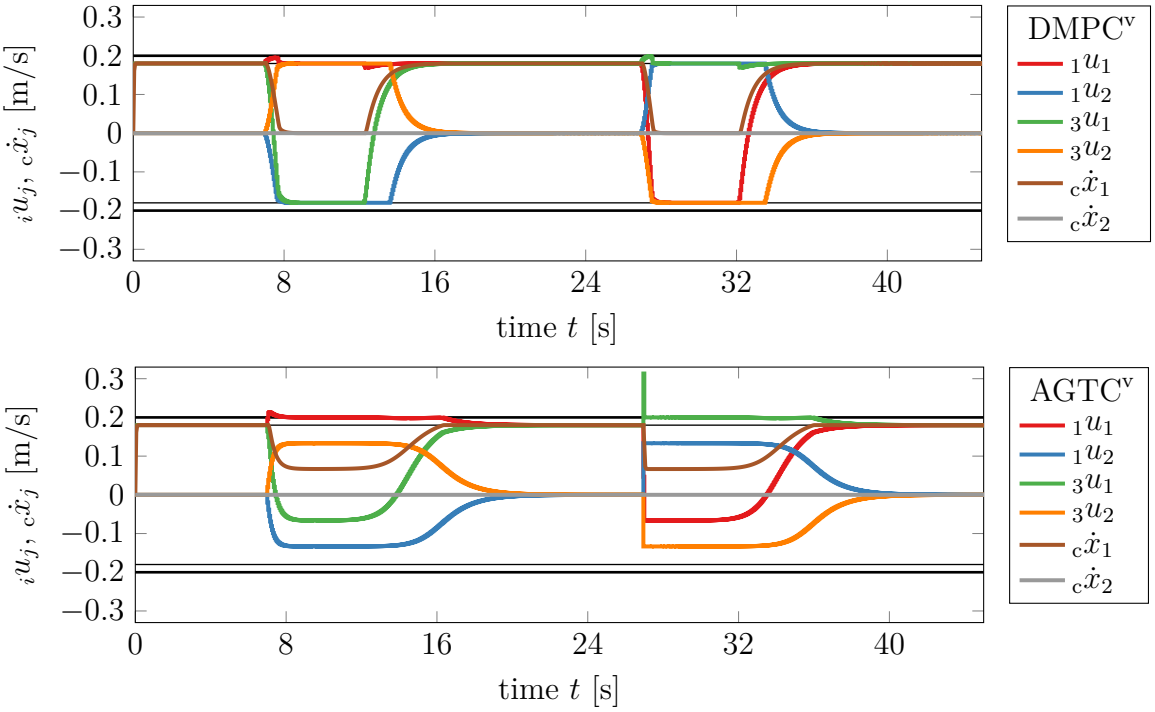


Figure 5.6: Comparison of the unsaturated control inputs and formation center velocity resulting from the DMPC^{v} and AGTC^{v} schemes in the second simulation scenario, which considers an evolving formation shape, with the DMPC^{v} results being shown in the upper plot

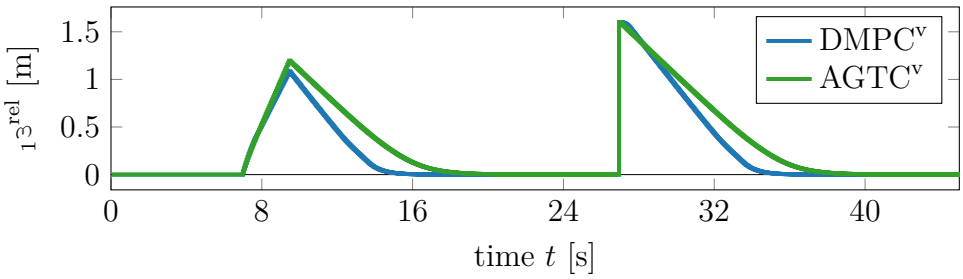


Figure 5.7: Relative control errors of robot 1 resulting from the application of the DMPC^{v} and AGTC^{v} schemes in the second simulation scenario, which considers an evolving formation shape

Experimental Analysis

In the experiments, the robotic agents use the same software architecture and, if not stated otherwise, identical control parameters as in the previous simulations. The robots' poses are tracked in real time by an external tracking system consisting of six Prime 13W OptiTrack cameras. A custom piece of software, running on a laboratory computer solely used for tracking purposes, takes the data from the tracking system and repackages it in a format identical to what is published by the simulation agent in simulations. The cameras are operated at a frequency of 100 Hz, which is also used as the publishing frequency of the pose information. The experiments employ the omnidirectional robot introduced in Section 4.3. The robots are connected to a common wireless network on which also the tracking information is available. Since, in its current configuration, the robots' onboard processing capabilities are somewhat limited, the control agents do not run on the robots themselves but on a portable laboratory computer. The latter has an Intel Core i7-9750H CPU operating at 2.6 GHz with six physical and twelve logical cores. The control agents publish the control inputs, which are given in the inertial frame of reference, on the network. Each robot receives its control inputs over the wireless network and rotates it into its body-fixed frame. For the latter, the onboard software of the hardware robots is subscribed to the data coming from the tracking system. The received control input is applied via the kinematics relations (4.5)-(4.8) and four independent PID controllers governing the motors' angular velocities. Since all data between the various participating programs is exchanged via communication, network time delays are present, and messages are, at times, lost or not received on time. Similarly, it does happen that the robots' wheels, with their small rollers, slip on the floor of the experimentation area. Therefore, the major disturbances and intricacies that are hardest to reflect in simulations are well-represented in the hardware setup. The experiment results are recorded with an overhead video camera. Furthermore, to provide additional information, a Matlab-based protocol agent is running in the network. It subscribes to and records all relevant information, including the control inputs commanded and the robot poses tracked. Naturally, it is not a perfect observer since some information may fall victim to lossy communication. Nevertheless, it is still a convenient way to gather the relevant information in one place, which would otherwise be distributed over the whole network.

The first experiment scenario puts to the test the DMPC$^{\mathrm{p}}$ and AGTC$^{\mathrm{p}}$ controllers by letting them track a very challenging trajectory with the geometric center of a four-robot formation. The formation's shape shall be a square with an edge length of 0.5 m, corresponding to the positioning of the first four robots in Figure 5.4 for $\ell_x = \ell_y = 0.5\,\mathrm{m}$. The trajectory is designed to saturate the input constraints so that the robots need to compromise between spending control effort for maintaining the formation shape and for tracking. It is given by

$$ {}_{\mathrm{c}}\boldsymbol{x}_{\mathrm{d}}(t) = \begin{bmatrix} 1.5\,\mathrm{m}\,\sin(\omega_{\mathrm{t}} t) & 0.5\,\mathrm{m}\,\sin(2\omega_{\mathrm{t}} t) \end{bmatrix}^{\mathsf{T}}, \tag{5.52} $$

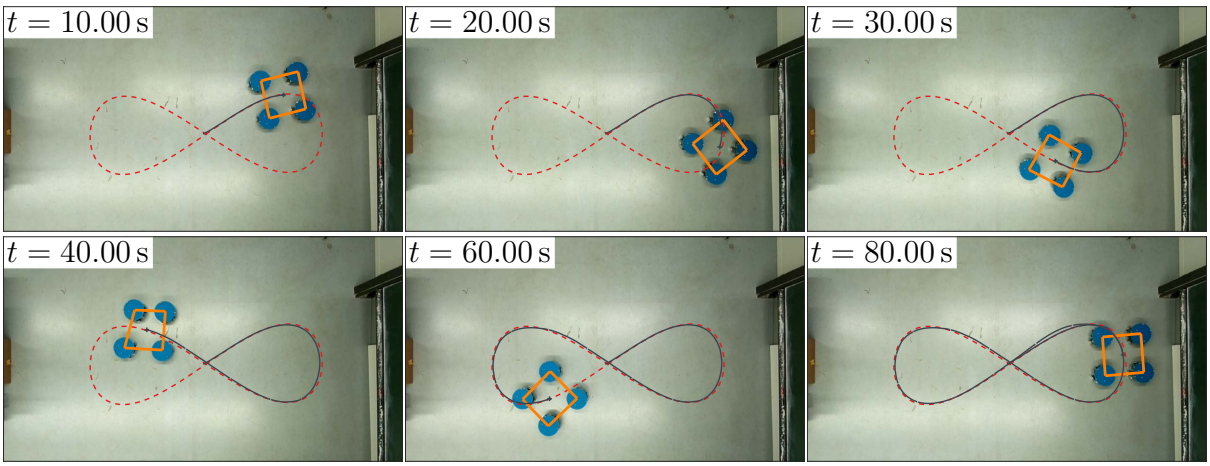


Figure 5.8: Impressions from the first experiment scenario, exemplarily run by the DMPC$^{\mathrm{p}}$ controller

which yields an ∞-shaped trajectory for a given angular velocity $\omega_{\mathrm{t}} > 0$. To increase the intricacy of the task, the formation is rotated continuously by letting the reference frame $\mathcal{K}_{\mathrm{F}}$ rotate with the angular velocity

$$\boldsymbol{\omega}_{\mathcal{K}_{\mathrm{I}}\mathcal{K}_{\mathrm{F}}} = \begin{bmatrix} 0 & 0 & \omega_{\mathrm{f}} \end{bmatrix}^{\mathsf{T}}. \tag{5.53}$$

For the moment, the angular velocities are set to $\omega_{\mathrm{t}} = 0.1\,\mathrm{rad/s}$ and $\omega_{\mathrm{f}} = 0.2\,\mathrm{rad/s}$. Consequently, both the desired relative positioning and the desired position of the formation center are modified continuously. Hence, different from the synthetic simulation scenario previously looked at in Figure 5.5, control errors occur along all edges of the graph-algebraic controller's communication graph. Initially, the robots are placed coarsely in formation around the origin. The experiment is run over a time span of 80 s. Exemplary impressions from the experiment can be found in Figure 5.8, which shows photographs of the experiment using the DMPC$^{\mathrm{p}}$ controller. The robots' top layers are covered in blue for improved visibility in photographs and video recordings. In this and all similar subsequent figures, the trajectories are mapped onto the images in an approximate manner, with the current formation shape being depicted in orange. The image corresponding to the time $t = 40\,\mathrm{s}$ shows that the robots do not, at all times, perfectly maintain their formation shape while following the trajectory. One may conjecture that this has to do with the input constraints. And, indeed, as the results from Figure 5.9 show, some robots' DMPC-based controllers make use of their full control authority to have a chance of following the trajectory. In consequence, the robots succeed in following the trajectory without falling behind too considerably, as the resulting trajectory of the formation's geometric center shows, cf. Figure 5.10. However, the results from the figure also suggest that the graph-algebraic controller fails to follow the trajectory, even with error gains increased beyond their standard setting of $\boldsymbol{K} = 3\,\boldsymbol{I}$. As the commanded control inputs depicted in Figure 5.11 show, even for $\boldsymbol{K} = 12\,\boldsymbol{I}$, the control inputs do not come close to satisfying the input

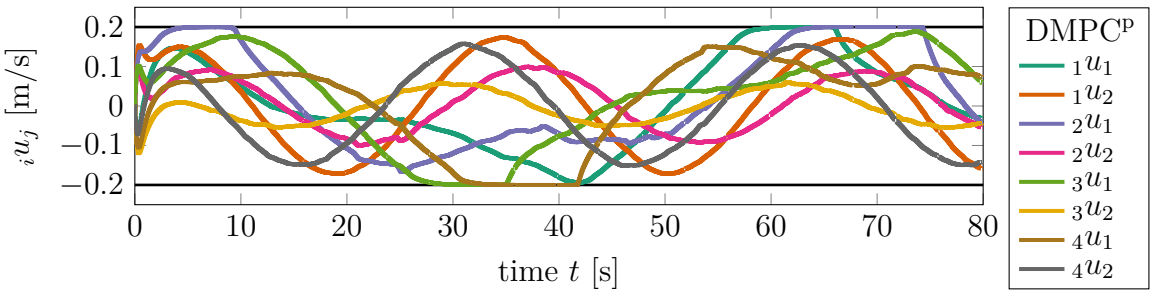


Figure 5.9: Control inputs commanded by the DMPC$^{\mathrm{p}}$ scheme in the first experiment scenario, which consists of following a fast ∞-shaped trajectory with a rotating formation

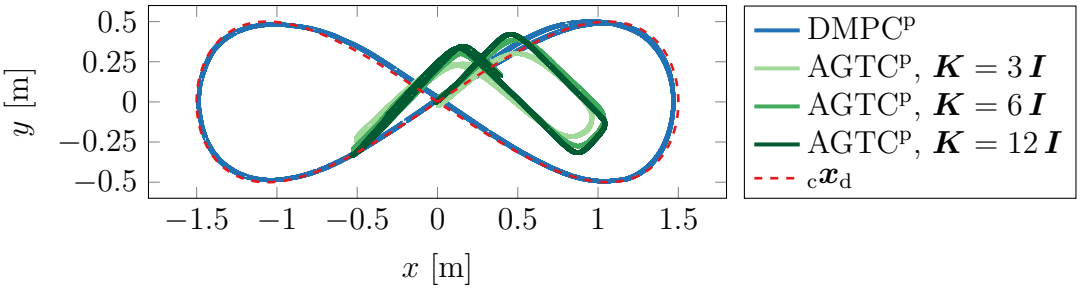


Figure 5.10: Comparison of the formation center's trajectories in the first experiment scenario, which consists of following a fast ∞-shaped trajectory with a rotating formation

bounds. This highlights that the controller's inherent conservativity, previously observed in theory and simulations, can also pose a problem in realistic settings where errors are present along all edges of the communication graph. The fact that the DMPC-based controller succeeds in compromising between maintaining the formation shape and following the trajectory, with the input constraints prohibiting a perfect performance, also becomes visible in the robots' relative formation errors. Inspecting the relative errors resulting from the DMPC$^{\mathrm{p}}$ scheme through the upper part of Figure 5.12 reveals that time spans of

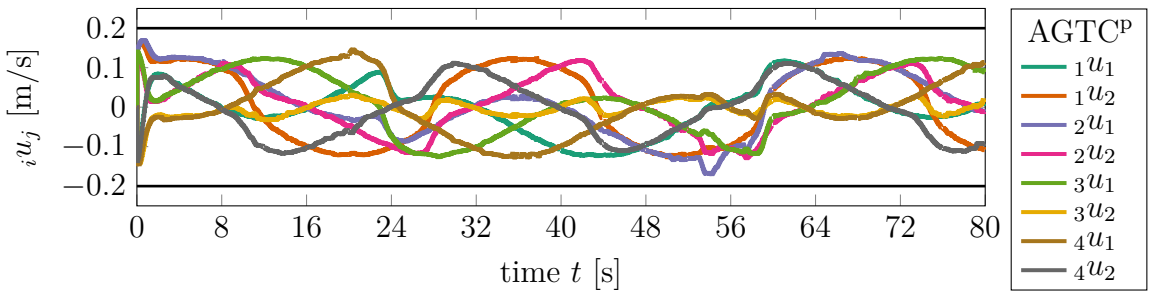


Figure 5.11: Control inputs commanded by the AGTC$^{\mathrm{p}}$ scheme for $\boldsymbol{K} = 12\,\boldsymbol{I}$ in the first experiment scenario, which consists of following a fast ∞-shaped trajectory with a rotating formation

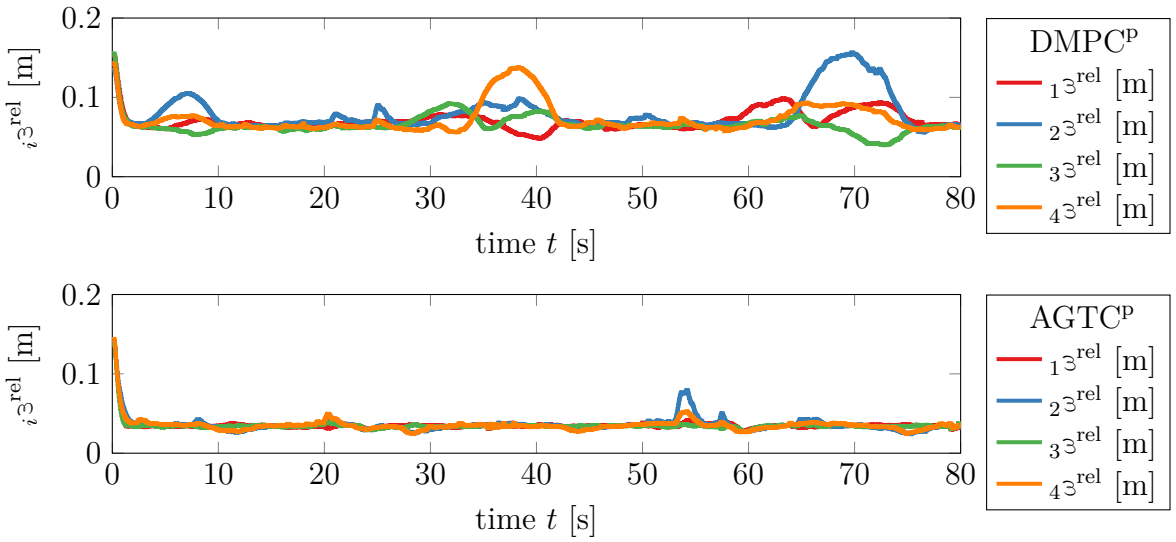


Figure 5.12: Relative control errors of the robots following a fast ∞-shaped trajectory with a rotating formation, with the AGTC$^{\text{p}}$ controller using an error gain of $\boldsymbol{K} = 12\,\boldsymbol{I}$

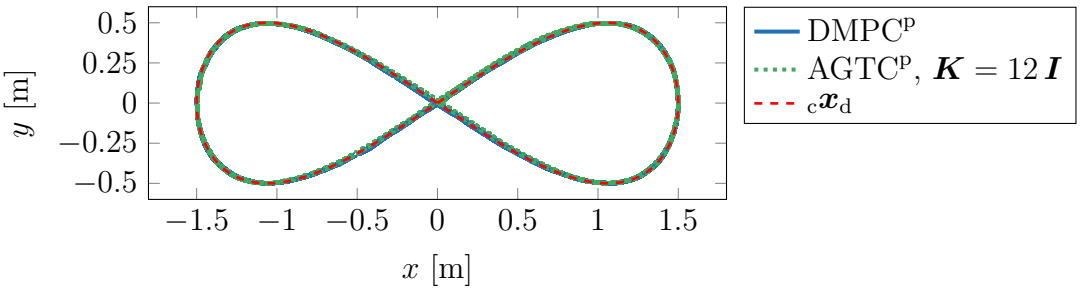


Figure 5.13: Comparison of the formation center's trajectories in a modified, slower version of the first experiment scenario

larger relative errors correspond to time spans where certain control inputs saturate the input constraints, cf. Figure 5.9. As Figure 5.12 shows, while failing to follow the desired formation center trajectory, the graph-algebraic controller fares better in keeping the relative errors low since more control authority is available for that task. Furthermore, the fact that, in this scenario, the AGTC$^{\text{p}}$ controller does not seem to be as overwhelmingly conservative as in the previous synthetic simulation scenario seems heartening regarding the practical usefulness of the controller. And, indeed, in a slower version of the considered scenario, it does succeed with tracking the desired formation center trajectory. Setting the angular velocity parameters to $\omega_{\text{t}} = 0.025\,\text{rad/s}$ and $\omega_{\text{f}} = 0.1\,\text{rad/s}$, and running the experiment over a time span of 300 s yields a tracking performance that is well-nigh perfect for both controllers, see Figure 5.13. Thus, in situations where its conservativity is not an issue, the AGTC$^{\text{p}}$ scheme can also be a valid choice that respects input constraints.

To conclude the treatise of formation control, it remains to be seen how the general setups

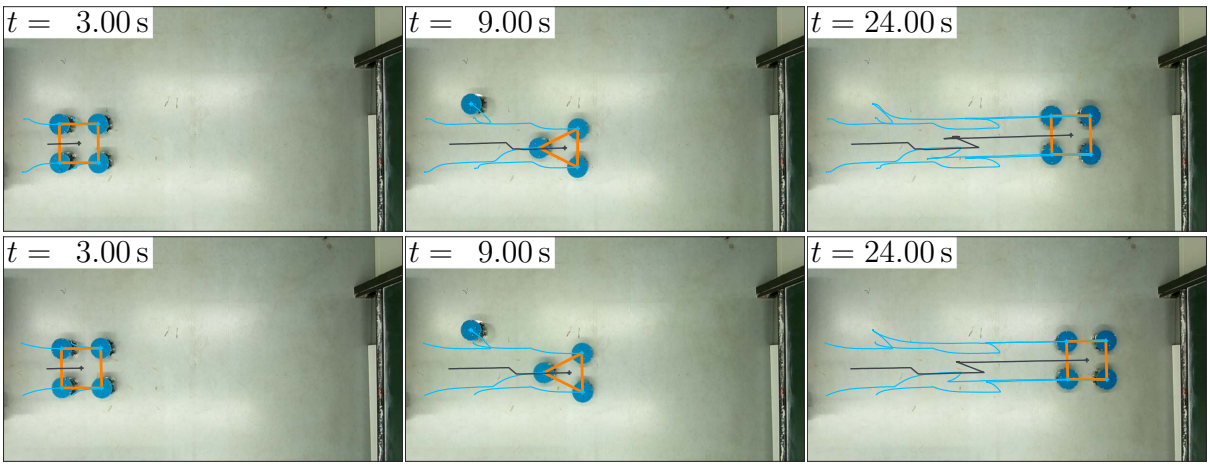


Figure 5.14: Impressions from the second experiment scenario with the results from the DMPC$^{\mathrm{v}}$ and AGTC$^{\mathrm{v}}$ controllers shown in the top and bottom rows, respectively

deal with changing numbers of active robots. Furthermore, the velocity-control schemes have not yet been inspected in experiments. In consequence, the final formation control experiment shall elucidate both aspects. To that end, once again, four robots cooperate in maintaining a square-shaped formation with an edge length of 0.5 m and are placed coarsely in formation initially. They shall move with a common velocity of ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}} = \begin{bmatrix} 0.18 & 0 \end{bmatrix}^{\mathsf{T}}$ m/s. However, for $5\,\mathrm{s} \leq t < 10\,\mathrm{s}$, robot 3 is not part of the formation and instead uses an auxiliary controller to steer clear of the formation. The remaining robots form a triangular formation, with the leading two robots keeping their centers' distance of 0.5 m. The formation's third robot shall move to the center of the former square formation shape's trailing edge. Impressions from the experiment are shown in Figure 5.14. The commanded control inputs depicted in Figure 5.15 confirm the observations made in the previous simulative studies, with the DMPC$^{\mathrm{v}}$ controller respecting the input bounds under all circumstances, whereas the AGTC$^{\mathrm{v}}$ scheme can violate them for short time intervals. In any case, Figure 5.16 reveals that both control schemes can swiftly reduce the relative formation errors when the formation shape changes.

All things considered, the experiments have shown that all proposed control schemes can deliver a usable performance in practical hardware experiments when used within the confines of their abilities. None of them has shown any particular susceptibility to the disturbances appearing in hardware experiments when compared with one another. However, the DMPC-based approaches have confirmed their advantages rooted in their theoretical setup, with constraints being respected and used to their fullest extents whenever necessary. The latter allows the DMPC-based position controller to track significantly faster trajectories for given input bounds. Moreover, apprehensions concerning the increased computational requirements of the DMPC-based schemes cannot be confirmed. Both the solution as well as the reformulation of the underlying optimization problems prove to be

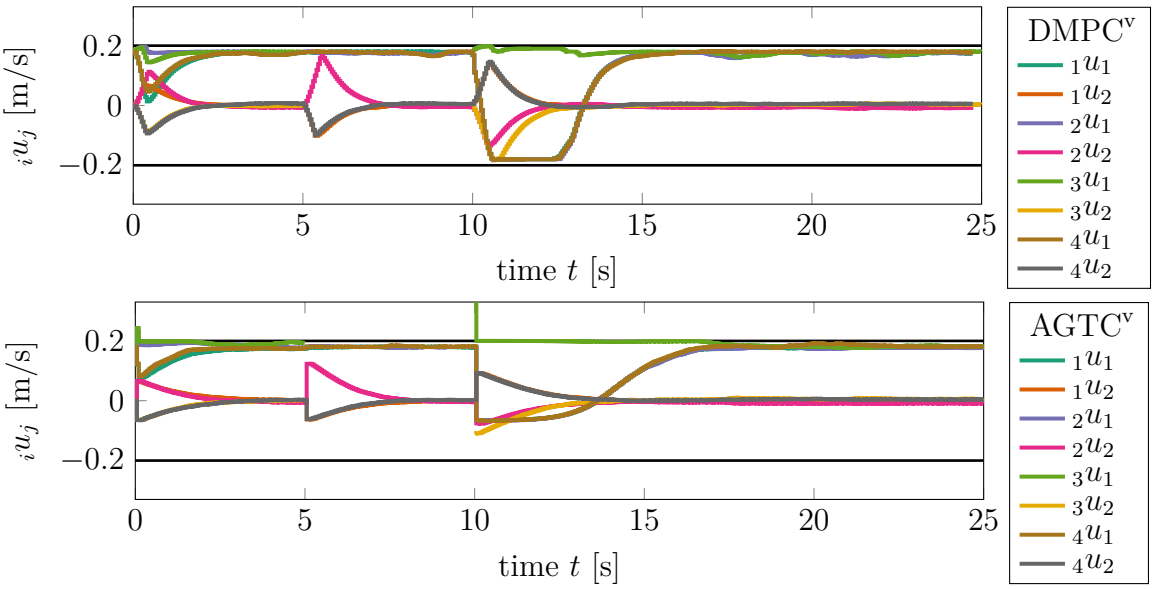


Figure 5.15: Control inputs commanded by the DMPC^{v} and AGTC^{v} schemes in the final experiment scenario

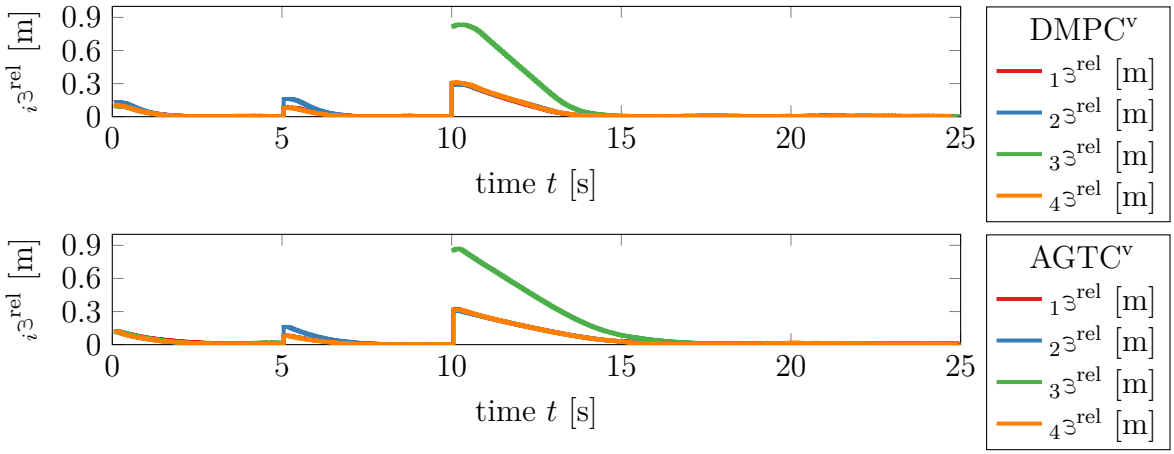


Figure 5.16: Relative control errors of the robots in the final experiment scenario for the DMPC^{v} and AGTC^{v} controllers

possible within the real-time requirements posed by an experiment. Furthermore, a more nuanced advantage does not manifest itself in the results but only in the experience of working with both approaches – and that is DMPC's ease and intuitiveness when it comes to tuning. For all these reasons, in the transportation scheme proposed in this thesis, DMPC-based formation control is employed. In addition, whenever precise positioning is required, e.g., when space for maneuvering is limited, the position-control setup is easier to parameterize than the velocity-control case. For maneuvering an object into a specific pose, the latter would require an outer control loop governing the formation's desired velocity, introducing additional control parameters that would require further tuning. Thus, henceforward, for transportation purposes, mostly the DMPC$^{\mathrm{p}}$ controller is used. This also suits the hardware experiments conducted in the thesis since the available experimentation area is of limited size, requiring precise maneuvering so that none of the robots goes astray and leaves the area covered by the tracking system.

5.2 Organization for Cooperative Transportation

Following the model task of cooperative transportation, it is still unclear how to come up with formations useful to transport an object, which constitutes the primary organizational task in this thesis. In more abstract manners, with DMPC-based formation control, a compelling and well-tunable method has been introduced that allows to coordinate the states of cooperating systems relative to one another and with regard to a global reference. The notion of organization now mostly reduces to the question of which relationship between the states is desirable and useful for solving a practical task.

Mathematically handling organization requires an additional formalization of the transportation task. In particular, this yields further requirements for the robotic system, in addition to those necessary for formation control, which have been stated and motivated at the outset of Section 5.1. Similar to formation control, the task is treated as a planar task. Regarding the object's shape, it is assumed that the robots know the shape in advance and that it is polygonal but potentially non-convex. Furthermore, each robot shall know its position relative to the object as well as the location of the object's center of mass, and the object's pose relative to a desired path or setpoint. Deformations of the object are not treated in this thesis, and, therefore, the object is assumed to be rigid. Moreover, the object shall not be subject to any kinematic constraints, i.e., it may be pushed in any direction. Fitting to the robot design introduced in Section 4.3, the robots shall have a circular footprint of radius r_{R}. Finally, each robot shall be equipped with distance sensors useful to navigate around the non-convex object safely. In an extension not treated subsequently, these sensors may be used to infer the object shape in a discovery phase executed before the transportation process begins. As will be seen, these additional requirements suffice to synthesize and attain formations serviceable to transportation.

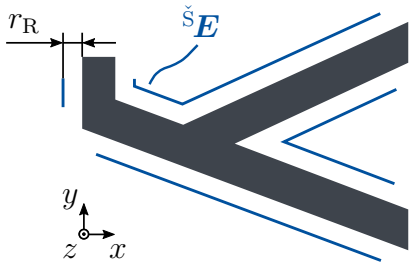


Figure 5.17: Illustration of a polygonal, non-convex object with the dilated object's edges ${}^{\check{\mathrm{S}}}\boldsymbol{E}$ shaded in blue

5.2.1 Formation Synthesis

As motivated previously, the formation synthesis problem will be solved through formulating and solving it as an optimization problem. Before discussing how the resulting problem can be solved in a distributed fashion, it is essential to analyze the resulting problem's properties. Thus, at first, the formation synthesis problem is formulated.

Formulating the Optimization Problem

Given the object's shape, to make the formation synthesis problem amenable to algorithmic treatment, a couple of preprocessing steps are helpful. Whereas the transportation task is two-dimensional, the formation synthesis task can be regarded as a one-dimensional problem of finding N_a robot positions along the edge of the object to be transported. Nevertheless, as will be seen, the intricacy of the problem is not to be underestimated.

During transportation, when the robots are in contact with the object, their centers have a distance of their radius to the edge of the object. Hence, in the first step, a dilated version of the object is calculated, with the edges being moved outward by the robot radius r_R. During transportation, the robots can aim to position their centers of mass along the dilated object's edges. Crucially, the robots shall manipulate the object through the forces they can exert in the direction of the inner normals of the object's edges. Potential friction forces in the tangential direction are not used since that is expected to be less robust. However, at pointed corners of the object, the normal direction is not uniquely defined – pushing the object at pointed corners may yield a hardly predictable behavior. Hence, at pointed corners, the edges of the dilated object are clipped by a certain safety distance, as illustrated in Figure 5.17. In contrast, if neighboring edges meet at a reflex inner angle, a robot may have one contact point with each of the edges simultaneously. The edges of the dilated object are described by the function ${}^{\check{\mathrm{S}}}\boldsymbol{E} : \mathcal{I}_\mathrm{e} \to \mathbb{R}^2$, giving all points along the edges in the coordinates of and relative to the frame of reference $\mathcal{K}_\mathrm{S}$, which is a body-fixed frame with its origin located in the object's center of mass C_S. It

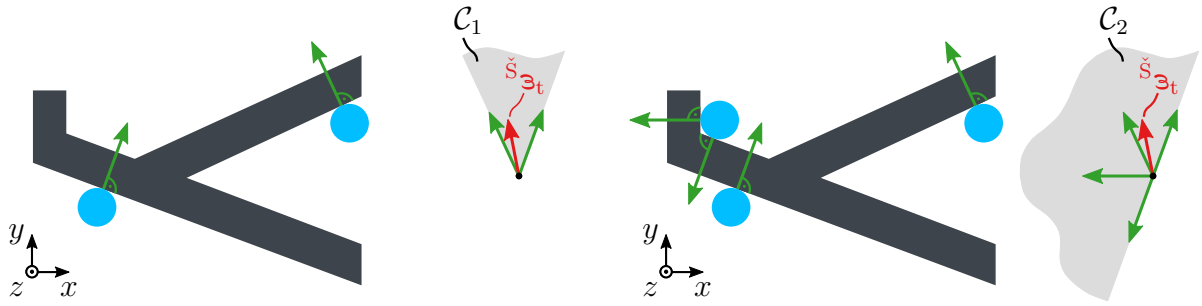


Figure 5.18: Two different formations of robots around an object, including the convex cones $\mathcal{C}_1$ and $\mathcal{C}_2$ generated by the corresponding set of inner contact normals

is parameterized on the closed interval $\mathcal{I}_\mathrm{e} \subset \mathbb{R}$. Subsequently, it is assumed that the object's current relative translational error is given by ${}^{\check{\mathrm{S}}}\mathfrak{z}_\mathrm{t} \in \mathbb{R}^2$, whereas the current rotational error is given by $\mathfrak{z}_\mathrm{r} \in \mathbb{R}$. The formation synthesis task can now be defined as finding N_a positions $w_1, \ldots, w_{N_\mathrm{a}}$ along the edge ${}^{\mathrm{S}}\boldsymbol{E}$. For algorithmic treatment, these positions are subsumed in the vector $\boldsymbol{w} \coloneqq \begin{bmatrix} w_1 \cdots w_{N_\mathrm{a}} \end{bmatrix}^\mathsf{T}$ in the following. Given a certain positioning defined by $\boldsymbol{w}$, the N_a robots may have $n_\mathrm{c}(\boldsymbol{w}) \geq N_\mathrm{a}$ contact points with the object. Evidently, for manipulating the pose of the object, two quantities are of central importance. Apart from the above-mentioned inner normals, for rotation, the contact points' lever arms with respect to the object's center of mass C_S are of interest. Henceforward, the lever arms corresponding to a configuration $\boldsymbol{w}$ are collected in the vector $\boldsymbol{\nu} \in \mathbb{R}^{n_\mathrm{c}(\boldsymbol{w})}$, whereas the columns of ${}^{\mathrm{S}}\boldsymbol{Z} \in \mathbb{R}^{2 \times n_\mathrm{c}(\boldsymbol{w})}$ shall correspond to the object's inner normal vectors at the contact points given in the coordinates of the frame $\mathcal{K}_\mathrm{S}$. The levers' signs are defined to be positive if the moment of a robot's pushing force at the corresponding contact point is a positive moment about the object's center of mass in the z-direction. In the opposite case, they are defined to be negative.

The first observation is that the translational error ${}^{\check{\mathrm{S}}}\mathfrak{z}_\mathrm{t}$ must be representable as a conic combination of the inner normal vectors so that the object can even be pushed into the direction of ${}^{\check{\mathrm{S}}}\mathfrak{z}_\mathrm{t}$. This means that there must exist a vector $\boldsymbol{\alpha} \in \mathbb{R}^{n_\mathrm{c}(\boldsymbol{w})}$ of coefficients such that

$$ {}^{\mathrm{S}}\boldsymbol{Z}(\boldsymbol{w})\,\boldsymbol{\alpha} = {}^{\check{\mathrm{S}}}\mathfrak{z}_\mathrm{t}, \quad \alpha_i \geq 0,\ i \in \{1, \ldots, n_\mathrm{c}(\boldsymbol{w})\}. \tag{5.54} $$

Such a vector of coefficients need not be unique. If it is postulated that the resultant force accelerating the object shall be proportional to the translational error ${}^{\check{\mathrm{S}}}\mathfrak{z}_\mathrm{t}$ instead of merely pointing in its direction, the entries of $\boldsymbol{\alpha}$ are proportional to the required pushing forces into the directions of the corresponding normal vectors in ${}^{\mathrm{S}}\boldsymbol{Z}(\boldsymbol{w})$.

The geometry of this line of thought is illustrated in Figure 5.18, which shows two different formations around the object, one with two robots and one with three robots, which are depicted as light-blue circles. As can be seen, the green normal vectors generate the

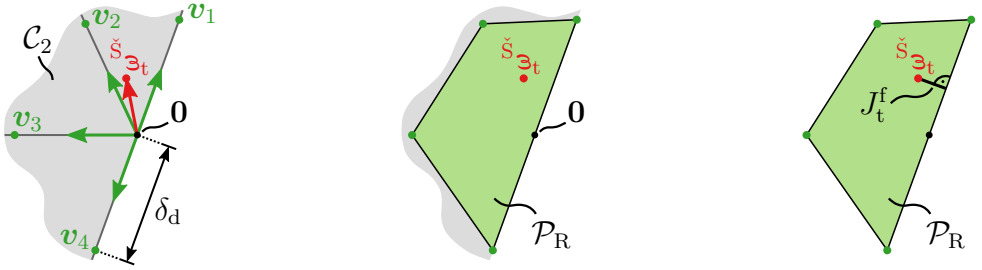


Figure 5.19: Illustration of the calculation of the robustness value $J_\mathrm{t}^\mathrm{f} = J_\mathrm{t}^\mathrm{f}(\boldsymbol{w})$ for the cone C_2 from the previous figure

cones $\mathcal{C}_1$ and $\mathcal{C}_2$, which contain all vectors that can be represented as a conic combination of the normals. Therefore, the cones stretch to infinity at their curved boundaries. The translational error vector ${}^\check{\mathrm{S}}\mathfrak{e}_\mathrm{t}$, which can be thought of as the relative vector pointing from the current to the desired position of the object's center of mass, is contained in both cones. Translation-wise, both formations are suitable to push the object in the demanded direction. When formulating an optimization problem describing the formation synthesis task, it hence seems prudent to require through the constraints that the translational error is contained in the cone generated by the normals since otherwise, the transportation will fail. However, even when fulfilling this constraint, formations may be of different quality. In Figure 5.18, the two-robot formation on the left-hand side is decidedly less robust to changes of the translational error. For instance, if the error vector is rotated a bit about the z-axis, it is no longer contained, which may happen quickly if a trajectory is tracked. Thus, during a transportation process, the lack of robustness of the configuration on the left-hand side may necessitate more frequent reorganizations around the object, which is undesirable because the transportation process will take longer and might be less accurate because the object can slide freely in more directions. Still, depending on the object, it may not be possible to improve upon this kind of robustness without additional robots. Consequently, when formulating an optimization problem describing the formation synthesis task, increasing the robustness with respect to changes of the translational error is an aim that should be incorporated into the cost function. However, this makes necessary a suitable measure to assign a numeric value to the robustness with regard to changes of the translational error.

To this end, this thesis proposes a convex hull-based approach that can be evaluated efficiently. Following Figure 5.19 from left to right, in the first step, each normal's directions are cut off at a specific distance $\delta_\mathrm{d} \gg \left\| {}^\check{\mathrm{S}}\mathfrak{e}_\mathrm{t} \right\|_2$, yielding the points $\boldsymbol{v}_i$, $i \in \{1, \ldots, n_\mathrm{c}(\boldsymbol{w})\}$. In the second step, the convex polytope $\mathcal{P}_\mathrm{R} = \mathrm{convhull}\left(\{\mathbf{0}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_\mathrm{c}(\boldsymbol{w})}\}\right)$ is calculated as the convex hull of the points $\boldsymbol{v}_i$ and the zero $\mathbf{0}$. Finally, under the assumption that the point corresponding to the error ${}^\check{\mathrm{S}}\mathfrak{e}_\mathrm{t}$ is contained in the polytope, the proposed robustness

value $J_{\mathrm{t}}^{\mathrm{f}}$ is calculated as the minimum distance of the point to the boundary of the polytope. The latter can be calculated efficiently from the convex polytope's half-space representation. With increasing value of $J_{\mathrm{t}}^{\mathrm{f}}$, the formation is more robust to changes of the translational error. Ideally, the underlying convex cone is equal to $\mathbb{R}^2$, meaning that the value of $J_{\mathrm{t}}^{\mathrm{f}}$ is mostly determined by and increasing with the parameter δ_{d}. Implementation-wise, this thesis employs the quickhull algorithm [BarberDobkinHuhdanpaa96] in its publicly available implementation to calculate convex hulls.

The previous disquisitions merely deal with the translational error. It must be ensured that the formation can rotate the object into the rotation direction currently necessary while simultaneously reducing the translational error. Thus, additionally to what has been demanded in Equation (5.54), one may require for the vector of coefficients $\boldsymbol{\alpha}$ that

$$\sum_{j=1}^{n_{\mathrm{c}}(\boldsymbol{w})} \alpha_j \left(\boldsymbol{\nu}(\boldsymbol{w})\right)_j = \mathfrak{z}_{\mathrm{r}}. \tag{5.55}$$

If, as above, the entries of $\boldsymbol{\alpha}$ are interpreted as being proportional to the required pushing forces, this means that the corresponding resultant moment about the object's center of mass is proportional to the rotational error $\mathfrak{z}_{\mathrm{r}}$. It is useful, although not immediately necessary, that a formation can rotate the object also into the opposite direction. This helps to prevent frequent reorganizations if the rotational error is small and may switch signs swiftly. The two cost terms

$$J_{\mathrm{r},1}^{\mathrm{f}} = -\sum_{j=1}^{n_{\mathrm{c}}(\boldsymbol{w})} \left|\left(\boldsymbol{\nu}(\boldsymbol{w})\right)_j\right|, \qquad J_{\mathrm{r},2}^{\mathrm{f}} = \left|\sum_{j=1}^{n_{\mathrm{c}}(\boldsymbol{w})} \left(\boldsymbol{\nu}(\boldsymbol{w})\right)_j\right| \tag{5.56}$$

jointly help achieve that goal, with the term $J_{\mathrm{r},1}^{\mathrm{f}}$ associating formations with lever arms of large absolute values with lower cost, whereas $J_{\mathrm{r},2}^{\mathrm{f}}$ ensures that they have different signs.

Having now dealt with the translational and rotational error, a final geometric requirement needs to be added to the constraints. To prevent collisions, all positions along the edge of the object should have a free Euclidean distance $d_{\min}$ that exceeds the robots' diameter, i.e., $d_{\min} > 2\,r_{\mathrm{R}}$. Finally, casting all the requirements and considerations into an optimization problem yields

$$\underset{\boldsymbol{w}\in\mathbb{R}^{N_{\mathrm{a}}}}{\text{minimize}} \quad -c_1^{\mathrm{f}}\, J_{\mathrm{t}}^{\mathrm{f}}(\boldsymbol{w}) + c_2^{\mathrm{f}}\, J_{\mathrm{r},1}^{\mathrm{f}}(\boldsymbol{w}) + c_3^{\mathrm{f}}\, J_{\mathrm{r},2}^{\mathrm{f}}(\boldsymbol{w}) \tag{5.57}$$

$$\text{subject to} \quad \exists\, \boldsymbol{\alpha} \in \mathbb{R}^{n_{\mathrm{c}}(\boldsymbol{w})} \text{ satisfying Equations (5.54) and (5.55)}, \tag{5.58}$$

$$\mathrm{frD}\left({}^{\check{\mathrm{S}}}\boldsymbol{E}(w_i),\ {}^{\check{\mathrm{S}}}\boldsymbol{E}(w_j)\right) \geq d_{\min} \quad \forall\, i \neq j. \tag{5.59}$$

Therein, c_1^{f}, c_2^{f}, and c_3^{f} are non-negative weights, and the free Euclidean distance $\mathrm{frD}\left({}^{\check{\mathrm{S}}}\boldsymbol{E}(w_i),\ {}^{\check{\mathrm{S}}}\boldsymbol{E}(w_j)\right)$ between its two arguments evaluates to infinity if there is no free line of sight between ${}^{\check{\mathrm{S}}}\boldsymbol{E}(w_i)$ and ${}^{\check{\mathrm{S}}}\boldsymbol{E}(w_j)$. The evaluation of constraint (5.59) becomes more expensive for increasing numbers of robots. However, it may be approximated by not

checking the free distance between all potential robot positions, but, e.g., only between those that are neighbors along the edge of the object in a one-dimensional sense. Furthermore, it is worth noting that the direct usage of Equation (5.54), with the coefficients of the conic combination being unbounded, can lead to degenerate cases that are not very useful in practice. For instance, in a two-robot formation, the corresponding normalized normal vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ may almost exactly oppose one another, i.e., $\boldsymbol{n}_2 \approx -\boldsymbol{n}_1$ but $\boldsymbol{n}_2 \neq -\boldsymbol{n}_1$. Indeed, in reality, they might even exactly oppose one another but may just not do so in the optimization problem due to numerics or due to an imprecise description of the object shape. The corresponding cone would then consist of almost an entire half-space of $\mathbb{R}^2$. However, some permitted motions, e.g., orthogonal to $\boldsymbol{n}_1$ into the half-space's interior, would then be almost impossible to execute practically. Mathematically, this corresponds to very large coefficients α_i being necessary in the conic combination. Hence, such degenerate cases can be prevented by additionally enforcing a maximum value for each coefficient's absolute value in the conic combination, i.e., $\|\boldsymbol{\alpha}\|_\infty \overset{!}{\leq} \alpha_{\max}$. This can also help choose the parameter δ_d in the convex hull's construction by merely choosing it considerably greater than $\alpha_{\max}$.

For the moment, it is assumed that a feasible solution $\boldsymbol{w}^\star$ of the optimization problem has been found and that the entry $\hat{w}_{\hat{\imath}}$ of $\boldsymbol{w}^\star$ is the formation position picked by robot $\hat{\imath}$. Then, the desired position of robot $\hat{\imath}$ relative to the object's center of mass can be obtained through ${}^{\check{\mathrm{S}}}\boldsymbol{E}(\hat{w}_{\hat{\imath}})$. However, to use this robot position in conjunction with the formation control setup from the previous section, the robot position must be given relative to the formation's geometric center. The corresponding nominal position of the formation center in the coordinates of and relative to the object's reference frame $\mathcal{K}_\mathrm{S}$ is given by ${}^{\check{\mathrm{S}}}_{\mathrm{c}}\boldsymbol{x}^\star = \frac{1}{N_\mathrm{a}}\sum_{i=1}^{N_\mathrm{a}} {}^{\check{\mathrm{S}}}\boldsymbol{E}(w_i^\star)$. Hence, robot $\hat{\imath}$'s desired relative position in the format required by the formation controllers can be obtained by ${}^{\check{\mathrm{F}}}_{\hat{\imath}}\boldsymbol{x}_\mathrm{d} = {}^{\mathrm{F}}\boldsymbol{S}_\mathrm{S}\left({}^{\check{\mathrm{S}}}\boldsymbol{E}(\hat{w}_{\hat{\imath}}) - {}^{\check{\mathrm{S}}}_{\mathrm{c}}\boldsymbol{x}^\star\right)$. Similarly, the offset between the formation center and the object's center of mass needs to be accounted for when following a trajectory with the object's center of mass since trajectory tracking is executed by controlling the formation center.

On another note, harking back to the beginning of the chapter, in the context of Figure 5.1, the question has been raised how to decide when the robots need to reorganize. An answer can be distilled from the formation synthesis problem since its constraints contain the strict requirements for a successful transportation. Therefore, the formation is reorganized if it does not satisfy the constraints anymore for the current translational and rotational errors. The reorganization happens by reinitializing the optimization problem and supplying it with the current errors. Then, the optimization problem is solved iteratively with a constant set of parameters, including the errors. However, this raises the question of how to solve the optimization problem, which involves two considerations.

Firstly, the optimization problem is a very intricate one. The problem is non-convex with potentially multiple local and global optima. The cost is, in general, even a discontinuous

function of the optimization variable. Altering the one-dimensional position w_i of one of the robots can not only change the direction of its contact normal in a discontinuous manner, but it may even happen that, e.g., the robot suddenly has two contact points instead of one. Furthermore, compared to optimization problems usually solved for optimal control, the problem is computationally intensive. For instance, evaluating the constraint (5.58) can be done by essentially checking the feasibility of a linear program that may require a similar effort as solving a linear program and may be implemented that way. In contrast, in the optimal control problems that have appeared in this thesis, an evaluation of the constraints merely necessitates a matrix-vector multiplication and a vectorized comparison. Beneficially, formation synthesis is not as time-critical of a task as dynamic control. A feasible, non-optimal formation is already serviceable for transportation, and the robots can wait until such a feasible formation has been found. This first consideration motivates the usage of augmented Lagrangian particle swarm optimization (ALPSO) as introduced in Section 3.1 due to it being a global optimization approach with practically non-existent requirements on the structure of the problem.

Secondly, precisely because of its complexity, the formation synthesis problem can benefit from a distributed solution. On the one hand, it can be acceptable for this slower-paced task to be executed on one leading robot, communicating the resulting formations to the other robots. On the other hand, if the computational capabilities of all robots can be leveraged to accelerate the solution process, the whole transportation process will benefit. This second consideration motivates to build upon [SedlaczekEberhard06] to realize a distributed, communication-enabled version of the ALPSO optimizer. In a sense, this can be thought of as a parallelization of the ALPSO algorithm for a distributed memory architecture. Therefore, it will be referred to as the distributed augmented Lagrangian PSO (DALPSO) algorithm subsequently.

Distributed Augmented Lagrangian PSO

Fortunately, the particle swarm algorithm itself is very accessible to parallelization. Recalling what has been said in Section 3.1, the basic algorithm consists of evaluating the cost function at a variety of particle positions and updating the particle positions according to a simple update rule, cf. Equations (3.15) and (3.16). In principle, the cost function evaluations can be done fully in parallel, and the only global information used by the iteration law is the position of the best current particle. Therefore, for computationally intensive optimization problems, for which an evaluation of the cost function takes much time compared to the exchange of data via communication, parallelization can have a significant benefit. The different program instances merely need to communicate their best current particle and the corresponding cost between iterations. Then, each program instance can determine the best particle position across all particle swarms via comparisons. The corresponding particle is referred to as the globally best particle subsequently. With

this setup, when communication is lossless, the distributed particle swarm algorithm yields the same results as the centralized one. It is also possible to communicate more rarely, which yields a different behavior than the centralized algorithm, with the individual program instances and corresponding particle populations behaving in a more decoupled manner in time spans without communication, leading to a more diverse search for optima.

However, augmented Lagrangian PSO complicates the process a bit since also Lagrange multiplier estimates and penalty factors are in play. To get an impression of the interdependencies and the communication requirements that follow from them, it is worthwhile to briefly inspect the modified optimization problem to be solved as well as the update laws of the multipliers and penalty factors. A more detailed derivation of the expressions is not necessary for that purpose and may instead be found in the literature, e.g., in [NocedalWright06] and [SedlaczekEberhard06], with the general augmented Lagrangian approach appearing in the former, and its application to PSO with all technical details in the latter.

Subsequently, the algorithm is formulated for a general nonlinear optimization problem as introduced in Section 3.1, with the optimization variable $\boldsymbol{z} \in \mathbb{R}^n$, the cost function $c(\boldsymbol{z})$, the inequality constraint functions $h_m(\boldsymbol{z})$, $m \in \{1, \dots, n_\mathrm{i}\}$, and the equality constraint functions $g_j(\boldsymbol{z})$, $j \in \{1, \dots, n_\mathrm{e}\}$. Inspired by the Lagrangian function from Equation (3.4), the augmented Lagrangian function can be defined in the form

$$\begin{aligned} L_\mathrm{A}(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{r}^\mathrm{i}, \boldsymbol{r}^\mathrm{e}) = c(\boldsymbol{z}) &+ \sum_{m=1}^{n_\mathrm{i}} \lambda_m \max\left(\left\{h_m(\boldsymbol{z}), -\frac{\lambda_m}{2\, r_m^\mathrm{i}}\right\}\right) + \sum_{j=1}^{n_\mathrm{e}} \mu_j\, g_j(\boldsymbol{z}) \\ &+ \sum_{m=1}^{n_\mathrm{i}} r_m^\mathrm{i} \max\left(\left\{h_m(\boldsymbol{z}), -\frac{\lambda_m}{2\, r_m^\mathrm{i}}\right\}\right)^2 + \sum_{j=1}^{n_\mathrm{e}} r_j^\mathrm{e}\, g_j^2(\boldsymbol{z}) \end{aligned} \tag{5.60}$$

with the penalty factors r_m^i and r_j^e corresponding to the inequalities and equalities and being subsumed in the vectors $\boldsymbol{r}^\mathrm{i}$ and $\boldsymbol{r}^\mathrm{e}$, respectively. Instead of solving the original optimization problem, the idea is to repeatedly solve an unconstrained problem with the cost function $L_\mathrm{A}(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{r}^\mathrm{i}, \boldsymbol{r}^\mathrm{e})$ instead of the original cost $c(\boldsymbol{z})$, updating the penalty factors and estimates for the Lagrange multipliers in-between.

This iterative approach works as follows. At iteration $k \in \mathbb{N}_0$, the notation $\boldsymbol{z}^{[k]}$ shall refer to the current candidate solution. The initial guess is denoted as $\boldsymbol{z}^{[-1]}$. Similarly, $\boldsymbol{\lambda}^{[k]}$, $\boldsymbol{\mu}^{[k]}$, $\boldsymbol{r}^{\mathrm{i},[k]}$, and $\boldsymbol{r}^{\mathrm{j},[k]}$ shall denote the current iterates for the multiplier estimates and the penalties. At $k = 0$, the multiplier estimates may be initialized with zeros, whereas the penalties are set to a positive constant. Then, in each iteration step, the unconstrained optimization problem with the cost function $L_\mathrm{A}(\boldsymbol{z}, \boldsymbol{\lambda}^{[k]}, \boldsymbol{\mu}^{[k]}, \boldsymbol{r}^{\mathrm{i},[k]}, \boldsymbol{r}^{\mathrm{e},[k]})$ is solved with an arbitrary, suitable optimization algorithm – in this thesis with particle swarm optimization, cf. Section 3.1. The result, i.e., the globally best particle, which need not necessarily be a fully converged solution, is used as the iterate $\boldsymbol{z}^{[k]}$. Between iterations, the Lagrange

multiplier estimates are updated by means of

$$\lambda_m^{[k+1]} = \begin{cases} 0 & \text{if } \lambda_m^{[k]} + 2\, r_m^{\mathrm{i},[k]}\ h_m(\boldsymbol{z}^{[k]}) \leq 0, \\ \lambda_m^{[k]} + 2\, r_m^{\mathrm{i},[k]} h_m(\boldsymbol{z}^{[k]}) & \text{if } \lambda_m^{[k]} + 2\, r_m^{\mathrm{i},[k]}\, h_m(\boldsymbol{z}^{[k]}) > 0, \end{cases} \tag{5.61}$$

$$\mu_j^{[k+1]} = \mu_j^{[k]} + 2\, r_j^{\mathrm{e},[k]}\, g_j(\boldsymbol{z}^{[k]}), \tag{5.62}$$

with them by tendency increasing with constraint violations of $\boldsymbol{z}^{[k]}$ [SedlaczekEberhard06]. Similarly, the penalty factors are updated in the form

$$r_m^{\mathrm{i},[k+1]} = \begin{cases} 2\, r_m^{\mathrm{i},[k]} & \text{if } h_m(\boldsymbol{z}^{[k]}) > h_m(\boldsymbol{z}^{[k-1]}) \text{ and } h_m(\boldsymbol{z}^{[k]}) > \varepsilon_{\mathrm{i}}, \\ \frac{1}{2}\, r_m^{\mathrm{i},[k]} & \text{if } h_m(\boldsymbol{z}^{[k]}) \leq \varepsilon_{\mathrm{i}}, \\ r_m^{\mathrm{i},[k]} & \text{else}, \end{cases} \tag{5.63}$$

$$r_j^{\mathrm{e},[k+1]} = \begin{cases} 2\, r_j^{\mathrm{e},[k]} & \text{if } |g_j(\boldsymbol{z}^{[k]})| > |g_j(\boldsymbol{z}^{[k-1]})| \text{ and } |g_j(\boldsymbol{z}^{[k]})| > \varepsilon_{\mathrm{e}}, \\ \frac{1}{2}\, r_j^{\mathrm{e},[k]} & \text{if } |g_j(\boldsymbol{z}^{[k]})| \leq \varepsilon_{\mathrm{e}}, \\ r_j^{\mathrm{e},[k]} & \text{else} \end{cases} \tag{5.64}$$

with the constraint tolerances ε_{i}, $\varepsilon_{\mathrm{e}} > 0$. Between iterations, the positions of the particles in the PSO algorithm are not reset. However, every time the multiplier estimates and penalties have changed, the recorded values of the augmented Lagrangian at their positions need to be recalculated. The recalculation can be done without additional evaluations of the cost and constraint functions. This requires to individually record the evaluated ingredients of the augmented Lagrangian, i.e., the constraint and cost functions' values at the particles' positions. Therefore, in a distributed calculation, these values have to be communicated along with each program instance's best particle if a reevaluation of the cost and constraints shall be prevented. In particular, the best particle's constraint function values are also needed for the updates of the multiplier estimates and penalties. Indeed, the update only needs those. Hence, also for the augmented Lagrangian PSO, it is possible to communicate after each iteration of the PSO solver, so that each program instance has the same best particle, leading to the same updates of the multiplier estimates and penalty factors. The latter is important to facilitate the comparison between the individual program instance's different solutions. With different multiplier estimates and penalties, the same particle would lead to different values of the augmented Lagrangian, prohibiting a well-defined determination of the globally best particle. However, in a crowded robotic network, it may be undesirable to communicate data after every internal PSO iteration and, thereby, to risk network congestion for an otherwise not very time-critical organizational task. Luckily, the comparability of the different ALPSO solutions is fully maintained if data is exchanged only before each update of the penalty factors and multipliers. This leads to a calculation and communication structure as depicted in Figure 5.20, where, for each constant set of multiplier estimates and penalty factors, the PSO algorithm is executed for n_{PSO} iteration steps. Outside of the PSO algorithm, the multiplier estimates

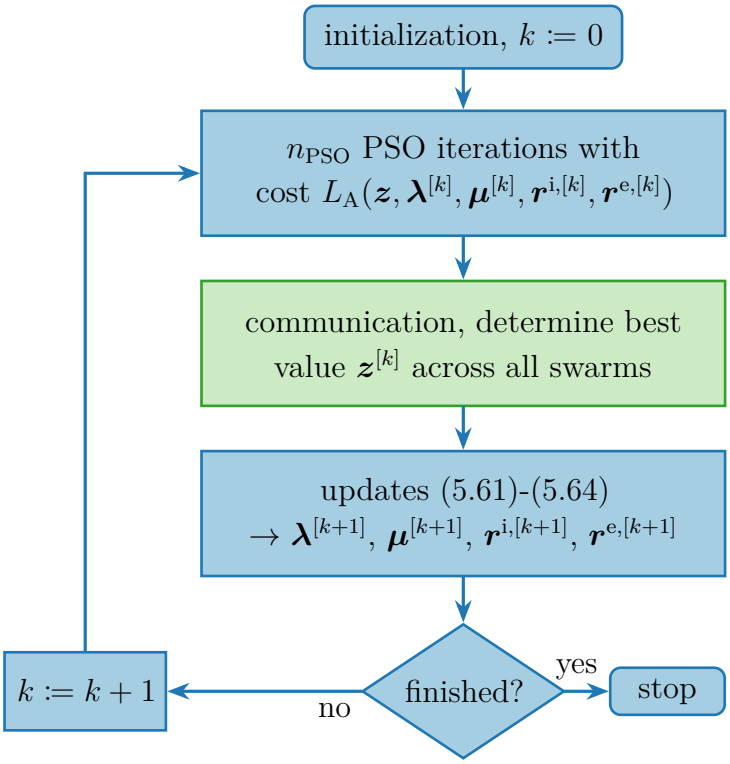


Figure 5.20: Flowchart of the distributed ALPSO algorithm

and penalties are updated. Since the PSO algorithm as well as the update of the multiplier estimates and penalties work in an iterative fashion, the overall algorithm has a doubly iterative structure, with the PSO algorithm performing inner, nested iterations. The optimization stops if a termination criterion is met, e.g., a convergence criterion or a maximum number of outer iterations n_{AL}. In the latter case, in total, the PSO problem is solved $n_{\mathrm{PSO}} \cdot n_{\mathrm{AL}}$ times. Although numerically unlikely, there may be multiple solutions with identical Lagrangian values across the particle swarms. In the case of equally good but different solutions, without loss of generality, the one coming from the program instance with the lowest identification number is chosen.

Having been formulated for a generic nonlinear optimization problem, this algorithm is readily applicable to the formation synthesis problem formulated above, with the constraint (5.58) being implemented as an equality constraint, evaluating to zero if it is satisfied and to one if it is not. Henceforward, in the organization agent responsible for formation synthesis, the algorithm is executed for a fixed number of inner and outer iterations n_{PSO} and n_{AL}, respectively. If this yields a feasible formation, it is published on a channel listened to by the robotic agents, which use the formation as their new aim formation. In this case, the ALPSO algorithm is executed again, without reinitializing the particles, multiplier estimates, and penalty factors, potentially leading to a better feasible point, which is again published on the network. This is repeated until the current

formation is infeasible according to the current translational and rotational errors. In this case, all particles across the swarms are reinitialized at random positions, and the algorithm is fed with new parameters, i.e., the current translational and rotational errors of the object. However, to hot-start the search process, the algorithm starts with the previous multiplier estimates and penalty factors. Similarly, if the previous best particle is better than all newly generated ones, it is added as the swarms' best particle. Subsequently, the organization agent dealing with formation synthesis is referred to as the formation agent.

If a robot joins or leaves the task solution, it does not contribute to the formation synthesis anymore. Implementation-wise, this is challenging to realize. The control agents publish on the network whether they are active or not, and, in turn, each control agent informs its formation agent about the activity status of the robots in the network. However, since messages may arrive at different time instances and may not be delivered successfully, there may be conflicting information on the network regarding which robot is currently active. If not implemented with care, this could lead to formation agents waiting indefinitely for messages that will never arrive because the corresponding formation agent is inactive. Furthermore, since the formation synthesis problem's size depends on the number of active robots, the formation agents may receive pieces of data of inconsistent sizes. The latter is actually used to resynchronize the formation agents' information and status after the size of the robotic network has changed. Every time a formation agent receives data of inconsistent size, it leaves its current iteration immediately, resets the algorithm, performs all inner iterations contained in the first outer iteration, and waits for incoming messages. This is done until messages with entirely consistent data have been received by all robots considered to be active. The robotic agents only consider feasible formations published by the formation agents when they fit in size to the number of robots they consider to be active. This strategy allows to deal with a changing robotic network in a reliable fashion.

On another note, some aspects of the optimization problem have a combinatorial character, e.g., which set of edges and hence contact normals are a useful combination to reduce the current errors. Thus, at a fixed number of iterations, it is to be expected that the algorithm's results become more reliable if more particles are used at the same time. Hence, assuming that each formation agent is run with a fixed number of particles, which it can evaluate efficiently and within an acceptable time span, the solution process should benefit from additional formation agents since they introduce additional particles into the network. Thus, to see whether additional formation agents can actually be beneficial in practice, a brief look at the DALPSO algorithm's performance in a representative formation synthesis scenario is warranted.

To that end, the formation synthesis for two robots around the object shape depicted in Figure 5.21 is examined. Evidently, a transportation task is usually more difficult for fewer robots since, at one point in time, they have fewer contact points and corresponding contact normals and levers at their disposal to manipulate the object. The object shall be

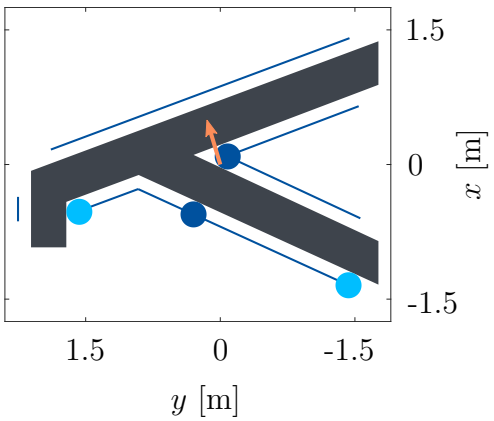


Figure 5.21: A λ-shaped object with two potential formations depicted in light blue and in dark blue with the desired translational movement depicted by means of an orange arrow

translated without rotating it, moving its center of mass from the base to the tip of the light-orange vector depicted in the image. The translational error is ${}^{\check{\mathrm{S}}}\mathfrak{e}_{\mathrm{t}} = \begin{bmatrix} 0.5 & 0.15 \end{bmatrix}^{\mathsf{T}}$ m. Each formation agent uses 15 particles, and a fixed number of $n_{\mathrm{AL}} = 40$ outer and $n_{\mathrm{PSO}} = 3$ inner iterations, respectively. Apart from the number of agents cooperating, all miscellaneous parameters are left constant. The weighting factors in the cost function are set to $c_1^{\mathrm{f}} = 40$, $c_2^{\mathrm{f}} = 10$, and $c_3^{\mathrm{f}} = 1$, whereas the parameter in the calculation of the robustness measure in the cost function is set to $\delta_{\mathrm{d}} = 20$. The formation's positions are required to have a free distance greater than twice the robots' diameter. Due to the optimization algorithm's stochastic properties, the experiment is run a thousand times each for one to six formation agents. For a two-robot formation, in a real-world scenario, usually a maximum of two formation agents, one for each robotic agent, would cooperate in the formation synthesis. Still, increasing the number further allows to obtain a better impression of the behavior of the algorithm. Furthermore, it demonstrates that the proposed software setup would allow additional robots to help with computation despite not partaking in the physical task. In the subsequent analysis, the problem is considered to be solved if the DALPSO algorithm has arrived at a feasible point, which need not be optimal. For all numbers of cooperating agents, in a share of the realizations, the problem has been solved in the first execution of the DALPSO algorithm, i.e., within n_{AL} outer iterations. However, when only one formation agent solves the problem, in 661 out of the 1000 realizations, the DALPSO algorithm had to be re-executed in a hot-started way, potentially multiple times. The number of trials, i.e., executions of the DALPSO algorithm, which it took the different numbers of agents to arrive at a feasible point is visualized in the plot on the left-hand side of Figure 5.22. The light-blue boxes span from the lower to the upper quartiles, whereas the dark-blue horizontal lines within the boxes mark the median values. Similarly, the dark-blue diamonds point out the averages, and the whiskers denote the minimum and maximum values. As the results show, by tendency, adding additional formation agents

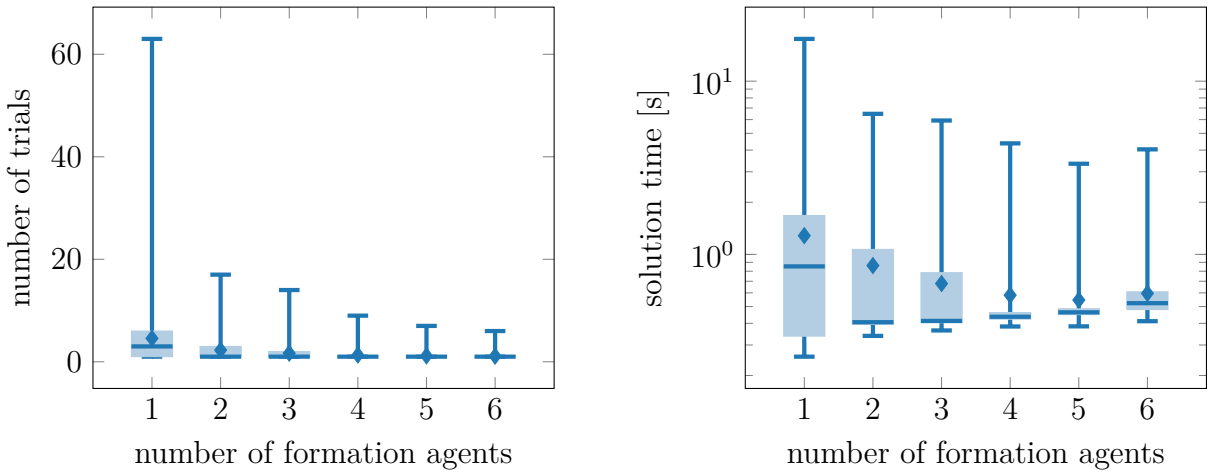


Figure 5.22: Box plots illustrating the solution performance of one to six formation agents solving an exemplary formation synthesis problem over 1000 realizations for each number of agents

more reliably yields a feasible solution within fewer trials. The median of the number of necessary trials is already at one when two agents cooperate. Similarly, the average number of necessary trials is monotonically decreasing for increasing numbers of cooperating agents. The solution times, illustrated in the same way on the right-hand side of Figure 5.22, paint a very similar picture. However, it does not make sense to add arbitrarily many formation agents due to the communication overhead. The optimal number of formation agents depends on the balance of the computational capabilities and the communication network's properties. The experiments have been run on a desktop computer with an Intel Core i7-8700 CPU operating at 3.2 GHz with six physical and twelve logical cores. Hence, communication between the formation agents happened on the same machine, yielding a comparatively low communication overhead. Thus, when running on networked robots, the network overhead will most certainly increase. However, part of its relative impact may be offset if the algorithm is run on robots with worse computational capabilities. The best cost achieved throughout the experiment was approximately -53.93, whereas the worst was about -15.47. The illustration from Figure 5.21 shows the formations associated with the lowest and highest costs in the form of light-blue and dark-blue circles, respectively. Although they are both feasible to translate the object into the required direction, it is clear that the light-blue formation would have an easier time to additionally rotate the object around its center of mass if it was required. Up to numerical accuracy, both formations have three contact points. Despite these encouraging results, the real test of the proposed algorithm and setup will be practical transportation scenarios involving a, potentially adaptive, robotic network. But before that will be looked at in the next chapter, some other matters have to be discussed. In particular, it is not yet clear which robot picks which position in the formation distributedly synthesized by the formation agents.

5.2.2 Negotiation

In an abstract sense, inter-robot negotiation is necessary to determine which robot solves which subtask in the network. Since all robots have equal capabilities, it would be possible to assign any robot to any task, merely avoiding conflicts, with every task being dealt with by exactly one robot. In this interpretation, it would be acceptable to let the robots agree upon any bijective mapping between the robots' identification numbers and the subtasks to deal with. However, for the present transportation task, where the subtasks are essentially defined by different positions around the object, this simple approach would be inefficient and complicate the robots' individual navigation around the object. On the one hand, robots may be assigned to formation positions far away from their current position, making the organization and reorganization process take more time. On the other hand, probably more critically, this may lead to a chaotic organization process, with the robots moving in different directions around the object, which would require a sophisticated navigation scheme avoiding collisions as well as deadlock situations in which two robots stop moving because they keep giving each other right of way.

Consequently, this thesis' negotiation scheme, which has already proven its usefulness in [EbelEberhard18] and [EbelEtAl21], takes into account the geometric nature of the task. As for the formation synthesis, the position negotiation is simplified by working with the N_a one-dimensional positions along the dilated object's edges as contained in the vector $\boldsymbol{w}^\star$, which is a feasible point of the formation synthesis problem (5.57)-(5.59). Subsequently, fitting to the notation used for formation control, robot $\hat{\imath}$'s relative position in the object-fixed reference frame is denoted as ${}^{\check{\mathrm{S}}}_{\hat{\imath}}\boldsymbol{x}$. In the first step, each robot projects its current position onto a point along the dilated object's edges to which it has the minimal Euclidean distance among all points of the edges. The set of such points is given by

$$\mathcal{P}\Big({}^{\check{\mathrm{S}}}\boldsymbol{E}, {}^{\check{\mathrm{S}}}_{\hat{\imath}}\boldsymbol{x}\Big) \coloneqq \Big\{ w \in \mathcal{I}_\mathrm{e} \;\Big|\; \forall\, \boldsymbol{y} \in {}^{\check{\mathrm{S}}}\boldsymbol{E}(\mathcal{I}_\mathrm{e}) \colon \Big\| {}^{\check{\mathrm{S}}}\boldsymbol{E}(w) - {}^{\check{\mathrm{S}}}_{\hat{\imath}}\boldsymbol{x} \Big\|_2 \leq \Big\| \boldsymbol{y} - {}^{\check{\mathrm{S}}}_{\hat{\imath}}\boldsymbol{x} \Big\|_2 \Big\}. \tag{5.65}$$

If this set contains more than one point, an arbitrary one is picked in a consistent manner, e.g., the one corresponding to the smallest one-dimensional position along the edges. In the following, the chosen one-dimensional position is denoted as $p^\mathrm{c}_{\hat{\imath}} \in \mathcal{P}\Big({}^{\check{\mathrm{S}}}\boldsymbol{E}, {}^{\check{\mathrm{S}}}_{\hat{\imath}}\boldsymbol{x}\Big)$. To facilitate the negotiation process, the robots communicate additional data. In each time step, robot $\hat{\imath}$ publishes its current closest one-dimensional position $p^\mathrm{c}_{\hat{\imath}}$ as well as, if already available, the latest goal position $\bar{w}_{\hat{\imath}}$ picked in a previous time step. It receives the quantities p^c_j and $\bar{w}_j$ from the other robots $j \neq \hat{\imath}$. The negotiation works most swiftly if the robots send and receive this data in every time step. It makes sense to send and receive it together in one message with the data communicated for the DMPC controller since that is published in every time step, too. Hence, the position negotiation is run in the control agent, bridging the gap between the data received from the organization agent and the dynamic controllers.

Proceeding with the negotiation process, given its own projected position as well as those received from the other robots, robot $\hat{\imath}$ sorts them in ascending order, i.e., it determines

a mapping $\pi\colon \{1,\dots,N_\mathrm{a}\} \to \mathcal{R}_\mathrm{a}$ such that $p^\mathrm{c}_{\pi(j)} \leq p^\mathrm{c}_{\pi(k)}$ for all $j<k,\ j,k \in \{1,\dots,N_\mathrm{a}\}$. If multiple robots have the same projected position, they are ordered according to their identification numbers as the second priority. The neighbors $\mathcal{N}_{\hat{\imath}}$ of robot $\hat{\imath}$ are now defined to be the robots corresponding to the two neighbors of the robot's projected position $p^\mathrm{c}_{\hat{\imath}}$ in the sequence $\left(p^\mathrm{c}_{\pi(N_\mathrm{a})}, p^\mathrm{c}_{\pi(1)}, p^\mathrm{c}_{\pi(2)}, \dots, p^\mathrm{c}_{\pi(N_\mathrm{a})}\right)$. The setup of the sequence ensures that each robot has two neighbors. In particular, robots $\pi(1)$ and $\pi(N_\mathrm{a})$ are neighbors of one another, fitting the toroidal character of the parameterized object edge. In the following, the two neighbors of robot $\hat{\imath}$ shall be given by $\mathcal{N}_{\hat{\imath}} =: \{n_1, n_2\}$. To avoid technicalities, it is subsequently assumed that robot $\hat{\imath}$'s position appears somewhere in the middle of the sequence, with its neighbors' projected positions satisfying $p^\mathrm{c}_{n_1} \leq p^\mathrm{c}_{\hat{\imath}} \leq p^\mathrm{c}_{n_2}$. With this, the idea is to define a safe zone $\mathcal{I}^\mathrm{f}_{\hat{\imath}} \subset \mathcal{I}_\mathrm{e}$ in which robot $\hat{\imath}$ can operate freely, picking a goal position that is available therein. A prudent choice is the set

$$\mathcal{I}^\mathrm{f}_{\hat{\imath}} \coloneqq \left\{ w \in \mathcal{I}_\mathrm{e} \mid \max\left(\{\bar{w}_{n_1}, p^\mathrm{c}_{n_1}\}\right) < w < \min\left(\{\bar{w}_{n_2}, p^\mathrm{c}_{n_2}\}\right) \right\}, \tag{5.66}$$

which is the open interval between either the projected positions or the previously picked goal positions of the neighbors, picking whichever is independently more restrictive. The robot now picks its new goal position $\hat{w}_{\hat{\imath}}$ from the solution $\boldsymbol{w}^\star$ of the formation synthesis problem by evaluating

$$\hat{w}_{\hat{\imath}} \coloneqq \max_{w^\star_i \in \mathcal{I}^\mathrm{f}_{\hat{\imath}}} \left(\left\{ w^\star_1, w^\star_2, \dots, w^\star_{N_\mathrm{a}} \right\} \right). \tag{5.67}$$

If the optimization problem (5.67) is infeasible, the arithmetic mean of the interval boundaries of $\mathcal{I}^\mathrm{f}_{\hat{\imath}}$ is chosen. If $\mathcal{I}^\mathrm{f}_{\hat{\imath}}$ is empty, robot $\hat{\imath}$ may simply stay stationary until it is not. The maximization in (5.67) lets the robots explore the whole boundary of the object. For instance, if all robots' initial projected positions are smaller than all entries of $\boldsymbol{w}^\star$, they will move into position in-order, without overtaking one another along the edges and instead waiting until a desirable position $w^\star_i$ appears in the window of operation defined by $\mathcal{I}^\mathrm{f}_{\hat{\imath}}$. This also defines a natural movement direction around the object, corresponding to increasing parameters of ${}^\check{\mathrm{S}}\boldsymbol{E}(\cdot)$ and minimizing the risk of collisions. Nevertheless, if all robots solve the problem (5.67) simultaneously, conflicts may arise with robots picking the same position if a new formation has just been synthesized. Fortunately, this can be solved by only letting non-neighboring robots pick their goal positions in parallel. Depending on whether the number of robots cooperating is even or uneven, due to the parameterized edges' toroidal structure, it takes either two or three time-steps until every robot has picked a goal position in a conflict-free manner. Robot $\hat{\imath}$'s two-dimensional goal position ${}^\check{\mathrm{S}}_{\hat{\imath}}\boldsymbol{g}_\mathrm{f} \in \mathbb{R}^2$ can finally be calculated by simply evaluating the parameterization of the dilated object's edges, i.e., ${}^\check{\mathrm{S}}_{\hat{\imath}}\boldsymbol{g}_\mathrm{f} \coloneqq {}^\check{\mathrm{S}}\boldsymbol{E}(\hat{w}_{\hat{\imath}})$.

In practice, for increased safety, the interval $\mathcal{I}^\mathrm{f}_{\hat{\imath}}$ may be clipped by a safety distance at both ends, e.g., by a value greater than the robot radius but smaller than the minimum distance d_min appearing in the formation synthesis problem (5.57)-(5.59). Furthermore, with the objects transported potentially being non-convex, it need not be trivial to

navigate from the robot's current position to its goal position ${}^{\check{\mathrm{S}}}_{i}\boldsymbol{g}_{\mathrm{f}}$ since the line of sight may be blocked by the object. In particular, for this reason, it is not generally possible to immediately use the formation controller. Instead, the robots use an individual navigation scheme to move into formation, informing the other robots when they are in formation so that dynamic formation control can be switched on. However, a general navigation scheme, planning shortest paths around the object, may not let the robot navigate into the natural movement direction along the object's edges as implied by the negotiation scheme. A practical way to enforce the general movement direction is to not supply the navigation scheme with the goal position corresponding to $\hat{w}_i$ but to a one-dimensional position into the direction of $\hat{w}_i$, using a small look-ahead distance. As the applications in Chapter 6 show, the scheme, implemented this way, works in a very robust fashion, letting the robots pick positions and therefore organize in a self-coordinated, conflict-free manner. Henceforth, it may also serve as a blueprint to solve other negotiation and task-allocation schemes that operate on a one-dimensional ordered set with a similar toroidal character. Extensions to higher dimensions may be similarly interesting.

5.3 Mapping and Path Planning

In this section, the navigation of an individual robot around the object is treated first, with navigation and path planning for the transported object through an obstacle-ridden environment following subsequently. Navigation and path planning are very well-researched fields and not the main focus of the thesis. However, the design of appropriate methods is still necessary to obtain a functioning all-around scheme, especially since the cooperative transportation setting does bring about a few characteristic features and requirements. Using similar foundational techniques, a look at individual navigation, despite being common to robotics, serves as a good preparation for the more intricate global navigation.

5.3.1 Individual Navigation

As delineated in Section 4.2, the evasion of obstacles is time-critical and, therefore, best run in the control agent alongside real-time control. Hence, the individual navigation, including mapping and path planning, need to be executed reliably within the control sampling time. Consequently, at best, the calculation effort is upper-bounded by a constant. Therefore, a discretization-based approach, operating on an equidistant grid, is employed. While it will always be an approximation due to its finite resolution, the latter also means that the calculation effort is strictly upper-bounded if the resolution is fixed. A comparatively high accuracy is desirable around the object since most activity will happen in the vicinity of the object's edges. However, when joining the transportation task, the robots may start far away from the object, meaning that a large area would have to be discretized, which

runs contrary to achieving a high accuracy around the object at a fixed resolution. This conflict can be resolved as follows.

Each robot uses information from its distance sensors as well as the object's known shape to build its personal grid-based map at a predefined resolution. The map moves and rotates with the object, i.e., with its body-fixed frame of reference $\mathcal{K}_S$. The map's scale varies with the distance of the robot to the object's center of mass so that differently large areas can be covered using the same resolution. Hence, the number of grid cells per area, and thereby the map's accuracy, increases as the robot comes closer to the object. The map's minimum size contains merely the object plus just enough of its surroundings to allow for safe maneuvering of the robot around the object.

Apart from this scaling trick, the approach builds upon the very classic idea of constructing a connectivity graph from a dilated version of the map, containing all obstacles in an enlarged form so that the circular robot can access the remaining unoccupied grid cells without collisions. The map employed is a binary map and can be equivalently thought of as a pixelated image with the pixels taking two different values, depending on whether the corresponding grid cell is considered to be occupied or not. Thus, in the first step, the object is drawn into the image. Then, the point-cloud data representing points of objects sensed by the robot's distance sensors are added. It is enough to only consider measured points closer to the robot than its goal position. Similarly, for this thesis' local navigation purposes, it suffices to only consider the current distance measurements without memorizing sensed obstacles. However, the map-based approach would allow for an extension in that regard. The dilated map is constructed by enlarging all obstacles and the transported object by the robot radius plus potentially an additional safety distance. Implementation-wise, being a typical operation in computer vision with optimized libraries being available, the dilation process and similar operations can be done efficiently for pixelated maps. The applications in this thesis use the OpenCV library [KaehlerBradski16] to manipulate and visualize all kinds of pixelated maps that are used for navigation purposes.

The connectivity graph is constructed so that every center of an unoccupied pixel in the dilated map corresponds to a node in the graph. Each node is adjacent to at most eight other nodes, which correspond to neighboring pixels that are unoccupied. Hence, it is assumed that the robot can move from one pixel to the other horizontally, vertically, and diagonally. The A* algorithm as introduced in Section 2.2.1 is used to find the shortest path from the start to the goal. In general, for a dilation radius greater than the robot radius, the optimal position within the formation will be inside the area covered by the dilated object and hence not accessible. In this case, the actual goal position is projected onto the nearest free grid cell. The path is then planned to the latter and completed by adding a straight-line segment to the actual goal position. Instead of the cell center corresponding to the goal position, the robot plans the path precisely to the goal's position within the cell. Similarly, if the robot's current position is inside a blocked region of the

dilated map, it is also projected to the closest free position as a starting point for the graph search algorithm. The resulting path is tracked by picking a point ahead by a certain distance and using it as a setpoint for a linear MPC controller for tracking. The latter can be thought of as a simpler, non-distributed version of the cooperative distributed controller introduced in Section 5.1.1. A schematic example showing the principle of the map building and path planning process at a deliberately low resolution is given in Figure 5.23, for which the steps can easily be retraced by hand. Actual, representative outputs of the resulting implementation for two robots moving into formation are given in Figure 5.24. These can be viewed in real time during the simulation or experiment. In the pixelated maps, the robots are depicted in light blue, the dilated object in light gray, the original object in dark gray, and the current planned path is drawn in dark blue. Therein, the distance-based scaling of the map becomes apparent.

Indeed, if the object's desired path is known a priori, the methods described so far already constitute a functioning and versatile transportation scheme. A known path is followed by tracking a path point that is a certain distance ahead of the object pose's projection onto the path. As a matter of fact, most of the applications later in the thesis will focus on the setting with an a priori known path, simply because it allows to inspect the properties and interaction of the organization and control schemes without any further schemes clouding the judgement by introducing additional complexities. It is organization and control that are particularly interesting and different in distributed robotics, with organization even becoming relevant only because of the cooperative aspect. Still, to conclude the mostly methodological part of the thesis, a tailor-made global navigation scheme is introduced. It enables a self-reliant transportation through obstacle-ridden environments.

5.3.2 Global Navigation

Global navigation is dealt with by a separate organization agent, which subscribes to distance sensor data published on the network. Apart from the measured distance values and the corresponding measurement directions, the distance sensor data also contains the position from which the measurement was taken. It is assumed that the global navigation agent knows the object's pose, which is realized by subscribing to the channel on which the object's pose is published. Furthermore, the goal of the transportation, which is the aim pose the object shall have at the end, is also known to the global navigation agent. The global mapping agent runs continuously and concurrently to the control agents, planning paths from the latest received pose of the object to the goal pose. Indeed, due to it not being a time-critical task, global navigation is run within only one of the robotic agents. Its algorithmic design and implementation do not treat data coming from its corresponding robotic agent any different than data received from the other agents. Hence, it may be run on any of the robotic agents or potentially even on a supporting robot or drone accompanying and supporting the transportation process without getting involved

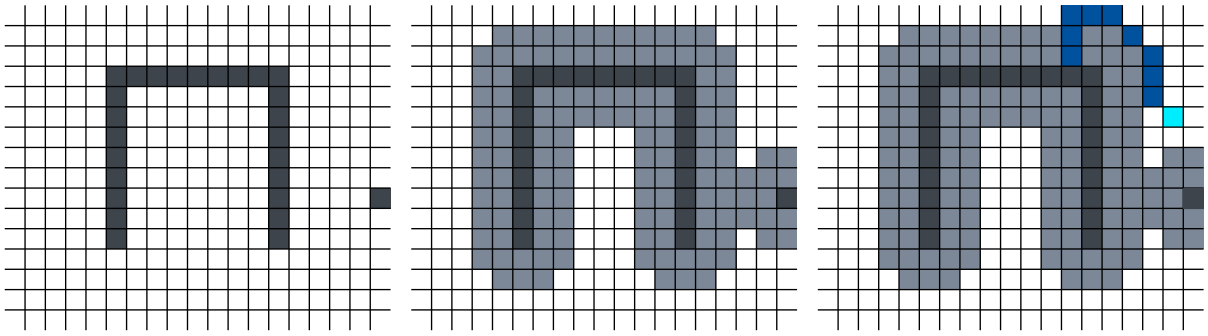


Figure 5.23: Illustrative pixelated maps depicting, from left to right, the U-shaped transported object with a single-pixel obstacle, the dilated version of the map, and a light-blue robot with a dark-blue planned path to its goal position

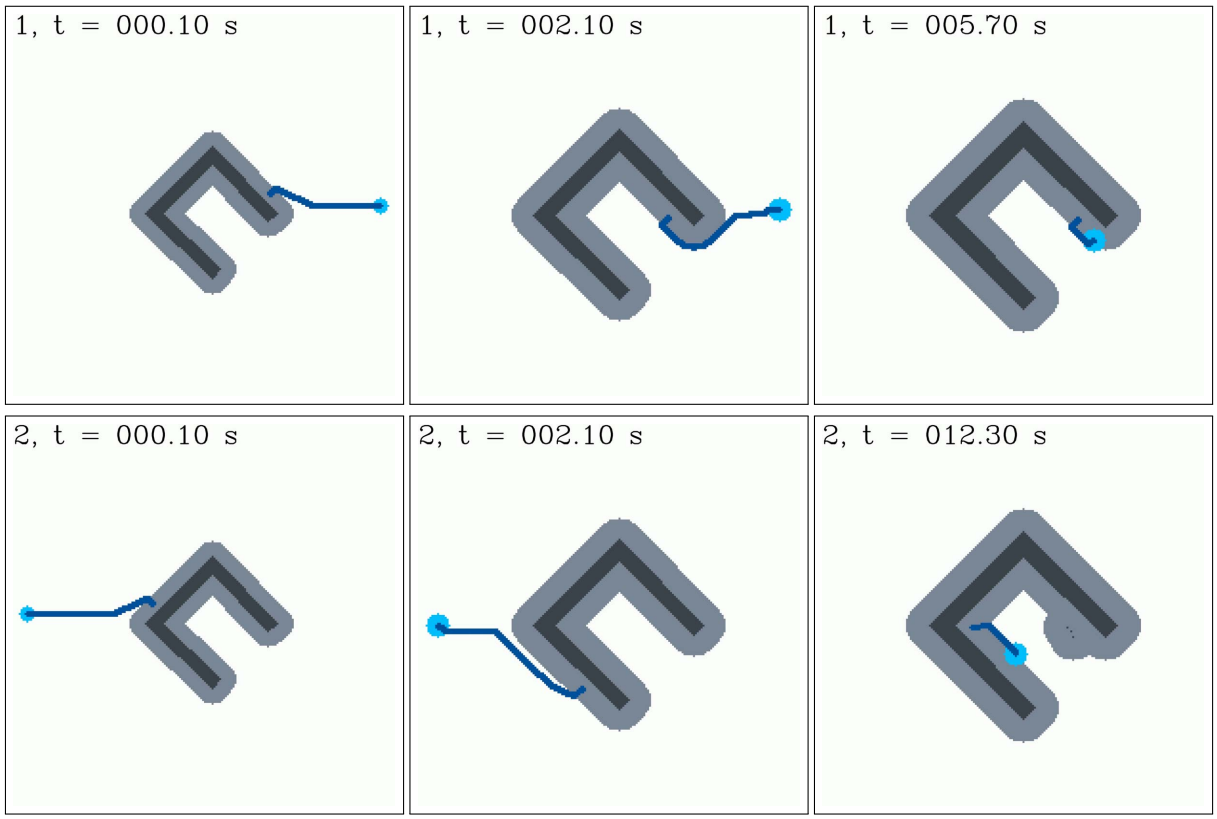


Figure 5.24: Program output showing the pixelated maps for the simulated local navigation of two robots, with those of robot 1 depicted in the upper row from left to right, whereas those of robot 2 are shown in the lower row

physically. If, for instance, the robot running the global navigation agent breaks down, it would be conceivable that the remaining robots use their still-running control agents to safely pause transportation until global navigation can be resumed on another robot. The control agents subscribe to the path resulting from global mapping and path planning, which is published by the global navigation agent. It is then tracked by the formation in the same way as a known path. The proposed scheme can deal both with predefined maps and maps constructed online from sensor data. Mixtures of both are also possible, with parts of the workspace being known a priori, whereas some unknown obstacles are added online through sensor data.

In any case, global navigation is decidedly more intricate than the local problem discussed previously. In unknown workspace environments, sensor information must be incorporated during operation to map and memorize the workspace and potential non-convex obstacles contained therein. Furthermore, the transported object's shape is not as simple as the robots' circular shape. It may be non-convex, and it is, in general, not rotation-symmetric. However, rotating it may be necessary to find a path that reaches the goal point. Consequently, checking for collisions between possible configurations of the object and obstacles cannot be circumvented by merely constructing a dilated map. Also, the simple fact that rotation does matter means that the description of the object's pose needs three generalized coordinates. Hence, the path planning problem becomes three dimensional, with the third dimension corresponding to the object's orientation. Finally, it is not sufficient if the planned path is free of collisions between the transported object and the obstacles. The robots need some additional space to maneuver between the object and the obstacles. Therefore, to allow the robots to maneuver, the path planning algorithm needs to reserve an area around the object.

Fortunately, all these challenges can be dealt with by a scheme expanding upon the techniques that have been useful already in Section 5.3.1. Once again, a pixelated, two-dimensional map of the workspace is constructed with a fixed resolution and, this time, also with a fixed scale. All a priori known obstacles are added to it at the start. Whenever new sensor information is received, it is also added to the map. This is done by using the full information brought about by the measurement. Not only are cells marked as occupied if a point is measured in their area, but they are also marked as being unoccupied if the measurements allow concluding so. If the sensors pick up points in the area around the object reserved for robot movement, they are not added to the map since they most likely belong to one of the robots instead of to an obstacle. Moreover, measurements within the area of the transported object are also not added since the map shall only contain obstacles. If the a priori known obstacles are known in a precise fashion, points measured in the vicinity of known obstacles may be disregarded, too. Similarly, the robot position from which the measurement was taken can be marked as free on the map since the robot cannot have been inside an obstacle.

In a copy of the map, all objects are dilated by a dilation radius greater than the robot's radius. For increased safety, uncertainties, e.g., in the form of measurement inaccuracies, may be taken into account by a larger dilation radius. So far, the strategy seems similar to the one described for individual navigation. However, the connectivity graph's construction is more intricate since the graph does not follow immediately from the two-dimensional map. Different from before, the path is planned on a three-dimensional grid. Two of its dimensions still correspond to the directions of the plane the robots and objects move in and, thus, to the two-dimensional map of the workspace. Hence, each layer of the grid has the same resolution as the workspace map. The third dimension corresponds to different orientations of the object, with the grid cells' centers discretizing the angle with respect to a reference orientation in an equidistant manner between $-180°$ and $180°$. The grid's outer edges in rotation direction correspond to angles of $-180°$ and $180°$ and hence encode the same orientation of the object, giving rise to a toroidal character of the grid.

The idea is to construct the connectivity graph from this grid. It shall contain a node for each cell center that corresponds to a collision-free object pose. Each node is adjacent to at most 26 other nodes, which correspond to collision-free, neighboring grid cells, including neighbors in the diagonal directions of the grid. Due to the grid's toroidal character in the rotation direction, the nodes in the upper-most grid layer, corresponding to angles close to $180°$, are suitably connected to the nodes in the lowest grid layer, which correspond to angles close to $-180°$. Once more, the A* algorithm is used to find a shortest path through the graph, leading from the current object pose to the goal pose. The connectivity graph's construction is only done implicitly while the A* algorithm searches for the shortest path. Whenever the algorithm requires the information whether the object pose symbolized by a grid cell is free of collisions, i.e., whether there actually is a corresponding node in the connectivity graph, an explicit collision test is performed. This can be done in an efficient manner. To this end, in a preprocessing step, pixelated snapshots of the object in all the different orientations encoded by the grid centers are recorded. These snapshots are of the same spatial pixel density as the one used for the workspace map. The snapshots are then dilated with a dilation radius that consists of the robot radius plus a safety distance and a further, additional distance that accounts for the space the robots need to maneuver around the object while reorganizing. Whenever it becomes necessary to check for collisions, the dilated object snapshots are superimposed onto the dilated map of the workspace. Then, checking for collisions comes down to a simple binary per-pixel operation that, once more, can be evaluated efficiently. This idea is sketched in Figure 5.25.

In the spatial directions, the edge weights of the connectivity graph are chosen as the Euclidean distances between the grid centers. Naturally, there is no universal notion of distance when comparing rotations and translations. Hence, the relation of the edge weights between a translation by a unit of length and a rotation by a unit of rotation is a tuning parameter. The same consideration applies to the design of the resulting path's parameterization.

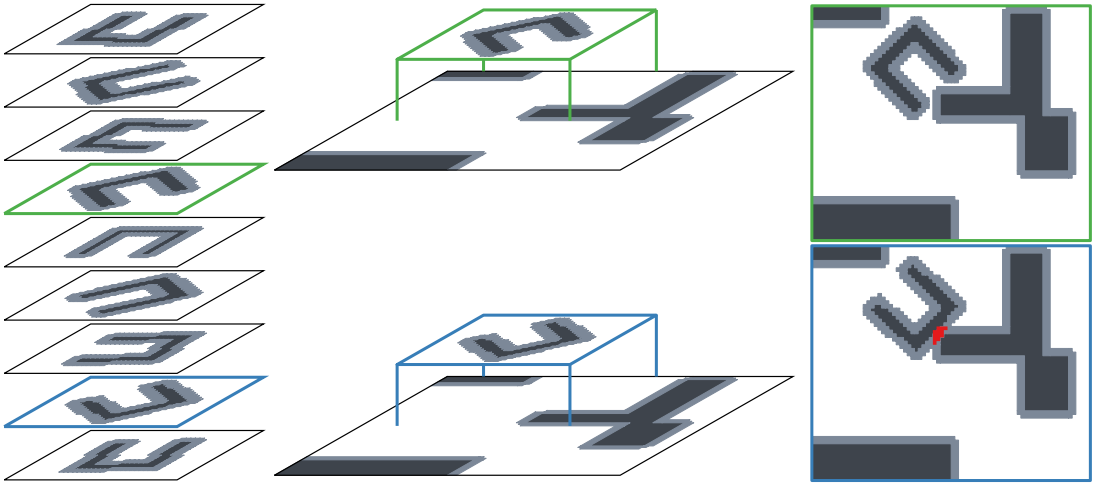


Figure 5.25: Principle of checking for collisions for global path planning, with pixels in collision shaded in red

It is worth noting that, even though only one agent deals with global navigation, the concurrent software architecture still brings about challenges. For instance, it needs to be ascertained that the workspace map is not modified while the A* algorithm uses it in collision queries since this may lead to undefined behavior. Hence, the A* algorithm uses a static copy of the map, with the original map not being modified until the data has been copied. The proposed scheme's full functionality and performance will become apparent in intricate simulation scenarios examined in the following chapter. In the meantime, to point out an example, the object transportation path depicted in Example 2 early on in the thesis has been planned with the method described here. Therein, the obstacles have been known a priori.

Chapter 6

Results from Cooperative Transportation

When looked at superficially, the challenge in cooperative robotics is the physical cooperation of the robots. However, beyond the physical level, tackling a practical task in the field also usually requires the finely coordinated cooperation of different methods and software components. Consequently, while some of the involved methods have already been put to work in several examples and simulative as well as experimental analyses, the interplay of all components and methods needs investigation. Hence, in this chapter, all that has been proposed, deliberated, and analyzed in the preceding chapters is put into service for cooperative transportation, revealing whether the proposed approach can deliver on its promises. At first, a simulative investigation is performed to examine the fundamental qualities of the scheme. These include whether the scheme can accommodate objects of intricate non-convex shapes and whether it can deal with different numbers of robots cooperating as well as with robots joining and leaving the transportation process. The scenarios assume that the object shall follow a known path. Subsequently, exemplary experiment scenarios are used to study whether the involvement of robotic hardware raises unprecedented problems not reflected in the simulations. Scenarios with different object shapes also highlight the versatility of the scheme. Finally, since the area available for experimentation is of limited size, navigation and transportation through obstacle-ridden environments are examined at the example of large-scale simulation scenarios.

6.1 Simulative Investigation

All simulations in this chapter use the software structure devised in Section 4.2 and summarized in Figure 4.2, with each robotic agent consisting of a control agent and a formation agent. As previously in the formation control simulations, the simulation agent,

which is a Matlab program, employs the robot model from Section 4.3.2. The contact forces between the robots and the transported object are calculated using a penalty-force approach. The object is modeled by means of the dynamics $\boldsymbol{M}_\mathrm{o}\ddot{\boldsymbol{q}}_\mathrm{o} + \boldsymbol{D}_\mathrm{o}\dot{\boldsymbol{q}}_\mathrm{o} = \boldsymbol{f}_\mathrm{o}$, with the object's pose being described by $\boldsymbol{q}_\mathrm{o} \in \mathbb{R}^3$, and $\boldsymbol{M}_\mathrm{o}$ and $\boldsymbol{D}_\mathrm{o}$ being the diagonal mass and damping matrices, respectively. The vector $\boldsymbol{f}_\mathrm{o} \in \mathbb{R}^3$ contains the contact forces and the resulting moment acting on the object. The sensor simulation agent simulates the robots' distance sensors, with them having a limited accuracy and range. As far as not stated otherwise, the sensors are simulated with a range of 1.25 m, with the robot obtaining 64 distance measurements equally distributed around the robot's perimeter and the measurements having a radial accuracy of 0.05 m. For an efficient evaluation, in a preprocessing step, the constrained Delaunay triangulation of the object is calculated and saved, making it straightforward to check algorithmically whether a measurement point is contained in the object. The maximum directional robot velocity commanded by the controllers is set to $u_\mathrm{max} = 0.4\,\mathrm{m/s}$, and the motor moments are saturated at an absolute value of 0.05 N m. As before, the motor controllers employ a sampling time of 0.01 s, whereas the DMPC controllers run with a sampling time of $T_\mathrm{s} = 0.05\,\mathrm{s}$, which is the sampling time also used for recording the results. The DMPC controllers use a horizon of length $H = 20$. The weighting matrices in their cost functions are set to $\boldsymbol{R} = 0.6\,\boldsymbol{I}$, $\boldsymbol{T} = 1 \cdot 10^4\,\boldsymbol{\mathcal{D}}\boldsymbol{I}$, and $\boldsymbol{\mathcal{D}} = \boldsymbol{I}$. The formation agents use the same parameters as in Section 5.2.1, albeit with 80 particles per agent. Some of the scenarios studied in this section are based on scenarios that have turned out to be challenging and instructive in preceding studies of the transportation task [EbelEberhard18, EbelEberhard19, EbelEtAl21]. In contrast to this thesis, the preliminary work uses predecessor schemes and employs a simpler, holonomic model for the simulation of the omnidirectional mobile robots.

The transportation task's character and the proposed scheme's virtues already become evident in the first considered scenario, with only a few robots cooperating. In this scenario, a U-shaped, and therefore non-convex, object shall be transported along a path consisting of straight-line segments. At each corner of the path, the object shall be rotated by $+90°$ or $-90°$ so that during translational movement, always the same edge of the object is facing forward. The object is of mass $m_\mathrm{o} = 4\,\mathrm{kg}$ and the diagonal of the damping matrix is set to $\mathrm{diag}(\boldsymbol{D}_\mathrm{o}) \coloneqq \begin{bmatrix}10\,\mathrm{kg/s} & 10\,\mathrm{kg/s} & 10\,\mathrm{kg\,m^2/s}\end{bmatrix}$. It is assumed that the object's mass is evenly distributed, with its center of mass, therefore, being located outside of the object. Initially, the object's center of mass is placed on the first path point at $x = y = 0$. The left-hand side of Figure 6.1 shows the simulation result for two cooperating robots. The iconography in these and subsequent similar figures is the same as in previous figures, with the transported object and the trajectory of its center of mass drawn in dark gray. Along the path, different snapshots of the robots and the object are depicted. They correspond to the times printed alongside them, with darker colors associated with later time steps. To indicate the robots' initial positions, the robots' trajectories from the start to the time point of the first snapshot are depicted in light blue. As can be seen, the path is

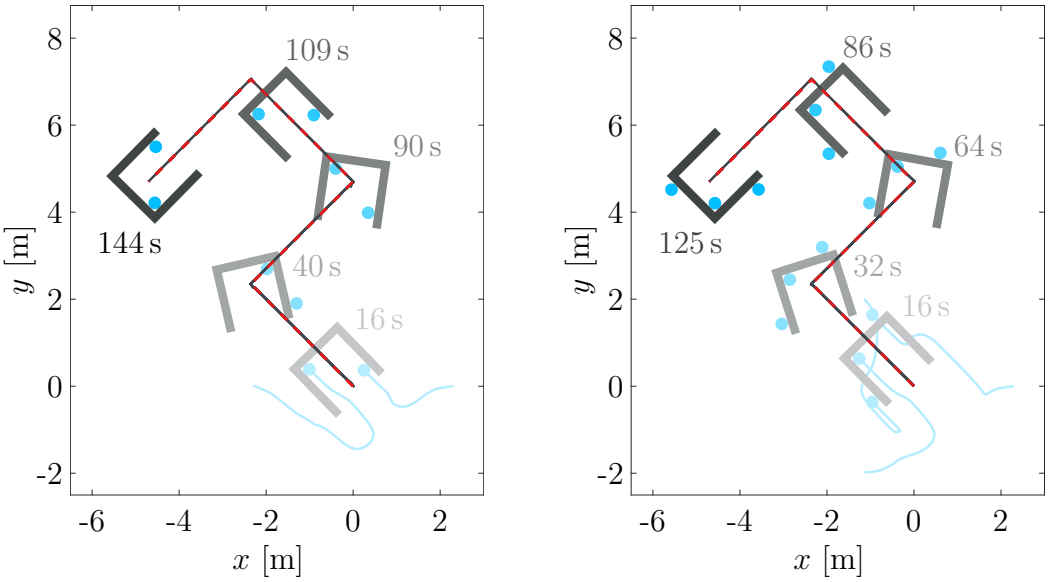


Figure 6.1: Simulation results for two and three robots transporting a U-shaped object

tracked very accurately with the object's center of mass. The robots reorganize multiple times to manipulate the object's pose as necessary. Quantitatively, the maximum and median positional path tracking errors evaluate to $\mathfrak{e}_{\mathrm{p}}^{\max} \approx 0.042\,\mathrm{m}$ and $\mathfrak{e}_{\mathrm{p}}^{\mathrm{med}} \approx 0.0038\,\mathrm{m}$, respectively. These are determined by calculating the minimum Euclidean distance of the object's center of mass to the path in each time step between the first movement of the object until the end of the path is reached. Hence, in this simulation devoid of exogenous disturbances, the object barely leaves the path, showing that the DMPC$^{\mathrm{p}}$ scheme is indeed able to position the robots and thereby the object very accurately. Until the end of the path is reached, the robots reorganize twice, making them take approximately 132 s to reach the goal when neglecting the time it takes them to get into formation initially since that mostly depends on how well the initial positions of the robots fit the synthesized formation. In contrast, in the three-robot result on the right-hand side of Figure 6.1, the formation is not reorganized at all. Consequently, the robots reach the end of the path a bit more quickly, with the transportation taking about 114 s after initial formation acquisition. However, the quantitative accuracy, which is already pristine for two robots, is not perceptibly different, with the maximum and median positional tracking errors $\mathfrak{e}_{\mathrm{p}}^{\max} \approx 0.019\,\mathrm{m}$ and $\mathfrak{e}_{\mathrm{p}}^{\mathrm{med}} \approx 0.0042\,\mathrm{m}$, respectively. Subsequently, quantitative performance figures are only given if they provide additional insight. As it will turn out throughout the analysis, it is mainly the more qualitative differences that reveal most about the properties of the proposed scheme, with the quantitative accuracy usually easily meeting the demands and, thus, not being a very valuable differentiator between different scenarios and settings.

In that regard, as a more subtle difference, after having reached the end of the path, the two-robot group still reorganizes, trying to perfect the positioning of the object. In contrast,

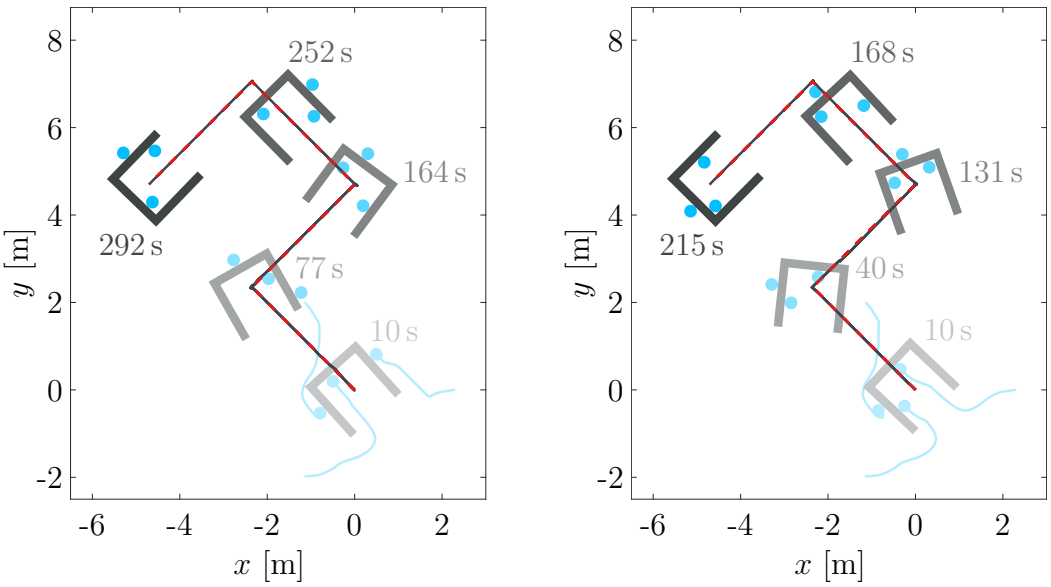


Figure 6.2: Two simulation results for three robots transporting a U-shaped object, with the formation agents picking random feasible formations

the three-robot group can fully control the object's pose without reorganizing. Hence, it seems that, for three robots, transporting the U-shaped object along the prescribed path is a comparatively simple task. Nevertheless, part of the perceived ease of transportation can be attributed to the setup of the formation synthesis optimization problem. This can be seen by running simulations where just any feasible formation is randomly picked by the formation agents, which is realized by simply setting the cost function to zero. Two exemplary results for this setup are depicted in Figure 6.2. Even at first glance, a qualitative difference between the formations depicted in Figure 6.2 and those from Figure 6.1 becomes apparent. The ones from the latter, using the proper cost function, always have large levers at their disposal, with the robots always being located in corners or at the ends of the edges of the dilated object. This is not the case for the formations from Figure 6.2. Similarly, the variety of inner normal directions is usually greater in the formations resulting from the proper cost function, making visible the influence of the translational robustness term in the cost. Of course, since the modified formation agents pick any feasible formation, there is a chance that they pick more favorable ones, potentially even the same one as on the right-hand side of Figure 6.1. However, this is not any more likely than picking one of the more unfavorable formations from Figure 6.2. Nevertheless, the number of necessary reorganizations may be very different depending on which formations are picked. For instance, in the left-hand result from Figure 6.2, the robots reorganize 14 times until they reach the path's end, whereas four reorganizations happen in the right-hand case. On the obverse side, even in the case of the full cost function, due to the nature of the optimization problem, the formation agents may converge to an optimum that is only a local one. For instance, there may even be multiple local

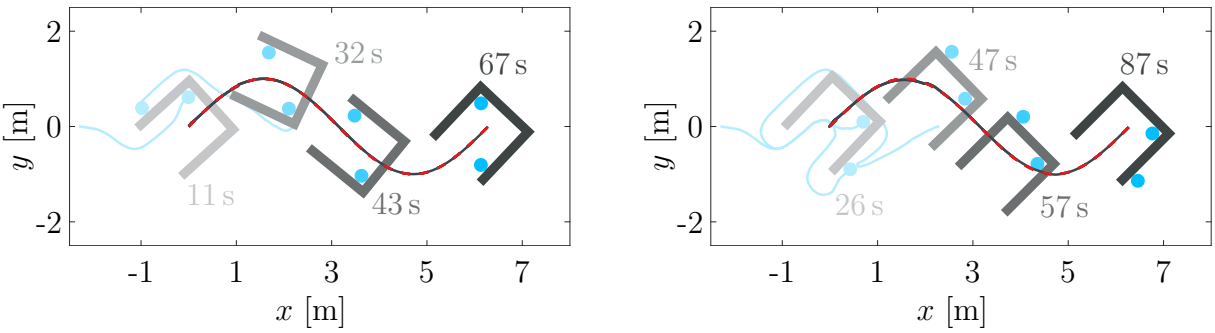


Figure 6.3: Simulation results for two robots transporting a U-shaped object, with the object being rotated continuously in the left-hand side result, whereas the desired orientation is kept constant on the right-hand side

minima of identical cost in the case of symmetry of the problem. Similarly, despite the best efforts encoded in the cost function, a low-cost formation may still be useful in one situation but not in another. After all, the future course of the path is not explicitly included in the optimization. However, even in the worst case from 100 simulation runs, the three-robot group using the original cost function has at most needed two reorganizations to reach the goal, reliably yielding a very satisfactory tracking performance, with the robots always having levers that are of maximal absolute value along their respective edges. Nevertheless, having given a bit of intuition on the influence of the setup of the formation synthesis problem, it is not the aim here to delve into a statistical analysis of the influence of cost function settings and parameters on the transportation performance for a very specific scenario. It is of more immediate methodological value to investigate the scheme's versatility for a constant set of parameters but over a wider range of scenarios. After all, robotics is all about automating behavior without the human-lead tuning of parameters to specific scenarios. In that sense, it is not crucial that the robots complete a specific scenario with a minimal number of reorganizations, with the control and organization parameters being tuned accordingly, but rather that the transportation is successful with satisfactory accuracy in a wide range of different scenarios. Thus, henceforward, always the same setup of the formation synthesis problem is used, with the same set of parameters as in the simulations from Figure 6.1.

In the previous simulation results, most reorganizations happen around the corners of the path, where the object needs to be rotated in-place, making it seem that this kind of cornered path is rather challenging for transportation. Nevertheless, it should be examined whether a good transportation performance can also be obtained for paths that require the simultaneous translation and rotation of the object. Such a scenario can be found on the left-hand side of Figure 6.3, where also the transportation direction changes in a smooth manner. The desired orientation of the object is once again designed such that always the same edge of the U-shaped object faces the direction of translation. The formation is

reorganized twice until the path's end, with the robots delivering a smooth transportation performance. Indeed, it may seem that rotating the object along the path in this way might benefit the transportation since the local pushing direction relative to the object-fixed reference frame is constant. Hence, it might be more challenging to actually move the object along this sinusoidal path while keeping its orientation constant. A result for this case is depicted on the figure's right-hand side, with the robots also reorganizing twice before reaching the end. Evidently, the robots tend to pick different formations than in the left-hand case, with the formations enabling a similarly successful transportation.

These findings cannot be generalized in a natural manner to completely different setups with objects of different shapes. After all, there is no obvious explicit functional relationship between the object shape and, e.g., the number of robots necessary for a reliable and accurate transportation performance along specific types of paths. Hence, the depicted results need not be representative for different kinds of transportation tasks. However, if a group of robots ever faces a situation within a transportation scenario where the transportation performance becomes unsatisfactory, e.g., because they do not find a formation favorable for transportation or because they need to reorganize too often at the expense of transportation time and accuracy, distributed robotics allows to let further robots join the transportation task. With every robot added, the formation can have at its disposal at least one additional contact normal, with the formation agents striving to enlarge the cone spanned by the inner normals while including large levers of both signs. Hence, if enough robots are present, the formation will be able to translate and rotate the object in any direction without reorganizing. Thus, leveraging the flexibility of distributed robotics is one way to ensure a successful transportation. In consequence, it is prudent to examine whether the proposed approach indeed scales well to larger numbers of robots. It has already been seen in Section 5.1 that the DMPC-based formation controller does scale in that regard. However, it will still be interesting to see whether the overall transportation scheme also works well for a larger group of robots, without, e.g., negotiation, organization, and navigation around the object succumbing into chaos. To this end, a larger, Σ-shaped object of mass $m_\mathrm{o} = 8\,\mathrm{kg}$ is transported along a path that combines aspects from the previous scenarios. At first, the object shall be translated along a straight-line segment, followed by a sinusoidal section. In both of these segments, the object's orientation shall stay the same. In the third section of the path, the object shall be moved along a straight line while rotating it linearly, reaching a rotation of $90°$ at the end of the path. A collection of exemplary results for increasing numbers of robots is depicted in Figure 6.4. Once again, as more robots are added, the robots acquire full control over the pose of the object, leading to a very consistent and almost perfect tracking performance without reorganizations. Only the two-robot group does reorganize in these results, acquiring a new formation eleven times until the goal is reached and also concluding the transportation successfully.

While the results looked at so far provide great promise with regard to the scheme's ability to accommodate different scenarios and numbers of robots, it is not yet clear whether

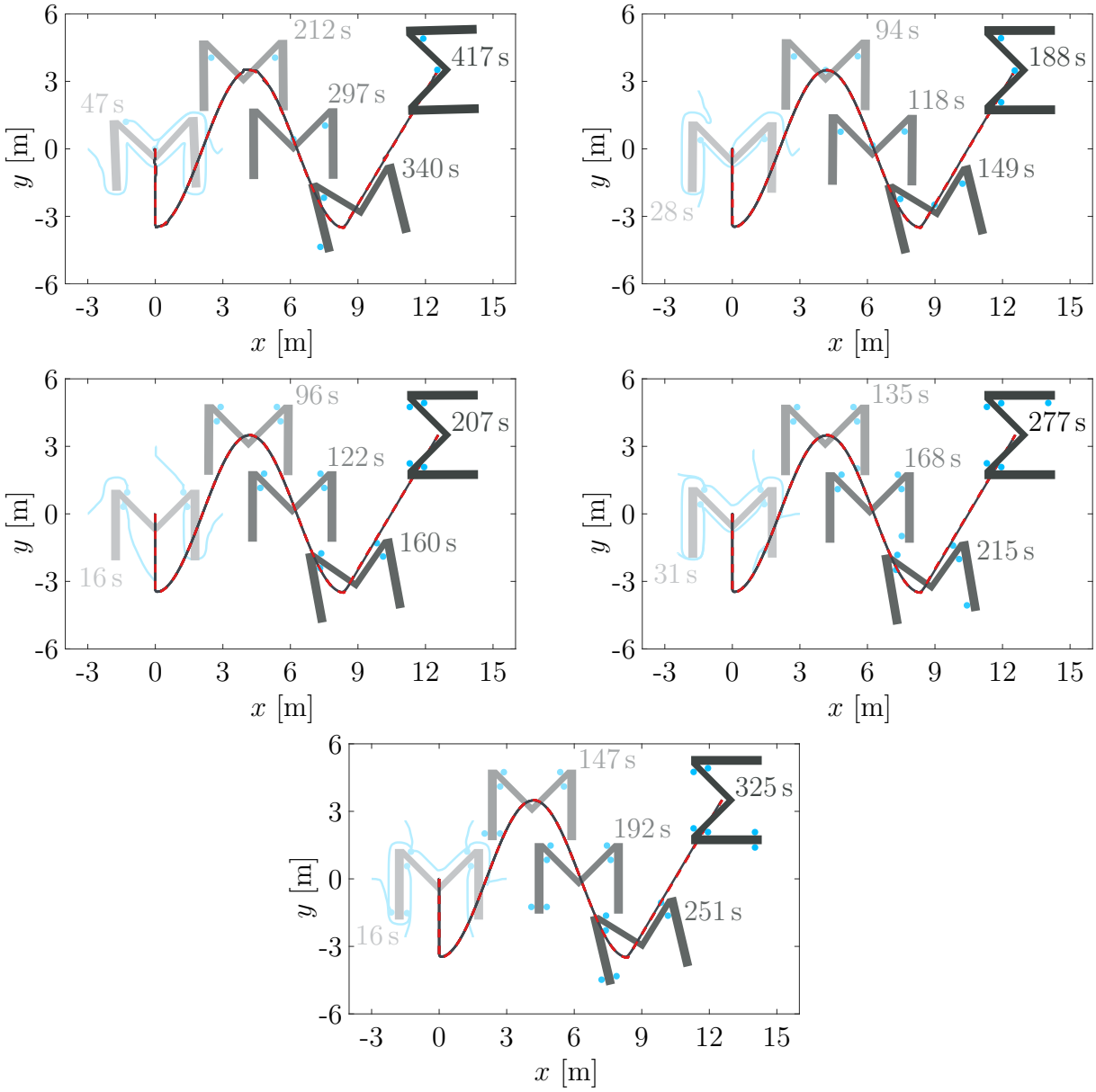


Figure 6.4: Simulation results for two to six robots transporting a Σ-shaped object

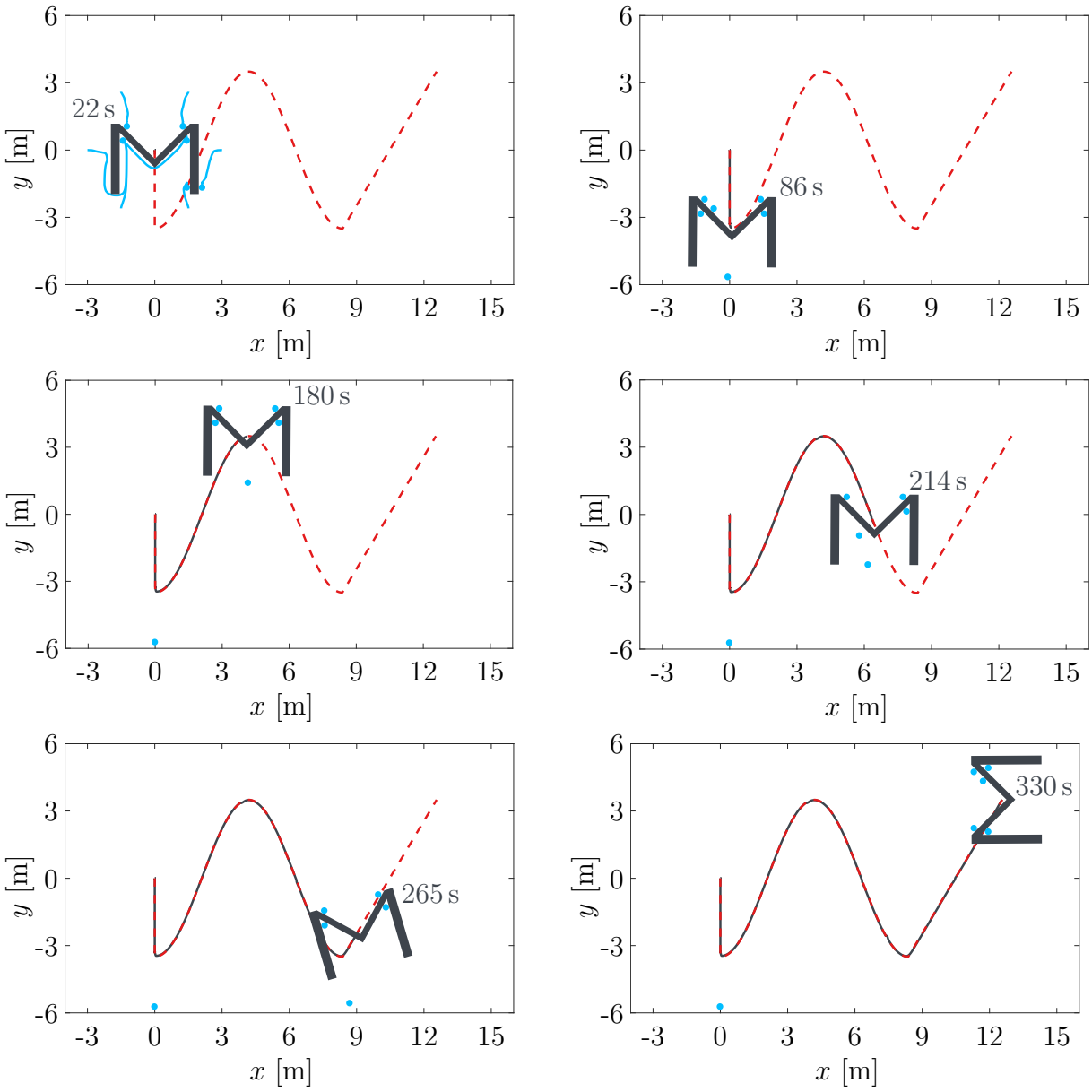


Figure 6.5: Simulation results of a variably-sized group of robots transporting a Σ-shaped object

it can actually deliver on the characteristic flexibility promised by distributed robotics, with robots joining and leaving the cooperative task at system runtime. Consequently, Figure 6.5 takes another look at the previous scenario with the Σ-shaped object, but with five reorganizations of the robotic network happening in the course of the simulation. In total, at three time instances, one of the robots leaves the transportation process, with the robots leaving 56 s, 157 s, and 210 s after the start. The second and third robots to leave later rejoin at the time instances 226 s and 280 s, respectively. Robots not partaking in the cooperative task are set up to use the signals from their distance sensors to steer clear of any robot or object to avoid getting in the way of the transportation process. Knowing

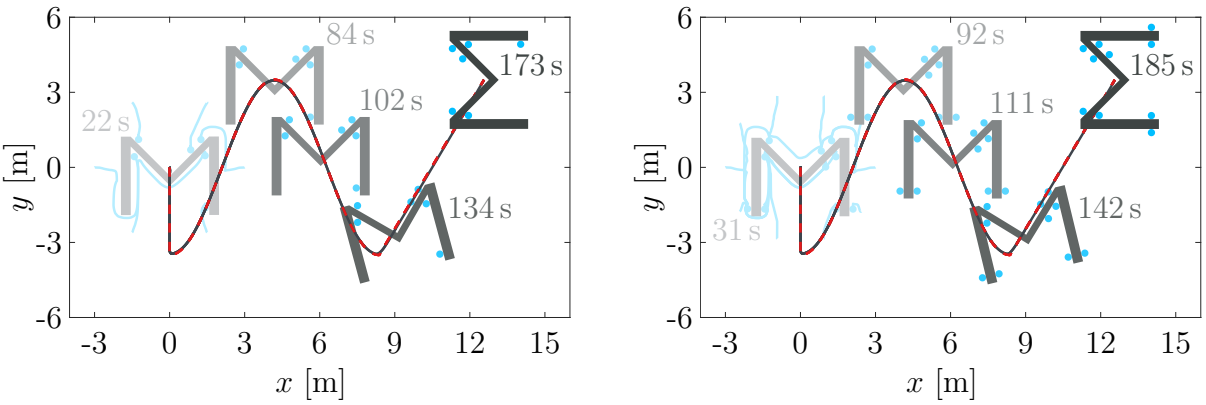


Figure 6.6: Simulation result for six and ten robots transporting a Σ-shaped object using the DMPC$^{\mathrm{v}}$ controller

from previous scenarios that the formation controller and the formation agents work well for different numbers of robots, this poses a challenge mostly to the proposed software architecture and the amenability of the employed methods for quick reconfiguration. Indeed, it is made evident by the results that the modular software architecture does deliver on its promise.

Interestingly, in the results from Figure 6.4, the transportation takes increasingly more time for the groups of four to six robots than it does for three robots, despite no reorganizations happening in all of these cases. This is not surprising since it has already been observed in Section 5.1.4 that the DMPC$^{\mathrm{p}}$ scheme does become more conservative for larger numbers of robots. This is already a scenario where the object needs to be transported across a relatively large distance. In contrast, it has been stated previously that the applications in this thesis focus more on precise positioning instead of fast long-distance transportation due to the confinements of the experimentation area available for hardware experiments. Nevertheless, if the focus is different, the DMPC$^{\mathrm{v}}$ scheme can be employed. Results from a proof-of-concept simulation that uses the DMPC$^{\mathrm{v}}$ controller for transportation purposes are illustrated on the left-hand side of Figure 6.6. The employed control parameters are mostly the same as those for the DMPC$^{\mathrm{p}}$ controller, except for the maximum change of the control input being set to $\Delta u_{\max} = 0.04\,\mathrm{m/s}$ and the control input weight being chosen to $\boldsymbol{R} = 0.1\,\boldsymbol{I}$. Therein, to track a path with the DMPC$^{\mathrm{v}}$ scheme, the current desired formation velocity ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ is chosen such that it points into the direction of the path point currently tracked. The velocity's absolute value is chosen depending on the amount of direction change within a window reaching ahead by 0.5 m along the path's arc length. The absolute value reaches 0.2 m/s in straight-line segments of the path. By nature, the choice of ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ determines the trade-off between transportation speed and tracking accuracy. As the results show, for the chosen setup, the object is transported much more swiftly than with the DMPC$^{\mathrm{p}}$ controller, reaching the goal after about 151 s when neglecting the initial organization phase, which results in an average progression velocity of about 0.19 m/s

along the path's arc length. In this example, the tracking accuracy is also pristine. Still, the quantitative influence of the desired common velocity ${}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{d}}$ on the tracking accuracy is not a priori clear in an explicit manner and thus not as easily tunable as when all control parameters are contained within the same optimal control problem. Nevertheless, the results strongly suggest that the DMPC$^{\mathrm{v}}$ scheme may be a worthwhile choice for very large numbers of robots or long-distance transportation, especially if the formation's common velocity is controlled in a way that is in accordance with the accuracy needs of the specific application. In addition, the examined results indicate that the interplay between the robots, including organization and negotiation, function well also for larger numbers of robots.

It is worth noting that, except for degenerate cases, geometrically, six robots should always be sufficient to be able to find a formation suitable for transportation without reorganization. In principle, four of the robot positions could be picked to obtain contact normals spanning a convex cone covering the whole plane, and the two remaining robot positions could be chosen to allow for rotations in the clockwise and counter-clockwise directions, respectively. Nevertheless, it is still interesting to see whether the proposed scheme and software architecture do scale to even larger numbers of robots. In cases where, e.g., the object is too heavy to be accelerated quickly enough with one of the robot's propulsion forces, additional robots with the same pushing direction can benefit the transportation process. Although such variations of the transportation task are not considered here, the scalability of the proposed approach to even more robots would mean that also such a scenario could be accounted for. Therefore, on the right-hand side of Figure 6.6, ten robots engage in transporting the Σ-shaped object using the DMPC$^{\mathrm{v}}$ controller as in the previous result. The robots automatically arrange successfully around the object, despite its edges being considerably more crowded due to the additional robots. As expected for this scenario, the additional robots do not perceptibly alter the transportation accuracy, with the average translational velocity of the object amounting to a similar value of 0.18 m/s. Nevertheless, the very satisfactory performance in this 10-robot-scenario shows that there is no fundamental issue preventing the scaling of the proposed methods to larger numbers of robots.

A remaining question now is whether the proposed approach can also deal well with the disturbances and challenges arising from real-world hardware experiments. The subsequent section is devoted to answering this question while also considering additional object shapes, giving a glimpse of the scheme's versatility. Toward the end of the section, the scheme's limitations, as they have become apparent during experimentation, are discussed.

6.2 Experimental Investigation

All hardware experiments in this thesis use a constant set of parameters in the optimization problem of the DMPC controller and in the formation synthesis optimization problem. In particular, these parameters are mostly identical to those used in the simulation scenarios considered in the previous section. One of the few changes is that the sampling time of the DMPC controller is set to 0.1 s instead of 0.05 s to more comfortably meet the real-time requirements with the processing hardware available. The only other change to the DMPC controller is that the robots' maximum admissible directional velocity is set to 0.2 m/s in absolute value. Due to its ease of parameterization, the DMPC$^{\text{p}}$ controller is used throughout the section. Since the considered hardware robots do not have onboard distance sensors, these continue to be simulated, with their range being reduced to 1.0 m, helping to limit the calculation time of the sensor simulation. The experimentation area available is of an approximate size of 4 m by 3 m, so the distance sensors can still cover a good portion of the area. The general experimental setup is very similar to the setup of the previously conducted formation control experiments from Section 5.1.4, with the same tracking system and robotic hardware being used. The operating frequency of the tracking system's cameras is set to 50 Hz, and, apart from the robots, it also tracks and publishes the pose of the transported object. Figure 6.7 shows photographs of the robots as used in the experiments as well as exemplary objects equipped with markers to be trackable by the tracking system. The diameter of the robots' two upper layers is slightly smaller than that of the lower layer so that only the lower layer comes in contact with the object. Once again, due to limited onboard processing power, the control and organization agents are not run on the robots themselves but on laboratory computers. The execution of the various agents is split across two different computers, with one being the portable laboratory computer used previously for formation control and the additional one having an Intel Xeon E31245 CPU operating four physical and eight logical cores at a frequency

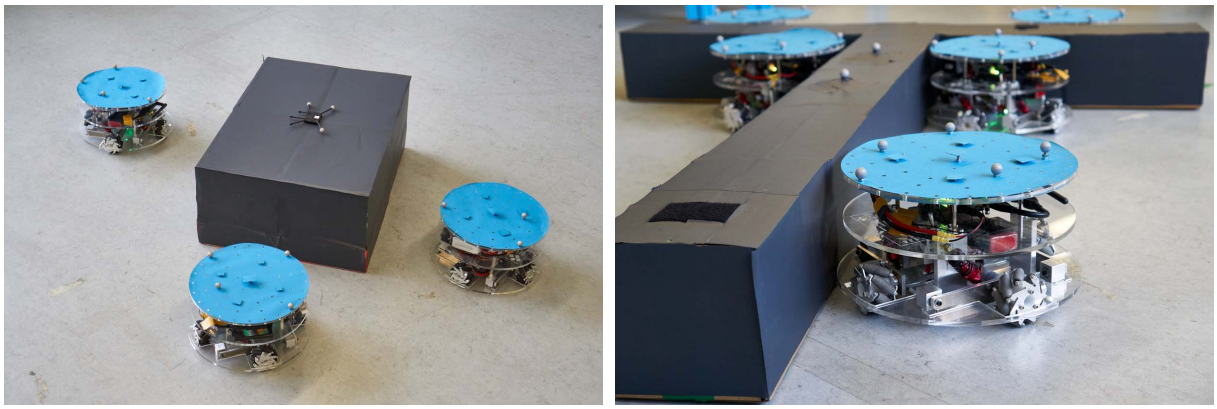


Figure 6.7: Photographs of the employed robots with objects to be transported

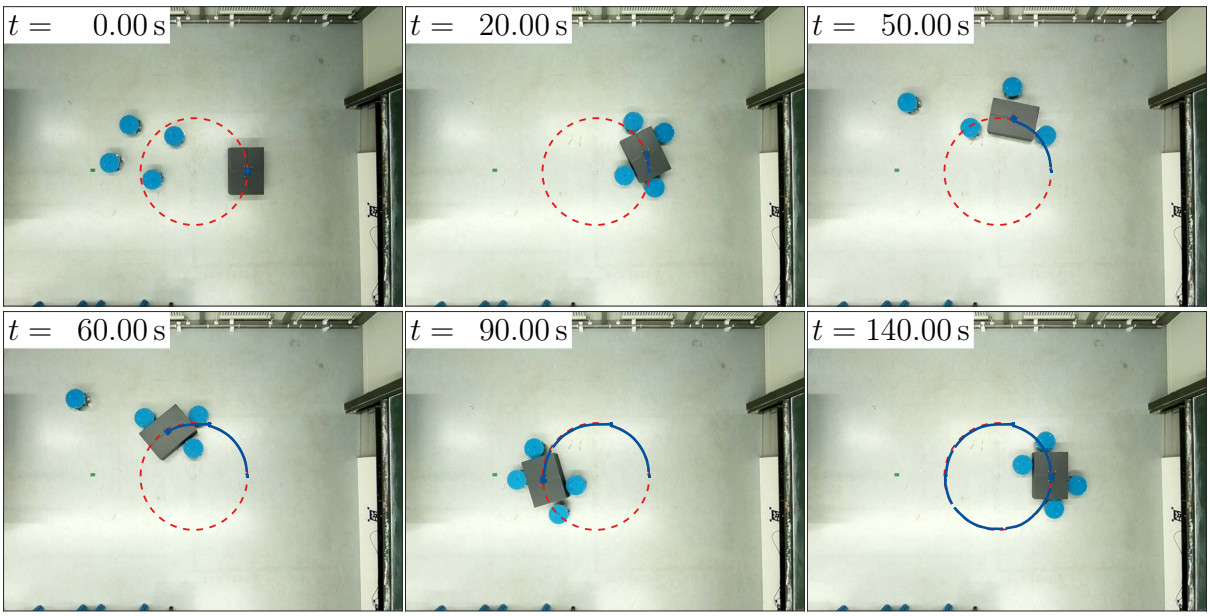


Figure 6.8: Four robots engaging in the transportation of a rectangular object, with one of the robots not participating for a certain amount of time

of 3.3 GHz. The purpose of this is twofold. On the one hand, with the control and organization agents, two different programs, which are multi-threaded themselves, need to be executed for each robot. Hence, especially for larger numbers of robots, increasing the number of real-time critical threads far beyond a single computer's physical resources can lead to unreliable cycle times of the involved programs. This is circumvented by adding more processing resources to the robotic network. On the other hand, the setup exemplifies that the involved programs can run anywhere on the network. Despite the real-time critical programs running on external computers, as before, all information to be exchanged is communicated via network communication through UDP multicast, with realistic communication delays present in the network. Hence, as envisioned in Section 4.2, the transition from simulations to hardware experiments indeed consists of replacing the simulator with the real-world robots and the tracking system, with it being transparent to the organization and control agents whether they govern the behavior of real or of simulated robots. This nurtures the hope that many of the favorable properties observed in simulations transfer well to real-world hardware experiments.

Consequently, with basically the same organization and control software being employed, one of the least surprising properties to transfer well to experiments is the flexibility of the robotic network, with the possibility to let robots join or leave the transportation process. Therefore, in the first experiment, precisely this kind of scenario is considered. Impressions from the experiment are illustrated in Figure 6.8, with the reference path dashed in red and the past trajectory of the transported object drawn in dark blue. Both are mapped onto the camera image in an approximate manner. Therein, four robots

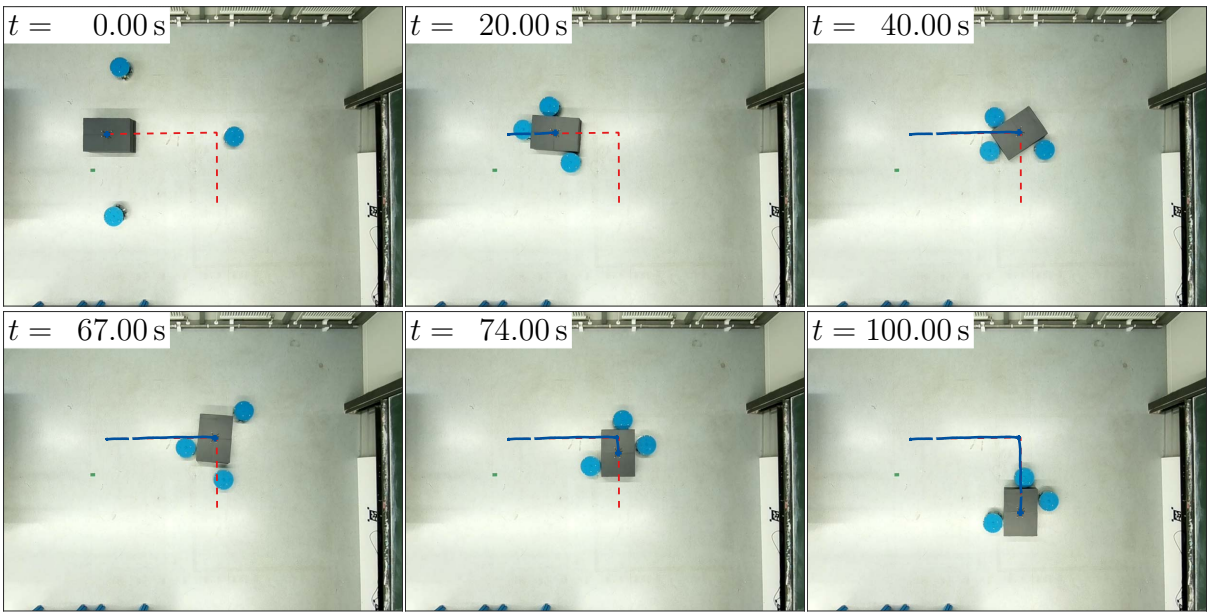


Figure 6.9: Three robots transporting a rectangular object along a path that either requires pure translation or pure rotation

transport a rectangular object along a circular path, rotating the object at a constant rate and completing one full rotation along the circle. This and all following objects are manufactured using cardboard boxes that slide across the laboratory floor. Initially, all four robots transport the object, but about 30 s after the experiment's start, the robot at the rectangle's leading edge leaves the transportation process. The remaining robots then reorganize, as can be seen in the third image of the figure. Then, they continue to transport the object until about 67 s after the start, when the non-participating robot receives a message to rejoin the transportation. Having reorganized once again, all four robots pursue the transportation task together and complete it successfully. It is worth noting that the robots are not preprogrammed in any way to reorganize, so they do not anticipate any alteration of the robotic network. Instead, the messages letting a robot join or leave are sent spontaneously by the event agent at the experimenter's discretion. Nevertheless, as is the case in simulations, robots joining or leaving is not the only reason why the robots need to reorganize around the object. An example of this can be seen in the experiment result depicted in Figure 6.9. Three robots shall transport the rectangular object along a path consisting of two straight-line segments, translating the object without rotating it. Only in the corner of the path, the robots shall rotate the object by 90° in a counterclockwise fashion. Due to the formation chosen initially, the robots need to reorganize to push the object downwards along the second straight-line segment. Evidently, the formation agents self-reliantly agree on a suitable formation, the robots negotiate their goal positions within the formation in a conflict-free manner and, therefore, after reorganizing, the transportation is concluded successfully. Furthermore, together, the

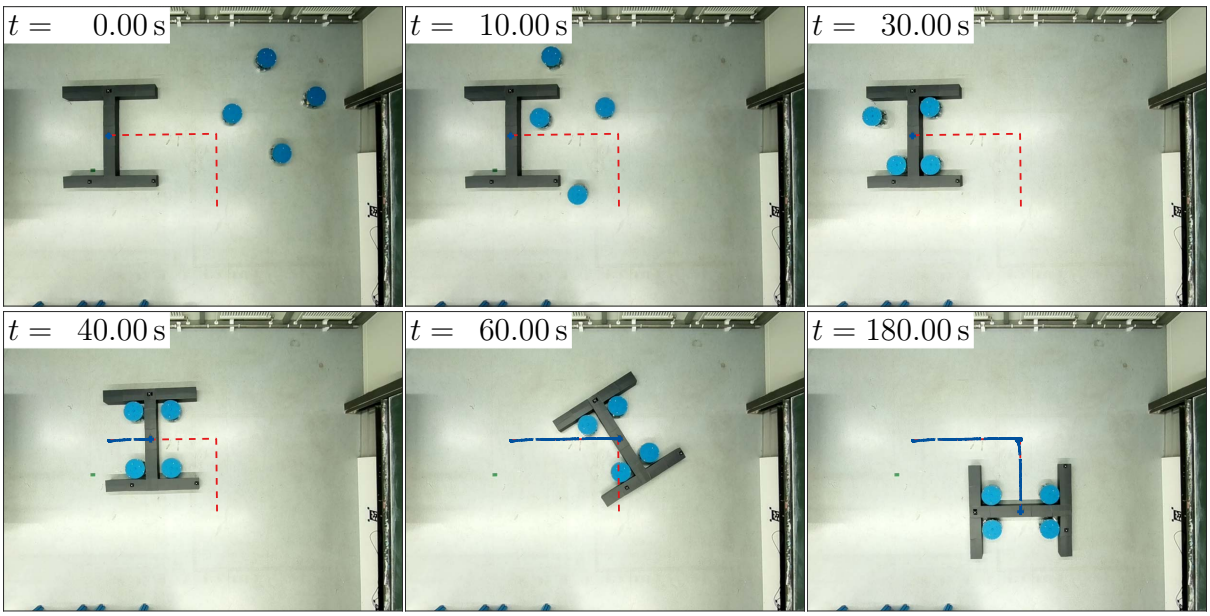


Figure 6.10: Four robots transporting a non-convex object along a path that either requires pure translation or pure rotation

results from Figures 6.8 and 6.9 show that, also in practice, the proposed scheme is capable of transportation both along paths requiring the simultaneous translation and rotation as well as along paths requiring the pure translation or the pure rotation of the object. Of course, the shape of the transported object in those scenarios is simple. However, it is worth pointing out that there exists published research focusing explicitly on the task of pushing rectangular boxes [MataricNilssonSimsarian95, KubeZhang96, YamadaSaito01, WangSilva06, RahimiEtAl19]. So it is reassuring to see that the proposed scheme can also naturally accommodate this widely studied case. Nevertheless, what remains to be shown is that the approach of this thesis can deliver on its promise of transporting more complicated, non-convex objects also in experiments.

Figure 6.10 depicts a result in this regard, with the transportation happening along the edgy path from the previous result. The first row of images gives an impression of the initial organization process. It becomes evident that the robots need not be positioned neatly around the object but can acquire a formation around a non-convex object also when positioned farther off from even just one side of the object. Having organized safely, the robots transport the object along the path without any large deviations. While such a path may seem simple at first glance, its practical relevance is not to be underestimated. It is easily conceivable that a path composed of segments of straight-line translation and in-place rotation can already be very effective in transporting an object through an obstacle-ridden environment. Since the straight-line segments can be small, using more general types of paths may bring little advantage with regard to the types of environments that can be successfully traversed. Nevertheless, the results for non-convex objects also extend well to

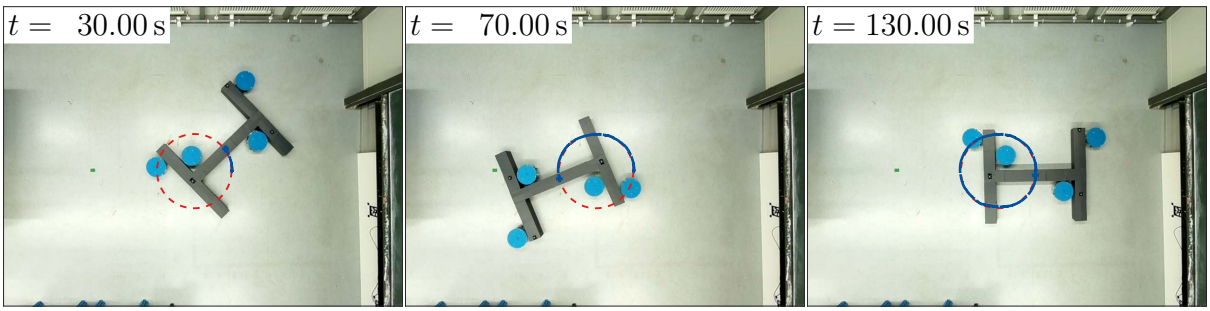


Figure 6.11: Four robots transporting a non-convex object along a circular path

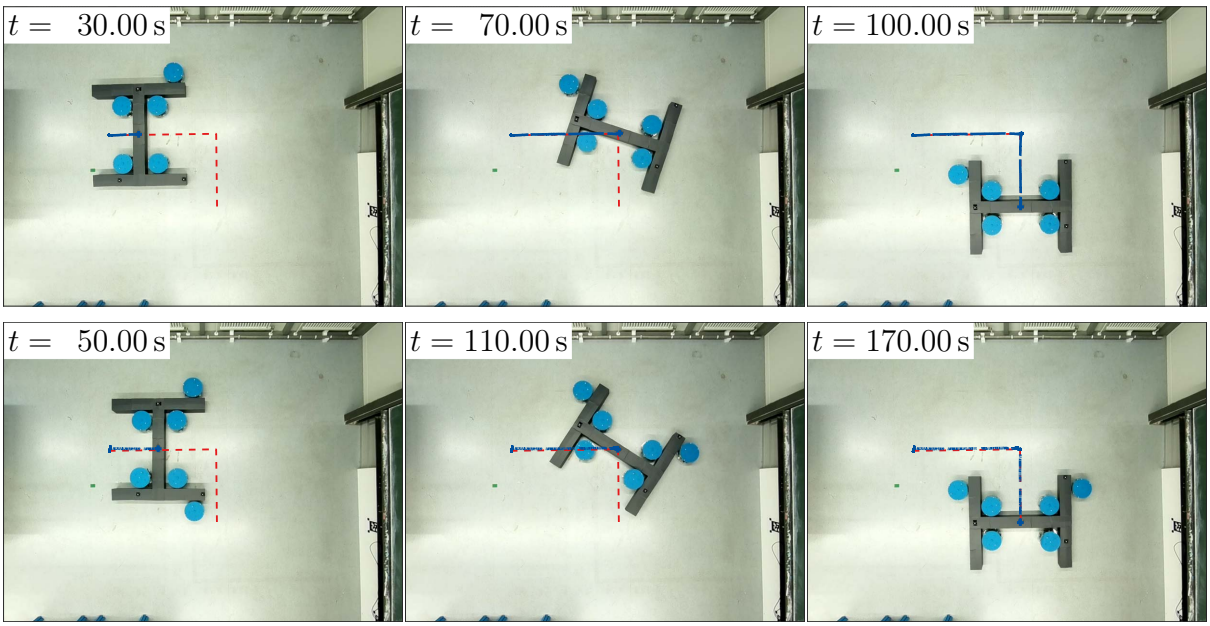


Figure 6.12: Groups of five and six robots transporting a non-convex object along a path that either requires pure translation or pure rotation

other types of paths, as exemplified by Figure 6.11. Furthermore, although not necessary for transportation, additional robots can be put to use, as shown in Figure 6.12, with five robots cooperating in the upper row and six in the lower row of the experiment snapshots. Thus, all things considered, the results so far already show that the proposed scheme's characteristic properties seem to transfer well to hardware experiments. Nonetheless, to get an impression of the scheme's versatility, additional experiments have been conducted. Their results are depicted in Figures 6.13-6.15, which show that the scheme can naturally accommodate objects of different shapes, with the robots transporting U-, S-, and T-shaped objects in a very successful manner.

Despite the results indicating that the scheme fulfills its design goals, a treatise on experimental results can only be complete when also the scheme's limitations are discussed. Indeed, judging by the experience from experimentation, the key limitation is connected to

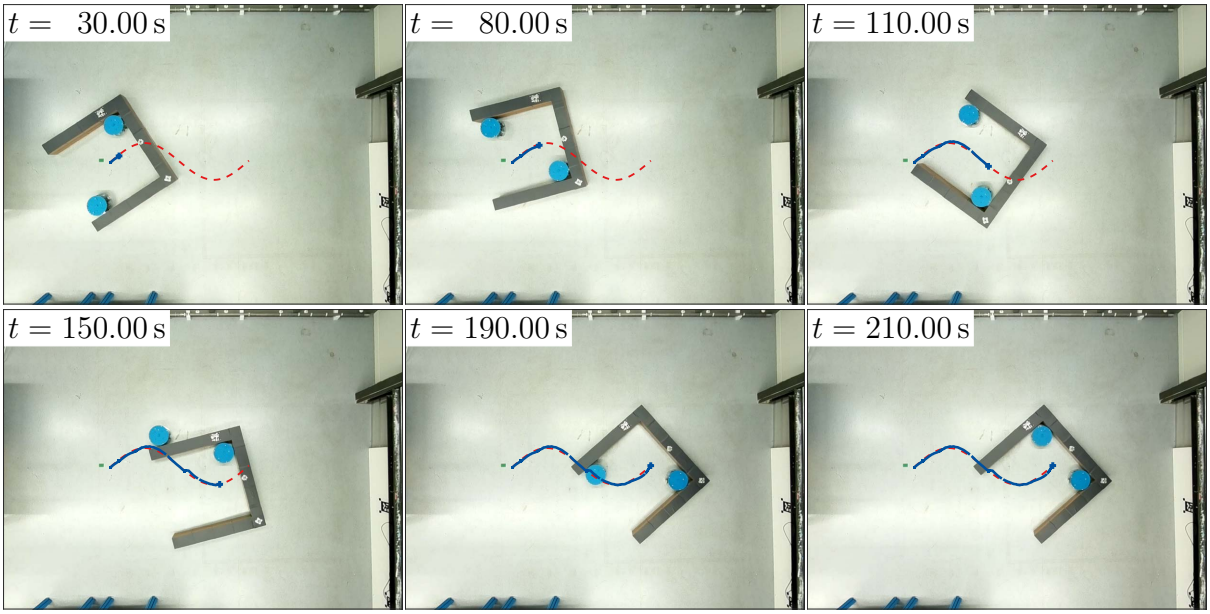


Figure 6.13: Two robots transporting a U-shaped object

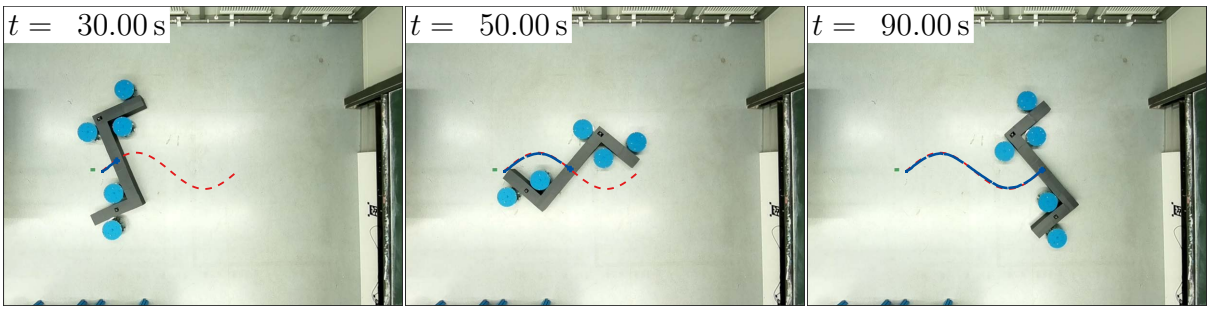


Figure 6.14: Five robots transporting an S-shaped object

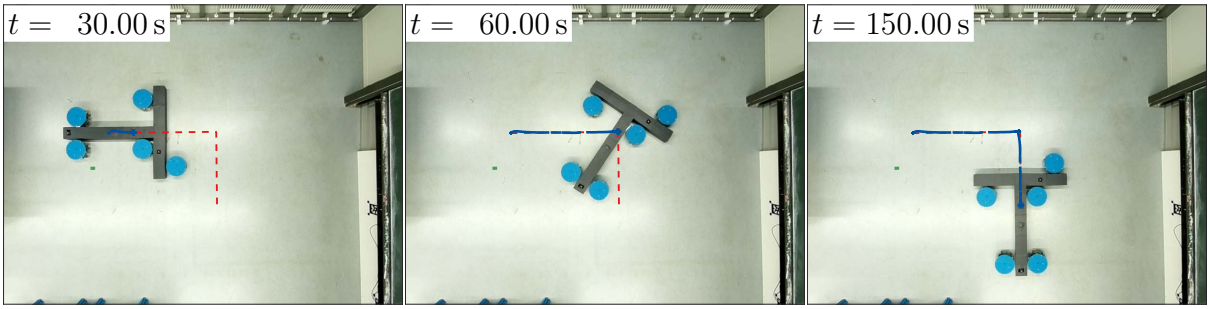


Figure 6.15: Five robots transporting a T-shaped object

the limited pushing forces of the robots. In particular, since the wheels' rollers are small, with their material being rather slippery, especially when dusty, and the robots being comparatively small and lightweight, resulting in small normal forces between the wheels and the ground, the attainable propulsion forces are limited. In consequence, it has been observed in the experiments that the robots' small wheels sometimes slip with the robot not managing to push the object, at least for a limited time frame. Similarly, for instance at the end of the experiment from Figure 6.11, there is a remaining error of about 4.5 cm in the direction tangential to the path at the goal point while there is no perceptible motion of the object anymore, despite not all control inputs vanishing. At times, robots struggle to start moving the transported object, overcoming the object's sticking friction, whereas transportation then may run more smoothly when the object is in motion. However, the results provide solace in the sense that the scheme seems to be able to deal well with this kind of disturbance. Through the feedback obtained via communication from the other robots, due to the DMPC formation controller, the robots try to move in a coordinated manner. Therefore, if the subset of the robots responsible for providing the majority of the pushing force is struggling, the other robots do, in effect, wait until the pushing robots accomplish to move the object. Furthermore, no matter how the robots and the scheme are designed, it is always possible to overwhelm the robots' pushing capabilities. Nevertheless, the experimental insight suggests potential measures to, in future research, further enhance the transportation capabilities. A very straightforward possibility is to alter the robot design, building a heavier robot with larger wheels providing better traction. However, most likely, this would increase the robots' size, causing the need for a larger experimentation area to perform insightful experiments. Nonetheless, this presents a purely technical solution approach not requiring any structural change of the proposed scheme. Similarly, a form of onboard traction control could help enhance the attainable propulsion forces. With regard to modifications of the scheme, a different formulation of the formation synthesis optimization problem may be worthwhile to push heavier objects. For instance, the cost function could be modified to maximize the sums of the robots' exertable pushing forces in the required directions. Another idea could be to explicitly include the object's dynamic properties in the formulation of the distributed controller. However, since, for instance, the object's friction-related properties may be unknown or changing during experimentation, this would most likely require online learning. As a more direct measure, it may very well be that the transportation performance could be improved further by directly tuning the formation control parameters, potentially adapting them individually to the object to be transported. As stated, this thesis refrains from doing so to instead show the scheme's versatility without engaging in any arduous tuning process. Nevertheless, in specific applications, where always the same types of objects need to be transported, application-specific tuning may still be a viable way to perfect the path tracking performance. In any case, the mere fact that the only noteworthy difficulty that has become visible during experimentation is mainly caused by the physical limitations of

the hardware robots, which classical engineering solutions could enhance, is a testament to the potential of the proposed scheme and architecture.

With the witnessed variety of the transportation scenarios in the simulations and experiments in mind, this is an excellent opportunity to pause for a moment and set in relation the proposed scheme's virtues to approaches that have appeared previously in the literature. It has already been noted that many studies in which robots shall cooperatively push objects limit their investigations to rectangular objects. Furthermore, even though some of the studies in the literature rely on reinforcement learning-based methods, e.g., [KovačŽivkovićBašić04, RahimiEtAl19], a generalization to different object shapes does not seem straightforward. Even if done successfully, it would most likely mean that the transportation strategy would have to be retrained for new object shapes. Even beyond approaches to box-pushing, it seems that practically all previous work on cooperative, non-prehensile transportation with mobile robots is limited to very specific object shapes. Intriguingly, the comparatively recent survey [TuciAlkilabiAkanyeti18] on cooperative transportation does not list a single non-prehensile transportation scheme that can deal with object shapes as general as this thesis's approach. Most only consider one particular shape of the transported object, with there being no pushing-based scheme that can deal with different non-convex object shapes in an automated manner. It is also common that only fixed numbers of robots are considered. Consequently, there seems to be a strong indication that the scheme presented in this thesis is genuinely unique in its versatility, which is realized by dint of the optimization-based approaches to organization and control.

Having seen that the proposed scheme is capable of moving objects along known paths, a remaining question needing further inspection is whether the proposed global navigation scheme can plan paths allowing a safe transportation through obstacle-ridden environments. These investigations are performed simulatively to allow for large-scale scenarios beyond the size of the hardware setup. It is worth noting that, in principle, to all aspects of the scheme examined so far, it is not important where the path to be followed originates from. Therefore, in a way, global navigation is decoupled from the other aspects, meaning that different navigation schemes could be used without changing any other aspect, method, or piece of software in the robotic network.

6.3 Transportation Through Obstacle-Ridden Environments

The simulations in this section use the same settings as those conducted in Section 6.1, with the objects' masses being set to $m_\mathrm{o} = 4\,\mathrm{kg}$ and, once again, being evenly distributed over each object's volume. However, different than previously, a global navigation agent is running in the robotic network. In the first scenario, the robots transport a U-shaped

object through an environment where all obstacles are known, i.e., the global navigation agent is supplied with a map of the environment containing all obstacles. The same map is also used in the simulation, meaning that the map represents the environment in an exact fashion. In more general applications, without necessitating any alteration of the proposed navigation scheme, the provided map may be a more coarse, outer approximation of the real world. The binary map is chosen to be quadratic, with 400 cells in each direction, which corresponds to the first two dimensions of the three-dimensional grid to be used for planning. The third dimension, corresponding to the orientation of the object, is chosen to have 181 cells. The object's center of mass shall be moved from the start position at $\begin{bmatrix}7.25 & 1.75\end{bmatrix}^\mathsf{T}$ m to the goal located at $\begin{bmatrix}0 & -5.25\end{bmatrix}^\mathsf{T}$ m. At the end of the path, the object's open end shall face into the negative x-direction. Three robots cooperate in the transportation, although one of the robots joins and leaves multiple times during the transportation. It leaves about 48 s after the start, rejoining about 147 s into the simulation, and once again leaves and rejoins about 316 s and 355 s into the transportation process, respectively. The simulation results are depicted in Figure 6.16, with the U-shaped object and the environment's obstacles being drawn in dark gray. Figure 6.17 provides insight into the operation of the global navigation agent. Therein, areas covered by the object itself during the planned motion are shaded in dark blue, with the planned path of the object's center of mass drawn on top of it. Around the latter and around the obstacles, areas shaded in lighter colors correspond to the dilated versions of the obstacles and the object. The results indicate that the global navigation agent can deal well with the given environment, devising a path that the formation can follow with typical accuracy. Due to the constructive, geometric formulation of the global navigation process and the already investigated path-following properties of the proposed scheme, this certainly is not unanticipated. Nevertheless, it remains to be seen whether the global navigator can indeed deal with unknown obstacles. To that end, a second scenario is considered. It is designed to include both known and unknown obstacles to demonstrate that such a setting can also be accounted for. The latter can be of significant practical value, e.g., with robots regularly operating in a warehouse or manufacturing facility where certain obstacles are always present and always in the same locations, for instance because they are load-bearing parts of the building. Thus, these fixed structures could be mapped once at a high accuracy so that the robots only have to register in real-time non-permanent obstacles. Therefore, if the non-permanent obstacles do not significantly influence the general direction of transportation, the duration of the transportation can potentially be reduced greatly. Furthermore, the influence of using lower-quality sensors in real-time is reduced if most major obstacles are already registered at high accuracy. Due to this motivation, the final result of this thesis considers an environment containing four rhombic obstacles, forming four narrow passageways, with the upper obstacle being unknown initially. The start position $\begin{bmatrix}-8 & 8\end{bmatrix}^\mathsf{T}$ m and the goal position $\begin{bmatrix}7.5 & 7.5\end{bmatrix}^\mathsf{T}$ m are chosen so that the object needs to be rotated in-between the rhombic obstacles. In the goal point, the longer sides of

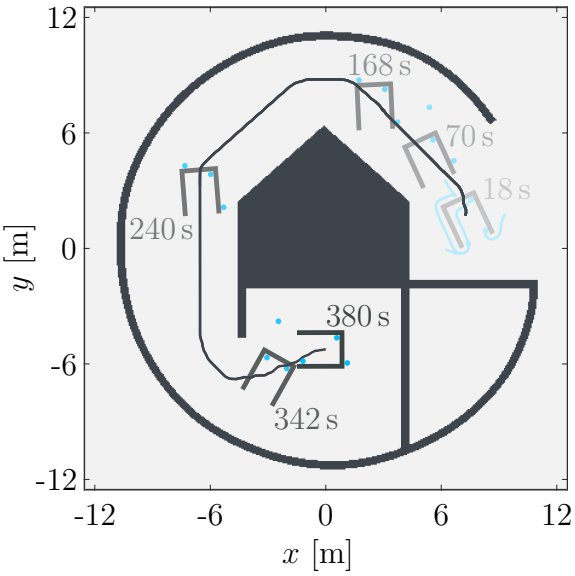


Figure 6.16: Simulation results for up to three robots transporting a U-shaped object through a known environment

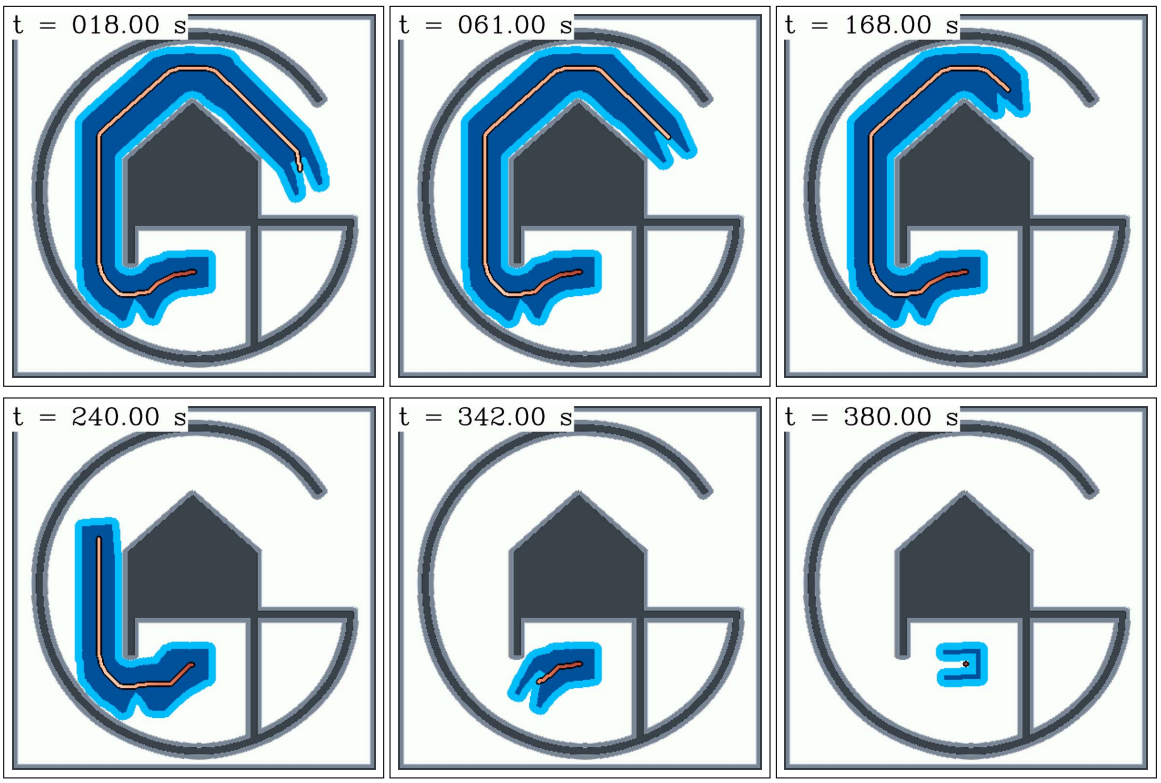


Figure 6.17: Snapshots of the program output from the global navigation agent in the simulation scenario from Figure 6.16

the rectangular object shall be orthogonal to the x-direction. Navigation happens on a grid with 645 cells in the x-direction, 480 cells in the y-direction, and 361 cells that discretize the object's orientation. Once again, some of the six robots cooperating join and leave, with robots 5, 4, and 6 leaving 30 s, 60 s, 105 s into the transportation process, respectively. Robots 4 and 5 rejoin 80 s and 100 s after the start. Consequently, the robots complete the transportation with fewer robots than engaged in the task initially. A simulation result of this scenario is illustrated in Figure 6.18, with the navigation agent's corresponding output being given in Figure 6.19. Evidently, the robots successfully discover the upper obstacle during transportation. Furthermore, the navigation agent achieves to find a suitable path for the object through the narrow passageways, backing up the object into the direction of the lower obstacle to maneuver from the upper left to the upper right passageway. Once again, path tracking is more than accurate enough to avoid collisions with the obstacles. Hence, it can be concluded that the transportation scheme tested in the previous sections can be coupled to a mapping and navigation scheme to allow the safe transportation of obstacles through environments that may contain unknown obstacles.

Notwithstanding these results, some natural constraints on the applicability of the navigation scheme have to be mentioned. For a successful transportation in the presence of unknown obstacles, there must always be at least one robot that can perceive nearby obstacles without them being occluded by the object being transported. Similarly, when applied to real-world hardware, there is an adverse relationship between the required sensor range and the processing capabilities of the robot running the global navigation agent. If the sensor range is excessively short, it could be that the robots collide with an unknown obstacle because the navigation agent is still busy with its calculations. Clearly, this can be alleviated by reducing the admissible speed of the formation or even of the robots themselves. Of course, these restrictions are natural to the task of navigation and not exclusive to what has been studied here. Beneficially, the modular, distributed software architecture does allow for a technical solution to the occlusion problem without having to modify the remaining part of the organization and control loop. Since the global navigation agent can run on any entity in the network, it could be run on an additional robot, thereby playing to the strengths of distributed robotics. For instance, it would be conceivable to have a surveillance drone fly above the robots, being able to clearly perceive the environment around the object from its vantage point. Even in more general terms, leveraging the advantages of a heterogeneous robotic swarm composed of robots with different capabilities may be a fruitful topic for future research. Following this line of thought toward a more dynamic interaction between dissimilar robots, it is conceivable that flying robots could pick up objects to transport them over obstacles insurmountable by ground-based ones. Indeed, a successor project, building upon the work conducted for this thesis, strives to realize advantages of this kind. Therefore, to render a precise, summarizing picture of the foundations laid for future work, it is time to draw final conclusions from this thesis' disquisition on cooperative mobile robotics.

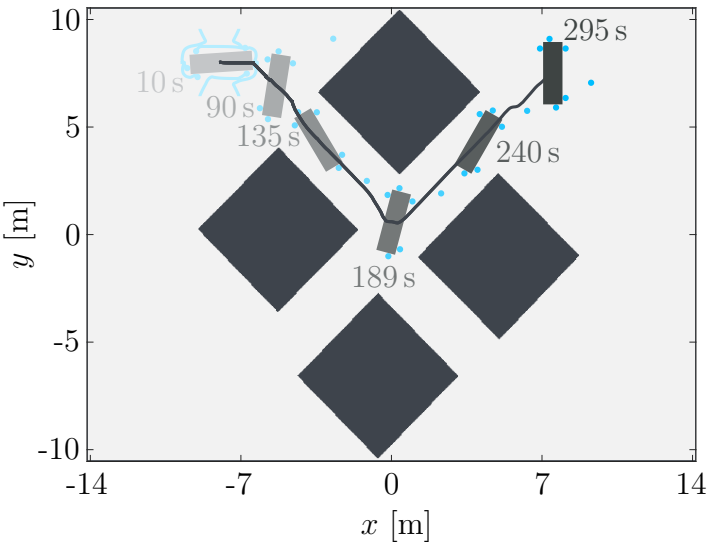


Figure 6.18: Simulation results for up to six robots transporting a rectangular object through an only partly known environment

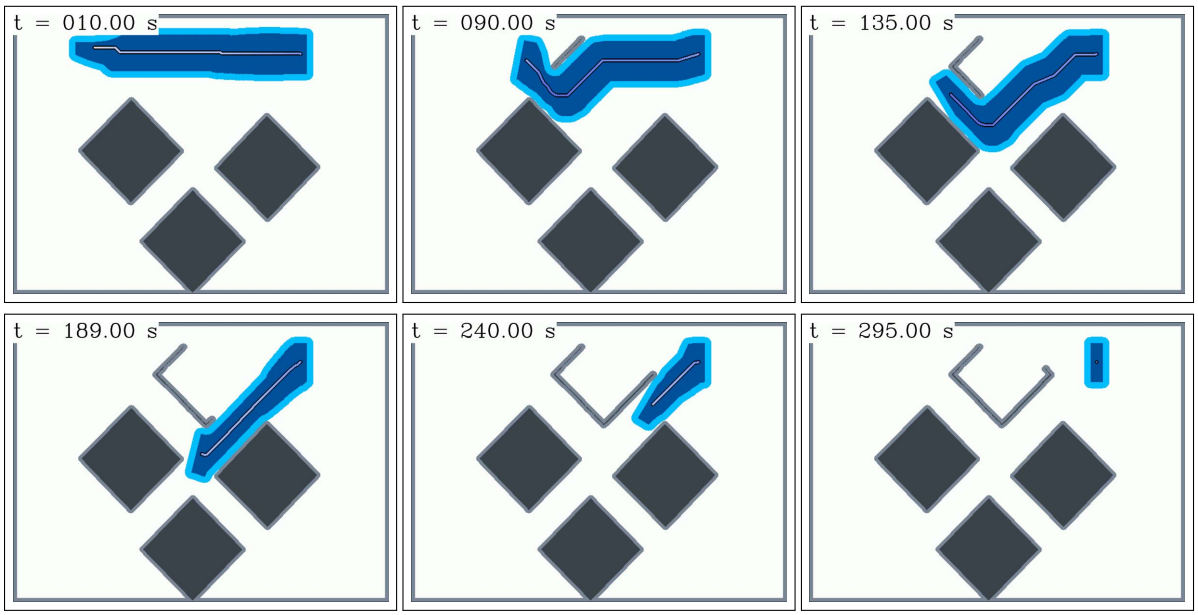


Figure 6.19: Snapshots of the program output from the global navigation agent in the simulation scenario from Figure 6.18

Chapter 7

Conclusion and Outlook

Cooperative distributed robotics can be as challenging to engineers as it is promising in furthering the state of the art of what is achievable with regard to robotic automation. A characteristic trait even of non-distributed robotics is its interdisciplinary nature, bringing together control engineering, mechanical modeling, algorithmics, as well as hardware and software design and development. In distributed robotics, problems related to some of these fields become even more intricate due to corresponding solution approaches needing to be inherently distributed, whereas communication and concurrency introduce a whole new set of challenges. However, the results obtained in this thesis for the cooperative, pushing-based transportation of polygonal objects show that facing these challenges can be very much worth the effort. The simulations and hardware experiments that have been presented demonstrate that all promised advantages of distributed robotics and control can be put into practice. This includes the robotic network adapting entirely self-reliantly to different objects to be transported, to different paths to transport them along, to different numbers of robots engaging in the transportation, and to robots joining and leaving in the midst of transportation. This kind of plug-and-play control is rarely demonstrated with real hardware involved.

Beneficially, the scheme developed for the task implicitly makes clear that the interconnected array of challenges can be atomized into sub-aspects that can be dealt with independently after having defined clear interfaces. This allows building sub-approaches rooted in theory already available in corresponding research areas that cater to a more focused set of challenges. For instance, the devised scheme uses a distributed model predictive controller as a formation controller and relies on distributed optimization to synthesize formations useful to manipulate the object as currently necessary. By using optimization-based approaches for the key challenges, the resulting scheme can adapt to different scenarios in a completely self-reliant manner, without arduous, scenario-dependent parameter tuning being necessary. The experience in developing the scheme has also shown that it is intriguingly intuitive to formulate the tasks to be solved in the form

of optimization problems, with the parameters that remain in the problems having a clear meaning by construction. All these aspects have led the cooperative transportation approach devised in this thesis to be uniquely versatile, by far surpassing all schemes known to the author that cater to a similar, pushing-based transportation task. Furthermore, all aspects of the task are dealt with in an encompassing manner. This means that also challenges such as self-reliant task allocation are solved by the proposed scheme – after all, since there is no centralized decision-making instance, the robots need to negotiate self-reliantly which robot takes which position in the formation. Methods tailor-made for the local and global navigation needs of the task were also introduced.

Furthermore, in its quest to master the cooperative transportation task, this thesis makes a number of contributions beyond the versatile transportation scheme itself. This includes the introduction and mechanical modeling of a mobile robot design for research on distributed control. The robot is very maneuverable since it can move omnidirectionally, nurturing the hope that it can be employed in a wide variety of investigations, beyond the transportation task that it was primarily designed for. In addition, as a reaction to the challenges faced when developing a complicated distributed system, a general software architecture was proposed based on requirements typical and characteristic to distributed robotics. This includes the insight to distinguish distributed control from less time-critical distributed organization tasks and has lead to a very modular software architecture based on multicast communication. After all, it is not only real-time automatic control that needs to be distributed when robots shall cooperatively and self-reliantly solve a practical task. The architecture allows different pieces of software to run concurrently at different or even variable sampling rates on any device in the network. In particular, the very same software stack can be used for simulations and hardware experiments, with it being transparent to the robot software whether it governs a simulated or a real robot. This helped significantly to smooth the transition from simulations to hardware experiments in the work on this thesis. Furthermore, two variants of a formation control setup based on distributed model predictive control were proposed and analyzed independently from the transportation task since the setup may be of value in all robotics applications where formation control can be of use. The two variants either allow to control the absolute position of the formation or the common velocity of the formation while adhering to input constraints. Similarly, for comparison purposes, a more traditional setup, relying on results from algebraic graph theory, but modified to respect input constraints, was introduced. It has been found that only the predictive control-based approach allows to intuitively weigh differently the control goals of moving the formation center and of maintaining the formation shape without the consideration of input constraints leading to superfluous conservativity. Robotics researchers may, therefore, consider the proposed predictive controllers as an alternative to more widely used types of formation controllers. Moreover, in response to the properties of the optimization problem solved to devise formations, a custom, distributed stochastic optimizer, that can handle constraints, was proposed based on augmented Lagrangian

particle swarm optimization [SedlaczekEberhard06]. In the future, it may prove to be a very useful tool also in the solution of other organizational problems or even in very large-scale optimization tasks as they may appear, e.g., in structural optimization.

These contributions set the stage for worthwhile areas of future research. Clearly, motivated by this thesis's promising results, a direct continuation of the work may be fruitful. In particular, the approach could be developed further to solve tasks even more engaging from a mechanical perspective, e.g., robots cooperatively transporting or manipulating a deformable object. To that end, it may be promising to equip the robots with force-sensing capabilities and to employ more complex mechanical models within the organization and control schemes, beyond mostly kinematic considerations. However, research in this area may quickly reach territory where not all aspects of the dynamics can be modeled appropriately solely based on first principles. It may become necessary to infer parts of the model using learning methods, potentially even at system runtime. For instance, the robots might try to identify specific mechanical properties of the transported deformable object. Tasks like this may also benefit significantly from augmenting the robotic swarm with additional sensing capabilities. Furthermore, the devised software architecture may be employed to deal with cooperative robotic tasks beyond transportation and manipulation, such as the cooperative search for targets [HuEtAl14, TangEtAl18].

Another research direction may seek the cooperation between robots of different propulsion, manipulation, and perception capabilities, venturing toward a heterogeneous robotic swarm. In the same way, when focusing on formation control as a building block of higher-arching cooperative behavior, it may have merit to extend the proposed setup based on distributed model predictive control toward robots with more severe non-holonomic kinematic constraints. While already interesting on its own, this may serve as a good stepping stone to allow the usage of simpler robots for transportation tasks or other cooperative benchmark problems. In that regard, differentially-driven robots may be a simple and popular alternative to more complicated, omnidirectional robots.

On another note, in this thesis, imperfections of communication merely acted as a disturbance that was not inspected in greater detail. Indeed, with the given network setup, the scheme worked well despite message losses and communication delays. Nevertheless, both in the interest of possible applications in a productive environment and for the usage with a less reliable network, research work in this direction can be very valuable. Recent results to be found in the literature nurture the hope that continued progress will be made in this area in the future, see, e.g., [JacobEtAl16, BaumannEtAl19, LinsenmayerHertneckAllgöwer21].

Appendix

A.1 Expressions in the DMPC Formation Controller

The matrices in the multi-parametric quadratic program representation of the DMPC formation controller from Section 5.1.1 and Equations (5.20)-(5.25) are given by

$$\boldsymbol{H} = 2\left(\boldsymbol{M_R} + \boldsymbol{M}_{\boldsymbol{U}_\mathrm{e}}^\mathsf{T}\boldsymbol{M_Q}\boldsymbol{M}_{\boldsymbol{U}_\mathrm{e}} + \boldsymbol{S}_\mathrm{L}^\mathsf{T}\boldsymbol{M}_\theta^\mathsf{T}\boldsymbol{C}^\mathsf{T}\boldsymbol{T}\boldsymbol{C}\boldsymbol{M}_\theta\boldsymbol{S}_\mathrm{L}\right), \tag{A.1}$$

$$\boldsymbol{F} = \begin{bmatrix} 2\,\boldsymbol{M}_{\boldsymbol{A}}^\mathsf{T}\boldsymbol{M_Q}\boldsymbol{M}_{\boldsymbol{U}_\mathrm{e}} \\ -2\,\boldsymbol{T}\boldsymbol{C}\boldsymbol{M}_\theta\boldsymbol{S}_\mathrm{L} \\ 2\,\boldsymbol{M}_{\boldsymbol{A}\boldsymbol{B}_\mathrm{p}}^\mathsf{T}\boldsymbol{M_Q}\boldsymbol{M}_{\boldsymbol{U}_\mathrm{e}} \end{bmatrix}, \tag{A.2}$$

$$\boldsymbol{M}_\mathrm{t1} = \begin{bmatrix} \boldsymbol{A}^{H-1}\boldsymbol{B}_\mathrm{e} & \boldsymbol{A}^{H-2}\boldsymbol{B}_\mathrm{e} & \cdots & \boldsymbol{B}_\mathrm{e} & \mathbf{0} \end{bmatrix} - \boldsymbol{M}_\theta\boldsymbol{S}_\mathrm{L}, \tag{A.3}$$

$$\boldsymbol{M}_\mathrm{t2} = -\begin{bmatrix} \boldsymbol{A}^{H} & \mathbf{0} & \boldsymbol{A}^{H-1}\boldsymbol{B}_\mathrm{p} & \boldsymbol{A}^{H-2}\boldsymbol{B}_\mathrm{p} & \cdots & \boldsymbol{B}_\mathrm{p} & \mathbf{0} \end{bmatrix}, \tag{A.4}$$

$$\boldsymbol{M}_\mathrm{d} = \begin{bmatrix} \bar{\boldsymbol{M}}_\mathrm{d} & \mathbf{0}_{2(H-1)\times n_\theta} \end{bmatrix} \tag{A.5}$$

with the matrix of zeros $\mathbf{0}_{a\times b} \in \mathbb{R}^{a\times b}$. Therein, the matrices

$$\boldsymbol{M_A} = \begin{bmatrix} \boldsymbol{A}^0 \\ \vdots \\ \boldsymbol{A}^{H-1} \end{bmatrix} \in \mathbb{R}^{n_x H \times n_x}, \tag{A.6}$$

$$\boldsymbol{M}_{\boldsymbol{A}\boldsymbol{B}_\mathrm{e}} = \begin{bmatrix} \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{B}_\mathrm{e} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \vdots & \vdots \\ \boldsymbol{A}\boldsymbol{B}_\mathrm{e} & \boldsymbol{B}_\mathrm{e} & \mathbf{0} & \cdots & \mathbf{0} & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} & \vdots & \vdots \\ \boldsymbol{A}^{H-2}\boldsymbol{B}_\mathrm{e} & \boldsymbol{A}^{H-3}\boldsymbol{B}_\mathrm{e} & \ldots & \ldots & \boldsymbol{B}_\mathrm{e} & \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n_x H \times (2H+n_\theta)}, \tag{A.7}$$

$$\boldsymbol{M}_{\boldsymbol{AB}_\mathrm{p}} = \begin{bmatrix} \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{B}_\mathrm{p} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \vdots & \vdots \\ \boldsymbol{A}\boldsymbol{B}_\mathrm{p} & \boldsymbol{B}_\mathrm{p} & \mathbf{0} & \cdots & \mathbf{0} & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} & \vdots & \vdots \\ \boldsymbol{A}^{H-2}\boldsymbol{B}_\mathrm{p} & \boldsymbol{A}^{H-3}\boldsymbol{B}_\mathrm{p} & \dots & \dots & \boldsymbol{B}_\mathrm{p} & \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n_x H \times (2(N-1)H+2)} \tag{A.8}$$

are useful in the state prediction over the horizon, which can be used together with the matrices

$$\boldsymbol{M}_{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{Q} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{Q} \end{bmatrix} \in \mathbb{R}^{Hn_x \times Hn_x}, \tag{A.9}$$

$$\boldsymbol{M}_{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R} & & & \mathbf{0} \\ & \ddots & & \\ & & \boldsymbol{R} & \\ \mathbf{0} & & & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(2H+n_\theta)\times(2H+n_\theta)}, \tag{A.10}$$

$$\bar{\boldsymbol{M}}_\theta = \begin{bmatrix} \boldsymbol{M}_\theta & & & \\ & \ddots & & \\ & & & \boldsymbol{M}_\theta \end{bmatrix} \in \mathbb{R}^{(Hn_x)\times(Hn_\theta)}, \tag{A.11}$$

$$\boldsymbol{S}_\mathrm{L} = \begin{bmatrix} \mathbf{0} & \dots & \dots & \boldsymbol{I}_{n_\theta} \end{bmatrix} \in \mathbb{R}^{n_\theta \times (2H+n_\theta)}, \tag{A.12}$$

$$\bar{\boldsymbol{S}}_\mathrm{L} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{I}_{n_\theta} \\ \vdots & & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{I}_{n_\theta} \end{bmatrix} \in \mathbb{R}^{Hn_\theta \times (2H+n_\theta)}, \tag{A.13}$$

$$\boldsymbol{M}_{\boldsymbol{U}_\mathrm{e}} = \boldsymbol{M}_{\boldsymbol{AB}_\mathrm{e}} - \bar{\boldsymbol{M}}_\theta \bar{\boldsymbol{S}}_\mathrm{L} \tag{A.14}$$

to express the cost function and terminal equality constraints in matrix-vector form. Finally, the matrix

$$\bar{\boldsymbol{M}}_\mathrm{d} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \otimes \boldsymbol{I}_2 \in \mathbb{R}^{2(H-1)\times(2H)} \tag{A.15}$$

is used in the inequality constraints through expression (A.5).

Abbreviations, Symbols, and Notation

In the following, all important and reoccurring abbreviations and symbols are given to provide an overview of the notation used throughout the thesis. In general, vectors and matrices are printed in bold math font, whereas scalar quantities are displayed in normal math font. Sets are usually written in calligraphic math font. Quantities only appearing within one section are usually only introduced and defined therein.

Abbreviations

AGTC	formation controller based on algebraic graph theory
AGTC$^{\text{p}}$	graph-algebraic formation controller controlling the formation's position
AGTC$^{\text{v}}$	graph-algebraic formation controller controlling the formation's common velocity
ALPSO	augmented Lagrangian particle swarm optimization
CPU	central processing unit
DALPSO	distributed augmented Lagrangian particle swarm optimization
DARE	discrete-time algebraic Riccati equation
DMPC	distributed model predictive control
DMPC$^{\text{p}}$	DMPC-based formation controller controlling the formation's position
DMPC$^{\text{v}}$	DMPC-based formation controller controlling the formation's common velocity
KKT	Karush Kuhn Tucker
LQR	linear quadratic regulator
MIMO	multiple-input multiple-output
MPC	model predictive control
PI	proportional integral
PID	proportional integral derivative
PSO	particle swarm optimization
ROS	robot operating system
SISO	single-input single-output
TCP	transmission control protocol
UDP	user datagram protocol

Latin Minuscules

$\ell(\boldsymbol{x}, \boldsymbol{u})$	stage cost of a predictive controller
$n_{\mathcal{E}}$	number of edges of a graph
n_u	dimensionality of the input vector of a dynamical system
$n_{\mathcal{V}}$	number of nodes of a graph
n_x	dimensionality of the state vector of a dynamical system
n_y	dimensionality of the output vector of a dynamical system

$\boldsymbol{o}_\alpha$ absolute position of the origin of the reference frame $\mathcal{K}_\alpha$, given in the coordinates of and relative to the inertial frame of reference

$\boldsymbol{q}$ vector of generalized coordinates of a multibody system

r_R radius of the circular footprint of the omnidirectional mobile robot

$\boldsymbol{u}$ input vector of a dynamical system

$\boldsymbol{u}(\cdot \,|\, t)$ input sequence planned by a predictive controller at time step t

$\boldsymbol{u}^\star(\cdot \,|\, t)$ optimal input sequence planned by a predictive controller at time step t

u_max absolute value of the maximum admissible directional velocity of the considered robot

Δu_max absolute value of the maximum desired change of the directional velocity of the considered robot

${}_\mathrm{c}\boldsymbol{v}_\mathrm{d}$ desired common velocity of the formation

$\check{w}$ weight function in a weighted, directed graph

$\boldsymbol{x}$ state vector of a dynamical system

$\boldsymbol{x}(\cdot \,|\, t)$ state sequence predicted by a predictive controller at time step t

$\boldsymbol{x}^\star(\cdot \,|\, t)$ optimal state sequence predicted by a predictive controller at time step t

${}_\mathrm{c}\boldsymbol{x}$ position of the geometric formation center

${}_\mathrm{c}\boldsymbol{x}_\mathrm{d}$ desired position of the geometric formation center

${}^{\check{\mathrm{F}}}_{i}\boldsymbol{x}_\mathrm{d}$ desired relative position of the ith robot in the formation

$\boldsymbol{y}$ output of a dynamical system

Latin Majuscules

$\boldsymbol{\mathcal{A}}_{\vec{\mathcal{G}}}$ adjacency matrix of the weighted directed graph $\vec{\mathcal{G}}$

$\boldsymbol{\mathcal{B}}_{\vec{\mathcal{G}}}$ incidence matrix of the directed graph $\vec{\mathcal{G}}$

C_S center of mass of the transported object

$\boldsymbol{\mathcal{D}}$ diagonal weighting matrix appearing in the stage cost of the DMPC formation controller

$\mathcal{E}$ set of edges of an undirected graph

$\vec{\mathcal{E}}$ set of edges of a directed graph

${}^{\check{\mathrm{S}}}\boldsymbol{E}$ function describing the edges of the dilated object

$\mathcal{G}$ undirected, or simple, graph

$\vec{\mathcal{G}}$ directed graph

H length of the prediction horizon of a predictive controller

$\boldsymbol{I}$ identity matrix

$\boldsymbol{I}_n$ identity matrix with n rows and columns

$\mathcal{I}$ interval, $\mathcal{I} \subset \mathbb{R}$

J cost function of a predictive controller

J_f terminal cost of a predictive controller

$\boldsymbol{K}$ diagonal gain matrix appearing in the AGTC formation controllers

$\mathcal{K}_\alpha$ reference frame, with α being an identifier

$\mathcal{K}_\mathrm{F}$ formation-fixed reference frame

$\mathcal{K}_\mathrm{I}$ inertial frame of reference

$\mathcal{K}_\mathrm{R}$ body-fixed reference frame, fixed to the center of mass of a mobile robot

$\mathcal{K}_\mathrm{S}$ body-fixed reference frame, fixed to the center of mass of the transported object

L Lagrangian function

L_A augmented Lagrangian function

$\boldsymbol{L}$ Laplacian matrix of a graph

N_a number of active robots

O_α origin of the reference frame $\mathcal{K}_\alpha$, with α being an identifier

$\mathcal{P}$ convex polytope

$\boldsymbol{Q}$ weighting matrix weighing the states in the stage cost of a predictive controller

$\boldsymbol{R}$ weighting matrix weighing the inputs in the stage cost of a predictive controller

$\mathcal{R}$ set of robot indices

$\mathcal{R}_\mathrm{a}$ index set of the active robots

${}^\mathrm{I}\boldsymbol{S}_\alpha$ rotation matrix transforming coordinates of a vector given in the basis of $\mathcal{K}_\alpha$ to a coordinate vector given in the basis of $\mathcal{K}_\mathrm{I}$

$\boldsymbol{T}$ weighting matrix weighing the deviation of the artificial steady state from the setpoint in the stage cost of a predictive controller for tracking

T_s sampling time of the formation controllers

$\mathcal{U}$ input constraint set of a predictive controller

V Lyapunov function (candidate)

$\mathcal{V}$ set of nodes of a graph

$\mathcal{X}$ state constraint set of a predictive controller

$\mathcal{X}_\mathrm{f}$ terminal set of a predictive controller

Non-Latin Characters

${}_i\mathfrak{Z}^\mathrm{rel}$ relative formation control error of the ith robot

$\boldsymbol{\omega}$ angular velocity vector

General Mathematical Expressions and Symbols

$\mathbf{1}_n$ vector of ones of dimension n

$\tilde{\boldsymbol{a}}$ skew-symmetric matrix corresponding to the vector $\boldsymbol{a} \in \mathbb{R}^3$

$\boldsymbol{A} \otimes \boldsymbol{B}$ Kronecker product of the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

$\mathbb{R}_+$ set of all positive real numbers

$\|\boldsymbol{v}\|^2_{\boldsymbol{V}}$ squared, weighted norm, evaluating to $\boldsymbol{v}^\mathsf{T}\boldsymbol{V}\boldsymbol{v}$ for a vector $\boldsymbol{v}$ and a symmetric, positive semi-definite matrix $\boldsymbol{V}$ of fitting dimensions

$\boldsymbol{V}_{i,:}$ the ith row of matrix $\boldsymbol{V}$

$\boldsymbol{V}_{:,j}$ the jth column of matrix $\boldsymbol{V}$

$\boldsymbol{V}_{i_1:i_2,:}$	matrix block of matrix $\boldsymbol{V}$ including rows i_1 to $i_2 \geq i_1$
$\boldsymbol{V}_{:,j_1:j_2}$	matrix block of matrix $\boldsymbol{V}$ including columns j_1 to $j_2 \geq j_1$
${}^{\alpha}\boldsymbol{z}$	a kinematic quantity $\boldsymbol{z}$ given in the coordinates of the reference frame $\mathcal{K}_\alpha$
${}^{\check{\alpha}}\boldsymbol{z}$	a kinematic quantity $\boldsymbol{z}$ given in the coordinates of and relative to the reference frame $\mathcal{K}_\alpha$
${}_{i}\bullet$	a quantity $\bullet$ of the ith robot

Bibliography

[AlvaradoEtAl11] Alvarado, I.; Limon, D.; Muñoz de la Peña, D.; Maestre, J.; Ridao, M.; Scheu, H.; Marquardt, W.; Negenborn, R.; Schutter, B.D.; Valencia, F.; Espinosa, J.: A Comparative Analysis of Distributed MPC Techniques Applied to the HD-MPC Four-Tank Benchmark. Journal of Process Control, Vol. 21, No. 5, pp. 800–815, 2011.

[AnderssonEtAl19] Andersson, J.A.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M.: CasADi: A Software Framework for Nonlinear Optimization and Optimal Control. Mathematical Programming Computation, Vol. 11, No. 1, pp. 1–36, 2019.

[AraiPagelloParker02] Arai, T.; Pagello, E.; Parker, L.E.: Guest Editorial: Advances in Multirobot Systems. IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, pp. 655–661, 2002.

[AswaniEtAl13] Aswani, A.; Gonzalez, H.; Sastry, S.S.; Tomlin, C.: Provably Safe and Robust Learning-Based Model Predictive Control. Automatica, Vol. 49, No. 5, pp. 1216–1226, 2013.

[BächleHentzeltGraichen13] Bächle, T.; Hentzelt, S.; Graichen, K.: Nonlinear Model Predictive Control of a Magnetic Levitation System. Control Engineering Practice, Vol. 21, No. 9, pp. 1250–1258, 2013.

[BaiArcakWen11] Bai, H.; Arcak, M.; Wen, J.: Cooperative Control Design: A Systematic, Passivity-Based Approach. New York: Springer Science+Business Media, 2011.

[BarberDobkinHuhdanpaa96] Barber, C.B.; Dobkin, D.P.; Huhdanpaa, H.: The Quickhull Algorithm for Convex Hulls. ACM Transactions on Mathematical Software, Vol. 22, No. 4, p. 469–483, 1996.

[BaumannEtAl19] Baumann, D.; Mager, F.; Jacob, R.; Thiele, L.; Zimmerling, M.; Trimpe, S.: Fast Feedback Control over Multi-Hop Wireless Networks with Mode Changes and Stability Guarantees. ACM Transactions on Cyber-Physical Systems, Vol. 4, No. 2, 2019.

[BeagleBoard.org Foundation] BeagleBoard.org Foundation: BagleBoard.org - Blue. Available at https://beagleboard.org/blue. Accessed on Mar. 26, 2021.

[BemporadEtAl02] Bemporad, A.; Morari, M.; Dua, V.; Pistikopoulos, E.N.: The Explicit Linear Quadratic Regulator for Constrained Systems. Automatica, Vol. 38, No. 1, pp. 3–20, 2002.

[BerberichEtAl21] Berberich, J.; Köhler, J.; Müller, M.A.; Allgöwer, F.: Data-Driven Model Predictive Control with Stability and Robustness Guarantees. IEEE Transactions on Automatic Control, Vol. 66, No. 4, pp. 1702–1717, 2021.

[BergEtAl08] de Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M.: Computational Geometry: Algorithms and Applications. Berlin: Springer-Verlag, 3rd Edn., 2008.

[BertsekasTsitsiklis89] Bertsekas, D.P.; Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Englewood Cliffs: Prentice-Hall, 1989.

[BeschastnikhEtAl16] Beschastnikh, I.; Wang, P.; Brun, Y.; Ernst, M.D.: Debugging Distributed Systems: Challenges and Options for Validation and Debugging. Queue, Vol. 14, No. 2, pp. 91–110, 2016.

[BocciaGrüneWorthmann14] Boccia, A.; Grüne, L.; Worthmann, K.: Stability and Feasibility of State Constrained MPC without Stabilizing Terminal Constraints. Systems & Control Letters, Vol. 72, pp. 14–21, 2014.

[BottemaRoth90] Bottema, O.; Roth, B.: Theoretical Kinematics. New York: Dover Publications, 1990.

[BoydVandenberghe04] Boyd, S.; Vandenberghe, L.: Convex Optimization. Cambridge: Cambridge University Press, 2004.

[BraunEtAl16] Braun, P.; Grüne, L.; Kellett, C.; Weller, S.; Worthmann, K.: A Distributed Optimization Algorithm for the Predictive Control of Smart Grids. IEEE Transactions on Automatic Control, Vol. 61, No. 12, pp. 3898–3911, 2016.

[BraunEtAl18] Braun, P.; Faulwasser, T.; Grüne, L.; Kellett, C.M.; Weller, S.R.; Worthmann, K.: Hierarchical Distributed ADMM for Predictive Control with Applications in Power Networks. IFAC Journal of Systems and Control, Vol. 3, pp. 10–22, 2018.

[BulloCortésMartínez09] Bullo, F.; Cortés, J.; Martínez, S.: Distributed Control of Robotic Networks. Princeton: Princeton University Press, 2009.

[BurkVölzGraichen21] Burk, D.; Völz, A.; Graichen, K.: A Modular Framework for Distributed Model Predictive Control of Nonlinear Continuous-Time Systems (GRAMPC-D). Optimization and Engineering, 2021. DOI: 10.1007/s11081-021-09605-3.

[ChenAllgöwer98] Chen, H.; Allgöwer, F.: A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability. Automatica, Vol. 34, No. 10, pp. 1205–1217, 1998.

[ChenEtAl15] Chen, J.; Gauci, M.; Li, W.; Kolling, A.; Groß, R.: Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots. IEEE Transactions on Robotics, Vol. 31, No. 2, pp. 307–321, 2015.

[Chew89] Chew, P.L.: Constrained Delaunay Triangulations. Algorithmica, Vol. 4, pp. 97–108, 1989.

[ChristofidesEtAl13] Christofides, P.D.; Scattolini, R.; Muñoz de la Peña, D.; Liu, J.: Distributed Model Predictive Control: A Tutorial Review and Future Research Directions. Computers & Chemical Engineering, Vol. 51, pp. 21–41, 2013.

[DaiEtAl16] Dai, Y.; Kim, Y.; Wee, S.; Lee, D.; Lee, S.: Symmetric Caging Formation for Convex Polygonal Object Transportation by Multiple Mobile Robots Based on Fuzzy Sliding Mode Control. ISA Transactions, Vol. 60, pp. 321–332, 2016.

[DehghaniMenhaj16] Dehghani, M.A.; Menhaj, M.B.: Communication Free Leader-Follower Formation Control of Unmanned Aircraft Systems. Robotics and Autonomous Systems, Vol. 80, pp. 69–75, 2016.

[DickersonLapin91] Dickerson, S.; Lapin, B.: Control of an Omni-Directional Robotic Vehicle with Mecanum Wheels. In Proceedings of the National Telesystems Conference, pp. 323–328, Atlanta, 1991.

[Dijkstra59] Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, Vol. 1, No. 1, pp. 269–271, 1959.

[Dijkstra65] Dijkstra, E.: Solution of a Problem in Concurrent Programming Control. Communications of the ACM, Vol. 8, No. 9, p. 569, 1965.

[DoratoLevis71] Dorato, P.; Levis, A.: Optimal Linear Regulators: The Discrete-Time Case. IEEE Transactions on Automatic Control, Vol. 16, No. 6, pp. 613–620, 1971.

[EbelEberhard18] Ebel, H.; Eberhard, P.: Distributed Decision Making and Control for Cooperative Transportation Using Mobile Robots. In Tan, Y.; Shi, Y.; Tang, Q. (Eds.): Advances in Swarm Intelligence: 9th International Conference, ICSI 2018, Shanghai, pp. 89–101, Cham: Springer International Publishing, 2018.

[EbelEberhard19] Ebel, H.; Eberhard, P.: Optimization-Driven Control and Organization of a Robot Swarm for Cooperative Transportation. In Proceedings of the 8th IFAC Symposium on Mechatronic Systems (MECHATRONICS), Vienna, 2019.

[EbelEberhard21] Ebel, H.; Eberhard, P.: A Comparative Look at Two Formation Control Approaches Based on Optimization and Algebraic Graph Theory. Robotics and Autonomous Systems, Vol. 136, p. 103686, 2021.

[EbelEtAl21] Ebel, H.; Luo, W.; Yu, F.; Tang, Q.; Eberhard, P.: Design and Experimental Validation of a Distributed Cooperative Transportation Scheme. IEEE Transactions on Automation Science and Engineering, Vol. 18, No. 3, pp. 1157–1169, 2021.

[EbelSharafian ArdakaniEberhard17a] Ebel, H.; Sharafian Ardakani, E.; Eberhard, P.: Comparison of Distributed Model Predictive Control Approaches for Transporting a Load by a Formation of Mobile Robots. In Proceedings of the 8th Eccomas Thematic Conference on Multibody Dynamics, pp. 1–10, Prague, 2017.

[EbelSharafian ArdakaniEberhard17b] Ebel, H.; Sharafian Ardakani, E.; Eberhard, P.: Distributed Model Predictive Formation Control with Discretization-Free Path Planning for Transporting a Load. Robotics and Autonomous Systems, Vol. 96, pp. 211–223, 2017.

[EbelWahrenLuo20] Ebel, H.; Wahren, F.; Luo, W.: A Holonomic Extensible Robotic Agent for Research on Distributed Control. Anleitung AN-59, Institute of Engineering and Computational Mechanics, University of Stuttgart, 2020.

[EllisDurandChristofides14] Ellis, M.; Durand, H.; Christofides, P.D.: A Tutorial Review of Economic Model Predictive Control Methods. Journal of Process Control, Vol. 24, No. 8, pp. 1156–1178, 2014.

[ErősEtAl19] Erős, E.; Dahl, M.; Bengtsson, K.; Hanna, A.; Falkman, P.: A ROS2-Based Communication Architecture for Control in Collaborative and Intelligent Automation Systems. Procedia Manufacturing, Vol. 38, pp. 349–357, 2019.

[EschmannEbelEberhard21] Eschmann, H.; Ebel, H.; Eberhard, P.: Trajectory Tracking of an Omnidirectional Mobile Robot Using Gaussian Process Regression. Accepted for publication in at - Automatisierungstechnik, 2021.

[FaíscaDuaPistikopoulos07] Faísca, N.P.; Dua, V.; Pistikopoulos, E.N.: Multiparametric Linear and Quadratic Programming. In Pistikopoulos, E.N.; Georgiadis, M.C.; Dua, V. (Eds.): Multi-Parametric Programming, pp. 1–23. Weinheim: John Wiley & Sons, 2007.

[FarinaEtAl20] Farina, F.; Camisa, A.; Testa, A.; Notarnicola, I.; Notarstefano, G.: DISROPT: A Python Framework for Distributed Optimization. IFAC-PapersOnLine, Vol. 53, No. 2, pp. 2666–2671, 2020.

[FarivarnejadWilsonBerman16] Farivarnejad, H.; Wilson, S.; Berman, S.: Decentralized Sliding Mode Control for Autonomous Collective Transport by Multi-Robot Systems.

In Proceedings of the IEEE 55th Conference on Decision and Control (CDC), pp. 1826–1833, Las Vegas, 2016.

[FerramoscaEtAl13] Ferramosca, A.; Limon, D.; Alvarado, I.; Camacho, E.: Cooperative Distributed MPC for Tracking. Automatica, Vol. 49, No. 4, pp. 906–914, 2013.

[FerreauEtAl14] Ferreau, H.J.; Kirches, C.; Potschka, A.; Bock, H.G.; Diehl, M.: qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming. Mathematical Programming Computation, Vol. 6, No. 4, pp. 327–363, 2014.

[GilbertTan91] Gilbert, E.; Tan, K.: Linear Systems with State and Control Constraints: The Theory and Application of Maximal Output Admissible Sets. IEEE Transactions on Automatic Control, Vol. 36, No. 9, pp. 1008–1020, 1991.

[GodsilRoyle01] Godsil, C.; Royle, G.F.: Algebraic Graph Theory. New York: Springer-Verlag, 2001.

[GraetzMichaels18] Graetz, G.; Michaels, G.: Robots at Work. The Review of Economics and Statistics, Vol. 100, No. 5, pp. 753–768, 2018.

[GrauEtAl17] Grau, A.; Indri, M.; Bello, L.L.; Sauter, T.: Industrial Robotics in Factory Automation: From the Early Stage to the Internet of Things. In Proceedings of the IECON 2017 – 43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 6159–6164, Beijing, 2017.

[GroßMondadaDorigo06] Groß, R.; Mondada, F.; Dorigo, M.: Transport of an Object by Six Pre-Attached Robots Interacting via Physical Links. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp. 1317–1323, Orlando, 2006.

[Grünbaum03] Grünbaum, B.: Convex Polytopes. New York: Springer-Verlag, 2nd Edn., 2003.

[GrünbaumShephard69] Grünbaum, B.; Shephard, G.C.: Convex Polytopes. Bulletin of the London Mathematical Society, Vol. 1, No. 3, pp. 257–300, 1969.

[Hägele16] Hägele, M.: Robots Conquer the World [Turning Point]. IEEE Robotics & Automation Magazine, Vol. 23, No. 1, pp. 120–118, 2016.

[HartNilssonRaphael68] Hart, P.E.; Nilsson, N.J.; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, Vol. 4, No. 2, pp. 100–107, 1968.

[HazarikaDixit18] Hazarika, S.M.; Dixit, U.S.: Robotics: History, Trends, and Future Directions. In Davim, J.P. (Ed.): Introduction to Mechanical Engineering, pp. 213–239. Cham: Springer International Publishing, 2018.

[HertneckEtAl18] Hertneck, M.; Köhler; Trimpe, S.; Allgöwer, F.: Learning an Approximate Model Predictive Controller with Guarantees. IEEE Control Systems Letters, Vol. 2, No. 3, pp. 543–548, 2018.

[HewingKabzanZeilinger19] Hewing, L.; Kabzan, J.; Zeilinger, M.: Cautious Model Predictive Control Using Gaussian Process Regression. IEEE Transactions on Control Systems Technology, Vol. 28, No. 6, pp. 2736–2743, 2019.

[HolfeldEtAl16] Holfeld, B.; Wieruch, D.; Wirth, T.; Thiele, L.; Ashraf, S.A.; Huschke, J.; Aktas, I.; Ansari, J.: Wireless Communication for Factory Automation: An Opportunity for LTE and 5G Systems. IEEE Communications Magazine, Vol. 54, No. 6, pp. 36–43, 2016.

[HornungEtAl13] Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W.: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. Autonomous Robots, Vol. 34, pp. 189–206, 2013.

[HouskaFerreauDiehl11a] Houska, B.; Ferreau, H.J.; Diehl, M.: ACADO Toolkit – An Open-Source Framework for Automatic Control and Dynamic Optimization. Optimal Control Applications and Methods, Vol. 32, No. 3, pp. 298–312, 2011.

[HouskaFerreauDiehl11b] Houska, B.; Ferreau, H.J.; Diehl, M.: An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. Automatica, Vol. 47, No. 10, pp. 2279–2285, 2011.

[HuangOlsonMoore10] Huang, A.S.; Olson, E.; Moore, D.C.: LCM: Lightweight Communications and Marshalling. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4057–4062, Taipei, 2010.

[HuangZhang16] Huang, J.; Zhang, C.: Debugging Concurrent Software: Advances and Challenges. Journal of Computer Science and Technology, Vol. 31, No. 5, pp. 861–868, 2016.

[HuEtAl14] Hu, J.; Xie, L.; Xu, J.; Xu, Z.: Multi-Agent Cooperative Target Search. Sensors, Vol. 14, No. 6, pp. 9408–9428, 2014.

[Ilon75] Ilon, B.E.: Wheels for a Course Stable Selfpropelling Vehicle Movable in Any Desired Direction on the Ground or Some Other Base, US Patent No. 3,876,255, 1975.

[JacobEtAl16] Jacob, R.; Zimmerling, M.; Huang, P.; Beutel, J.; Thiele, L.: End-to-End Real-Time Guarantees in Wireless Cyber-Physical Systems. In Proceedings of the 2016 IEEE Real-Time Systems Symposium (RTSS), pp. 167–178, Porto, 2016.

[KaehlerBradski16] Kaehler, A.; Bradski, G.: Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. Sebastopol: O'Reilly Media, 2016.

[KalmanBertram60a] Kalman, R.; Bertram, J.: Control System Analysis and Design Via the "Second Method" of Lyapunov: I Continuous-Time Systems. Journal of Basic Engineering, Vol. 82, No. 2, pp. 371–393, 1960.

[KalmanBertram60b] Kalman, R.; Bertram, J.: Control System Analysis and Design Via the "Second Method" of Lyapunov: II Discrete-Time Systems. Journal of Basic Engineering, Vol. 82, No. 2, pp. 394–400, 1960.

[KäpernickGraichen14] Käpernick, B.; Graichen, K.: The Gradient Based Nonlinear Model Predictive Control Software GRAMPC. In Proceedings of the 2014 European Control Conference (ECC), pp. 1170–1175, Strasbourg, 2014.

[KavrakiLaValle16] Kavraki, L.E.; LaValle, S.M.: Motion Planning. In Siciliano, B.; Khatib, O. (Eds.): Springer Handbook of Robotics, pp. 139–162. Heidelberg: Springer-Verlag, 2nd Edn., 2016.

[KehoeEtAl15] Kehoe, B.; Patil, S.; Abbeel, P.; Goldberg, K.: A Survey of Research on Cloud Robotics and Automation. IEEE Transactions on Automation Science and Engineering, Vol. 12, No. 2, pp. 398–409, 2015.

[KennedyEberhart95] Kennedy, J.; Eberhart, R.: Particle Swarm Optimization. In Proceedings of the International Conference on Neural Networks, Vol. 4, pp. 1942–1948, Perth, 1995.

[Khalil02] Khalil, H.K.: Nonlinear Systems. Upper Saddle River: Prentice Hall, 2002.

[KöhlerMüllerAllgöwer18] Köhler, P.N.; Müller, M.A.; Allgöwer, F.: A Distributed Economic MPC Framework for Cooperative Control under Conflicting Objectives. Automatica, Vol. 96, pp. 368–379, 2018.

[KovačŽivkovićBašić04] Kovač, K.; Živković, I.; Bašić, B.D.: Simulation of Multi-Robot Reinforcement Learning for Box-Pushing Problem. In Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, pp. 603–606, Dubrovnik, 2004.

[KromannEtAl19] Kromann, L.; Malchow-Møller, N.; Skaksen, J.R.; Sørensen, A.: Automation and Productivity – a Cross-Country, Cross-Industry Comparison. Industrial and Corporate Change, Vol. 29, No. 2, pp. 265–287, 2019.

[KubeZhang96] Kube, C.R.; Zhang, H.: The Use of Perceptual Cues in Multi-Robot Box-Pushing. In Proceedings of the 1996 IEEE Intemational Conference on Robotics and Automation, pp. 2085–2090, Minneapolis, 1996.

[KwakernaakSivan72] Kwakernaak, H.; Sivan, R.: Linear Optimal Control Systems. New York: John Wiley & Sons, 1972.

[Laub79] Laub, A.: A Schur Method for Solving Algebraic Riccati Equations. IEEE Transactions on Automatic Control, Vol. 24, No. 6, pp. 913–921, 1979.

[LimonCalliessMaciejowski17] Limon, D.; Calliess, J.; Maciejowski, J.: Learning-Based Nonlinear Model Predictive Control. IFAC-PapersOnLine, Vol. 50, No. 1, pp. 7769–7776, 2017.

[LimonEtAl06] Limon, D.; Alamo, T.; Salas, F.; Camacho, E.: On the Stability of Constrained MPC without Terminal Constraint. IEEE Transactions on Automatic Control, Vol. 51, No. 5, pp. 832–836, 2006.

[LimonEtAl08] Limon, D.; Alvarado, I.; Alamo, T.; Camacho, E.: MPC for Tracking Piecewise Constant References for Constrained Linear Systems. Automatica, Vol. 44, No. 9, pp. 2382–2387, 2008.

[LimonEtAl10] Limon, D.; Alvarado, I.; Alamo, T.; Camacho, E.: Robust Tube-Based MPC for Tracking of Constrained Linear Systems with Additive Disturbances. Journal of Process Control, Vol. 20, No. 3, pp. 248–260, 2010.

[LinsenmayerHertneckAllgöwer21] Linsenmayer, S.; Hertneck, M.; Allgöwer, F.: Linear Weakly Hard Real-Time Control Systems: Time- and Event-Triggered Stabilization. IEEE Transactions on Automatic Control, Vol. 66, No. 4, pp. 1932–1939, 2021.

[LiuEtAl10] Liu, J.; Chen, X.; Muñoz de la Peña, D.; Christofides, P.D.: Sequential and Iterative Architectures for Distributed Model Predictive Control of Nonlinear Process Systems. AIChE Journal, Vol. 56, No. 8, pp. 2137–2149, 2010.

[LynchPark17] Lynch, K.M.; Park, F.C.: Modern Robotics: Mechanics, Planning, and Control. Cambridge: Cambridge University Press, 2017.

[Maciejowski01] Maciejowski, J.: Predictive Control with Constraints. Harlow: Pearson Education, 2001.

[MagniScattolini06] Magni, L.; Scattolini, R.: Stabilizing Decentralized Model Predictive Control of Nonlinear Systems. Automatica, Vol. 42, No. 7, pp. 1231–1236, 2006.

[MataréNilssonSimsarian95] Matarić, M.J.; Nilsson, M.; Simsarian, K.T.: Cooperative Multi-Robot Box-Pushing. In Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, pp. 556–561, Pittsburgh, 1995.

[MayneEtAl00] Mayne, D.; Rawlings, J.; Rao, C.; Scokaert, P.: Constrained Model Predictive Control: Stability and Optimality. Automatica, Vol. 36, No. 6, pp. 789–814, 2000.

[MayneMichalska90] Mayne, D.Q.; Michalska, H.: Receding Horizon Control of Nonlinear Systems. IEEE Transactions on Automatic Control, Vol. 35, No. 7, pp. 814–824, 1990.

[MayneSeronRaković05] Mayne, D.; Seron, M.; Raković, S.: Robust Model Predictive Control of Constrained Linear Systems with Bounded Disturbances. Automatica, Vol. 41, No. 2, pp. 219–224, 2005.

[MesbahiEgerstedt10] Mesbahi, M.; Egerstedt, M.: Graph Theoretic Methods in Multiagent Networks. Princeton: Princeton University Press, 2010.

[MinguezLamirauxLaumond16] Minguez, J.; Lamiraux, F.; Laumond, J.P.: Motion Planning and Obstacle Avoidance. In Siciliano, B.; Khatib, O. (Eds.): Springer Handbook of Robotics, pp. 1177–1202. Heidelberg: Springer-Verlag, 2nd Edn., 2016.

[MitzeMönnigmann20] Mitze, R.; Mönnigmann, M.: A Dynamic Programming Approach to Solving Constrained Linear–Quadratic Optimal Control Problems. Automatica, Vol. 120, p. 109132, 2020.

[MiyataEtAl02] Miyata, N.; Ota, J.; Arai, T.; Asama, H.: Cooperative Transport by Multiple Mobile Robots in Unknown Static Environments Associated with Real-Time Task Assignment. IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, pp. 769–780, 2002.

[MönnigmannOttenJost15] Mönnigmann, M.; Otten, J.; Jost, M.: Nonlinear MPC Defines Implicit Regional Optimal Control Laws. IFAC-PapersOnLine, Vol. 48, No. 23, pp. 142–147, 2015.

[MüllerAllgöwer14] Müller, M.A.; Allgöwer, F.: Distributed Economic MPC: A Framework for Cooperative Control Problems. IFAC Proceedings Volumes, Vol. 47, No. 3, pp. 1029–1034, 2014.

[MüllerRebleAllgöwer12] Müller, M.A.; Reble, M.; Allgöwer, F.: Cooperative Control of Dynamically Decoupled Systems via Distributed Model Predictive Control. International Journal of Robust and Nonlinear Control, Vol. 22, No. 12, pp. 1376–1397, 2012.

[MurraySastry93] Murray, R.; Sastry, S.: Nonholonomic Motion Planning: Steering Using Sinusoids. IEEE Transactions on Automatic Control, Vol. 38, No. 5, pp. 700–716, 1993.

[NegenbornMaestre14] Negenborn, R.R.; Maestre, J.M.: Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities. IEEE Control Systems Magazine, Vol. 34, No. 4, pp. 87–97, 2014.

[NeumannKitts16] Neumann, M.A.; Kitts, C.A.: A Hybrid Multirobot Control Architecture for Object Transport. IEEE/ASME Transactions on Mechatronics, Vol. 21, No. 6, pp. 2983–2988, 2016.

[NocedalWright06] Nocedal, J.; Wright, S.J.: Numerical Optimization. New York: Springer Science+Business Media, 2nd Edn., 2006.

[NotarstefanoNotarnicolaCamisa19] Notarstefano, G.; Notarnicola, I.; Camisa, A.: Distributed Optimization for Smart Cyber-Physical Networks. Foundations and Trends in Systems and Control, Vol. 7, No. 3, p. 253–383, 2019.

[O'Rourke98] O'Rourke, J.: Computational Geometry in C. New York: Cambridge University Press, 2nd Edn., 1998.

[Pabst19] Pabst, M.: Experimental Study on Control of a Mobile Manipulator for a Pick-and-Place Task. Bachelorarbeit BSC-113, Institute of Engineering and Computational Mechanics, University of Stuttgart, 2019.

[PoliKennedyBlackwell07] Poli, R.; Kennedy, J.; Blackwell, T.: Particle Swarm Optimization. Swarm Intelligence, Vol. 1, pp. 33–57, 2007.

[PrimbsNevistić00] Primbs, J.A.; Nevistić, V.: Feasibility and Stability of Constrained Finite Receding Horizon Control. Automatica, Vol. 36, No. 7, pp. 965–971, 2000.

[PuckEtAl20] Puck, L.; Keller, P.; Schnell, T.; Plasberg, C.; Tanev, A.; Heppner, G.; Roennau, A.; Dillmann, R.: Distributed and Synchronized Setup Towards Real-Time Robotic Control Using ROS2 on Linux. In Proceedings of the 16th IEEE International Conference on Automation Science and Engineering (CASE), pp. 1287–1293, Hong Kong, 2020.

[QinBadgwell03] Qin, S.J.; Badgwell, T.A.: A Survey of Industrial Model Predictive Control Technology. Control Engineering Practice, Vol. 11, No. 7, pp. 733–764, 2003.

[QuigleyEtAl09] Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A.: ROS: An Open-Source Robot Operating System. In Proceedings of the Open-Source Software Workshop of the International Conference on Robotics and Automation (ICRA), Kobe, 2009.

[RahimiEtAl19] Rahimi, M.; Gibb, S.; Shen, Y.; La, H.M.: A Comparison of Various Approaches to Reinforcement Learning Algorithms for Multi-Robot Box Pushing.

In Fujita, H.; Nguyen, D.C.; Vu, N.P.; Banh, T.L.; Puta, H.H. (Eds.): Advances in Engineering Research and Application, pp. 16–30, Cham: Springer International Publishing, 2019.

[RaimondoMagniScattolini07] Raimondo, D.; Magni, L.; Scattolini, R.: Decentralized MPC of Nonlinear Systems: An Input-to-State Stability Approach. International Journal of Robust and Nonlinear Control, Vol. 17, No. 17, pp. 1651–1667, 2007.

[Raković12] Raković, S.V.: Invention of Prediction Structures and Categorization of Robust MPC Syntheses. IFAC Proceedings Volumes, Vol. 45, No. 17, pp. 245–273, 2012.

[Rao09] Rao, S.S.: Modern Methods of Optimization. In Rao, S.S. (Ed.): Engineering Optimization, pp. 693–736. Hoboken: John Wiley & Sons, 2009.

[RawlingsMayneDiehl17] Rawlings, J.B.; Mayne, D.Q.; Diehl, M.M.: Model Predictive Control: Theory, Computation, and Design. Santa Barbara: Nob Hill Publishing, 2nd Edn., 2017.

[RichardsHow07] Richards, A.; How, J.: Robust Distributed Model Predictive Control. International Journal of Control, Vol. 80, No. 9, pp. 1517–1531, 2007.

[SachsEtAl18] Sachs, J.; Wikstrom, G.; Dudda, T.; Baldemair, R.; Kittichokechai, K.: 5G Radio Network Design for Ultra-Reliable Low-Latency Communication. IEEE Network, Vol. 32, No. 2, pp. 24–31, 2018.

[SavelaTurjaOksanen18] Savela, N.; Turja, T.; Oksanen, A.: Social Acceptance of Robots in Different Occupational Fields: A Systematic Literature Review. International Journal of Social Robotics, Vol. 10, pp. 493–502, 2018.

[Scattolini09] Scattolini, R.: Architectures for Distributed and Hierarchical Model Predictive Control – A Review. Journal of Process Control, Vol. 19, No. 5, pp. 723–731, 2009.

[SchiehlenEberhard14] Schiehlen, W.; Eberhard, P.: Applied Dynamics. Cham: Springer International Publishing, 2014.

[Schneider93] Schneider, R.: Convex Bodies: The Brunn-Minkowski Theory. Encyclopedia of Mathematics and its Applications. Cambridge: Cambridge University Press, 1993.

[Schnelle18] Schnelle, F.: Modellprädiktive Ansätze zur Regelung von unteraktuierten Mehrkörpersystemen (in German). Dissertation, Schriften aus dem Institut für Technische und Numerische Mechanik der Universität Stuttgart, Vol. 54. Aachen: Shaker Verlag, 2018.

[SchwertassekWallrapp99] Schwertassek, R.; Wallrapp, O.: Dynamik flexibler Mehrkörpersysteme: Methoden der Mechanik zum rechnergestützten Entwurf und zur Analyse mechatronischer Systeme (in German). Braunschweig: Vieweg, 1999.

[ScokaertMayneRawlings99] Scokaert, P.; Mayne, D.; Rawlings, J.: Suboptimal Model Predictive Control (Feasibility Implies Stability). IEEE Transactions on Automatic Control, Vol. 44, No. 3, pp. 648–654, 1999.

[ScokaertRawlings98] Scokaert, P.; Rawlings, J.: Constrained Linear Quadratic Regulation. IEEE Transactions on Automatic Control, Vol. 43, No. 8, pp. 1163–1169, 1998.

[SedlaczekEberhard06] Sedlaczek, K.; Eberhard, P.: Using Augmented Lagrangian Particle Swarm Optimization for Constrained Problems in Engineering. Structural and Multidisciplinary Optimization, Vol. 32, No. 4, pp. 277–286, 2006.

[Shabana97] Shabana, A.A.: Definition of the Slopes and the Finite Element Absolute Nodal Coordinate Formulation. Multibody System Dynamics, Vol. 1, pp. 339–348, 1997.

[Shabana20] Shabana, A.A.: Dynamics of Multibody Systems. Cambridge: Cambridge University Press, 5th Edn., 2020.

[Sharir89] Sharir, M.: Algorithmic Motion Planning in Robotics. Computer, Vol. 22, No. 3, pp. 9–19, 1989.

[ShiEberhart98] Shi, Y.; Eberhart, R.: A Modified Particle Swarm Optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, pp. 69–73, Anchorage, 1998.

[Sibson78] Sibson, R.: Locally Equiangular Triangulations. The Computer Journal, Vol. 21, No. 3, pp. 243–245, 1978.

[Sloan93] Sloan, S.: A Fast Algorithm for Generating Constrained Delaunay Triangulations. Computers & Structures, Vol. 47, No. 3, pp. 441–450, 1993.

[StewartEtAl10] Stewart, B.T.; Venkat, A.N.; Rawlings, J.B.; Wright, S.J.; Pannocchia, G.: Cooperative Distributed Model Predictive Control. Systems & Control Letters, Vol. 59, No. 8, pp. 460–469, 2010.

[StewartWrightRawlings11] Stewart, B.T.; Wright, S.J.; Rawlings, J.B.: Cooperative Distributed Model Predictive Control for Nonlinear Systems. Journal of Process Control, Vol. 21, No. 5, pp. 698–704, 2011.

[TangEtAl18] Tang, Q.; Ding, L.; Yu, F.; Zhang, Y.; Li, Y.; Tu, H.: Swarm Robots Search for Multiple Targets Based on an Improved Grouping Strategy. IEEE/ACM

Transactions on Computational Biology and Bioinformatics, Vol. 15, No. 6, pp. 1943–1950, 2018.

[TiderkoHoellerRöhling16] Tiderko, A.; Hoeller, F.; Röhling, T.: The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems. In Koubaa, A. (Ed.): Robot Operating System (ROS): The Complete Reference, Vol. 1, pp. 629–650. Cham: Springer International Publishing, 2016.

[TøndelJohansenBemporad03] Tøndel, P.; Johansen, T.A.; Bemporad, A.: An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions. Automatica, Vol. 39, No. 3, pp. 489–497, 2003.

[TuciAlkilabiAkanyeti18] Tuci, E.; Alkilabi, M.H.M.; Akanyeti, O.: Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art. Frontiers in Robotics and AI, Vol. 5, 2018.

[van den BerghEngelbrecht06] van den Bergh, F.; Engelbrecht, A.: A Study of Particle Swarm Optimization Particle Trajectories. Information Sciences, Vol. 176, No. 8, pp. 937–971, 2006.

[Wahren20] Wahren, F.: Implementierung, experimentelle Umsetzung und Erprobung eines Kartierungs- und Lokalisierungsalgorithmus für mobile Roboter mit LIDAR-Sensoren (in German). Bachelorarbeit BSC-123, Institute of Engineering and Computational Mechanics, University of Stuttgart, 2020.

[WampflerSaleckerWittenburg89] Wampfler, G.; Salecker, M.; Wittenburg, J.: Kinematics, Dynamics, and Control of Omnidirectional Vehicles with Mecanum Wheels. Mechanics of Structures and Machines, Vol. 17, No. 2, pp. 165–177, 1989.

[WangDavison73] Wang, S.H.; Davison, E.: On the Stabilization of Decentralized Control Systems. IEEE Transactions on Automatic Control, Vol. 18, No. 5, pp. 473–478, 1973.

[WangSchwager16] Wang, Z.; Schwager, M.: Multi-Robot Manipulation without Communication. In Chong, N.Y.; Cho, Y.J. (Eds.): Distributed Autonomous Robotic Systems, pp. 135–149, Tokyo: Springer Japan, 2016.

[WangSilva06] Wang, Y.; de Silva, C.W.: Multi-Robot Box-Pushing: Single-Agent Q-Learning vs. Team Q-Learning. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3694–3699, Beijing, 2006.

[WenHeZhu18] Wen, J.; He, L.; Zhu, F.: Swarm Robotics Control and Communications: Imminent Challenges for Next Generation Smart Logistics. IEEE Communications Magazine, Vol. 56, No. 7, pp. 102–107, 2018.

[Woernle16] Woernle, C.: Mehrkörpersysteme: Eine Einführung in die Kinematik und Dynamik von Systemen starrer Körper (in German). Berlin: Springer Vieweg, 2016.

[YamadaSaito01] Yamada, S.; Saito, J.: Adaptive Action Selection without Explicit Communication for Multirobot Box-Pushing. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 31, No. 3, pp. 398–404, 2001.

[ZeidisZimmermann19] Zeidis, I.; Zimmermann, K.: Dynamics of a Four-Wheeled Mobile Robot with Mecanum Wheels. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 99, No. 12, p. e201900173, 2019.

[ZimmermannEtAl16] Zimmermann, K.; Zeidis, I.; Schale, F.; Flores-Alvarez, P.A.: Mechanics of Mobile Robots with Mecanum Wheels. In Parenti-Castelli, V.; Schiehlen, W. (Eds.): ROMANSY 21 - Robot Design, Dynamics and Control, pp. 103–111, Cham: Springer International Publishing, 2016.