TU WIEN Informatics

Coordination and Control of Robotic Multi-agent Systems in Confined Environments

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Ing. Dipl.-Ing. Andreas Brandstätter, BSc
Registration Number 0927824

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Dr.h.c. Radu Grosu

The dissertation has been reviewed by:

Scott A. Smolka

Martin Saska

Vienna, January 15, 2025

Andreas Brandstätter

Erklärung zur Verfassung der Arbeit

Ing. Dipl.-Ing. Andreas Brandstätter, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 15. Jänner 2025

Andreas Brandstätter

Acknowledgements

I would like to express my sincere gratitude to my advisor, Univ.Prof. Radu Grosu, who already mentored me throughout my master's thesis and now my doctoral thesis, gave me enough freedom and flexibility to pursue my ideas and research interests and always greatly supported these.

I am thankful for the excellent collaboration with my co-authors, especially Luigi Berducci, Axel Brunnbauer, Ashish Tiwari, Distinguished Professor Scott A. Smolka, Professor Scott D. Stoller, Ramin Hasani, and Mathias Lechner, on carrying out joint research and sharing the highs and also depths of academic publishing.

I am grateful for the heartfelt welcome and support by Distinguished Professor Scott A. Smolka and Professor Scott D. Stoller during my research exchange at the College of Engineering and Applied Sciences, Department of Computer Science at Stony Brook University, New York. This international exchange was kindly supported by a scholarship from the Austrian Marshall Plan Foundation.

I want to thank Distinguished Professor Scott A. Smolka, and Associate Professor Martin Saska for reviewing my dissertation, as well as Univ.Prof. Christian Bettstetter and Assistant Prof.x Katta Spiel for serving on my proficiency evaluation committee.

Even though teaching obligations were sometimes costly in terms of time, I am grateful for the experience and what I learned while teaching and organizing *Operating Systems* and *Autonomous Racing Cars* lectures. For sharing this duty, I thank my Univ.Ass.-colleagues Michael Platzer, David Lung, and Florian Mihola.

I am grateful for the extremely nice work environment provided by my colleagues at the Research Unit of Cyber-Physical Systems at the Institute of Computer Engineering, Faculty of Informatics of TU Wien. Foremost to our office management, Elisa Di Cristo, for all her support and the sunshine that she brings to the office. The further professors at our Research Unit, Ao.Univ.Prof. Puschner Peter, and Univ.Prof. Bartocci Ezio for their advice and for always lending me a sympathetic ear.

My roommates Emad Jacob Maroun and Francesco Pontiggia for the calm work environment and nice discussions. Michele Chiari for being the best pasta chef and making Thursdays truly enjoyable. All colleagues, in addition to those already mentioned, especially Mónika Farsang, Felix Resch, Luigi Berducci, Stefan Schupp, Drishti Yadav, Axel Brunnbauer, Luca Di Stefano, Mahum Naseer, and Zlatan Tucaković for their pleasant company at multiple occasions of cake cutting experiments, as well as out-of-office activities.

During my time at the Research Unit of Cyber-Physical Systems, we formed the amazing F1Tenth team *Scuderia Segfault*, which I am very grateful to be part of. The team is not only highly successful in winning top podium placements at international races but is genuinely a group of fellows. I thank Felix Resch, Luigi Berducci, Mónika Farsang, Moritz Christamentl, Agnes Poks, Larisa Clement, Mihai Stanusoiu, Elisa Di Cristo, Dennis Erdogan, Philipp Mandl, Philipp Gratzer, Piet Carl Kaul, Samuel Lechner, Fabian Kresse, Daniel Scheuchenstuhl, and Stefan Ulmer for a great time in jointly working on the cars and participating in numerous international races.

I thank my family and friends, amongst others, especially those I met at the technical college St. Pölten and my studies at TU Wien, those from Club Alpbach Lower Austria, from Voluntary Fire Brigade Sieghartskirchen, District Fire Brigade Association of Tulln and State Fire Brigade Association of Lower Austria, for all the motivating words during my studies.

Last but by no means least, I am very grateful to my parents, who continuously supported me throughout my entire studies.

Abstract

In nature, ants are well known for collectively carrying loads, birds fly in large flocks, and fish gather in schools. Such collective behavior is not only relevant in nature, but also for an increasing number of robotic systems. Multi-agent Systems (MAS) can provide benefits for a wide range of application areas, as groups of robots can collectively perform tasks that are way beyond the capabilities of individual agents. In this thesis, we introduce methods to coordinate and control robotic agents in both antagonistic and collaborative MAS in confined environments. We exemplarily work on two different scenarios, which are autonomous driving respectively racing, and formation control for a flock of quadcopters. For each of these scenarios, we use a three-pronged approach consisting of theoretical analysis and reasoning, implementation and simulation, and hardware experiments.

For the autonomous racing task, we use machine learning to drive an agent in a timed racing scenario. We show that in simulation, the model-based deep Reinforcement Learning (RL) algorithm outperforms a number of other model-free RL algorithms, and we empirically demonstrate that this control method is able to successfully transfer the learned policy from simulation to a real-world test environment. In our collaborative MAS consisting of a group of drones, we control quadcopters to form and maintain a flock formation. We introduce Spatial Predictive Control (SPC) as a fully distributed control method that is based only on the position of the individual drone itself and on those of neighboring drones. Hardware experiments demonstrate SPC's robustness against a potential sim-to-real transfer gap and its capability to perform properly in the presence of significant sensor noise and the extra latency of positional and control signals. For scenarios, where even positional observations are not possible, we present Distributed Distance-based Control (DDC), which is fully distributed and solely based on scalar distance measurements and local position estimation. To the best of our knowledge, we are the first to demonstrate such a controller on aerial MAS and perform experiments with real hardware drones.

Contents

CHAPTER 1

Introduction

"Alone we can do so little; together we can do so much.", says Helen Keller, an American author and lecturer. This is not only true for humans, but also for a great number of animals and even robotic systems. Ants are well known for collectively carrying heavy loads, birds fly in large flocks or V-formations, and fish gather in schools to protect against predators. Also in robotics, Multi-agent Systems (MAS) can provide a wide range of benefits for very different application areas, as robots can collectively perform tasks that are way beyond the capabilities of individual agents. In this thesis, we introduce methods to coordinate and control MAS in confined environments.

As benefits rarely come without additional challenges, this is also true for MAS. First, coordinating and controlling MAS needs appropriate methods for perception and localization. For most control methods, agents also need to be aware of the position of other agents, to take proper actions. As this might be difficult to achieve in practice, in this thesis, we work on obviating this requirement at least partially. Secondly, some control methods might require individual agents to exchange information among them. This requires careful consideration in real-world applications, as typically transmission bandwidth is limited, has only a certain range, and might even be interrupted due to external influences. To address this issue, we seek ways to limit necessary inter-agent communication for our coordination method to a minimum. Lastly, the individual agents use this information, which was perceived by themselves and received from other agents, to infer control actions. These control actions are subject to their task-specific objective, such that they form and maintain the respective formation. Note, that this formation is not governed by a central coordinator or master controller, but it solely emerges by the individual agent's actions. In our thesis, we propose control methods that are more efficient in reaching and maintaining such formations.

1.1 Motivation: Biological Multi-agent Systems

When looking at ants, we see that they can achieve incredible things when working together. They collectively carry loads much heavier than their own body weight [Gel15], when transporting food or building materials. For this collective motion there is a high degree of coordination required, which demands efficient communication between ants. For *Paratrechina longicornis* ants, there is no single leader, but navigation, while collectively carrying, is a distributed effort, as it emerges by the joint behavior of the individual ants [Gel15]. Ants are also able to build floating rafts out of their interlinked bodies to survive floods [MTH11] and float effortlessly for multiple days. These rafts of *Solenopsis invicta* ants demonstrate self-assembly and self-healing abilities, which makes the overall ant colony behave like a superorganism.

Flocks of starlings are more resilient to predators as they would be alone [Goo17]. Collective behavior observed in starling murmurations of *Sturnus vulgaris* is reported to be an effective anti-predator adaptation. Large numbers of birds are safer when gathering at suitable roosting sites together, while in contrast, they would be more vulnerable flying to the roost individually. Multiple advantages are reported for murmuring at the roosting site, which are the dilution effect, higher vigilance leading to a detection effect, and also predator confusion [Goo17]. For birds, V-shaped formation greatly helps to reduce energy consumption when flying over long distances [Por14; LS70]. Groups of northern bald ibises *Geronticus eremita* which fly in a V-shaped flock, do not only place themselves in aerodynamically optimum positions but also synchronize their flapping to maximize the upwash benefit [Por14].

While cooperation is beneficial in the aforementioned examples, we might also observe antagonistic behavior within groups of animals. Competition for resources, such as food and water, is one of the most obvious reasons for antagonistic behavior between individuals [SF51; Geo13]. Aggressive behavior in such competitive scenarios can be considered as an optimization problem, where such strategies evolve when the benefits of aggression outweigh its costs and when the benefit/cost ratio is higher than other, non-aggressive, behavior [SF51]. Social structures and hierarchy fights, as well as inter-male fighting to win mates, are observed between animals [McG86]. Such behaviors involve elaborated strategies and careful consideration of benefits versus risks of the confrontation [Pay98].

Further examples of biological multi-agent systems include swarms of bees, fish schooling, herds of mammals, and many more [Goo17]. The aforementioned examples of cooperative behaviour of animals show the benefit of collaboration that helps to achieve a goal better or more efficiently. In some cases it would even be impossible to achieve the goal individually, so the emergent behavior of a group of individuals is crucial to solve the task. In engineering, there was always the idea to mimic nature in certain aspects, including collective behavior. Therefore also, the collaboration of individual robots is a highly relevant topic relating to research in the area of MAS. Analogously, also antagonistic behavior has its counterparts in engineering applications when individual agents compete to solve a specific task faster, more efficiently, or more reliably.

1.2 Robotic Multi-agent Systems

Since Josef Čapek coined the term *robot* [Čap20] in 1920, robots have been used for a multitude of tasks in various application areas. These range from industrial automation, inspection and maintenance in hazardous environments, transport and mobility, to entertainment and arts. Even though in many of these applications, more than one robot is used, this is not necessarily a MAS. For the context of this thesis, we consider a group of robots as a MAS subject to the following key properties.

- **Emergent Behaviour from Individual Control Actions:**
  Control actions should be derived individually by each agent in a distributed manner without any central coordinator or master controller. Features and behavioral patterns of the overall system will emerge from the actions of individual agents rather than being imposed by an external control entity. Actions of agents are based on sensed information, optionally further information that is exchanged among the agents or from other sources, and the task-specific objective. This property is important for a reactive and resilient system, as it does not require a single entity that processes all relevant information to determine every agent's control action. If there were such a central coordinator or master controller, the robots could be considered as some type of distributed actuator, but not individual agents.

- **Agents as Partially-Observable Markov Decision Processes (POMDPs):**
  In a MAS, each individual agent can be described by a Partially-Observable Markov Decision Process (POMDP). In this description, the state space includes the individual agent's own state, as well as the state of other agents and the environment. Each of the state variables might be continuous and infinite in value and time domain and, therefore, should be appropriately mapped to a discrete set with the required precision and value domain. Most importantly, it is, in most cases, impossible to sense (measure) the entire state, which implies only partial observability. To derive an appropriate action, maximizing the agent's reward, it is therefore necessary to estimate a belief state based on the available observations.

- **Variable Number of Agents:**
  In general, the system should be capable of dealing with any number of agents. However, in practical scenarios, the maximum number of agents can be limited by different factors such as spatial constraints, the cost of robots, or hardware resource limitations. Nevertheless, the system design and algorithms itself should neither limit the minimum nor the maximum number of agents by an arbitrary number. This makes it possible to scale the system to larger and smaller application scenarios. It is also a necessary property to make the overall system resilient against the fault of individual agents. While in a cooperative scenario, an unlimited number of agents should be able to work together to achieve a certain goal, also in antagonistic scenarios, the number of competing agents should not be restricted.

- **Interchangeable Agents:**
  The system shall not depend on the presence of a designated leader or any kind of distinctive master agent. In the simplest case, all agents are completely identical and employ the same control scheme. Also, their control actions should react to every other agent's behavior in the same way, which makes agents interchangeable without influencing the behavior of the whole system. In more complex scenarios, there could be different types and categories of robots (e.g. ground-based robots and aerial robots). However, there should be no fixed subset of agents, where others depend on their presence and/or actions.

- **Individual Sensing Capabilities:**
  Each individual agent should have sensing capabilities to gather information about its own state and the environment. The state of an agent could be the position or pose, but it also includes further information such as internal temperature or battery level. Relevant information about the environment includes potential obstacles, surface conditions, other objects, visual impressions, and further information relevant to any specific task. Most importantly, the environment explicitly includes other agents, where distance, position, and pose could be relevant information to capture with sensors. Having sensing capabilities is also a necessary condition for the first property of this list, which is to determine its own control actions. Note that the full system state is, in general, only partially observable, which relates to the property of describing *agents as POMDPs*.

From these aforementioned properties, one might conclude, that individual agents are totally independent and solely base their control actions on information gathered by their own sensors. However, if not stated otherwise in the specific chapter of this thesis, we do not make such strong assumptions about MAS, but allow to have the following abilities:

- **Information Exchange:**
  Agents might exchange information (e.g., over radio transmission protocols) with each other. This can include its own state information, any information about the environment, or derived control actions. It is also possible to re-transmit information that was sent by another agent, building some sort of relay network to exchange information over longer distances. However, as in practice, bandwidth is limited, agents might only exchange a subset of their available information with a limited update rate. Furthermore, they could limit their communication to agents, which are in close vicinity, often referred to as *neighbors*. This is especially relevant, as otherwise the property *variable number of agents* would lead to an unmanageable quadratic growth of messages from every to every agent.

- **Support Infrastructure:**
  Global-Navigation Satellite System (GNSS) is a well-known and widely used method to determine its own position at nearly any point on the Earth's surface. For outdoor applications, this is state-of-the-art for robot localization. There exist comparable types of indoor localization systems [Fer17; Lao18], which can be used to determine robot positions. We consider both types of such systems as support infrastructure, which can be available to a MAS for a given scenario unless otherwise stated in the respective chapter of this thesis. Other types of comparable support infrastructure could be passive or active optical markers, as well as radio transmission markers.

- **Off-board Computation:**
  While in general, we assume for all agents of a MAS to derive *individual control actions* in a distributed manner without any central coordinator or master controller, we relaxed this property on an implementation level only. For some hardware implementation tests, we send all the necessary information from the agents to a computer and execute our algorithm implementation there. Results are then again sent to the agents to carry out these actions. One might consider this approach as central coordinator or master controller, but actually, all computations on the computer are done in individual software nodes without exchanging any information between them. We took this approach to simplify implementation and allow for deeper debugging and logging capabilities of algorithmic internals. With additional porting effort, but without any algorithmic modification, the very same implementation could actually be executed on the robot hardware, obviating the need for this off-board computation. Therefore this is not an actual property of the MAS, but only of our proof-of-concept implementations.

1.3 Application Areas

There are many application areas that benefit from the emergent properties when using robotic MAS. These MAS can either be collaborative or antagonistic — for both types, we find a multitude of examples in literature. Apart from that, we also notice applications that deploy multiple robots that do not feature the properties of a MAS introduced in Section 1.2, which we also briefly present in Section 1.3.3.

1.3.1 Collaborative Robotic MAS

Robotic MAS can collectively perform tasks that are way beyond the capabilities of individual agents, as shown by multiple surveys [Rin21; Chu18a]. For Search-and-Rescue (SAR) applications, there is a wide variety of collaborative multi-agent approaches using ground, aerial, floating, and underwater vehicles [Que20b]. Specific examples are drone fleets that can help rescue operations in disaster scenarios [Câm14]. Collaborative mapping can be used in earthquake-damaged buildings via ground and aerial robots [Mic14]. Further scenarios for multi-agent systems include transportation, where they can collectively carry a heavy load while still being much more agile than a single larger drone [MFK11; LK18].

Inspection of large and complex structures can be done by aerial MAS [Mia20; Jin20]. MAS can also play a role in environmental monitoring, space exploration, agriculture, entertainment, and industrial maintenance [Sch20].

1.3.2 Antagonisitc Robotic MAS

There are applications and scenarios where robotic agents are in competition with each other and, therefore, might exhibit antagonistic behavior. A quintessential example of such a competitive scenario is autonomous racing [Bet22]. If there is no coordination with other road users, autonomous driving, in general, might also be considered competitive to some extent, while individual agents still need to obey the traffic rules. Autonomous delivery of goods using aerial vehicles [Lem21] or ground-based vehicles [Li20] is another example of when multiple competing operators deploy their autonomous robots in the same airspace, respectively streets.

1.3.3 Centrally Controlled Multi-robot Systems

In 2012, there was the first large-scale light show using Unmanned Aerial Vehicles (UAVs) demonstrated at the open-air music festival *Klangwolke* in Linz, Austria [Hör12]. At this time, 50 quadcopters equipped with multicolor LEDs were used to form a 3D display in the night sky. Thereafter many more light shows were performed with up to thousands of UAVs [ZCI21]. Unlike the aforementioned description of MAS, where agents dynamically place themselves such that the respective formation is reached, these light shows follow a pre-determined movement schedule. The formations and movement patterns are generated beforehand and fully pre-programmed [Wen22; Hua21; NK22]. For the context of our work, we, therefore, do not consider such systems with a central coordinator as MAS.

In warehouses and distribution centers, robotic handling systems are increasingly applied. There are automated robotic handling systems, such as shuttle-based storage and retrieval systems, shuttle-based compact storage systems, and robotic mobile fulfillment systems [ADR19]. Large logistics companies operate fleets of hundreds of mobile robots to move shelves in warehouses [Mou02; DW08]. These robots are used to move the shelves from their storage location to picking stations and vice-versa. For such a warehouse system, a centralized optimization approach is used to optimize the path for each robot based on information about the location of all goods on the shelves, the allocation of tasks to picking stations, and joint robot trajectory planning [LCH19; Bar20]. As the trajectories of such robot fleets are centrally controlled, we also do not consider these systems as MAS, in the context of our work.

1.3.4 MAS in Confined Environments

Every robotic system deployed in any real-world scenario is subject to limitations imposed by its environment. Notably, these are space constraints given by the designated operation area. In outdoor scenarios, e.g., SAR [Câm14], this area is defined by the given search perimeter and restricts where UAVs shall operate. Similarly, for any application in an inhabited area, e.g., drone shows in cities [Hör12], the allowed operation area is given by regulatory constraints. For indoor applications, space constraints are even more confined [Raz21]. Despite the limitation of the available space, robots are also hindered in their free movement by static or dynamic obstacles. These obstacles could be, e.g., trees [Loq21; Zho22], structural building elements and furniture [Raz21], as well as artificial obstacles [Son23]. In this thesis, we, therefore, deal with obstacle avoidance as well as compactness metrics to assess if a group of robots fits in a certain confined volume and is able to operate under these environmental constraints.

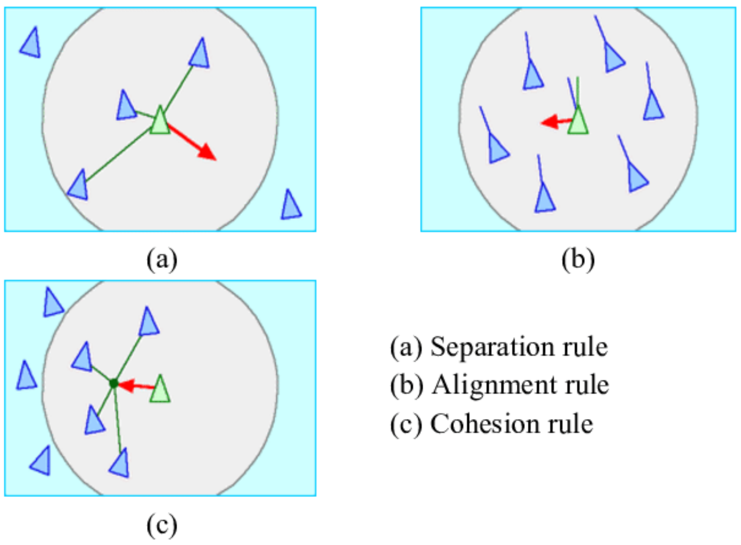


Figure 1.1: Flocking model introduced by Reynolds [Rey87] with force terms (a) separation, (b) alignment, and (c) cohesion (picture taken from [AYS14]).

1.4 State of the Art

Before dealing with the control of multiple agents, we consider the competitive MAS scenario of autonomous racing, where the task is to control a single racing agent in competition with other opponent agents. Then, we look at descriptions of aerial MAS formations and their control methods and strategies. Subsequently, we summarize what observations are used for such control methods, and what information is exchanged between the individual agents for such systems.

1.4.1 Control for Competitive MAS

Autonomous racing is a quintessential example of MAS in a competitive scenario. For the objective of completing laps without collision and reaching the target as fast as possible, individual agents behave antagonistically toward other participating racing agents. The rising popularity of self-driving cars has led to rapid growth in this research field in recent years [Bet22]. Autonomous racing is also highly relevant as it is related to automated driving in urban and general environments and, therefore, pushing research that can eventually be transferred to these application areas.

Traditional control divides this problem into independent sub-problems: perception, planning, and control. Recent success in the context of Formula Student [Kab19b; Kab19a; And20] has been achieved by Model Predictive Control (MPC) and by engineering the perception pipeline using sensor fusion of LiDAR and RGB cameras. Several approaches plan an optimal trajectory [VT05; RNH15; TC13; Váz20].

A large amount of research dealing with Reinforcement Learning (RL) in the context of autonomous racing made use of camera images and adopted model-free methods [Jar18; RMD07; Ken19]. In [Fuc20], the authors used Soft Actor-Critic (SAC) to control a racing car in simulation by feeding the controller with state information and ad-hoc features about the road curvature. Another recurrent trend for autonomous racing consists of combining MPC and deep RL [BB20b; Wil17]. In most of these works, the agent is assumed to be an Markov Decision Process (MDP), that is, to have a fully observable state. In contrast, in this thesis we assume that the state is only partially observable, that is, the model of the agent is a POMDP.

1.4.2 Formation Control of Aerial MAS

All collaborative MAS applications require some *control method* to establish a certain formation and maintain it throughout the operation of the system. In order to work with MAS formations, it is first necessary to *describe* such formations and their specific properties for the respective application area. This description of how the formation of agents should look is not necessarily linked to a specific control method. However most papers describe a formation and also propose an associated control method to achieve a MAS formation based on their given description. Therefore, also in this section, we consider *description* and *control method* together.

Reynolds [Rey87] was the first to propose a flocking model, using cohesion, separation, and velocity alignment force terms (see Figure 1.1) to compute agent accelerations. Even though it originally described flocking for computer animation, its principles are directly applicable to actual robotic systems. Reynolds model was extensively studied [ER10] and adapted for different application areas [Chu18a]. Alternative flocking models, based on cost functions, are considered in [Olf06; Meh18; Mar14; SSF19; BLM08]. In [TJP03], a formulation based on a cost function that is defined in terms of scalar distances to other agents only, is introduced. Gradient optimization for robot control has been studied in [Sch09]. Other formulations consider swarm control in the context of formation rigidity [SG17; Poz21; Zel15].

In most of these approaches, flocks are described using point models. This means that physical properties of agents (e.g., quadcopters) such as mass and inertia, are not taken into account for evaluation. In addition to these largely theoretical approaches, in [Vás18; SSF21; SSF22], flocking controllers are implemented and tested on real hardware. In contrast to these works, we use a detailed drone model for our simulations and furthermore perform experiments with actual hardware drones in this thesis. The approach of [SSF21; SSF22] involves the use of model-predictive control, which is known to be computationally rather expensive and requires a physical model of the agents. We, in contrast, propose control methods that obviate the need for such computationally expensive calculations.

1.4.3 Observations and Communication of MAS

As introduced, MAS might include the perception of other agents and some method of localization in order to determine each agent's position. MAS coordination is more challenging if observations are limited, and therefore perception and/or localization is not available or limited. This means the whole system state is only partially observable by the individual agents, which is also reflected in the description of agents as POMDPs.

In [Lu23; Sas17; Moh18; Whe20], agents are equipped with cameras and/or LiDAR sensors that are used for (relative) localization. Other works [OA14; Ara16; KPA17; LXW18; WWP10; BS20], including the survey [OPA15], assume individual agents can sense relative positions of their neighboring agents with respect to their own local coordinate systems. One example of a system to measure relative displacement vectors among a group of quadcopters, called UVDAR, is using cameras and ultraviolet led markers [Wal19; Wal18; WSF18]. This system was demonstrated indoors as well as outdoors for a variable number of quadcopters. However, sensors to determine this information are not necessarily available in every scenario. In this thesis, we therefore deal with agents as POMDPs, where the state is only partially observable.

If agents are able to interchange messages, they can communicate their distance measurements, acceleration, and possibly further information. In [AY11; ST08; ZWG19; GLX20; JAH20], message exchange (transmitting acceleration, angular velocity, or other data) is used between at least some of the agents. As there might exist scenarios where such information exchange is at least temporarily not available, we consider the case of very limited, respectively non-existent, information exchange in this thesis.

1.5 Methodology

The research conducted in this thesis is in the area of robotics and Cyber-Physical Systems (CPS). For CPS, physical components and software are deeply interconnected, as they interact in both directions. These interactions of CPS and the physical environment do not only depend on the value correctness of calculations but also on their timely behavior. This aspect is especially relevant when dealing with real-time constraints of certain systems. Despite this timing aspect, which is mostly neglected in pure software-based applications, also imperfections of models are of special interest in this research area. To describe the interactions of a CPS with its environment, we can create a model describing a subset of the real world, which is relevant to the specific scenario. However, every model needs to make abstractions or simplifications to be practically tractable. Even if the model is very detailed, there might be aspects of the real world that are hard to measure exactly and, therefore, cannot be modeled perfectly. This leads to an inherent imperfection of the model. When such a model is used to simulate real-world scenarios, this leads to a so-called sim2real transfer gap. To address these issues, we use a three-pronged approach consisting of theoretical analysis and reasoning, implementation and simulation, and hardware experiments.

- **Theoretical Analysis and Reasoning:**
  The basis for our research is a thorough literature study in order to analyze the current state-of-the-art of control methods, relevant measurement and sensor technology, means of communication, and physical properties of the robotic agents. Following this, we identify which aspects of the current state-of-the-art can be improved or are yet incomplete or missing. Based on this, we formulate research questions, as introduced in Section 1.6. For these questions, we first give a mathematical formulation of the problem and try to model the scenario consisting of the MAS agents and their environment. We seek to develop a new approach addressing the stated research question and describe it in a high-level language or as a purely mathematical formulation using our model. This part can be considered as employing a *rationalist* [Ede07] approach since we use mathematical and logical reasoning to deduct attributes of our control approach under the given properties of our robot model and environment model. To assess the performance of a control method, we define relevant quality metrics for the respective scenario. These are, e.g., lap time and collision-safety distance for autonomous racing (refer to Chapter 5) or compactness, inter-agent distance, and obstacle clearance for drone flocking (refer to Chapters 2 to 4). In the stage of theoretical analysis, we are not necessarily able to give guarantees for our control methods w.r.t to these quality metrics.

- **Implementation and Simulation:**
The new approach that we first developed and described in a high-level language or as a purely mathematical formulation is thereafter implemented as a software program. This software can then be used in an appropriate simulation environment, which emulates the physical model of our robotic agents and their environment. In this part of our research, we conduct extensive experiments in this simulation environment. This part, as well as the following hardware experiments, can be considered as employing a *technocratic* [Ede07] approach, as we gather a posteriori knowledge by these executions of simulations. On the one hand, we intuitively can asses if the implementation actually controls agents in the expected way. On the other hand, we use the aforementioned quality metrics to quantitatively assess the controller's performance. If thresholds for certain quality metrics are violated, we consider the controller as not functional, and we need to revise its design and/or implementation. This could imply that another iteration of theoretical analysis is needed in order to come up with a modified control formulation. In other cases, it might be sufficient to fix potential implementation mistakes or to adjust some parameters of the controller. Using the simulation environment, we are also able to quantitatively evaluate the implementation over a range of different parameters, to search for its most optimal values w.r.t. to certain quality metrics. We furthermore use simulations to compare our developed controllers to other previously published controller designs and based on quality metrics, we quantitatively compare them.

- **Hardware Experiments:**
As the aforementioned sim2real transfer gap is a critical concern for any type of robotic system, we perform experiments with real hardware robots. For autonomous racing, we use a model race car and build several different race tracks to drive in our lecture hall at the university (refer to Chapter 5). For drone flocking, we use up to seventeen *Crazyflie 2.1* quadcopters to perform experiments at various indoor locations that are big enough, including a fire department's vehicle hall (refer to Chapters 2 to 4). In these experiments, the controller's performance is assessed by the very same quality metrics as in the simulation. These hardware experiments follow the principle of *empirical falsification* [Pop59]. Repeated and structured experiments are performed for a set of pre-defined task-specific scenarios and a variation of configurations (e.g., number of agents). While these experiments cannot formally prove any properties of the controllers, we can demonstrate their fitness to apply them to real hardware robotic systems and actual physical environments.

1.6 Contributions of this Thesis

The contributions of this dissertation to advance the state of the art in research on MAS are as follows: At first, we consider the competitive MAS scenario of autonomous racing, as introduced in Section 1.4.1, where we present a RL approach to control a single POMDP agent, competing for the fastest lap time. Given the different MAS formations and control methods introduced in Section 1.4.2 and available types of observations and communication in Section 1.4.3, we seek ways to improve on these. We present a control method for a swarm of drones that only needs positional observations of neighboring drones and does not require the physical model of the agents. We furthermore present another control method that even does not need positional observations but only scalar distance measurements to neighboring drones.

1.6.1 Reinforcement Learning for Autonomous-Racing POMDPs

For autonomous racing there exist a large number of engineered solutions based on traditional control. These solve the independent sub-problems of perception and/or localization, planning, and control. Monte Carlo localization [Fox99] is an efficient method to determine the position of a robot on a map based on sensor (e.g. LiDAR) measurements. Optimal trajectories can be pre-planned, w.r.t to different metrics (e.g. minimum lap time, distance, curvature) [VT05; RNH15; TC13; Váz20]. Subsequently it remains to employ a control method to follow this trajectory. The *Pure Pursuit Path Tracking Algorithm* [Cou92] is an example of such a method, which is widely used in F1tenth racing [OKe20; Agn20].

However, deploying deep RL agents in the real world is difficult. This is because they require running a significantly large amount of episodes to obtain reasonable performance [SB18]. This performance is only tractable in simulation environments. Subsequently, the agents should also overcome the challenges of transferring learned dynamics from simulation to the real world. We set out to design deep RL models that are able to learn to autonomously complete time-lap racing tasks. Considering the real-world conditions, we formalize the problem as a POMDP. This is necessary, as the state is not directly observable by the available sensor data and therefore, we need to infer a policy that gives a mapping from the history of observations (respectively belief states) to actions.

In our publication [Bru22] (see Chapter 5), we address the following research question:

RQ 1:
How to design deep reinforcement learning models that are able to learn to autonomously complete time-lap racing tasks?

1.6.2 MAS Control Without a Plant-Dynamics Model

On the one hand, many control methods for aerial MAS (see Section 1.4.2) describe the agents as point models. These works assume agents to be single points with a specific mass and describe the behavior as a double integrator when forces are acting on this point. Some even simpler models do not consider mass and inertia but only model it as single integrator dynamics based on the agent's velocities and positions. These models clearly neglect the detailed physical behavior of actual aerial robots, such as quadcopters and fixed-wing aircraft. When control methods based on a point model are described in theory only, it is unclear whether and to what extent they are applicable to real hardware.

On the other hand, there are controllers, such as MPC, which use detailed models of the agents. However, MPC is computationally expensive and, therefore, not applicable for all robots, as embedded systems typically have limited onboard computation capabilities. This can be either due to hardware restrictions, but also for energy efficiency reasons.

In our work, we, therefore, investigate the gap between these two approaches. Given an actual robotic system with its associated physical behavior, we seek to design an optimized controller without internally using the physical model of the agents.

In our publication [Bra23] (see Chapter 2), we address the following research question:

RQ 2:
How to design a distributed controller that minimizes a given positional cost function for robotic agents with black-box positional low-level-controllers, no available model of the plant dynamics, and only positional observations of their environment?

1.6.3 MAS Control Under Limited Observations

Most control methods, introduced in Section 1.4.2, require knowledge of the position of all agents, or at least of neighboring agents. Where the neighborhood of an agent can, for example, be defined as a subset of all the drones within a certain distance. Some control methods also require other drone's velocities or accelerations. This has two implications w.r.t observations and communication (also refer to Section 1.4.3).

First, agents need to determine their own position, velocity, and acceleration in a global coordinate system. GNSS and comparable types of indoor localization systems [Fer17; Lao18], can determine such position vectors. However, these systems might not be installed or are currently unavailable at some locations. Moreover, they are unlikely to be available in some applications (e.g., underwater SAR and cave exploration). If a positioning system is not available, it remains to use onboard sensors to determine relative position vectors from one agent to another. Such methods were also introduced in Section 1.4.3.

Second, agents might communicate this information either point-to-point, via multi- or broadcast messaging. It enables individual agents to control their actions based on local measurements as well as information from other agents. Such exchange of information needs to be done in a coordinated manner (e.g., to avoid collisions on the radio transmission carrier frequency). It takes time until the message is delivered, and transmissions might get lost. There also might exist scenarios where such information exchange is at least temporarily not available.

In our work, we therefore seek to develop a MAS formation controller with only limited observations (scalar distance measurements) and limited (or without) information exchange. This was partially discussed in our publication [Bra22c] (see Chapter 3) and subsequently more extensively addressed in our paper [Bra24] (see Chapter 4), to answer the following research question:

RQ 3:
How can the controller design for distance-based cost functions be generalized such that control actions are chosen based only on scalar distances without knowledge of relative position vectors?

1.7 Publications

The remainder of this cumulative dissertation consists of the following published papers:

- **Chapter 2:**
  Publication: Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Multi-Agent Spatial Predictive Control with Application to Drone Flocking“. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 1221–1227. DOI: `10.1109/ICRA48891.2023.10160617`

  Conference ranking: A* / A1 [1]

  Abstract: We introduce *Spatial Predictive Control* (SPC), a technique for solving the following problem: given a collection of robotic agents with black-box Positional Low-Level Controllers (PLLCs) and a mission-specific distributed cost function, how can a distributed controller achieve and maintain cost-function minimization without a plant model and only positional observations of the environment? Our fully distributed SPC controller is based strictly on the position of the agent itself and on those of its neighboring agents. This information is used in every time step to compute the gradient of the cost function and to perform a spatial look-ahead to predict the best next target position for the PLLC. Using a simulation environment, we show that SPC outperforms Potential Field Controllers, a related class of controllers, on the drone flocking problem. We also show that SPC works on real hardware, and is therefore able to cope with the potential sim-to-real transfer gap. We demonstrate its performance using as many as 16 Crazyflie 2.1 drones in a number of scenarios, including obstacle avoidance.

  Author's contribution: Survey of the state of the art and formulation of research question. Theoretical and conceptual design of the proposed method. Mathematical formulation of Spatial Predictive Control (SPC) and analysis of cost function terms' derivative. Conceptual design and implementation of simulation environment and scenario definitions. Software implementation of control method and drone model in the simulation environment. Analysis of the controller's behavior and properties in simulation. Refinement of controller design and parameters and repeated improvements in simulation. Conceptual design and implementation of hardware setup for experimental evaluation on real hardware quadcopters. Portation and adaption of the controller's software implementation to be used on the hardware quadcopters. Execution and evaluation of hardware experiments. Presentation of current status at regular meetings with other co-authors to discuss the progress of the work. Preparing the graphics and presenting the work in written form.

[1] IEEE International Conference on Robotics and Automation (ICRA) is ranked A* according to CORE2023 [Edu23] and A1 according to Qualis [Edu12].

- **Chapter 3:**

  Publication: Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Towards Drone Flocking Using Relative Distance Measurements“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning.* Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2022, pp. 97–109. ISBN: 978-3-031-19759-8. DOI: `10.1007/978-3-031-19759-8_7`

  Conference ranking: C / B4 [2]

  Abstract: We introduce a method to form and maintain a flock of drones only based on relative distance measurements. This means our approach is able to work in GPS-denied environments. It is fully distributed and therefore does not need any information exchange between the individual drones. Relative distance measurements to other drones and information about its own relative movement are used to estimate the current state of the environment. This makes it possible to perform lookahead and estimate the next state for any potential next movement. A distributed cost function is then used to determine the best next action in every time step. Using a high-fidelity simulation environment, we show that our approach is able to form and maintain a flock for a set of drones.

  Author's contribution: Analysis of potential direction of future work with regards to the previously published work and survey of the state of the art, and thereafter formulation of research question. Theoretical and conceptual design of the proposed method for controlling a MAS without position information. Sketch and implementation of first proof-of-concept studies for the proposed control methods. Conceptual design and implementation of simulation environment and scenario definitions. Software implementation of control method in the simulation environment using the drone model from previous works. Analysis and evaluation of the controller's behavior and properties in simulation. Presentation of current status at regular meetings with other co-authors to discuss the progress of the work. Preparing the graphics and presenting the work in written form.

[2] International Symposium on Leveraging Applications of Formal Methods Verification and Validation (ISoLA) is ranked C according to CORE2023 [Edu23] and B4 according to Qualis [Edu12].

- **Chapter 4:**

  Publication: Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements“. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 12193–12200. DOI: `10.1109/ICRA57147.2024.10610147`

  Conference ranking: A* / A1 [3]

  Abstract: We present *distributed distance-based control* (DDC), a novel approach for controlling a multi-agent system, such that it achieves a desired formation, in a resource-constrained setting. Our controller is fully distributed and only requires local state-estimation and scalar measurements of inter-agent distances. It does not require an external localization system or inter-agent exchange of state information. Our approach uses spatial-predictive control (SPC), to optimize a cost function given strictly in terms of inter-agent distances and the distance to the target location. In DDC, each agent continuously learns and updates a very abstract model of the actual system, in the form of a dictionary of three independent key-value pairs $(\Delta\vec{s}, \Delta\boldsymbol{d})$, where $\Delta\boldsymbol{d}$ is the partial derivative of the distance measurements along a spatial direction $\Delta\vec{s}$. This is sufficient for an agent to choose the best next action. We validate our approach by using DDC to control a collection of Crazyflie drones to achieve formation flight and reach a target while maintaining flock formation.

  Author's contribution: Analysis of directions for future work in our previously published paper. Theoretical and conceptual design of the identified improvements, which lead to the development of the DDC approach. Software implementation of DDC in simulation environment employing the previously used drone model. Analysis and evaluation of the controller's behavior and properties in simulation. Refinement of controller design and parameters and repeated improvements in simulation. Conceptual design and implementation of hardware setup for experimental evaluation on real hardware quadcopters. Portation and adaption of the controller's software implementation to be used on the hardware quadcopters. Execution and evaluation of extensive hardware experiments. Presentation of current status at regular meetings with other co-authors to discuss the progress of the work. Preparing the graphics and presenting the work in written form.

[3] IEEE International Conference on Robotics and Automation (ICRA) is ranked A* according to CORE2023 [Edu23] and A1 according to Qualis [Edu12].

- **Chapter 5:**

  Publication: Axel Brunnbauer*, Luigi Berducci*, Andreas Brandstätter*, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. „Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing“. In: *2022 International Conference on Robotics and Automation (ICRA)*. (* indicates equal contribution). 2022, pp. 7513–7520. DOI: `10.1109/ICRA46639.2022.9811650`

  Conference ranking: A* / A1 [4]

  Abstract: World models learn behaviors in a latent imagination space to enhance the sample-efficiency of deep RL algorithms. While learning world models for high-dimensional observations (e.g., pixel inputs) has become practicable on standard RL benchmarks and some games, their effectiveness in real-world robotics applications has not been explored. In this paper, we investigate how such agents generalize to real-world autonomous vehicle control tasks, where advanced model-free deep RL algorithms fail. In particular, we set up a series of time-lap tasks for an F1TENTH racing robot, equipped with a high-dimensional LiDAR sensor, on a set of test tracks with a gradual increase in their complexity. In this continuous-control setting, we show that model-based agents capable of learning in imagination substantially outperform model-free agents with respect to performance, sample efficiency, successful task completion, and generalization. Moreover, we show that the generalization ability of model-based agents strongly depends on the choice of their *observation model*. We provide extensive empirical evidence for the effectiveness of world models provided with long enough memory horizons in sim2real tasks.

  Author's contribution: The main focus is on the implementation, demonstration, evaluation, and analysis of sim2real transfer capabilities by experimentation using a hardware model race car. Comparing the simulated properties with the demonstrated abilities of the agent in a number of different real-world experiments. Responsible for the hardware setup, including building and operating the model race car hardware platform, consisting of chassis, sensors, actuators, and embedded computing platform. All software on the model race car, which is required to operate the car and to execute the same inference node, as in simulation: Linux operating system, drivers, and the Robot Operating System (ROS) stack. Porting and adapting the implemented control agent to run properly on the model race car hardware platform. Responsible for the track setup, including all race track infrastructure with track barriers, network, and camera setup, to conduct the hardware experiments. Conducting and overseeing the hardware experiments in the lecture hall, recording and evaluating the associated results to demonstrate the sim2real capability of the proposed RL algorithm. Iterative improvements on the overall system design and parameters based on hardware experiment results. Present and discuss the findings and conclusions in written and graphical form.

[4] IEEE International Conference on Robotics and Automation (ICRA) is ranked A* according to CORE2023 [Edu23] and A1 according to Qualis [Edu12].

CHAPTER 2

Multi-Agent Spatial Predictive Control with Application to Drone Flocking

This work was published as:
Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Multi-Agent Spatial Predictive Control with Application to Drone Flocking“. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 1221–1227. DOI: 10.1109/ICRA48891.2023.10160617

By the following authors with their affiliations:
Andreas Brandstätter at CPS, Technische Universität Wien (TU Wien), Austria
Scott A. Smolka at Department of Computer Science, Stony Brook University, USA
Scott D. Stoller at Department of Computer Science, Stony Brook University, USA
Ashish Tiwari at Microsoft, USA
Radu Grosu at CPS, Technische Universität Wien (TU Wien), Austria

A pre-print of this paper is available as:
Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. *Multi-Agent Spatial Predictive Control with Application to Drone Flocking (Extended Version)*. 2022. DOI: 10.48550/ARXIV.2203.16960

Abstract

We introduce *Spatial Predictive Control* (SPC), a technique for solving the following problem: given a collection of robotic agents with black-box Positional Low-Level Controllers (PLLCs) and a mission-specific distributed cost function, how can a distributed controller achieve and maintain cost-function minimization without a plant model and only positional observations of the environment? Our fully distributed SPC controller is based strictly on the position of the agent itself and on those of its neighboring agents. This information is used in every time step to compute the gradient of the cost function and to perform a spatial look-ahead to predict the best next target position for the PLLC. Using a simulation environment, we show that SPC outperforms Potential Field Controllers, a related class of controllers, on the drone flocking problem. We also show that SPC works on real hardware, and is therefore able to cope with the potential sim-to-real transfer gap. We demonstrate its performance using as many as 16 Crazyflie 2.1 drones in a number of scenarios, including obstacle avoidance.

2.1 Introduction

A collection of drones can perform tasks that cannot be accomplished by individual drones alone [Chu18b]. It can, for example, carry a heavy load while still being much more agile than a single larger drone [MFK11; LK18]. In search-and-rescue applications, the drones can explore unknown terrain by covering individual paths that jointly cover the entire area [Câm14; BS18; Mic14]. These collective maneuvers can be expressed as the problem of minimizing a *positional cost function*, i.e., a cost function that depends on the positions of the drones (and possibly information about their environment). Such a problem formulation requires a method to localize each drone within a common reference frame, e.g. a Global-Navigation Satellite System (GNSS) or an indoor localization system.

Off-the-shelf drones, such as Crazyflie [Gie17], DJI [SZ 21], and Parrot [Par21], come equipped with a *Positional Low-Level Controller (PLLC)*. Such a controller takes a position argument as input and maneuvers the drone to this position, where it then hovers. PLLCs are common in other types of robotic systems, including the Landshark [Bla21] and Taurob [Tau21] unmanned ground vehicles, and the Bluefin®-12 [Gen21] unmanned underwater vehicle. Unfortunately, the PLLC's code is often proprietary, and the exact parameters of the physical drone model might not be available. Since the PLLC and physical drone together form the plant to be controlled, a dynamic model of the plant is often unavailable, for one or both of these reasons.

In this paper, we address the following problem: *Design a distributed controller that minimizes a given positional cost function for robotic agents with black-box PLLCs, no available model of the plant dynamics, and only positional observations of their environment.*

To solve this problem, we introduce *Spatial Predictive Control* (SPC), a novel distributed high-level approach to multi-agent control. In SPC, each agent's controller identifies

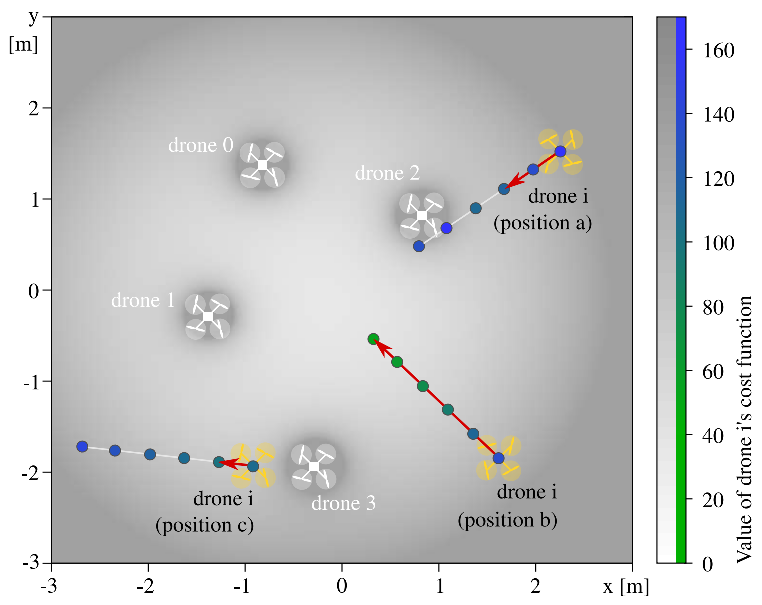


Figure 2.1: Given the positions shown for drones 0 to 3, the greyscale heatmap indicates the value of the cost function for a drone i at each point, if it was placed at that point. The direction in which drone i should move is determined by the gradient of the cost function (shown for positions a, b, and c). SPC evaluates the cost at the spatial lookahead (indicated by the colored dots) along this direction and chooses the best value for drone i's next position (the red arrow).

N equally-spaced points within a maximum look-ahead distance $\epsilon \cdot N$ from its current position in the direction of the negative gradient of a cost function c. The SPC controller then computes the value of c for each of these points and chooses the one with minimal cost as the target location to be sent to the PLLC. The PLLC makes a best effort to reach this location, while in the next time-step, SPC provides an updated target.

SPC vs MPC. To solve the stated problem, one might consider the PLLC as part of the plant and design a Model Predictive Control (MPC) for the high-level control. Such an approach is not applicable since neither a dynamic model of the physical plant nor the internals of the PLLC are available. Even if an approximate dynamic model could be obtained using system identification techniques, and if the code for the PLLC is available (e.g., for Crazyflies), MPC remains a computationally expensive method [NM14]. This is especially relevant for embedded processors with limited computing capabilities. MPC needs to calculate the predicted behavior for the plant model over a specified time horizon in order to search for an optimal control input. In contrast, SPC does not require a plant model and avoids extensive prediction calculations.

SPC vs Planning. Given that (robotic) agents are equipped with PLLCs, one might ask if a controller is actually needed, or would a planning-based approach suffice. Based on the initial positions of agents and obstacles, a plan of way-points could be generated for the PLLC to follow. Since, however, the environment is constantly changing due to the movement of other agents (and possibly obstacles), such a plan would become quickly outdated. This is exactly the type of problem we address with SPC: in every time step, we use the observations of the environment to calculate the next input to the PLLC; the PLLC makes a best effort to reach this position. Hence, the key difference between SPC and planning is the granularity of the time horizon: planning uses long time horizons for calculating trajectories, whereas in our approach, feedback from the plant in every time step is used to recalculate the desired next position.

SPC vs PFC. Potential Field Controllers (PFCs) are well-known controllers for mobile robots. Prior work [TJP03; Sch09] has considered their application to flocking. PFCs view the cost function as defining a potential field, and thus, PFCs use the gradient of the potential as the force (or acceleration) the controller needs to apply. There are two issues with using PFCs for our stated problem. First, when the environment has obstacles and a large number of moving drones, the cost function becomes time-varying and nonlinear (as in Figure 2.1), and *local* gradients become misleading. Second, in our setting, we can not set the acceleration directly because we only have access to the PLLC. Nevertheless, we can adapt and use PFC in our setting, but our experiments confirm that it performs poorly compared to SPC, which evaluates the cost function at *multiple* candidate future positions within the spatial look-ahead horizon to find the best next position (in terms of cost-function minimization).

Application to Drone Flocking: After introducing SPC as a general approach, we apply it to a distributed multi-agent system with the goal of achieving flock formation, and maintaining flock formation while moving to a specified target location, avoiding obstacles in the process.

The local cost function c for this problem (see Section 2.3.2) depends only on the locations of the drones. As illustrated for four drones in Figure 2.1, the negation of the gradient of c (see Section 2.3.3) suggests a direction of movement for drone i. The SPC algorithm (see Section 2.2) evaluates its local cost function at multiple points in that direction. As the figure illustrates, this enables the drone to see peaks and valleys in the cost function that may lie ahead.

The **main contributions of this paper** are:

- We introduce the novel concept of *Spatial Predictive Control*, a control methodology well suited for PLLCs.
- SPC is *model-free*. One only needs to be able to determine the cost at different locations along the direction of the cost function's gradient in order to apply it. Also, SPC does not need to measure the velocity or acceleration of neighboring drones.
- We evaluate SPC using drone flocking in a drone simulation environment.

- We further experimentally validate our approach by achieving flocking with real drone hardware in the form of off-the-shelf Crazyflie quadcopters in a number of different scenarios.

2.2 SPC for multi-agent systems

We describe the distributed control problem addressed in this paper and then present SPC for solving this problem.

2.2.1 Distributed Control for Distributed Cost Minimization in the Presence of PLLCs

We consider a multi-agent system consisting of a set D of agents. Every agent $i \in D$ has a state (x_{io}, x_{ih}), where x_{io} is the observable part of its state and x_{ih} is the hidden part of its state. Agent i has a control input u_i and its dynamics is assumed to be given by some unknown function f:

$$(\frac{dx_{io}(t)}{dt}, \frac{dx_{ih}(t)}{dt}) = f(t, x_{io}(t), x_{ih}(t), u_i(t)) \tag{2.1}$$

Agent i has access to the observable state of a subset $H_i \subseteq D$ of agents. H_i will be referred to as the *neighborhood* of i.

The objective for the multi-agent system is given in terms of a cost function $c(x_{io}, x_{Hio})$ that maps the observable state of agent i (x_{io}) and of its neighbors (x_{Hio}) to a non-negative real value. Here we use x_{Hio} as shorthand for $(x_{jo})_{j \in H_i}$. Agent i's goal is to minimize $c(x_{io}, x_{Hio})$.

In our setting, we do not have ability to directly set u_i. Instead, we can only set a *reference value* $x_{io}^{(r)}$ that is then used by some black-box, low-level controller $PLLC$ to internally set the control input.

$$u_i(t) = PLLC(t, x_{io}(t), x_{ih}(t), x_{io}^{(r)}) \tag{2.2}$$

Both the dynamics of each agent (function f) and the details of the PLLC (function $PLLC$) are unknown. The cost function c is given. We want to find a procedure that allows each agent to minimize its cost in the above setting.

2.2.2 Spatial Predictive Control (SPC)

Let $\nabla_{x_{io}} c(x_{io}, x_{Hio})$ denote the gradient of cost function c with respect to x_{io}. One way to minimize the cost $c(x_{io}, x_{Hio})$ would be to follow the negative of the gradient at every point. However, if the cost function is nonlinear (e.g., has many peaks), then gradients can be misleading. The key observation underlying SPC is that each agent should *look ahead in the observable state space*, in the direction of the negative gradient, to determine the reference value for its observable state. An SPC controller picks N equally-spaced points within a maximum look-ahead distance $\epsilon \cdot N$ from the current observable state and in the direction of the negative gradient, where ϵ and N are parameters of the controller. At any given state $(x_{io}(t), x_{Hio}(t))$, the set Q_i containing these equally-spaced points is given by:

$$Q_i = \left\{ x_{io}(t) - n \cdot \epsilon \cdot \frac{\nabla_{x_{io}} c(x_{io}(t), x_{Hio}(t))}{\|\nabla_{x_{io}} c(x_{io}(t), x_{Hio}(t))\|} \mid n = 0 \,..\, N \right\} \tag{2.3}$$

Here $\nabla_{x_{io}} c(x_{io}(t), x_{Hio}(t))$ denotes the evaluation of the gradient of c at the point $(x_{io}(t), x_{Hio}(t))$. Our spatial-predictive controller selects the point in Q_i with minimum cost as the next target position $x_{io}^{(r)}$ for agent i:

$$x_{io}^{(r)} = \underset{\tilde{x}_{io} \in Q_i}{\operatorname{argmin}} \left(c(\tilde{x}_{io}, x_{Hio}(t)) \right) \tag{2.4}$$

Note that the SPC controller recomputes the reference $x_{io}^{(r)}$ at each time step. This is important because this computation of the reference does not take into account the motion of the neighbors. However, SPC will respond to any change in neighbors' observable states in its next computation of the reference.

2.3 Application to Multi-Agent Flocking Problem

This section starts with background on flocking, describes how to use SPC for this application, and then presents metrics to assess the quality of a flocking controller.

2.3.1 The Flocking Problem

A set of agents D is in a flock formation if the distance between every pair of agents is "not too large and not too small." These requirements yield the first two terms of our cost function: *cohesion* and *separation*. These two terms are sufficient to cause the agents to form and maintain a flock formation.

In our model, drone i has access to positions of only a subset H_i of drones, namely its local neighbors. Hence, we define a local cost function, parameterized by i, which uses only the positions of drones in $\{i\} \cup H_i$.

2.3.2 Cost Function

Consider a drone i, i in D. Let p_j, when it appears in the local cost function of drone i, denote the position of drone j as known to drone i; this may differ from the actual position due to sensing error. Let p_{H_i} denote the tuple of positions of drones in H_i and let r_d be the radius of each drone. We define the cost function $c(p_i, p_{H_i})$ as:

$$\begin{aligned} c(p_i, p_{H_i}) = & c_{coh}(p_i, p_{H_i}) + c_{sep}(p_i, p_{H_i}) + \\ & c_{tar}(p_i, p_{H_i}) + c_{obs}(p_i) \end{aligned} \tag{2.5}$$

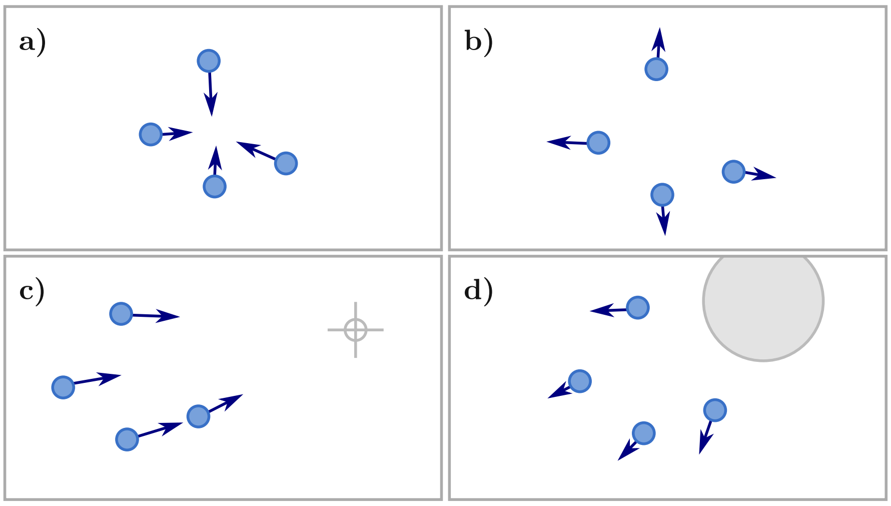


Figure 2.2: Directional movements (indicated by arrows) induced by cost-function terms: **a**: *Cohesion*, **b**: *separation*, **c**: *target seeking*, and **d**: *obstacle avoidance*.

The value of the *cohesion* term increases as drones drift apart, and the *separation* term increases as drones get closer together. Each term has a weight, denoted by a subscripted ω.

Cohesion term:

$$c_{coh}(p_i, p_{H_i}) = \omega_{coh} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} \|p_i - p_j\|^2 \tag{2.6}$$

Separation term:

$$c_{sep}(p_i, p_{H_i}) = \omega_{sep} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} \frac{1}{max(\|p_i - p_j\| - 2r_d, \hat{0})^2} \tag{2.7}$$

The function $max(., \hat{0})$ ensures positive values when there is sensor noise, but does not further influence the cost function; $\hat{0}$ denotes a very small positive value.

The mission-specific *target seeking* term sets a target location, denoted by p_{tar}, for the entire flock. The *obstacle avoidance* term prevents the drones from colliding with infinitely tall cylindrical objects. Let K denote the set of obstacles. For $k \in K$, let r_k denote the radius of obstacle k, and let p_k denote its center on the xy-plane.

Target-seeking term:

$$c_{tar}(p_i, p_{H_i}) = \omega_{tar} \cdot \left\| p_{tar} - \frac{p_i + \sum_{j \in H_i} p_j}{|H_i| + 1} \right\|^2 \tag{2.8}$$

Obstacle-avoidance term:

$$c_{obs}(p_i) = \omega_{obs} \cdot \frac{1}{|K|} \cdot \sum_{k \in K} \frac{1}{max(\|\mathcal{P}(p_i) - p_k\| - r_k - r_d, \hat{0})^2} \tag{2.9}$$

The function $\mathcal{P}(.)$ projects a vector to the xy-plane.

2.3.3 Gradient of the cost function

For SPC, the gradient of the cost function is required in Eq. (2.3), and is given by (for readability, we elide function arguments):

$$\nabla_{p_i} c = \nabla_{p_i} c_{coh} + \nabla_{p_i} c_{sep} + \nabla_{p_i} c_{tar} + \nabla_{p_i} c_{obs} \tag{2.10}$$

Cohesion gradient:

$$\nabla_{p_i} c_{coh} = 2 \cdot \omega_{coh} \cdot \left(p_i - \frac{1}{|H_i|} \cdot \sum_{j \in H_i} p_j \right) \tag{2.11}$$

Separation gradient:

$$\nabla_{p_i} c_{sep} = \frac{2 \cdot \omega_{sep}}{|H_i|} \cdot \sum_{j \in H_i} \frac{p_j - p_i}{(\|p_i - p_j\| - 2r_d)^3 \cdot \|p_i - p_j\|} \tag{2.12}$$

Target-seeking gradient:

$$\nabla_{p_i} c_{tar} = \frac{2 \cdot \omega_{tar}}{|H_i| + 1} \cdot \left(\frac{p_i + \sum_{j \in H_i} p_j}{|H_i| + 1} - p_{tar} \right) \tag{2.13}$$

Obstacle-avoidance gradient:

$$\nabla_{p_i} c_{obs} = \frac{2\omega_{obs}}{|K|} \cdot \sum_{k \in K} \frac{p_k - \mathcal{P}(p_i)}{(\|\mathcal{P}(p_i) - p_k\| - r_k - r_d)^3 \cdot \|\mathcal{P}(p_i) - p_k\|} \tag{2.14}$$

2.3.4 Flock-Formation Quality metrics

Collision avoidance: To avoid collisions, the distance between all pairs of drones must remain above a specified threshold $dist_{thr}$. We define a metric for the minimum distance between any pair of drones as follows:

$$dist_{min} = \min_{i,j \in D; i \neq j} \|p_i - p_j\| \tag{2.15}$$

We set $dist_{thr} = 2 \cdot r_d + r_{safety}$, where r_d is the radius of the drone, and r_{safety} is a safety margin.

Compactness: Compactness of the flock is measured by the maximum distance of any drone from the centroid of the flock. It is defined as follows:

$$comp_{max} = \max_{i \in D} \left\| \frac{\sum_{j \in D} p_j}{|D|} - p_i \right\| \tag{2.16}$$

It is expected to stay below some threshold $comp_{thr}$; otherwise, the drones are too far apart.

Obstacle clearance: Keeping a safe distance from obstacles is required to avoid collisions. We therefore measure the minimum distance from any drone to any obstacle:

$$clear_{obj} = \min_{i \in D; k \in K} \|\mathcal{P}(p_i) - p_k\| \tag{2.17}$$

For safety, this should always be greater than some threshold $clear_{thr} = r_d + r_k + r_{safety}$.

2.4 Experimental Evaluation

We evaluated SPC on the drone flocking problem using simulations and experiments with Crazyflie 2.1 drones.

2.4.1 Simulation Experiments

As a simulation framework, we use *crazys* [SAI18], which is based on the *Gazebo* [KH04] physics and visualization engine and the Robotic Operating System (ROS) [Sta]. Our SPC algorithm is implemented in C++ as a separate ROS node. It receives position messages from neighboring drones, and control messages, such as the target location or a stop command, from the human operator. It outputs a set-point to the PLLC. The SPC we implemented is fully distributed: there is no central optimizer and no further information is exchanged between ROS nodes. The SPC node calculates the gradient according to Eqs. (2.11)-(2.14). The spatial look-ahead parameter N is determined dynamically based on the distance to the target location, where $N^* \in \mathbb{N}^+$ is a system parameter:

$$N = \lceil N^* \cdot max(1, min(1.5 \cdot (\|p_i - p_{tar}\| + 0.5), 3)) \rceil \tag{2.18}$$

This allows the drones to more quickly reach distant target locations and reduces the controller's computational cost (by reducing $|Q_i|$) once the flock reaches the target. Thereafter, the set-point position $x_{io}^{(r)}$ is determined by Eqs. (2.3)-(2.4). Auxiliary functions, like hovering at the starting position, are also implemented in this node. In the simulations, we added Gaussian sensor noise, with $\sigma = 10cm$, for drone position measurements. Cost-function weights and controller parameters (Table 2.1) were determined empirically by analysis of the controller behavior. Note that the maximum look-ahead distance $\epsilon \cdot N$ should not be too large, to avoid "seeing through" other drones or obstacles. On the other hand, a small value for N reduces the granularity of the controller action space, leading to a bang–bang controller if $N = 1$.

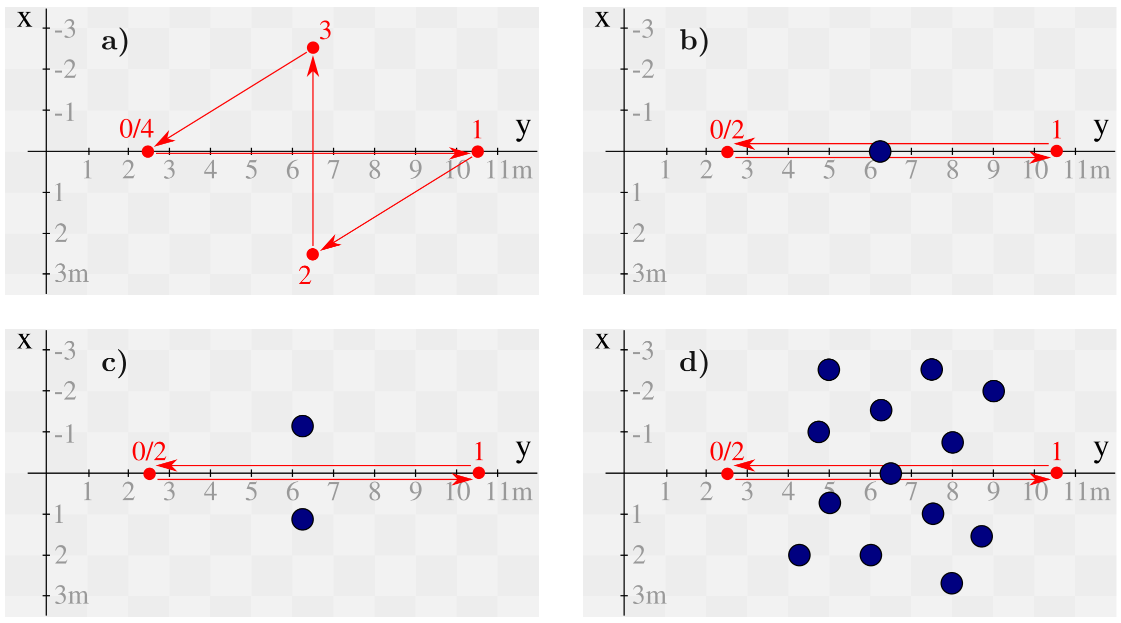


Figure 2.3: Experiments were performed using four different scenarios: **a**: without obstacles, **b**: with one obstacle, **c**: with 2 obstacles, and **d**: with 13 obstacles indicated in dark-blue. The direct path between the numbered target locations p_{tar} (red dots) is indicated with red arrows. (z-dimension is elided in these plots, since it is constant for all target locations.)

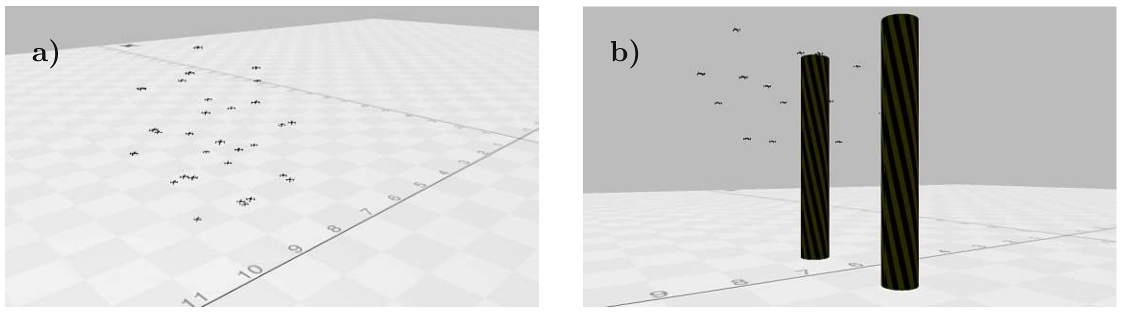


Figure 2.4: Simulation experiments using simulation environment. Snapshot of **a**: flock of 30 drones; and **b**: flock of 15 drones with 2 obstacles.

		PLLC A	PLLC B	Hardware
Cost weights	ω_{coh}	$20\,m^{-1}$	$20\,m^{-1}$	$20\,m^{-1}$
	ω_{sep}	$12\,m^{-1}$	$12\,m^{-1}$	$12\,m^{-1}$
	ω_{tar}	$150\,m^{-1}$	$150\,m^{-1}$	$150\,m^{-1}$
	ω_{obs}	$18\,m^{-1}$	$18\,m^{-1}$	$35\,m^{-1}$
SPC parameters	N^*	6	3	3
	ϵ	$0.05\,m$	$0.05\,m$	$0.04\,m$
PFC gain	k	$0.003\,m$	$0.0015\,m$	n.a.
Dimensions	r_d	$0.07\,m$	$0.07\,m$	$0.07\,m$
	r_k	$0.25\,m$	$0.25\,m$	$0.25\,m$
	r_{safety}	$0.06\,m$	$0.06\,m$	$0.06\,m$

Table 2.1: Parameters used in simulation experiments and hardware experiments.

To evaluate SPC and its implementation, we defined four path-based scenarios (trajectories), as shown in Figure 2.3. The end points on the path (shown in red) are provided in a timed sequence as target location p_{tar}. There are four scenarios: without obstacles (Figure 2.3a), with 1 obstacle (Figure 2.3b), with 2 obstacles (Figure 2.3c), and with 13 obstacles (Figure 2.3d). Simulations were conducted with flocks of size $|D| = 4$, 9, 15, and 30. Using radius $r_H = 0.9m$, the neighborhood is defined by:

$$H_i = \{j \in D \setminus \{i\} \wedge \|p_i - p_j\| < r_H\} \tag{2.19}$$

To check SPC's robustness to different PLLCs, we experimented with two PLLCs with different step responses. PLLC B reaches its set-point for x- and y-dimensions in less than half the time of PLLC A, while overshooting by about 50 % more. The PLLCs behave very similarly in the z-dimension. Figure 2.4 show snapshots of the simulations. A video is provided in the Supplementary Material and available at: https://youtu.be/iUkaYrnZz9k.

2.4.1.1 Results

The analysis of the quality metrics for *collision avoidance*, *compactness*, and *obstacle clearance* show that our SPC-based approach successfully maintains a stable flock. In Figure 2.5, metrics are plotted over time for three representative simulations. Data from the prefix of an execution, when the drones move from random starting positions into flock formation, are omitted when computing the metrics.

2.4.1.2 Computational complexity

The computation time of Eq. (2.4) is $O(|Q| \cdot (|H_i| + |K|))$. $|H_i|$ is bounded by $|D|$ and also depends on r_H. Introducing a concept of neighborhood for obstacles can reduce the computation time.

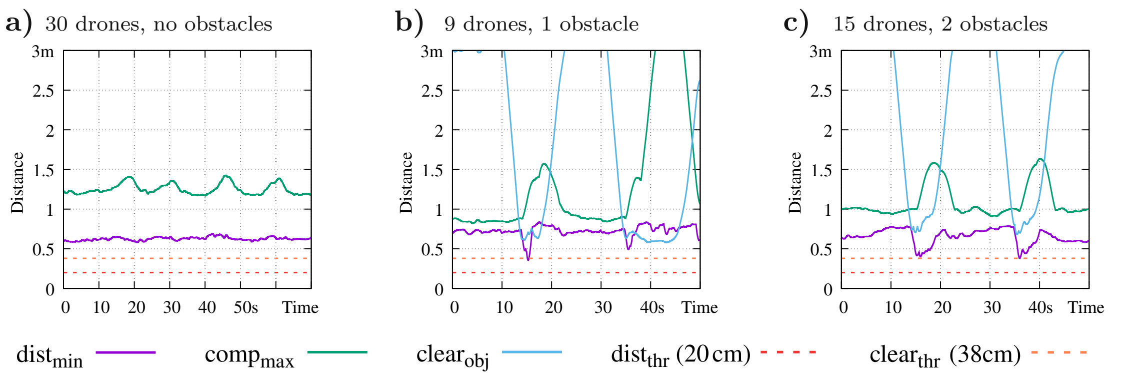


Figure 2.5: Quality metrics over time for SPC simulations using PLLC B for exemplary scenarios of **a**: 30 drones with 0 obstacles, **b**: 9 drones with 1 obstacle, and **c**: 15 drones with 2 obstacles. Results for other simulation experiments were very similar. While the flock is passing the obstacle(s) the metrics temporarily degrade, however values $dist_{min}$ and $clear_{obj}$ stay above the respective thresholds, meaning there are no collisions, throughout the whole simulation. Analogously $comp_{max}$ stays below the threshold (5m), indicating that a compact flock is continuously maintained.

2.4.1.3 Comparison with PFC

To compare SPC with [Sch09; TJP03], we also experimented with a PFC controller based solely on gradients. In this controller, the gradient vector $\nabla_{p_i} c$ is used to determine the next set-point $x_{io}^{(r)}$ for the PLLC as follows:

$$x_{io}^{(r)} = p_i - k \cdot \nabla_{p_i} c(p_i) \tag{2.20}$$

The control law stated in [TJP03] provides an acceleration vector, which we adapted in Eq. (2.20) to a positional variant as required by the PLLC. We determined the gain k empirically such that the target of the flock was reached within the same time as our SPC implementation. The gain determines how aggressively the controller moves the drone toward the target location. In the experiments detailed below, the gain is constant, as in [Sch09; TJP03]. We also briefly experimented with dynamic gain, where k is computed using a function similar to the one in Eq. (2.18). This had relatively small effects. Compared to the results with static gain reported below: collision avoidance improved slightly for some scenarios; obstacle clearance improved for some cases with PLLC A, while it worsened with PLLC B; and compactness improved moderately.

Figure 2.6 shows performance metrics for simulations of SPC and PFC controllers. While both perform reasonably well without obstacles, SPC's performance is superior in the presence of obstacles. This validates our hypothesis that SPC is particularly valuable when the cost function is more nonlinear (adding obstacles has that effect). Whenever a drone enters or leaves another drone's neighborhood, the cost function instantaneously changes its value; the gradient changes too. This causes the PFC controller to fail: in these simulations, we observed oscillating behavior and multiple collisions. SPC successfully deals with all of these situations. *In short, SPC is more robust to nonlinearities in the cost function and differences in the behavior of the PLLC.*

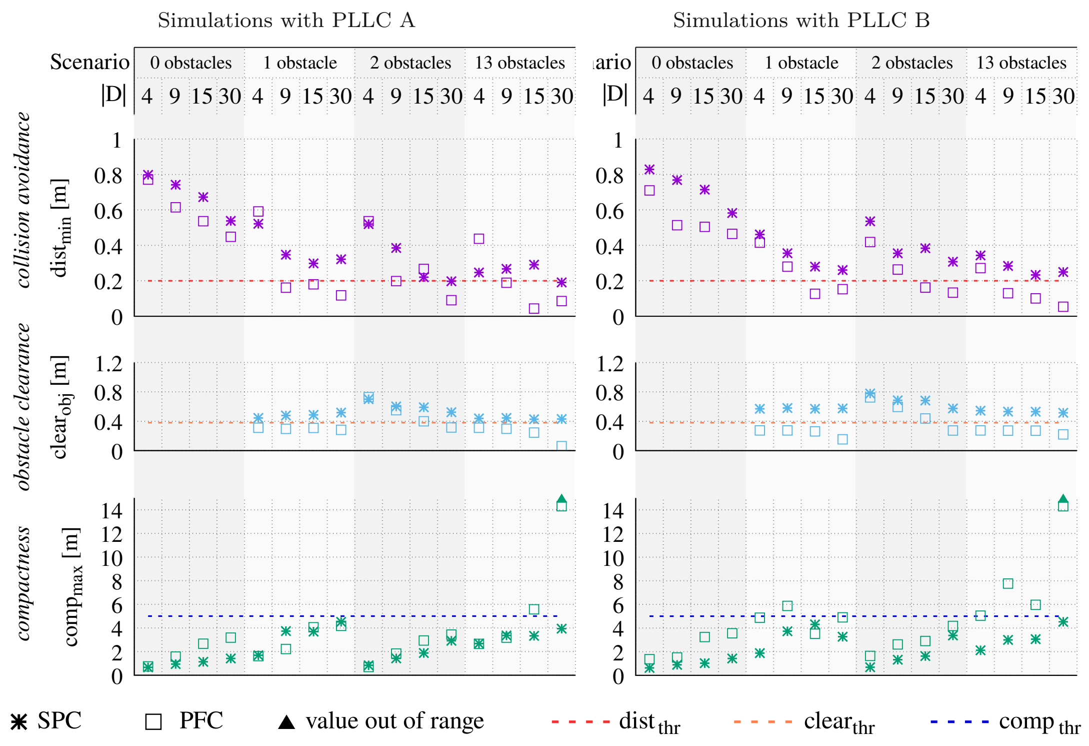


Figure 2.6: Performance comparison for SPC and PFC. Values are *min* (for collision avoidance and obstacle clearance) or *max* (for compactness) over the simulation duration. SPC satisfies $dist_{thr}$ in nearly every scenario, while PFC frequently violates it, especially in the presence of obstacles. SPC maintains $comp_{thr}$ in every scenario, while for PFC, $comp_{max}$ sometimes gets very high, even out of range.

2.4.2 Hardware Experiments

We also experimented with real drones, specifically, *Crazyflie 2.1*-quadcopters [Gie17]; see Figure 2.7a. For localization, we used the *Loco-Positioning* system [Bit21]. The drones seamlessly integrated with the localization system, resulting in a (internal) PLLC that enables a drone to hold its position at a given set-point. Stability, however, depends on both the accuracy of the localization system and on the mechanical limitations of the drone. When hovering at a given set-point, we observed noise in the drone's position in the range of 15 cm. This was also noted in [Que20a].

In the hardware implementation, we used ROS with the same software node as in Section 2.4.1, with only minor parameter modifications. This demonstrates the robustness of SPC with respect to a potential sim-to-real transfer gap. Since Crazyflies are incapable of running ROS on-board, we transmit the position updates to a PC that runs the controller and transmits the set-point position to the drone. Our experiments therefore also show that SPC is resilient to the additional delay introduced by radio transmission of position updates and set-point messages. Our controller, however, could be ported to run directly on ROS-capable drones, since we run it separately for each drone.

For the hardware experiments we used the same scenarios, as in the simulation experiments (Figure 2.3), except with 13 obstacles. Flocks of size $|D| = 2$, 4, 9, and 16 were used.

2.4.2.1 Results

Figures 2.7b and 2.7c show pictures of our experiments in a lecture hall. A video is provided in the Supplementary Materials. To show the drone movements for one example experiment with 16 drones, the recorded traces of the localization system are plotted in Figures 2.7d, 2.7e, and 2.7f.

Figure 2.8 presents performance metrics for our hardware experiments. Data from the prefix of an experiment, when the drones move from initial starting positions into flock formation, are omitted when computing the metrics. Figure 2.8 shows that our SPC-based approach successfully maintains a stable flock of Crazyflie drones by satisfying thresholds for *collision avoidance*, *compactness*, and *obstacle clearance* in nearly every scenario for the full duration of the experiment. Detailed plots for some critical scenarios are shown in Figure 2.8b, and Figure 2.8d. There are transient violations of the metrics, which are likely caused by measurement issues in the localization system; our controller, however, is able to promptly re-establish proper operation. Figure 2.9 provides a comparison of simulation and hardware experiments. It establishes that the controller performance metrics are slightly worse and noisier.

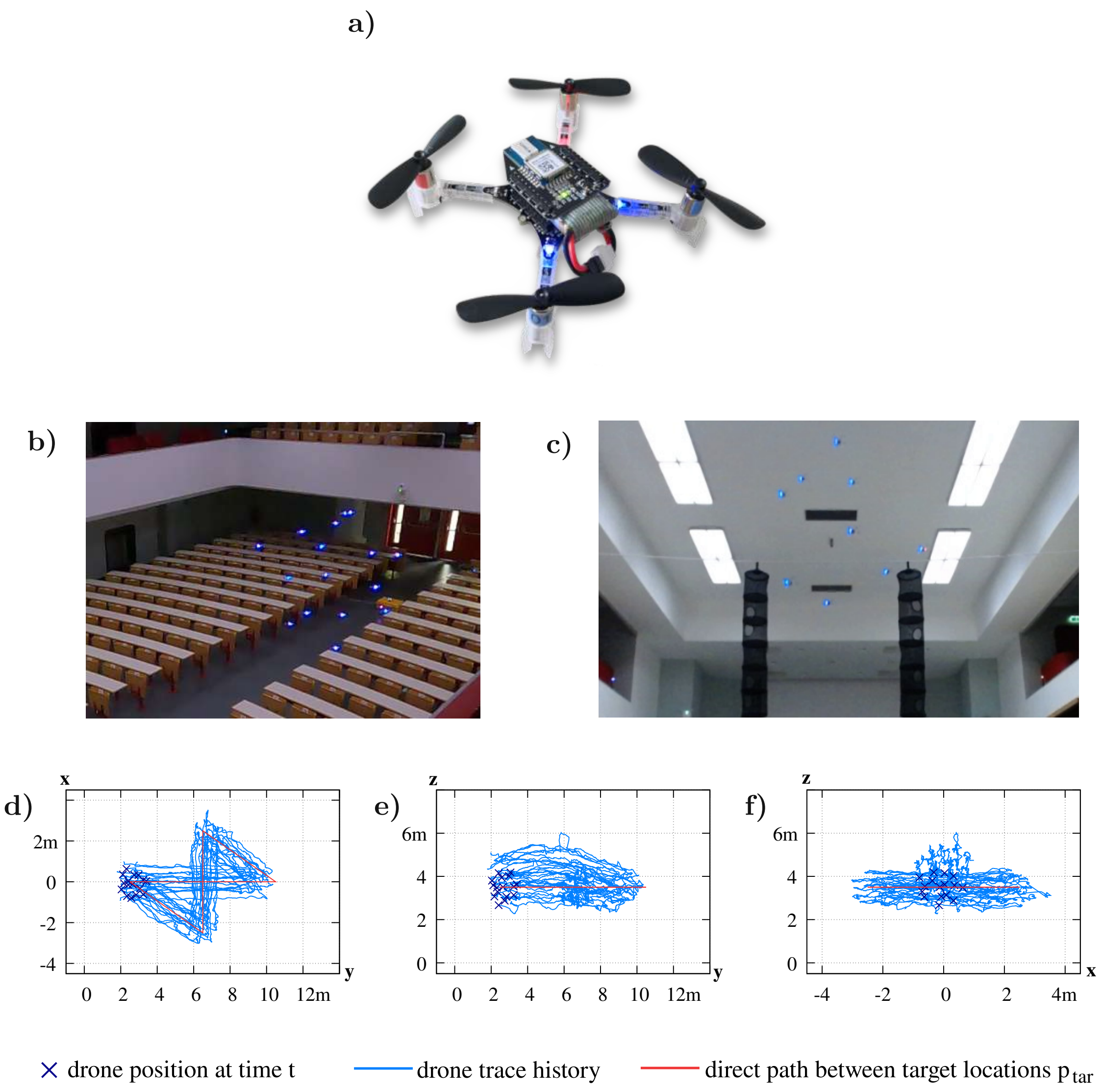


Figure 2.7: Hardware experiments. **a**: Crazyflie 2.1 quadcopters were used. **b**: A flock of 16 drones and **c**: 9 drones with 2 obstacles in our lecture hall (a video is in the Supplementary Materials). **d**, **e**, **f**: Recorded traces show the movements of the 16 drones for one exemplary experiment.

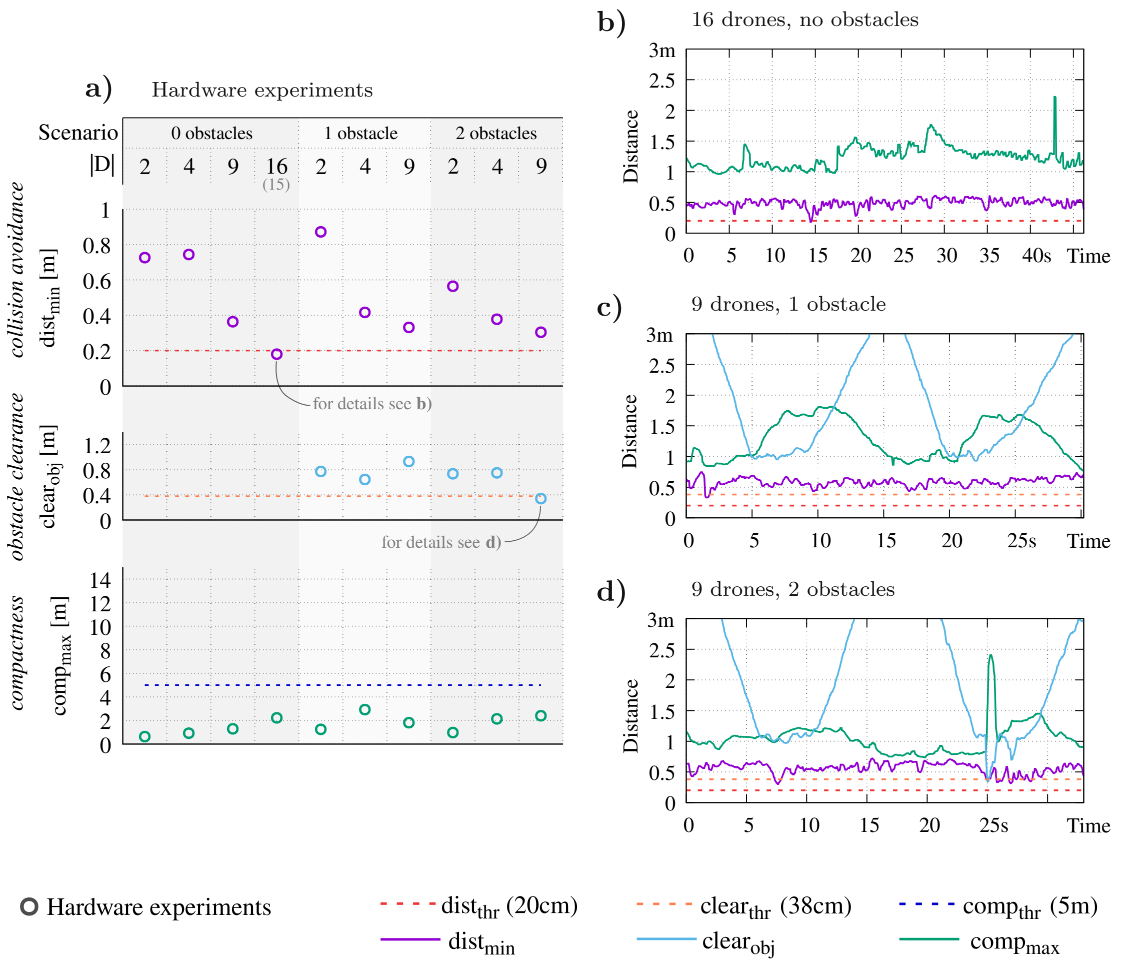


Figure 2.8: Performance metrics for hardware experiments. **a**: Values are *min* (for collision avoidance and obstacle clearance) or *max* (for compactness) over the experiment. Experiments show that the flock is properly maintained: $dist_{thr}$ is satisfied in every scenario but one (see transient violation at $t = 15s$ in **b**). Similarly for $clear_{thr}$ (see transient violation at $t = 25s$ in **d**). **b**, **c**, **d**: Metrics for the whole duration of the hardware experiment for selected scenarios.

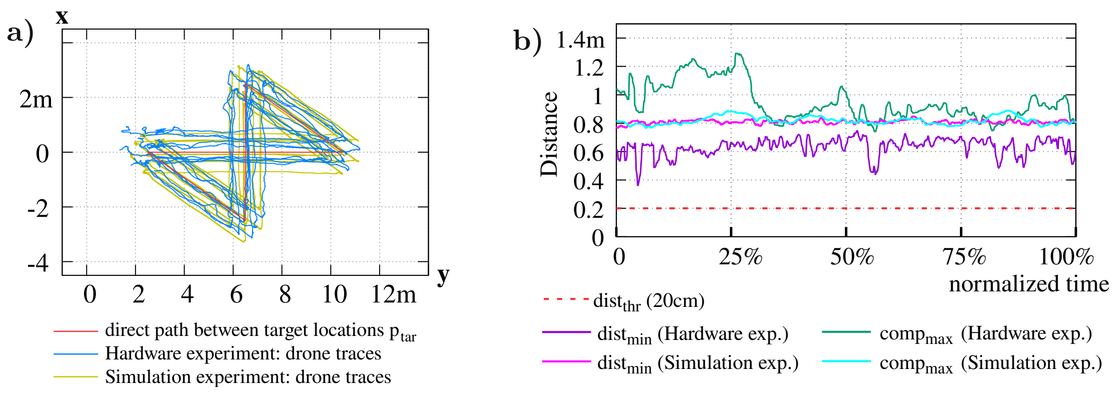


Figure 2.9: Comparison of simulation and hardware experiments for 9 drones without obstacles. **a**: Recorded traces show the movements of the drones. **b**: Metrics for the entire duration of the experiment. The hardware-experiment metrics are a bit more noisy. This also explains why the hardware-experiments metrics in Fig. 2.8 are slightly worse.

2.5 Related work

SPC can be viewed as combining features of MPC and PFC. MPC does a lookahead in time to decide the best control action. It requires a model of the system to compute states at future time points. Intuitively, MPC computes all the states that can be reached in k time steps using different control inputs, picks the best feasible trajectory, and returns the associated control action. In contrast, SPC ignores the system model and feasibility altogether and instead searches for good target states by enumerating promising candidates. Both MPC and SPC recompute their action in each time step using an optimization procedure to handle noise and variability in environment. PFC uses the gradient of the cost to pick the next action, just like SPC, but PFC does not perform any optimization.

Reynolds [Rey87] was the first to propose a flocking model, using cohesion, separation, and velocity alignment force terms to compute agent accelerations. Reynolds model was extensively studied [ER10] and adapted for different application areas [Chu18a]. Alternative flocking models are considered in [Olf06; Meh18; Mar14; SSF19; BLM08], and [TJP03]. Other formulations consider swarm control in the context of formation rigidity [SG17; Poz21; Zel15]. In these approaches, flocks are described using point models. This means that physical properties of agents (e.g., drones) such as mass and inertia, are not taken into account. In our work, we evaluate SPC on a realistic physical drone model, as well as on real hardware.

In addition to these largely theoretical approaches, in [Vás18; SSF21; SSF22], flocking controllers are implemented and tested on real hardware. However, the approach of [SSF21; SSF22] involve the use of model-predictive control, which is computationally more expensive than SPC. In contrast to SPC, [Vás18] requires the velocity of neighboring drones. Gradient optimization for robot control has been studied in [Sch09]. In contrast, SPC uses spatial look-ahead as opposed to pure gradient descent.

2.6 Conclusions

We introduced the concept of *Spatial Predictive Control* (SPC), and demonstrated its utility on the drone flocking problem. SPC is fully distributed. It is based only on the position of the individual drone itself, and on those of neighboring drones. This information is used to compute the gradient of the local cost function and to perform a spatial prediction for the best next action.

We performed an extensive experimental evaluation of SPC on the drone flocking problem. Our simulation experiments used a physics engine with a detailed drone model. Our results demonstrated SPC's ability to form and maintain a flock, avoid obstacles, and move the flock to multiple target locations. They also highlighted SPC's robustness to sensing noise and PLLC variability, and its role in the controller hierarchy.

We also evaluated the same controller implementation on a flock of Crazyflie 2.1 quadcopters in different scenarios, thereby demonstrating the effectiveness of SPC in controlling real hardware. Needing only a minor parameter adjustment, and no modifications to the control algorithm, SPC proved to be very robust in terms of a potential sim-to-real transfer gap. The hardware experiments also highlighted SPC's capability to perform properly in the presence of significant sensor noise introduced by the localization system and the extra latency introduced by radio transmission of positional and control signals.

We also experimentally compared SPC with a related PFC-based approach of [Sch09]. We found that SPC exhibits superior performance and stability, as its discrete search for an optimal solution enables it to avoid oscillations. SPC is a general technique for designing *middle-level controllers* sandwiched between high-level planners and PLLCs that often come integrated with the hardware.

2.7 Acknowledgments

We thank the anonymous reviewers for their valuable comments. Radu Grosu was partially supported by EU-H2020 Adaptness and AT-BMBWF DK-RES and CPS/IoT Ecosystem. This work was supported in part by NSF grants CCF-1954837, CCF-1918225, and CPS-1446832.

2.8 Additional Material

This section comprises additional figures which were not published in [Bra23], but can be found in the extended version [Bra22b].

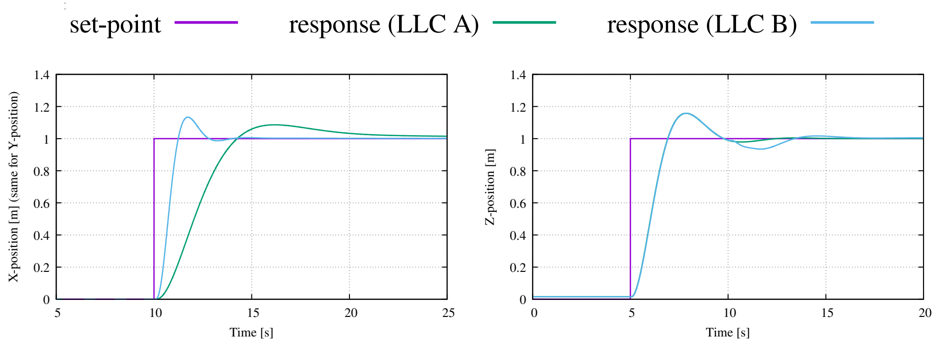


Figure 2.10: To check SPC's robustness to different PLLCs, we experimented with two PLLCs, with different step responses. PLLC B reaches its set-point for x- and y-dimensions in less than half the time of PLLC A, while overshooting more. The PLLCs behave very similarly in the z-dimension.

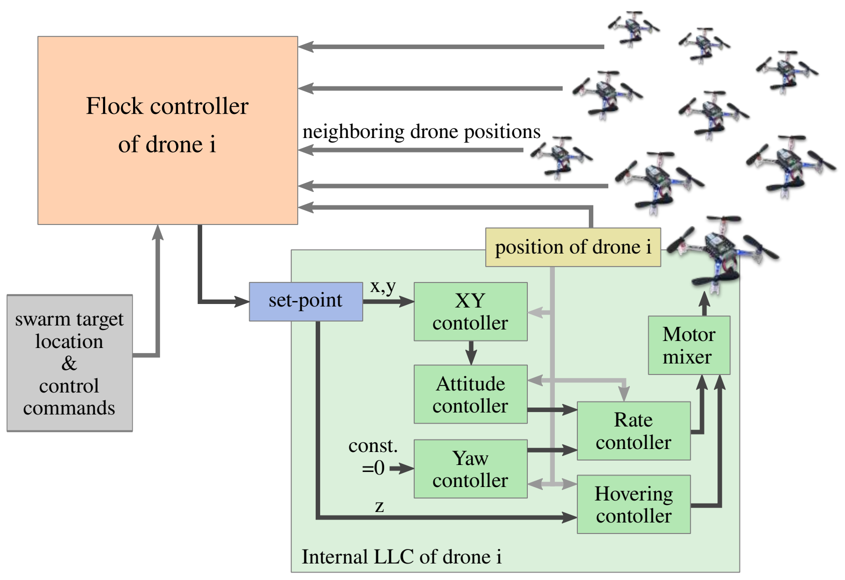


Figure 2.11: The ROS-node of the SPC controller for a drone i receives position messages of all drones and control messages (e.g. swarm target location). It outputs the set-point for the LLC.

CHAPTER 3

Towards Drone Flocking using Relative Distance Measurements

This work was published as:
Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Towards Drone Flocking Using Relative Distance Measurements“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning.* Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2022, pp. 97–109. ISBN: 978-3-031-19759-8. DOI: `10.1007/978-3-031-19759-8_7`

By the following authors with their affiliations:
Andreas Brandstätter at CPS, Technische Universität Wien (TU Wien), Austria
Scott A. Smolka at Department of Computer Science, Stony Brook University, USA
Scott D. Stoller at Department of Computer Science, Stony Brook University, USA
Ashish Tiwari at Microsoft, USA
Radu Grosu at CPS, Technische Universität Wien (TU Wien), Austria

A research report related to this paper is available as:
Andreas Brandstätter. *Research report: Coordination of a Flock of Drones using Ephemeral Environment Learning.* 2022. URL: `https://www.marshallplan.at/images/All-Papers/mp-2022/Brandst%C3%A4tter_Andreas.PDF`

Abstract

We introduce a method to form and maintain a flock of drones only based on relative distance measurements. This means our approach is able to work in GPS-denied environments. It is fully distributed and therefore does not need any information exchange between the individual drones. Relative distance measurements to other drones and information about its own relative movement are used to estimate the current state of the environment. This makes it possible to perform lookahead and estimate the next state for any potential next movement. A distributed cost function is then used to determine the best next action in every time step. Using a high-fidelity simulation environment, we show that our approach is able to form and maintain a flock for a set of drones.

3.1 Introduction

Flocking is a fundamental flight-formation problem. Birds flock for a variety of reasons, including foraging for food, protection from predators, communal warmth, and for mating purposes. Starling flocks can also perform high-speed pinpoint maneuvers, such as a 180° turn [Att]. Some types of flocks in nature have distinct leaders, such as queen bees, and queen ants. Other swarms are formed by animals that do not have a permanently defined leadership, such as starlings or herrings. Although flocking is a well-studied problem mathematically [Rey87; Meh18; CS07b; CS07a], its realization using actual drones is not nearly as mature (but see [Vás18; SSF21]).

Drone swarms, a quintessential example of a multi-agent system, can carry out tasks that cannot be accomplished by individual drones alone [Chu18b]. They can, for example, collectively carry a heavy load while still being much more agile than a single larger drone [MFK11; LK18]. In search-and-rescue applications, a swarm can explore unknown terrain by covering individual paths that jointly cover the entire area [Câm14; BS18; Mic14]. While flocking provides a number of advantages over individual flight, it also poses a significant challenge: the need for a distributed control mechanism that can maintain flock formation and its stability [Olf06]. These collective maneuvers can be expressed as the problem of minimizing a *positional cost function*, i.e., a cost function that depends on the positions of the drones (and possibly information about their environment). In our formulation, every agent is identical, which means there is no designated leader.

To work with such a positional cost function, an absolute localization system is needed. This can be an optical or radio-based system for indoor applications or GPS-based localization for outdoor scenarios. In this work, we study the problem for scenarios that lack an absolute localization system (GPS-denied environments). We only have the ability to measure the distance to other drones and to measure the acceleration and rotational velocity of the own drone using an onboard Inertial Measurement Unit (IMU). For flock formation, we observe that the positional cost function can be replaced by a function based solely on relative distances. This obviates the need for absolute localization. We propose a method to simultaneously learn properties of the environment (inter-agent

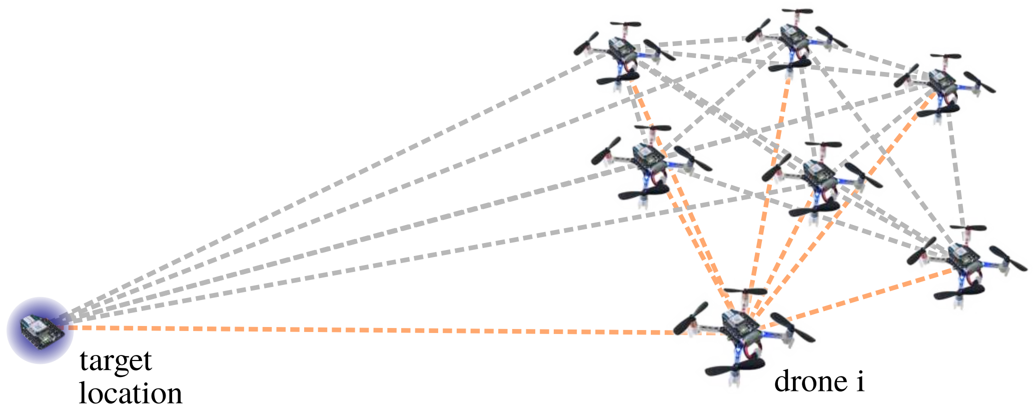


Figure 3.1: Our distributed controller forms and maintains a flock based on relative distance measurements to other agents of the flock. The target location is shown in blue. Distance measurements for drone i to other drones and to the target location are shown in orange.

distance changes), while at the same time maintaining the flock formation solely on relative distance information.

In this paper, we address the following **Challenge Problem:** *Design a distributed controller that forms and maintains a flock based solely on inter-agent distance measurements.*

To solve this problem, we introduce a method to estimate changes of the environment based on the observed changes for previous movements and thereafter use this information to minimize the cost-function over a set of candidate positions. We build upon our previous work that introduced Spatial Predictive Control (SPC) [Bra22b] to select the best next action from the set of candidate positions. However we have a substantially different problem here, since we have limited observation capability: in the previous work [Bra22b], absolute positions of all the drones were available; whereas in this work we can only measure relative distances. This also changes our possibilities how to apply SPC: whereas in the previous work it was possible to optimize the direction based on the cost function's gradient, we need to do a search on possible candidate positions in all directions in this work.

Our agent's observations consist of its own acceleration in three-dimensional space, rotational velocity along three axes, and the relative distance to other agents, as well as the distance to a fixed target location (as shown in Figure 3.1). (The target location is currently only used to counteract drifting tendencies of the whole flock.) There is no communication or central coordination, which makes our approach fully distributed. Our flocking objective is formulated as a cost function (see Section 3.2.2) which is based on these distance measurements. The corresponding action of each agent is a relative spatial vector, to which the drone should move, to minimize its cost function's value.

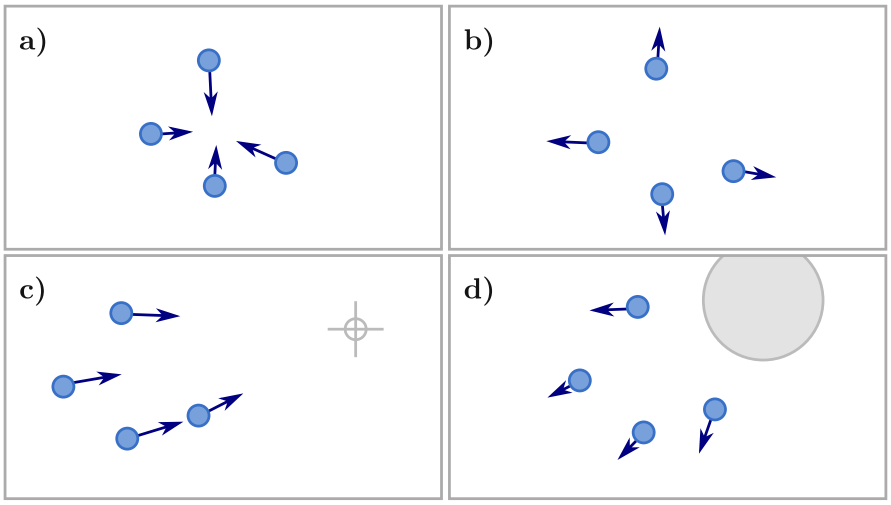


Figure 3.2: Directional movements (indicated by arrows) induced by cost-function terms: **a**: *Cohesion*, **b**: *separation*, **c**: *target seeking*, and **d**: *obstacle avoidance* (not implemented in our method yet).

Paper Outline: Section 3.2 describes our cost function for flocking with target seeking and related performance metrics. Section 3.3 introduces our method to represent environmental knowledge and thereafter describes our distributed flocking controller. Section 3.4 presents the results of our experimental evaluation. Section 3.5 considers related work. Section 3.6 offers our concluding remarks.

3.2 Drone Flocking

This section starts with background on flocking, introduces our cost function for flocking with target seeking, and then presents metrics to assess the quality of a flocking controller.

3.2.1 What is a Flock?

A set of agents, D, is in a flock formation if the distance between every pair of agents is range bounded; that is, the drones are neither too close to each other nor too far apart. Our approach to flock formation is based on defining a cost function such that the agents form a flock when the cost is minimized. The requirement that the inter-agent distance is range bounded is encoded as the first two terms of our cost function, namely the *cohesion* and *separation* terms shown in the next section. Note that the Reynolds rules for forming a flock [Rey87] also include a term for aligning the drone's velocities, apart from the cohesion and separation terms. By not including velocity alignment term, we potentially allow a few more behaviors, such as circling drones, but some of those behaviors are eliminated by our third term, namely the *target seeking* term.

3.2.2 Cost Function

Consider drones i and j, where $i, j \in D$. Let d_{ij}, when it appears in the local cost function of drone i, denote the distance between drone i and drone j as it appears to drone i; this may differ from the actual distance due to sensing error. Similarly l_i denotes the distance between drone i and the fixed target location p_{tar}. In all cases, distances are measured from the drone's center of the mass. Let r_{drone} denote the radius of each drone (specifically the radius of the circumscribed sphere including propellers). In our formulation for the cost function, drone i has access to distances of only a subset $H_i \subseteq D$ of drones, namely its local neighbors. Hence, we define a local cost function, parameterized by i, which uses only the distances to drones in H_i. However for now we only consider the case for global neighborhood, which is $H_i = D$. We plan to extend our experiments also to local neighborhood as future work (see Section 3.6.1). Let d_{H_i} denote the tuple of distances from drone i to drones in H_i. The cost function c we use in this paper is defined for every drone $i \in D$ as in Equation (3.1).

$$c(d_{H_i}, l_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) + c_{tar}(l_i) \tag{3.1}$$

The value of the *cohesion* term increases as drones drift apart, and the *separation* term increases as drones get closer. Each term has a weight, denoted by a subscripted ω.

Cohesion term:

$$c_{coh}(d_{H_i}) = \omega_{coh} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} {d_{ij}}^2 \tag{3.2}$$

Separation term:

$$c_{sep}(d_{H_i}) = \omega_{sep} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} \frac{1}{max(d_{ij} - 2r_{drone}, \hat{0})^2} \tag{3.3}$$

Here $\hat{0}$ denotes a very small positive value. The function $max(., \hat{0})$ ensures the denominator remains nonzero, especially because sensor noise can cause distance measurements to have errors.

To prevent the flock from moving in random directions, we currently use a *target seeking* term with a fixed target location, denoted by p_{tar}, for the entire flock. Here l_i denotes the distance between the center of drone i and the fixed target location p_{tar}.

Target seeking term:

$$c_{tar}(l_i) = \omega_{tar} \cdot {l_i}^2 \tag{3.4}$$

With only *cohesion* and *separation*, the whole flock would form and move in random directions and random locations in absolute world coordinates. This would make it of limited use in any real-world scenario. Our *target seeking* term avoids this behaviour. All drones use the same target location; thus, this last term assumes shared global knowledge of the target. The control algorithm will still be fully distributed. A way to avoid having a fixed target location would be to designate one of the drones as the leader of the flock. This leader could be equipped with additional sensors to get information about its absolute position, allowing it to employ a different control scheme. We leave that investigation for future work.

3.2.3 Flock-Formation Quality metrics

We define two quality metrics to assess the quality of the flock formation achieved by a flocking controller. To compute these quality metrics, we assume to have access to full ground truth information about the absolute positions of the drones. The position (center of mass) of drone i is denoted by p_i.

Collision avoidance: To avoid collisions, the distance between all pairs of drones $\texttt{distance}(D)$ must remain above a specified threshold $\texttt{distance}_{thr}$. We define the metric for the minimum distance between any pair of drones as follows:

$$\texttt{distance}(D) = \min_{i,j \in D; i \neq j} \|p_i - p_j\| \tag{3.5}$$

$$\texttt{distance}(D) \geq \texttt{distance}_{thr} \tag{3.6}$$

We set $\texttt{distance}_{thr} = 2 \cdot r_{drone} + r_{safety}$, where r_{safety} is a safety margin.

Compactness: Compactness of the flock is determined by the flock *radius*. Radius is defined as the maximum distance of any drone from the centroid of the flock:

$$\texttt{radius}(D) = \max_{i \in D} \left\| \frac{\sum_{j \in D} p_j}{|D|} - p_i \right\| \tag{3.7}$$

The drones are said to be in a compact flock formation if $\texttt{radius}(D)$ stays below some threshold $\texttt{radius}_{thr}$; otherwise the drones are too far apart, not forming a flock.

$$\texttt{radius}(D) \leq \texttt{radius}_{thr} \tag{3.8}$$

The value for $\texttt{radius}_{thr}$ is picked based on the drone model and other parameters governing the flock formation problem. We set it to $\texttt{radius}_{thr} = \frac{F \cdot r_{drone}}{\sqrt[3]{|D|}}$, where we use the drone radius r_{drone} to incorporate the physical size and multiply by a factor F.

3.3 Distributed Flocking Controller using Relative Distances

In our distributed approach to flock formation, each drone picks the best action at every time step. The action here is a target displacement vector. Each drone picks the optimal displacement vector for itself by looking ahead in different spatial directions and finding a location that would minimize the cost *if this drone moved there*. To perform this search, each drone needs capability to *estimate* the relative distances to other drones when it moves to different potential target locations. To perform this estimation, each drone stores some measurements from past time steps, which is described in Section 3.3.1. Thereafter, Section 3.3.2 shows how this stored knowledge is used by each drone to estimate relative distances of other drones for different possible displacements of itself.

3.3.1 Environmental knowledge representation

We describe the procedure from the perspective of Drone i. The "environment" for Drone i consists of the current distances to the neighboring drones (and the fixed target), as this is all the information Drone i needs to evaluate the cost function. To represent the knowledge of the environment, Drone i keeps two matrices, a $(k \times 3)$-matrix N and a $(k \times (|D|+1))$-matrix P for some $k \geq 3$. The j-th row of N is a displacement vector for Drone i. The j-th row of P is a vector of change in distances of every other drone and the target to Drone i (as seen by Drone i when it moved by displacement vector in j-th row of N). In particular, P_{lj} is the change in distance of Drone j (or target if $j = |D|+1$) as seen by Drone i when it moved by the vector N_{l*}. The notation N_{l*} denotes the l-th row vector of matrix N. Let us see how the matrices N and P are generated.

Each drone is capable of measuring its own acceleration vector in three dimensions $\vec{a_i}$. By integration, the velocity vector $\vec{v_i}$ can be derived. Drone i constructs the matrices N and P as follows:

1. Save the observations of time instant t. Let $d_{ij,t}$ denote the distances to Drone j, and let $l_{i,t}$ denote the distance to the fixed target, at this time instant t (as obtained from the sensors).

$$d_{ij,t} = d_{ij} \,|\, j \in H_i, t \tag{3.9}$$

$$l_{i,t} = l_i \,|\, t \tag{3.10}$$

2. Integrate velocity vector to keep track of its own position changes, which gives the displacement vector $\vec{u_i}$:

$$\vec{u_i} = \int_{t-\Delta t}^{t} \vec{v_i}\, dt \tag{3.11}$$

3. If the norm of the change in position is larger than a threshold $||\vec{u_i}|| > s_{thr}$, calculate the changes in distances as follows:

$$\bar{d_{ij}} = d_{ij,t} - d_{ij,t-\Delta t} \tag{3.12}$$

$$\bar{l_i} = d_{i,t} - d_{i,t-\Delta t} \tag{3.13}$$

Here $d_{ij,t-\Delta t}$ denotes the observed distance of Drone j at the previous time instant $t - \Delta t$. If the length of the displacement vector is smaller than the threshold, we go back to Step (1).

4. Add the displacement vector $\vec{u_i}$ of Drone i as a row vector in matrix N and add the vector $\langle \bar{d}_{i1}, \dots, \bar{d}_{i|D|}, \bar{l_i} \rangle$ as a row vector in matrix P. Note that we have assumed here that the neighborhood H_i of Drone i is the full set D, but the details can be easily adapted to the case when $H_i \subset D$.

5. The process starts again at (1) and we thus keep adding rows to the matrices N and P.

In this way, the matrix P reflects the available knowledge of how the distances to other drones and to the target change when Drone i moves along vector $\vec{u_i}$. Note that this data gets stale as time progresses, and the newly added rows clearly have more relevant and current information compared to the rows added earlier. Furthermore note that $\vec{u_i}$ is obtained by double integration and therefore it is prone to acceleration sensing errors, and also numerical errors. This influence is however limited, since integration times Δt are also small.

When the procedure above is followed, the matrices N and P keep getting bigger. Let N_{l*} denote the l-th row vector of matrix N. Let N_{a*}, N_{b*}, N_{c*} denote three displacement (row) vectors taken from the (most recent rows in) matrix N such that they are linearly independent – that is, they are all different from each other ($N_{a*} \neq N_{b*} \neq N_{c*}$), nonzero ($N_{a*} \neq \vec{0}$, $N_{b*} \neq \vec{0}$, $N_{c*} \neq \vec{0}$), and not in a common plane ($(N_{a*} \times N_{b*}) \cdot N_{c*} \neq \vec{0}$). These three vectors form a basis in the three-dimensional space. Using a basis transform it is therefore possible to estimate the change for distances for any movement vector $\vec{u}$. Specifically, if

$$\vec{u} = \lambda_a \cdot N_{a*} + \lambda_b \cdot N_{b*} + \lambda_c \cdot N_{c*} \tag{3.14}$$

then we can compute the estimated change in distances of each of the other drones, $\bar{d}(\vec{u})$, and the target, $\bar{l}(\vec{u})$ for this displacement $\vec{u}$ as follows:

$$\langle \bar{d}(\vec{u}), \bar{l}(\vec{u}) \rangle = \lambda_a \cdot P_a + \lambda_b \cdot P_b + \lambda_c \cdot P_c \tag{3.15}$$

(addition and multiplication in Equation (3.15) are applied element-wise on the vectors).

We have shown how the three vectors N_{a*}, N_{b*}, N_{c*} can be used to infer the expected change for any displacement vector $\vec{u}$ for Drone i. To ensure that three different vectors with meaningful data are present, our controller employs some optimizations in addition to the procedure described above. A special startup procedure with random movements is used to collect initial data. The three vectors (N_{a*}, N_{b*}, N_{c*}) and their associated data in P are continuously updated to avoid outdated information. However, a vector is only considered if the threshold s_{thr} is exceeded within a certain time limit. This avoids updates when the drone is moving very slowly over longer time-periods. To get the best quality of the prediction for any displacement $\vec{u}$, it is desirable to have the vectors (N_{a*}, N_{b*}, N_{c*}) ideally, but not necessarily, orthogonal to each other. This also influences which row (vector) gets replaced in the matrices N and P. As soon as one of the vectors gets outdated, a random movement in an orthogonal direction might be triggered to enhance the knowledge representation.

3.3.2 Distributed flocking controller

We now describe our control approach based on the cost function introduced in Section 3.2.2 and on the environmental knowledge representation described in Section 3.3.1.

The set of candidate positions Q is defined as follows:

$$Q = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid x \in \{-\epsilon_Q, 0, \epsilon_Q\}, y \in \{-\epsilon_Q, 0, \epsilon_Q\}, z \in \{-\epsilon_Q, 0, \epsilon_Q\} \right\} \tag{3.16}$$

This gives a set of 27 points on a equally spaced three dimensional grid. The spacing distance of this grid is ϵ_Q. Over this set Q the best action q_{next} is searched by minimizing the cost function c:

$$q_{next} = \underset{q \in Q}{\operatorname{argmin}} \{c(\widehat{d}_i(q), \widehat{l}_i(q))\} \tag{3.17}$$

If two candidate positions q_1 and q_2 both have the same minimum value for the cost function c, our implementation of argmin takes the last one based on the implementation of the enumeration. The function $\widehat{d}_i(q)$ estimates the distances to drones, where $\widehat{l}_i(q)$ estimates the distance to the target, if the action q is applied. For each $q \in Q$, the vector $\widehat{d}_i(q)$ (and the value $\widehat{l}_i(q)$) is calculated by first computing the estimates of the *change vector* $\bar{d}_i(q)$, and the change $\bar{l}_i(q)$ using Equation 3.15. Now the distances can be estimated by just adding the estimated change to the currently measured distances d_{i*} and l_i:

$$\widehat{d}_i(q) = d_{i*} + \bar{d}_i(q) \tag{3.18}$$

$$\widehat{l}_i(q) = l_i + \bar{l}_i(q) \tag{3.19}$$

Each drone minimizes its local cost function (Eq. 3.17) in order to recompute the desired set-point at every time step. As we similarly did in [Bra22b], this set-point is then handed off to a low-level controller that controls the thrust of the drone's motors so that it steers towards this set-point.

3.4 Experiments

We evaluated our method using simulation experiments. The goal of the experiments was to investigate and demonstrate the ability to form and maintain a stable flock while holding position at a target location.

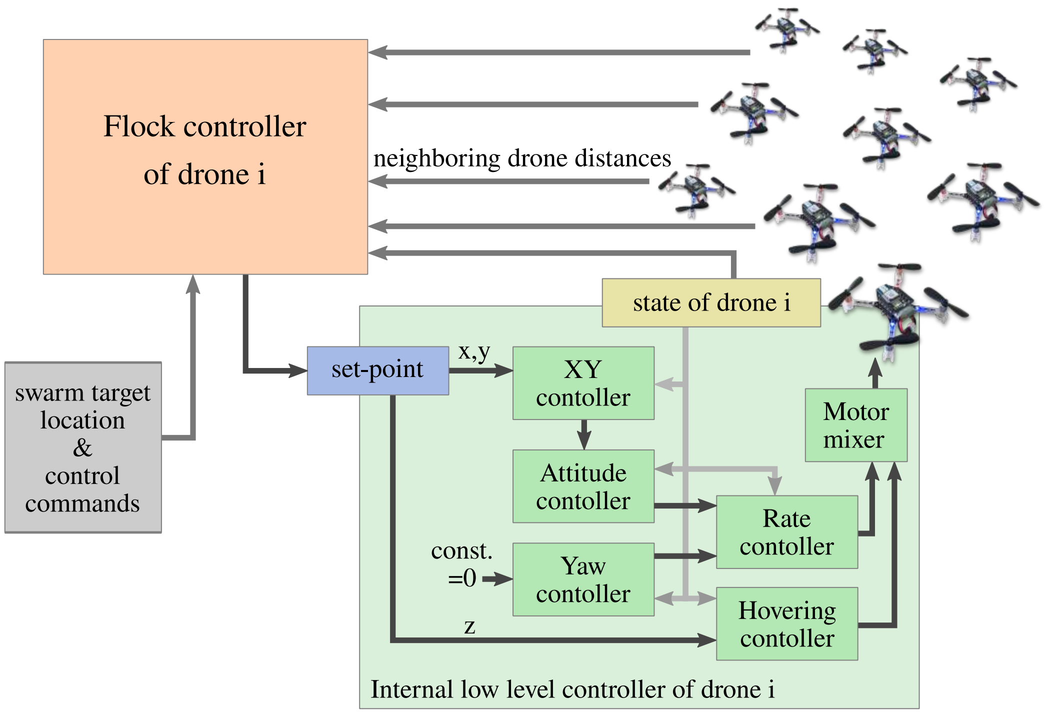


Figure 3.3: The ROS-node of the SPC controller for drone i receives distance measurements to neighboring drones and control messages (e.g. swarm target location, start/stop command). It outputs the set-point for the internal low level controller.

3.4.1 Simulation Experiments

As a simulation framework, we use *crazys* [SAI18], which is based on the *Gazebo* [KH04] physics and visualization engine and the Robot Operating System (ROS) [Sta]. Our algorithm is implemented in C++ as a separate ROS node. As shown in Figure 3.3, it receives the measured distances to neighboring drones, and control messages, such as the target location or a stop command, from the human operator. It calculates the best next action according to Equations (3.16)-(3.19). The parameter ϵ_Q is determined empirically and fixed throughout the whole simulation. Auxiliary functions, like hovering at the starting position, and landing after the experiments are finished, are also implemented in this node.

In order to evaluate the control mechanism and its implementation, we fixed the target location, as described above. This avoids drifting behaviour of the whole flock, which could not be detected by relative distance measurements in any way. Simulations were done with flocks of size $|D| = 5$, 9, and 15. Figure 3.4 shows a screenshot of a simulation with 5 drones. All simulations use global neighborhood ($H_i = D$) for now.

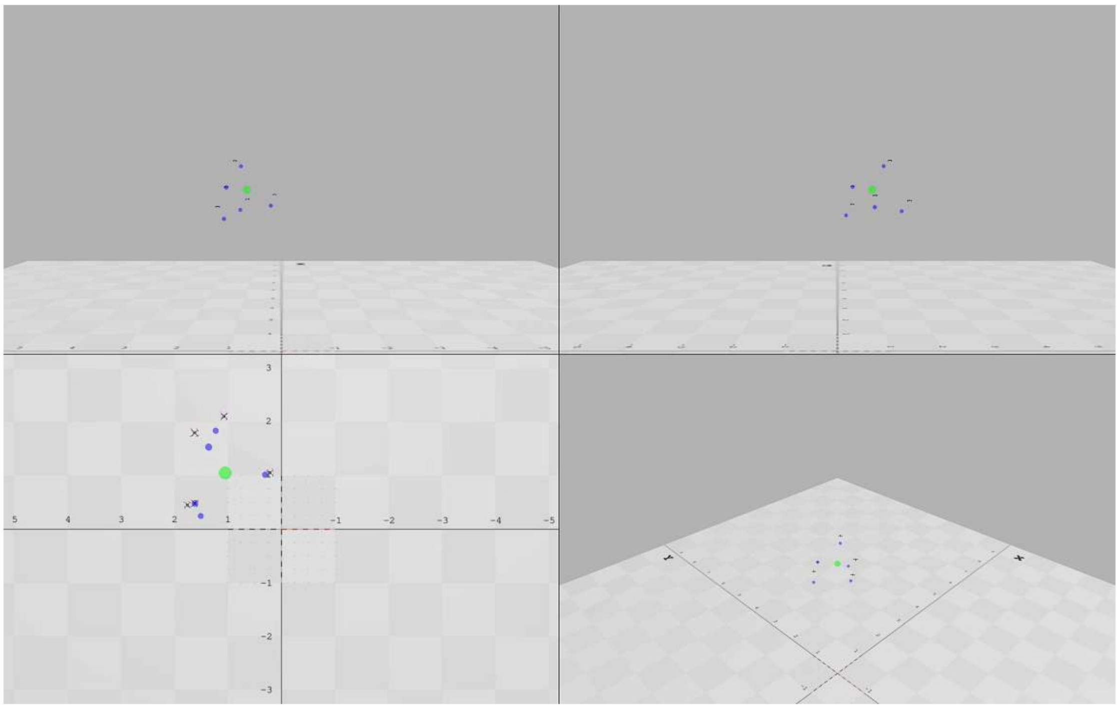


Figure 3.4: Screenshot of the end of the simulation with 5 drones. Shown from four different camera views after the flock reached its target. The green dot indicates the target location. The blue dots visualize the next action which is supplied to the lower level controller.

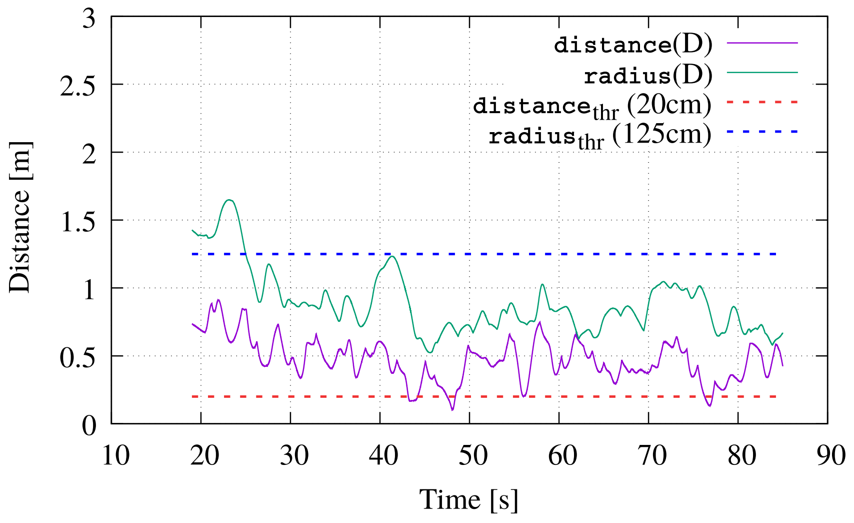


Figure 3.5: Quality metrics for simulation with 5 drones. Threshold $\texttt{distance}_{thr}$ for collision avoidance is satisfied most of the time. After settling in, the swarm radius remains below the threshold $\texttt{radius}_{thr}$, thus showing the ability to form a compact flock in the simulation. (Quality metric recordings start at $t = 19\,s$ after initialization procedure.)

3.4.2 Results

Early results show that our approach is able to properly form and maintain a flock with only relative position measurements. Figure 3.5 shows performance metrics over time for a simulation with 5 drones. The analysis of the quality metrics for *collision avoidance*, and *compactness* show that our control approach successfully maintains a stable flock (threshold $\texttt{distance}_{thr}$ is only violated for very short moments). Note that these results are already obtained before extensive controller tuning. Using carefully adjusted values for ω_{coh} and ω_{sep} should lead to even better results and maintain the threshold throughout the whole simulation.

3.5 Related work

Reynolds [Rey87] was the first to propose a flocking model, using cohesion, separation, and velocity alignment force terms to compute agent accelerations. Reynolds' model was extensively studied [ER10] and adapted for different application areas [Chu18a]. Alternative flocking models are considered in [Olf06], [Meh18], [Mar14], [SSF19], [TJP03], and [Sch09]. In all these approaches, absolute position measurements and/or inter-agent communication were available. In our work, we only work with relative distance measurements and a fully distributed formulation.

In addition to these largely theoretical approaches, in [Vás18] and [SSF21], flocking controllers are implemented and tested on real hardware. However, the approach of [SSF21] involves the use of Nonlinear Model Predictive Control (NMPC). In contrast to our work, [Vás18] also requires the velocity of neighboring drones.

3.6 Conclusions

We introduced a method to control a flock only based on relative position measurements to neighboring drones, and demonstrated its utility on the drone flocking problem. We performed simulation experiments using a physics engine with a detailed drone model. Our results demonstrated the ability to form and maintain a flock, and hold its position on a target location.

3.6.1 Future work

As we currently have only intermediate results of the experiments with limited number of agents, we plan to do more extensive testing with a wide set of different scenarios, including larger number of drones, and local neighborhood ($H_i \subset D$). Neighborhood might be defined by euclidean distance, or alternatively by topological distance, as introduced in [Bal08]. As further directions of future work, we plan to extend our approach with obstacle avoidance capabilities. We also plan to test it for moving target locations and various path tracking scenarios. To prepare for the transfer to real hardware we plan to introduce sensor noise in the simulation and test the robustness of our method to cope with such disturbances. As next goal it should then be implemented on real drones, specifically, *Crazyflie 2.1*-quadcopters [Gie17].

© 2022 Springer Nature Switzerland. Material from: Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Towards Drone Flocking Using Relative Distance Measurements“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2022, pp. 97–109. ISBN: 978-3-031-19759-8. DOI: 10.1007/978-3-031-19759-8_7.

CHAPTER 4

Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements

This work was published as:
Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements“. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 12193–12200. DOI: 10.1109/ICRA57147.2024.10610147

By the following authors with their affiliations:
Andreas Brandstätter at CPS, Technische Universität Wien (TU Wien), Austria
Scott A. Smolka at Department of Computer Science, Stony Brook University, USA
Scott D. Stoller at Department of Computer Science, Stony Brook University, USA
Ashish Tiwari at Microsoft, USA
Radu Grosu at CPS, Technische Universität Wien (TU Wien), Austria

Abstract

We present *distributed distance-based control* (DDC), a novel approach for controlling a multi-agent system, such that it achieves a desired formation, in a resource-constrained setting. Our controller is fully distributed and only requires local state-estimation and scalar measurements of inter-agent distances. It does not require an external localization system or inter-agent exchange of state information. Our approach uses spatial-predictive control (SPC), to optimize a cost function given strictly in terms of inter-agent distances and the distance to the target location. In DDC, each agent continuously learns and updates a very abstract model of the actual system, in the form of a dictionary of three independent key-value pairs $(\Delta\vec{s}, \Delta\boldsymbol{d})$, where $\Delta\boldsymbol{d}$ is the partial derivative of the distance measurements along a spatial direction $\Delta\vec{s}$. This is sufficient for an agent to choose the best next action. We validate our approach by using DDC to control a collection of Crazyflie drones to achieve formation flight and reach a target while maintaining flock formation.

4.1 Introduction

Multi-agent Systems (MAS) can collectively perform tasks which are beyond the abilities of individual agents. For Search-and-Rescue (SAR) applications, there is a wide variety of multi-agent approaches using ground, aerial, floating, and underwater vehicles [Que20b; Câm14; Mic14]. MASs can also play a role in environmental monitoring, space exploration, agriculture, entertainment, and industrial maintenance [Sch20]. All of these applications require a coordination and control method for formation establishment and maintenance.

MAS formations, such as flocking, can be described by a (distributed) cost function $c(\boldsymbol{x})$ over the state variables $\boldsymbol{x}$ of an agent, such that when each agent minimizes this cost, the system reaches the desired formation [Rey87; Olf06]. We call cost functions that are defined in terms of scalar distances $\boldsymbol{d} \subset \boldsymbol{x}$ to other agents only, *distance-based cost functions*. For flocking, formulations using distance-based cost functions can be found in [TJP03; Bra23]. However, it turns out that controllers, such as Potential Field Controller (PFC) [TJP03] and Spatial Predictive Control (SPC) [Bra23], choose their actions based on the cost-function's spatial gradient. Consequently, even for distance-based cost functions, existing approaches require knowledge of relative-position vectors to derive the spatial gradient, and subsequently the control actions.

*A **key challenge**, therefore, is to determine if the controller design for distance-based cost functions can be generalized such that control actions are chosen based only on scalar distances without knowledge of relative position vectors?*

While Global-Navigation Satellite System (GNSS) and comparable types of indoor localization systems [Fer17; Lao18], can determine relative position vectors, they might not be installed or currently available at some locations, and they are unlikely to be available in some applications (e.g., underwater SAR, cave exploration). This further motivates the above challenge.

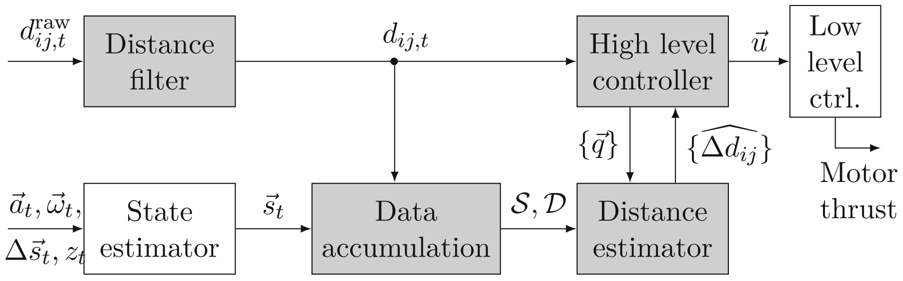


Figure 4.1: Block diagram of our DDC distributed formation controller, running locally on each agent. Grey blocks were developed by us; for the white blocks, we use third-party state-of-the-art implementations.

One approach to solve the above challenge is to estimate the relative positions of other agents. This can be accomplished by measuring distances only, and using a coordinated movement schedule in each time step, as in [CYA11]. One can then apply methods such as PFC or SPC to derive the actions. However, as shown by our experiments in Section 4.5.2, this approach is impractical, since it is slow (only a subset of agents can move simultaneously), and susceptible to sensing noise.

In this work, we propose a novel approach, called DDC (for *distributed distance-based control*), that obviates the estimation of relative positions. Instead, it directly uses estimated distance changes in the spatial gradient of the cost-function. A key idea is to use a first order Taylor approximation of such changes. Intuitively, each agent maintains a dictionary of three independent key-value pairs $(\Delta\vec{s}, \Delta\boldsymbol{d})$, representing the partial derivative $\Delta\boldsymbol{d}$ of distance measurements along spatial directions $\Delta\vec{s}$.

During an initial startup-phase, agents perform *exploration* in orthogonal (or independent) directions, to populate this dictionary. Agents then *exploit* this dictionary to estimate the expected change $\widehat{\Delta\boldsymbol{d}}$ when they displace themselves by $\Delta\vec{u}$, as predicted by the Taylor approximation. This enables them to estimate the new cost when moving by a certain vector. As in [Bra23; Bra22c], to find the set-point with minimal cost, each agent performs in every time-step, a search over a grid of points surrounding the current location.

Importantly, each agent continuously *updates* its copy of the dictionary at each time step with its last observation. In particular, agents replace stale information with newer information, and they also occasionally perform an exploration step, where they take a potentially non-optimal action, to maintain independence of the keys in this dictionary. The dictionary is reminiscent of an attention layer used in transformers.

Note that each agent implicitly makes the simplifying, but not generally accurate, assumption that other agents do not move. Under the constrained setting, this is our only recourse. Our experiments show that our approach is able to tolerate this inaccurate assumption. DDC passes the selected movement action to the agent's low-level controller that controls the motor thrust. Low-level controllers for quadcopters and their stability is studied in numerous works, such as [Gie17; MK11; Flo14; Gar17].

MAS DDC is suitable in resource-constrained settings, where individual agents: (1) Are only able to perform local state-estimation with onboard sensors, (2) Do not communicate any state information, and (3) Can only measure scalar distances to other agents (and to the fixed target location). The local state-estimation of an agent is comprised of its change in position relative to a prior position, velocity, acceleration, orientation and optionally altitude. For this, onboard sensors, such as inertial-measurement units (IMUs), optical flow modules, barometers, and altimeters are used. These sensors do not provide any information about other agents.

As a consequence of (2), there is no central coordinator, and we thus present a fully distributed approach. As another consequence of (2), triangulation, multilateration, rigid graph based methods and joint (global) state-estimation are not applicable. Scalar distance measurements to other agents (and to the fixed target location) contain significantly less information than relative position, pose, or angle measurements would do.

In our hardware implementation, distance measurements also turned out to be extremely noisy, providing only imperfect measurements. In contrast to Model Predictive Control (MPC) approaches, DDC does not need a physical model of the agents.

Summarising, the main contributions of this paper are:

- We introduce the concept of DDC, a novel formation control method, which is solely based on onboard sensing, and on scalar distance measurements.
- DDC is model-free, fully distributed, and does not require internal-state information of other agents.
- We evaluate DDC on the drone-flocking problem with target seeking, in a drone simulation environment.
- We experimentally validate our approach, by achieving flocking with real off-the-shelf quadcopters, *Crazyflie 2.1* [Gie17].

4.2 Related Work

In [Lu23; Sas17; Moh18; Whe20; Hep21], agents are able to localize themselves with cameras, LiDAR sensors and/or external systems. In our setting, these sensors are not available. Other works [OA14; Ara16; KPA17; LXW18; WWP10; BS20; He13], including the survey [OPA15], assume individual agents can sense relative positions of their neighboring agents with respect to their own local coordinate systems. This provides considerably more information than scalar distance measurements.

Position estimation and control for two-dimensional (2D) scenarios are studied in [JDA17; Cha18], but generalization to three-dimensional (3D) is not trivial. In [AY11; ST08; ZWG19; GLX20; JAH20], message exchange (transmitting acceleration, angular velocity, or other data) is used between at least some of the agents. DDC does not use such communication.

The work in [Stu09] makes use of *anchor nodes*, with fully known positions in a global reference frame. In [CS19], a centralized approach to localization is presented. In contrast, DDC is fully distributed. In [CYA09], a single leader moving with constant velocity is tracked by a single follower. In [Kan14], one leading and two following agents are described.

Formation control in [SS18] is based on distance measurements involving sinusoidal perturbations to the agent's actions. Perturbation frequencies are assumed to be pairwise distinct, which limits application to larger numbers of agents, if in practice the allowed range for frequencies is bounded by the agent's mechanical limitations. Our previous work in [Bra22c] sketched the idea of distance-based flocking using precise distance values, but it could not handle imperfect measurements due to significant sensor noise and it performed no hardware experiments.

In this paper, we therefore introduce DDC, which features a different method of data accumulation, additional signal filtering, the use of exploration steps to assure an orthogonal basis, and a new slack parameter for the separation term. We evaluated DDC for altitude-aided drone flocking on real hardware. The work in [CYA11] proposes a control scheme with three alternating periods: identification, control, and resting (where the agent needs to remain stationary). In Section 4.5.1 we provide a comparison of DDC to this method, which turns out to be very susceptible to sensing noise and much slower.

4.3 Formation Controller

Consider a collection of $|A|$ agents. Each agent i, where i ranges from 1 to $|A|$, runs a controller, consisting of the following blocks, as shown in Figure 4.1:

- **Distance filter**: As measurements of the scalar inter-agent distances coming from real hardware are imperfect (noisy), the raw measurements $d_{ij,t}^{\text{raw}}$ of the distance between agents i and j at time t are filtered.

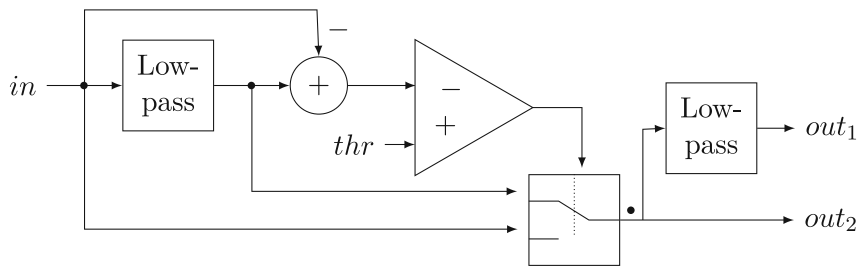


Figure 4.2: Rejection filter: When the input is within threshold *thr* of the low-passed signal, the signal is fed through. Otherwise the filter outputs the low-passed signal (out_2). Finally there is an optional second low-pass (out_1).

- **State estimator**: All sensor data that is available for the individual agent i (e.g. acceleration $\vec{a}_t$, rotational velocity $\vec{\omega}_t$) is processed by a standard Kalman filter (using the state-of-the-art implementation available for Crazyflies [Gie17]) for local state estimation. This is used to estimate relative position $\vec{s}_t$ (for Agent i), i.e., relative to its position at $t = 0$. Additional optional sensors that can directly measure displacement $\Delta\vec{s}_t$ and/or absolute elevation z_t can be incorporated when available.

- **Data accumulation**: As the system operates, historical data of the change in inter-agent distances $\Delta d_{ij,t}$ for displacements $\Delta\vec{s}_t$ is stored in two matrices $\mathcal{D}, \mathcal{S}$.

- **Distance estimator**: The data stored in $\mathcal{S}$ and $\mathcal{D}$ is used to (linearly) estimate the change in distances $\widehat{\Delta d_{ij}}$ (for all j) for any given candidate displacement $\vec{q}$.

- **High-level controller**: The cost $c(\boldsymbol{d})$ is evaluated on a set of candidate displacements $\{\vec{q}\}$. The best is chosen as $\vec{u}$.

- **Low-level controller**: We use a PID controller (as implemented in [Gie17; SAI18]) to set the motor thrust for $\vec{u}$.

4.3.1 Outlier Detection and Noise Filtering

Our hardware experiments use a collection of *Crazyflie 2.1* quadcopters [Gie17], equipped with the *DWM1000* Ultra-Wideband (UWB) hardware module [Qor21] on *Loco-positioning deck* with the software implementation of [Sha22], to sense the inter-agent distances. The radio signals are used solely for inter-agent distance measurements. No internal state information is exchanged over this channel. There are no fixed UWB beacons.

As noise in distance measurements is a critical concern for our system, we did an analysis of the noise model. It turns out that *outliers* are common, and there is some additive *random noise*. When looking at its energy spectrum, we found higher energy in low-frequency ranges, as compared to white noise. Such a noise profile was also reported in [Sha22].

Our filter tracks the input *in*, using an Infinite Impulse Response (IIR) low-pass filter; see Figure 4.2. *To detect outliers*, we subtract the input *in* from the output of this low-pass and compare it with a threshold *thr*. If it is below the threshold, we directly use *in* as output; otherwise, we feed the low-passed signal to the output out_2. After this rejection filter, we use an optional *IIR low-pass filter* to produce out_1. Cutoff frequencies and threshold were determined empirically. In order to retain changes due to real movements, the cutoff frequency must not be too low. Therefore low-frequency components of noise cannot be filtered effectively. We observe, however, that higher-frequency noise is also effectively suppressed by *linear regression* in our *data accumulation* block (step 3a in Section 4.3.2).

4.3.2 Data Accumulation in Key-Value Dictionary

Each agent i measures the current distances to neighboring agents (and the target) and estimates it's local position. For simplicity, we describe the distance estimation process from the perspective of agent i, only. The same procedure is analogously applied for estimating distance to the target.

Agent i stores the relevant data history in two matrices, a (3×3)-matrix $\mathcal{S}$ and a $(3\times|A|)$-matrix $\mathcal{D}$. Row-k of $\mathcal{S}$ is a *key* displacement vector $\Delta\vec{s}$ (in 3-dimensional space) for agent i. Row-k of $\mathcal{D}$ is a *value* vector $\Delta\boldsymbol{d}$ of change in distances to every other agent, as seen by agent i when it moved by displacement $\Delta\vec{s}$. In particular, $\mathcal{D}_{kj}$ is the change in distance to agent j as seen by agent i, when it moved by the vector $\mathcal{S}_{k*}$ ($\mathcal{S}_{k*}$ is $\mathcal{S}$'s k-th row vector). The dictionary thus represents partial derivatives $\mathcal{D}_{k*}$ of distance measurements along spatial directions $\mathcal{S}_{k*}$.

Let $d_{ij,t}$ be the distance to agent j at time t. Each agent's local state estimator is capable of estimating its own position $\vec{s}_t$ relative to its initial position at $t=0$. Agent i continuously updates the matrices $\mathcal{S}$ and $\mathcal{D}$ as follows:

0. Set t_0 and $\vec{s}_{t_0}$ to be the initial time and position of Agent i.

1. Initialize empty sets T and M_j for all $j \in A$.

2. Save the current time t and distances in sets: $T = T\cup\{t\}$, $M_j = M_j\cup\{d_{ij,t}\}$. Calculate the displacement vector $\Delta\vec{s}_t$ based on relative position information:

$$\Delta\vec{s}_t = \vec{s}_t - \vec{s}_{t_0} \tag{4.1}$$

3.a) If the norm of this displacement vector is larger than a given threshold, i.e. if $||\Delta\vec{s}_t|| > s_{thr}$, calculate the changes in distances by linear regression over the saved measurements within $\Delta t = t - t_0$ as follows:

$$\bar{d}_{ij} = \frac{1}{|T|} \sum\nolimits_{r\in T} d_{ij,r} \tag{4.2}$$

$$\bar{t} = \frac{1}{|T|} \sum\nolimits_{r\in T} r \tag{4.3}$$

$$\Delta d_{ij} = \frac{\sum_{r\in T} (r-\bar{t})\ (d_{ij,r}-\bar{d}_{ij})}{\sum_{r\in T} (r-\bar{t})^2}\ \Delta t \tag{4.4}$$

Note, that $|T| = |M_j|$. Go to Step (4).

b) If Δt is larger than a threshold $\Delta t > t_{thr}$, set t_0 to be t and s_{t_0} to be s_t, and go back to Step (1). In this case the agent did not move considerably within Δt, and we therefore discard such measurements.

c) Otherwise, go back to Step (2). Continue measuring.

4. Select the row k in $\mathcal{S}$, which is most similar to $\Delta\vec{s}_t$, by $k = \text{argmax}_{r\in\{1,2,3\}}\{|\mathcal{S}_{r*} \bullet \Delta\vec{s}_t|\}$. Replace row k in matrix $\mathcal{S}$ with the normalized displacement vector $\frac{\Delta\vec{s}_t}{||\Delta\vec{s}_t||}$, and replace row k in matrix $\mathcal{D}$ with the vector $\langle\frac{\Delta d_{i1}}{||\Delta\vec{s}_t||}, \ldots, \frac{\Delta d_{i|A|}}{||\Delta\vec{s}_t||}\rangle$.
5. Set t_0 to t and s_{t_0} to s_t, and start again at (1) and keep updating rows in the matrices $\mathcal{S}$ and $\mathcal{D}$, representing recent displacements of agent i and associated changes in distance measurements.

Note that relative position $\vec{s}_t$ is obtained by a state estimator based on Inertial Measurement Unit (IMU) data and similar sensors. State estimation is therefore prone to sensor noise, and might drift over time. However, our overall approach is immune to such drifts since it only depends on displacements $\Delta\vec{s}_t$ that happen in time duration Δt, which is bounded by t_{thr}.

4.3.3 Exploration

As the matrices $\mathcal{S}$ and $\mathcal{D}$ are initially empty, the agents first need to perform some exploration: move in some non-optimal direction, measure inter-agent distances, and populate the matrices with that information. This is done mainly in the startup phase. Later, the moves are the computed control actions. We also keep track of when each row was updated. If a row becomes older than some threshold t_{old}, it is deleted. Exploration is performed according to the following rules ($\times$ denotes the cross product of two vectors, and $\bullet$ denotes the dot product):

1. If all rows of $\mathcal{S}$ and $\mathcal{D}$ are empty: Sample three random variables as the components of vector $\vec{r}$. Apply action $\vec{q}_{expl,1} = w_{expl}\ \frac{\vec{r}}{||\vec{r}||}$, where w_{expl} is a scaling parameter.
2. Only $\mathcal{S}_{1*}$ is not empty: Sample random vector $\vec{r}$ as in step (1). Apply the vector $\vec{q}_{expl,2} = w_{expl}\ \frac{\mathcal{S}_{1*}\times\vec{r}}{||\mathcal{S}_{1*}\times\vec{r}||}$ as action (which is by construction orthogonal to $\mathcal{S}_{1*}$).
3. Only $\mathcal{S}_{1*}$, and $\mathcal{S}_{2*}$ are not empty: Apply the vector $\vec{q}_{expl,3} = w_{expl}\ \frac{\mathcal{S}_{1*}\times\mathcal{S}_{2*}}{||\mathcal{S}_{1*}\times\mathcal{S}_{2*}||}$ as action (which is by construction orthogonal to $\mathcal{S}_{1*}$ and $\mathcal{S}_{2*}$).
4. All entries in $\mathcal{S}$ are non-empty:
   a) If there exist two dependent rows k and $m \neq k$ (pointing in a similar direction $(|\mathcal{S}_{k*} \bullet \mathcal{S}_{m*}| > \kappa_{thr})$, where κ_{thr} is a parameter to quantify this similarity): Apply the vector $\vec{q}_{expl,4} = w_{expl}\ \frac{\mathcal{S}_{k*}\times\mathcal{S}_{m*}}{||\mathcal{S}_{k*}\times\mathcal{S}_{m*}||}$ as action (which is by construction orthogonal to these row vectors).
   b) If such rows do not exist, the vectors $\mathcal{S}_{1*}$, $\mathcal{S}_{2*}$, $\mathcal{S}_{3*}$ are linearly independent; i.e. they are all different ($\mathcal{S}_{1*} \neq \mathcal{S}_{2*} \neq \mathcal{S}_{3*}$, ensured by 4a), nonzero ($\mathcal{S}_{1*} \neq \vec{0}$, $\mathcal{S}_{2*} \neq \vec{0}$, $\mathcal{S}_{3*} \neq \vec{0}$, ensured by $||\Delta\vec{s}_t|| > s_{thr}$), and not in a common plane ($(\mathcal{S}_{1*} \times \mathcal{S}_{2*})\bullet\mathcal{S}_{3*} \neq \vec{0}$, ensured by 2, 3, and 4). $\vec{0}$ is used as shorthand notation for $(0, 0, 0)^T$. Exploration is finished and the controller can go to exploitation mode.

4.3.4 Exploitation

The dictionary represents the partial derivative of distance measurements along spatial directions $\mathcal{S}_{k*,k\in\{1,2,3\}}$ yielding associated values $\mathcal{D}_{k*}$. For any displacement vector $\vec{q}$, we compute the estimated change in distances to each other agent $\widehat{\Delta d_{i*}}(\vec{q})$ by a first order Taylor approximation:

$$\widehat{\Delta d_{i*}} = \lambda_1\ \mathcal{D}_{1*} + \lambda_2\ \mathcal{D}_{2*} + \lambda_3\ \mathcal{D}_{3*} \tag{4.5}$$

(here, addition and multiplication are applied element-wise).
Since $\mathcal{S}_{1*}$, $\mathcal{S}_{2*}$, $\mathcal{S}_{3*}$ form a basis, the unique coefficients λ_1, λ_2, λ_3 of the linear combination are given by:

$$\vec{q} = \lambda_1\ \mathcal{S}_{1*} + \lambda_2\ \mathcal{S}_{2*} + \lambda_3\ \mathcal{S}_{3*} \tag{4.6}$$

Likewise we can compute the estimated distances after the agent i would have moved by displacement vector $\vec{q}$:

$$\widehat{d_{ij}}(\vec{q}) = d_{ij} + \widehat{\Delta d_{ij}}(\vec{q}) \tag{4.7}$$

4.3.5 High-Level Controller

Our high-level controller works with any distance-based cost function. It makes use of distance measurements provided by our *distance estimator* to choose the best action from a set of candidate actions as the resulting action.

The set of candidate actions Q is defined as follows:

$$E = \left\{ (x,y,z)^T \,\middle|\, x,y,z \in \{-1,0,1\} \right\} \setminus \vec{0} \tag{4.8}$$

$$Q = \left\{ \epsilon_Q\, n \frac{\vec{q}}{\|\vec{q}\|} \,\middle|\, \vec{q} \in E, n \in \{1,..,N_Q\} \right\} \cup \vec{0} \tag{4.9}$$

This gives a set of $26 \cdot N_Q + 1$ points which are spaced by a distance of ϵ_Q each in every direction, including diagonals. The estimated distances, if action $\vec{q}$ is taken, are estimated by Eq. (4.7). Over the set Q, the best action $\vec{q}_{best}$ is chosen by minimizing cost function c:

$$\vec{q}_{best} = \underset{\vec{q}\in Q}{\operatorname{argmin}} \{ c(\widehat{d_{i*}}(\vec{q})) \} \tag{4.10}$$

Each agent computes the desired position set-point at every time s.t. its local cost function (Eq. 4.10) is minimized. This set-point is passed as input to a low-level controller to control the drone's propeller motors, as it is also done in [Bra23].

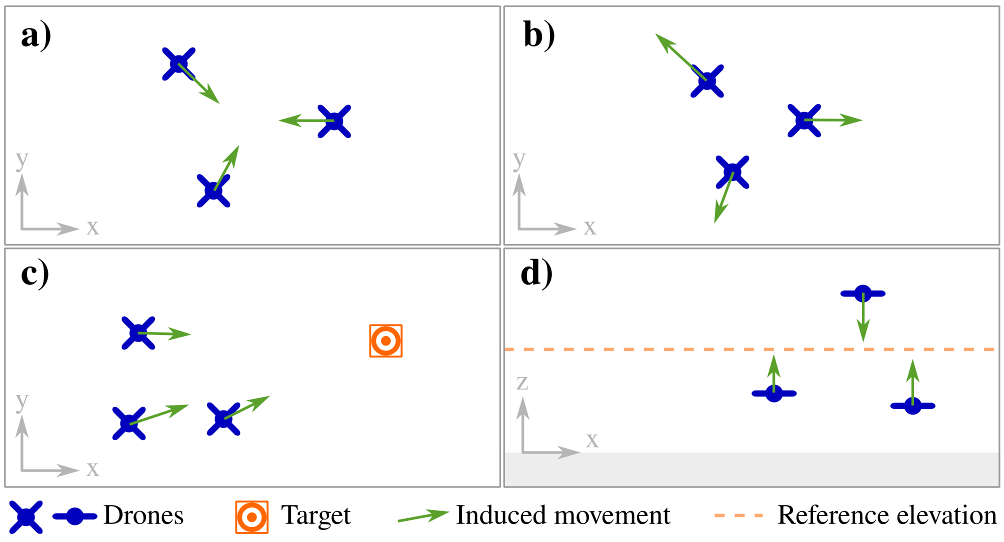


Figure 4.3: Cost-function terms for drone flocking: **a**: *cohesion*, **b**: *separation*, **c**: *target-seeking*, and **d**: *elevation*.

4.4 Application to Drone Flocking

In this section, we introduce a distance-based cost function for flocking and describe quality metrics of flock formations.

4.4.1 Drone Flocking

Let A denote the set of agents. They are in flock formation if the distance between every pair of agents is range bounded; that is, the agents are neither too close to each other nor too far apart. To formulate this concept, we define a cost function that each agent tries to minimize in order to achieve the formation. Cost function based formulations for flocking were studied in [Rey87; TJP03; Meh18]. These works show the usefulness of such a formulation for achieving flock formations and discuss properties w.r.t. to non-collision and non-dispersion.

Consider drones i and j. Let d_{ij} denote the distance between drones i and j, as it appears to drone i (measured from their centers of mass). Let l_i denote the distance between drone i and the target location p_{tar}. The radius of the circumscribed sphere of a drone is denoted by r_{drone}. Drone i has only access to distances of a subset $H_i \subseteq A$ of the drones, called its neighbourhood. Hence, the local cost function is parameterized by i, and uses only the distances to drones in H_i. We define H_i using a neighborhood radius parameter r_H, as follows:

$$H_i = \{j \,|\, d_{ij} < r_H, j \in A \setminus \{i\}\} \tag{4.11}$$

The tuple of distances from drone i to drones in H_i is denoted by d_{H_i}. Our cost function c_a (a for aerial) is defined for every drone $i \in A$ as follows:

$$c_a(d_{H_i}, l_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) + c_{tarA}(l_i) \tag{4.12}$$

a) Cohesion term:

$$c_{coh}(d_{H_i}) = \frac{\omega_{coh}}{|H_i|} \sum_{j \in H_i} {d_{ij}}^2 \tag{4.13}$$

As $c_{coh}(d_{H_i})$ increases when drones drift apart, this term keeps drones together. Each term includes a subscripted ω as a weight.

b) Separation term:

$$c_{sep}(d_{H_i}) = \frac{\omega_{sep}}{|H_i|} \sum_{j \in H_i} \frac{1}{max(d_{ij} - 2r_{drone} - \chi_{sep}, \hat{0})^2} \tag{4.14}$$

The *separation* term keeps drones apart, as it increases when drones get closer. Here $\hat{0}$ denotes a very small positive value. Function $max(., \hat{0})$ ensures a strictly positive denominator. The slack parameter χ_{sep} is used to influence the minimum distance between two drones. This parameter is different from previous works [Rey87; Bra23; Bra22c]. During experimental validation, this showed significant impact on avoiding collisions.

c) Aerial target-seeking term:

$$c_{tarA}(l_i) = \omega_{tar}\ l_i \tag{4.15}$$

Our *aerial target-seeking* lets us move the flock towards a specified target location. In simulation, we move the target location, while on real hardware, we switch between different targets, where only the active one is used in the cost function. As all agents have the same target location, we assume shared knowledge of that information. However, the control algorithm itself is still fully distributed. Instead of a target location, one of the drones could be designated as a leader. This leader could have additional sensors to obtain information about its absolute position, to employ an alternative control scheme.

4.4.2 Altitude-Aided Drone Flocking

The size of the indoor space constrains movement in all three dimensions, but typical-room height is most restrictive. We also want to place our target on the ground; as such, the flock should reach this target indirectly, at a certain altitude. We therefore describe an alternative cost function c_g (Eq. 4.16) for *altitude-aided drone flocking*. In real-world environments, altitude measurements can be gathered by barometers or altimeters (e.g. downward-facing range sensors).

$$\begin{aligned} c_g(d_{H_i}, l_i, z_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) +& \\ c_{tarG}(l_i, z_i) + c_{elev}(z_i)& \end{aligned} \tag{4.16}$$

d) Elevation term:

$$c_{elev}(z_i) = \omega_{elev}\ (\zeta_{elev} - z_i)^2 \tag{4.17}$$

Here z_i denotes the z-coordinate of the position of agent i, which is it's altitude. This term keeps the flock at a certain altitude, which is determined by the reference elevation ζ_{elev}.

e) Ground target-seeking term:
The distance l_i of drone i to the current target is projected to the x-y plane to compute the cost term c_{tarG}:

$$\tilde{l}_i(l_i, z_i) = \sqrt{max({l_i}^2 - {z_i}^2, 0)} \tag{4.18}$$

$$c_{tarG}(l_i, z_i) = c_{tarA}(\tilde{l}_i(l_i, z_i)) \tag{4.19}$$

4.4.3 Flock-Formation Quality Metrics

We define metrics, and associated constraints, to assess the flock quality achieved by DDC.

a) Collision avoidance: The distance between all pairs of drones $\mathrm{dist}(A)$ must remain above a specified threshold $\mathrm{dist}_{thr} = 2 \cdot r_{drone} + r_{safety}$, where r_{safety} is a safety margin.

$$\mathrm{dist}(A) = \min_{i,j \in A; i \neq j} d_{ij} \tag{4.20}$$

$$\mathrm{dist}(A) \geq \mathrm{dist}_{thr} \tag{4.21}$$

b) Compactness: The radius of the sphere circumscribing all agents would be the ideal metric. This would require us, however, to know each agent's position in a global coordinate system, which we do not have. We thus instead use the maximum radius of the pairwise inter-agent circumscribed spheres:

$$\mathrm{comp}(A) = \frac{1}{2} \max_{i,j \in A; i \neq j} d_{ij} \tag{4.22}$$

c) Target reaching: To assess the quality of target-seeking, we determine, for each target k, the average distance to the target:

$$\mathrm{tar}_k(A) = \frac{1}{|A|} \sum_{i \in A} l_{ik} \tag{4.23}$$

When using ground a target-seeking term, l_{ik} is replaced by $\tilde{l}_{ik}$.

4.5 Evaluation

We evaluate DDC's performance in simulation and on real hardware using *Crazyflie 2.1* quadcopters [Gie17]. In both cases, we use the same software implementation, with only minor adjustments of empirically determined parameters.

4.5.1 Simulation Experiments

As simulation environment for our experiments, we used *crazys* [SAI18]. This is based on the *Gazebo* [KH04] physics and visualization engine and the Robot Operating System (ROS) [Sta]. We implemented DDC (the gray blocks shown in Figure 4.1) in C++, as a separate ROS node. DDC receives measured distances $d_{ij,t}^{\mathrm{raw}}$, and relative position information $\vec{s}_t$. Based on the cost function, it calculates the next action $\vec{u}$ and passes it to the low-level controller. Controller parameters were determined empirically from a range of values for each parameter by manual inspection of performance metrics.

As noise is critical (cf. 4.3.1), we modeled it in simulation by a sum of low-passed white noise plus additional white noise. We tested different noise levels to assess DDC's robustness.

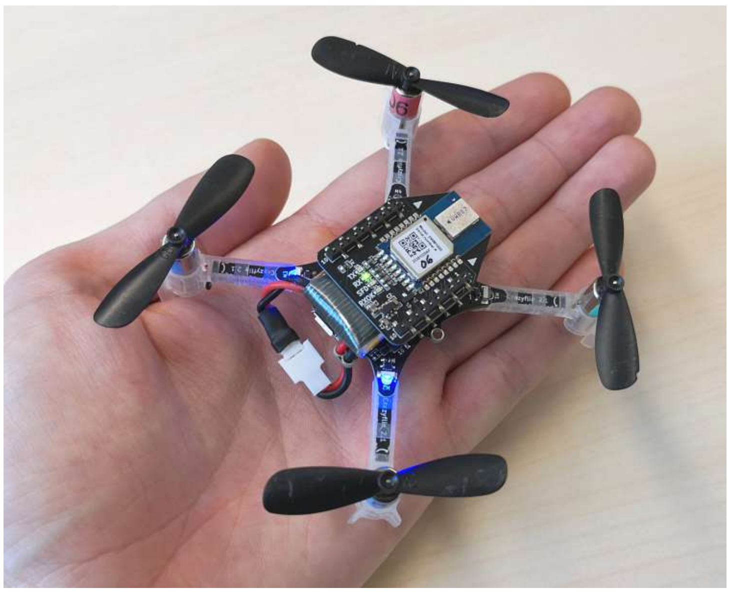
For evaluation, we used two positions sequentially supplied as target location: After the drones formed a flock around the starting position $(x, y, z) = (0, 0, \zeta_{elev})$, they move to $(10\text{m}, 0, \zeta_{elev})$ and then to $(10\text{m}, 10\text{m}, \zeta_{elev})$. We performed simulations with $|A| = 5$, 9, and 15 drones. Quality metrics over time for representative simulation runs are shown in Figure 4.5. The metrics for *compactness* (blue) and *collision avoidance* (orange) show that DDC successfully maintains a stable flock without collisions. The flock moves towards the target locations, as shown by the decreasing *target-reaching* metric (green). While moving, *compactness* (blue) is temporarily degraded.

4.5.2 Comparison to Cyclic Stop-And-Go Strategy

We compared DDC to the *cyclic stop-and-go strategy* [CYA11], consisting of three alternating periods: *identification*, *control*, and *resting*. This method allows only a subset of agents, which are not neighbours of each other, to move simultaneously (i.e. to perform *identification* or *control*). At the same time, all other agents are in *resting* phase, where they need to remain stationary. This is easy in simulation, but hard to achieve in the real world.

As [CYA11] does not provide an implementation, we took it upon ourselves to implement it: In the *identification* phase, we use three orthogonal movements and true-range multilateration [Fan86] to estimate other agent's relative positions. For the *control* phase, we use a variant of SPC [Bra23], with the same parameters and cost function weights as in DDC. For arbitration of the different phases, we allowed for central coordination,even though in our problem statement agents are not able to exchange such information. In our DDC approach, such coordination is not necessary. Comparison in Fig. 4.7 shows that this method is susceptible to sensing noise and much slower in reaching the target.

a)



b)

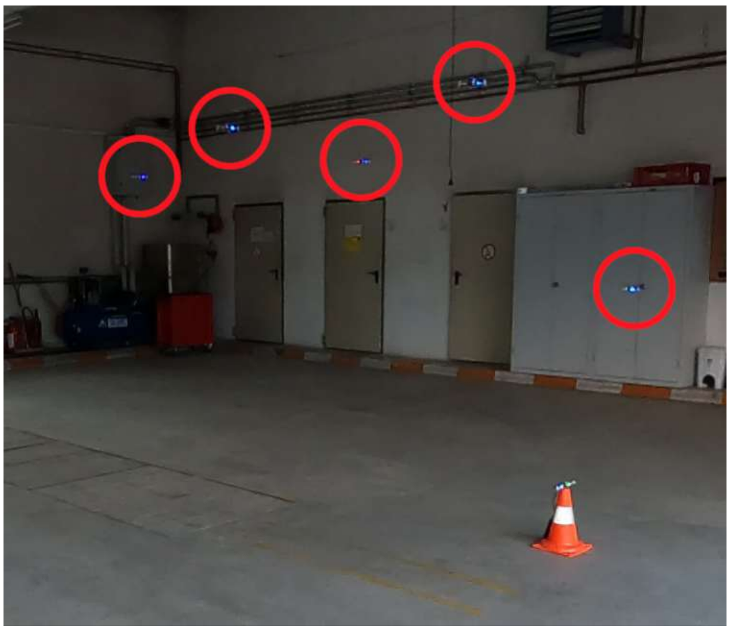


Figure 4.4: Drones used for hardware experiments: **a**: *Crazyflie 2.1* quadcopter **b**: Five drones (highlighted in red circles) while testing scenario Join.

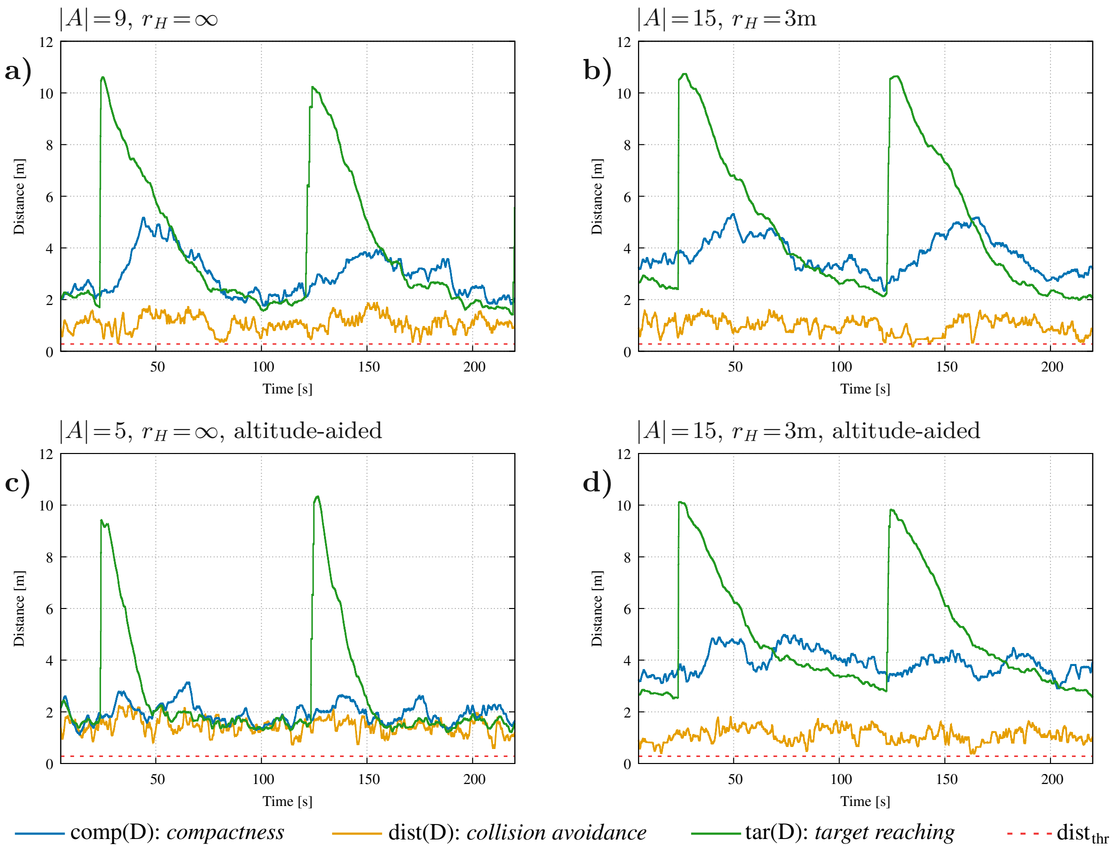


Figure 4.5: Metrics over time for representative simulation experiments. The target is updated with a new position at $t_1 = 20\,\text{s}$ and $t_2 = 120\,\text{s}$, resulting in a saw-tooth shape. **a**: drone flocking with 9 agents and global neighborhood, **b**: 15 agents and local neighborhood, **c**: altitude-aided drone flocking with 5 agents and global neighborhood, **d**: 15 agents and local neighborhood.

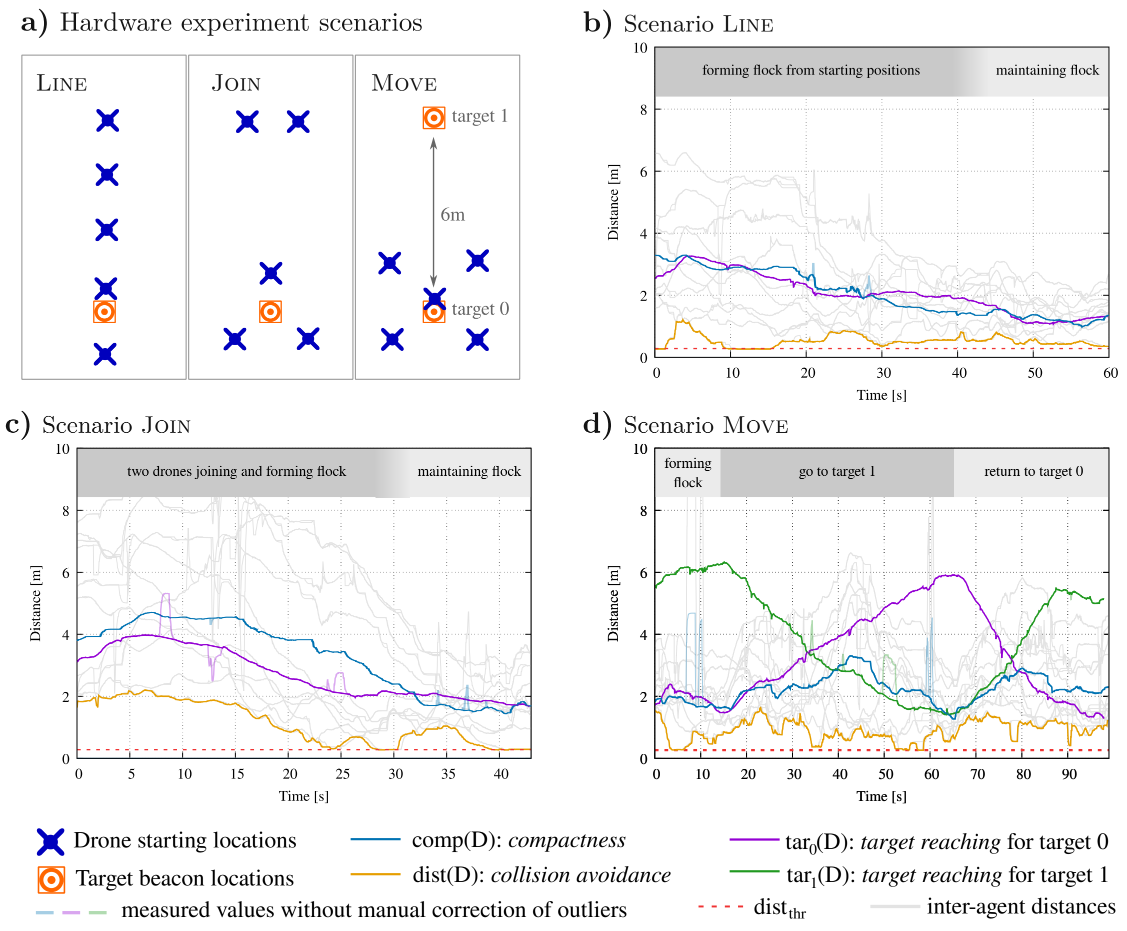


Figure 4.6: Hardware experiments. **a** Scenarios: Line, forming a flock from starting positions along a line; Join, two agents joining three agents at the target location; Move, a flock moves between two alternating target locations. Plots of metrics and inter-agent distances for scenarios **b**: Line, **c**: Join, and **d**: Move.

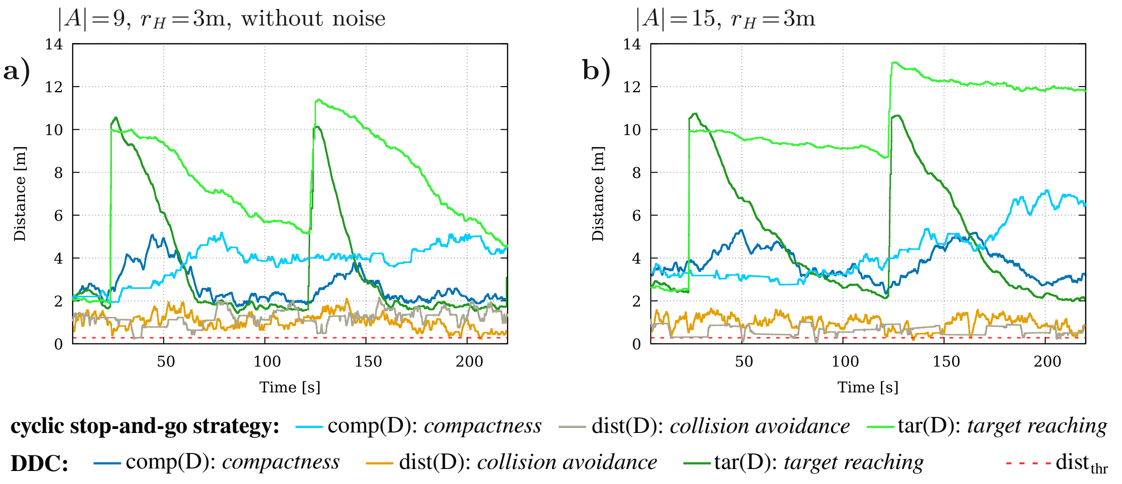


Figure 4.7: Comparison of DDC and cyclic stop-and-go strategy [CYA11], for simulations of drone flocking with local neighborhood. For the latter, *target reaching* (green) decays much more slowly, and *compactness* (blue) is considerably worse, than for DDC. **a**: 9 agents without distance measurement noise, **b**: 15 agents with same noise level as in all other simulations of DDC.

4.5.3 Hardware Experiments

We experimented with *Crazyflie 2.1* quadcopters [Gie17] with *Loco-positioning* deck, featuring *DWM1000* UWB modules [Qor21] for distance measurements and *Flow deck v2* (with z-ranging altimeter and optical flow module); see Figure 4.4. To determine inter-drone distances, we used a ranging software implementation [Sha22]. For each drone, the measured distances $d_{ij,t}^{\text{raw}}$ and relative position information $\vec{s}_t$ are transmitted to a computer. There, DDC is executed in a separate ROS node for each drone (same as in simulation), the next action $\vec{u}$ is computed, which is then transmitted to the drone. Even though it is executed on the same computer for all drones, DDC is fully distributed, as there is no additional information exchange between the individual nodes.

We evaluated DDC in three scenarios. In Figure 4.6, the starting locations and traces of representative experiments are visualized. In scenario Line, the drones start in a line. As the drones move into a more compact formation, the *compactness* metric improves (becomes lower) over time. For Join, two drones start further away from target 0. They join the other three, which are already closer to the target. For Move, the drones start around target 0, indicated by *forming flock*. Then, when moving towards target 1 in the next section, the average distance to the target continuously decreases. In the last section, the flock returns to target 0. While moving, *compactness* is temporarily degraded. In all of these scenarios, the metric for *collision avoidance* stays above the threshold. A video of these hardware experiments is available online at `https://youtu.be/InRoXIdc-WU`.

4.6 Conclusions

We introduced DDC, a MAS-formation-control approach which is fully distributed, and solely based on scalar-distance measurements, and local-position estimation. We demonstrated DDC on drone-flocking with target-seeking. To validate DDC, we took a two-pronged approach: (1) We performed simulation experiments using a physics engine with a detailed drone model, and (2) We performed experiments with real drones, specifically, *Crazyflie 2.1* quadcopters. Our results demonstrate DDC's ability to form and maintain a flock, and move towards a target location. To the best of our knowledge, we are the first to demonstrate such a controller on aerial MASs. While DDC is able to satisfy the specified performance metrics, future work will focus on ensuring that the flock reaches the target location by a given deadline and extending it with obstacle avoidance capabilities. We plan to perform analysis on stability and guarantees for non-collision and non-dispersion. We will also explore replacing our dictionary structure with more general attention-based models and learning techniques.

4.7 Acknowledgments

Andreas Brandstätter was supported by a research exchange scholarship from Austrian Marshall Plan Foundation. Radu Grosu was partially supported by EU-H2020 NimbleAI and AT-BMBWF DK-RES and CPS/IoT Ecosystem. This work was supported in part by NSF grants CCF-1954837, CCF-1918225, CPS-1446832, ITE-2134840, and OIA-2040599. We thank the volunteer fire department of Sieghartskirchen for letting us use their vehicle hall for our hardware experiments.

4.8 Recent Related Work

This section was not published in [Bra24], but provides additional related material and background information on more recent related work. To give a concise overview of related work and highlight the relevant aspects in comparison to our method, we present them in a tabular form. The columns of Tables 4.1 to 4.4 are as follows:

- **Observations**
  In our method, we only use scalar interagent distance measurements and onboard IMU data. In contrast, other works use relative position measurements, bearing angle measurements, visual observations, or a combination thereof. These types of measurements can provide significantly more information than only scalar interagent distances. Therefore, control methods relying on these are not directly comparable to our proposed method.

- **Communication**
  In our paper, we do not use any information exchange, i.e. communication, between the individual agents. In related works, however, agents exchange their state containing their own pose in a relative or global coordinate system, velocity, acceleration, actions, or parts thereof. They might also exchange observations about other agents, or relay information they received from others.

- **Dimensionality** (**Dim.** in column head)
  We work with aerial robots, more specifically drones, and therefore need a method for a 3D environment. Some other works use ground-based robots and, as such, only work in 2D environments. Unless explicitly shown, it is not clear whether these methods are applicable to 3D as well.

- **Hardware demonstrated** (**HW** in column head)
  In our paper, we actually implement our method using real hardware drones. We thereby experimentally demonstrate the fitness of this method to work with real aerial robotic systems. Many other papers lack this aspect, as they either describe a method in theory only or use simulations without actual hardware implementation. Some even knowingly neglect the physical properties of the robotic systems by describing the agents as simplified point models. For these works, it is therefore unclear whether they are actually applicable to real robotic systems.

- **Remarks**
  Other aspects that differentiate related works to our method could be, e.g., fully known positions of reference beacons/markers, limitations on the minimum/maximum number of agents, distinctive leader-follower configurations, and special requirements/limitations regarding the agents' motions.

Paper	Observations	Comm-unication	Dim.	HW	Remarks
Our work: Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements [Bra24]	interagent distances and IMU data	none	3D	yes	**ours**
Cooperative navigation of MAVs In GPS denied areas [ST08]	interagent distance and bearing	IMU data	2D	no	
Control for Localization of Targets using Range-only Sensors [Stu09]	distances to anchor nodes	none	3D	2D only	anchor nodes with known positions
Coordination with the leader in a robotic team without active communication [CYA09]	interagent distances	none	2D	no	leader-follower structure (constant velocity)
Decentralized control for satellite formation using local relative measurements only [WWP10]	relative positions and velocities	none	3D	no	
Range-only sensing for formation shape control and easy sensor network localization [AY11]	interagent distances	distance information	2D	no	rigid body framework
Formation control using range-only measurements [CYA11]	interagent distances	none	3D	no	cyclic stop-and-go strategy
Distributed formation control of mobile autonomous agents using relative position measurements [He13]	interagent distances	velocity	2D	no	leader-follower structure
Distance-based undirected formations of single-integrator and double-integrator modeled agents in n-dimensional space: DISTANCE-BASED UNDIRECTED FORMATIONS [OA14]	relative positions	none	3D	no	
Distance-based formation control with a single moving leader [Kan14]	relative positions	none	2D	no	leader-follower structure
Distributed Formation Stabilization Using Relative Position Measurements in Local Coordinates [Ara16]	relative positions	none	2D	no	
System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization [Sas17]	relative positions	none	3D	yes	relative position estimation based on cameras

Table 4.1: Relevant aspects of additional related work in comparison to our method (1/4).

Paper	Observations	Communication	Dim.	HW	Remarks
Distance-Based Cycle-Free Persistent Formation: Global Convergence and Experimental Test With a Group of Quadcopters [KPA17]	relative positions	none	2D	yes	
Simultaneous Velocity and Position Estimation via Distance-only Measurements with Application to Multi-Agent System Control [JDA17]	interagent distances	none	2D	no	circular motion needed
Adaptive 3D Distance-Based Formation Control of Multiagent Systems with Unknown Leader Velocity and Coplanar Initial Positions [LXW18]	relative positions	none	3D	no	leader-follower structure
Distance-based Formation Stabilization and Flocking Control for Distributed Multi-agent Systems [WG18]	relative position and velocity	none	3D	no	
Fast Mutual Relative Localization of UAVs using Ultraviolet LED Markers [WSF18]	relative positions	none	3D	yes	relative position estimation based on cameras
Relative Pose Estimation using Range-only Measurements with Large Initial Uncertainty [Cha18]	relative positions	odometry data	2D	yes	
Formation Shape Control Based on Distance Measurements Using Lie Bracket Approximations [SS18]	interagent distances	none	3D	no	assumes rigid formation, needs sinusoidal perturbations
Distributed adaptive control for distance-based formation and flocking control of multi-agent systems [ZWG19]	relative positions	velocity data	2D	no	
An Integrated Localization and Control Framework for Multi-Agent Formation [CS19]	interagent distances	distances	2D	no	
3-D Relative Localization of Mobile Systems Using Distance-Only Measurements via Semidefinite Optimization [JAH20]	interagent distances	relative position information	3D	no	
Distance-Based Multiagent Formation Control With Energy Constraints Using SDRE [BS20]	relative positions	none	3D	no	

Table 4.2: Relevant aspects of additional related work in comparison to our method (2/4).

Paper	**Observations**	**Comm-unication**	**Dim.**	**HW**	**Remarks**
Relative navigation of autonomous GPS-degraded micro air vehicles [Whe20]	visual odometry	-	3D	yes	single drone only
Ultra-Wideband and Odometry-Based Cooperative Relative Localization With Application to Multi-UAV Formation Control [GLX20]	interagent distances	velocities and position estimates	2D	yes	
Varying-coefficient Based Distributed Formation Control of Multiple UAVs [Hep21]	positions	positions	3D	yes	
Autonomous Flight for Multi-UAV in GPS-Denied Environment [Lu23]	LiDAR odometry	relative pose	3D	yes	
Non-cooperative Stochastic Target Encirclement by Anti-synchronization Control via Range-only Measurement [Liu23]	interagent distances	state and distances	2D	yes	target encirclement by 2 agents
A Distributed Technique for Localization of Agent Formations From Relative Range Measurements [CCW12]	interagent distances	distances	2D	no	static agents
Multi-vehicle formation using range-only measurement [Sun07]	interagent distances	none	2D	no	leader-follower structure
Minimization of the effect of noisy measurements on localization of multi-agent autonomous formations [SFA09]	interagent distances	none	3D	no	anchor nodes with known positions
An Elasticity Inspired method for multi-AUVs formation control using range-only measurements [XL14]	interagent distances	none	2D	no	leader-follower structure
Coherent swarming of unmanned micro aerial vehicles with minimum computational and communication requirements [BS17]	relative positions	none	3D	yes	relative position estimation based on cameras
Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information [Liu17]	interagent distances and IMU data	state estimates	2D	yes	centralized approach
Relative Position Estimation in Multi-Agent Systems Using Attitude-Coupled Range Measurements [Sha21]	interagent distances	none	3D	no	multiple range tags per agent

Table 4.3: Relevant aspects of additional related work in comparison to our method (3/4).

Paper	**Observations**	**Communication**	**Dim.**	**HW**	**Remarks**
Distributed Formation Estimation Via Pairwise Distance Measurements [Zie21]	interagent distances and odometry	odometry data, parts of state	3D	no	
Relative Position Estimation Between Two UWB Devices With IMUs [Cos21]	interagent distances and odometry	odometry data	3D	yes	HW demonstration for 2 agents
Relative Localization of Mobile Robots with Multiple Ultra-WideBand Ranging Measurements [Cao21]	interagent distances and odometry	none	2D	yes	multiple range tags per agent
Flexible and Resource-Efficient Multi-Robot Collaborative Visual-Inertial-Range Localization [NNX22]	interagent distances and odometry	odometry data	3D	yes	HW demonstration for 2 agents
Relative Transformation Estimation Based on Fusion of Odometry and UWB Ranging Data [NX23]	interagent distances and odometry	odometry data	3D	yes	HW demonstration for 2 agents
Real-time Relative Pose Estimation for Muti-UAV Systems Using Odometry and UWB Measurements [QY24]	interagent distances and IMU data	pose estimates	3D	no	
Fast Swarming of UAVs in GNSS-Denied Feature-Poor Environments Without Explicit Communication [Hor24]	relative positions	none	3D	yes	relative position estimation based on cameras
High-Performance Relative Localization Based on Key-Node Seeking Considering Aerial Drags Using Range and Odometry Measurements [CLD24]	interagent distances and IMU data	state estimates	3D	yes	centralized approach
Reconfigurable Multi-Robot Formation via Ultra-Wideband Ranging in Unknown Environments [Guo24]	interagent distances and odometry	odometry data	2D	yes	leader-follower structure
Where are You? Unscented Particle Filter for Single Range Relative Pose Estimation in Unobservable Motion Using UWB and VIO [Dur24]	interagent distances and odometry	odometry data	3D	yes	HW demonstration for 2 agents in 2D only

Table 4.4: Relevant aspects of additional related work in comparison to our method (4/4).

Summarizing, we conclude that our work differs from these related works in at least one relevant aspect, as shown in the respective columns of Tables 4.1 to 4.4.

As already laid out in Section 4.2, some works use cameras, LiDAR sensors, and/or external systems to localize neighboring agents. In our setting, such sensors are not available. Regarding agent's observations, the relative positions of their neighboring agents in their own local coordinate system are provided in some works. This is substantially different than scalar distance measurements and provides considerably more information. Position estimation and control for 2D is studied in some works; however, it lacks generalization to 3D. In some works, agents are allowed to exchange messages containing their distance measurements, acceleration, odometry, internal states, and possibly further information.

Noteworthily, there are a number of very recent papers, including [NNX22; NX23; QY24; CLD24; Guo24; Dur24], addressing the problem of relative localization using UWB ranging and IMU and/or Visual Inertial Odometry (VIO) odometry data. Using odometry data and distance measurements, these works combine individual trajectories, e.g., by sliding-window optimization, to infer the relative positions in their own coordinate system. For this, it is needed to exchange odometry data between individual agents. In our work, there is no such information exchange.

CHAPTER 5

Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing

This work was published as:
Axel Brunnbauer*, Luigi Berducci*, Andreas Brandstätter*, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. „Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing“. In: *2022 International Conference on Robotics and Automation (ICRA)*. (* indicates equal contribution). 2022, pp. 7513–7520. DOI: 10.1109/ICRA46639.2022.9811650

By the following authors with their affiliations:
Axel Brunnbauer* at CPS, Technische Universität Wien (TU Wien), Austria
Luigi Berducci* at CPS, Technische Universität Wien (TU Wien), Austria
Andreas Brandstätter* at CPS, Technische Universität Wien (TU Wien), Austria
Mathias Lechner at Institute of Science and Technology Austria (IST Austria)
Ramin Hasani at CSAIL, Massachusetts Institute of Technology (MIT), MA, USA
Daniela Rus at CSAIL, Massachusetts Institute of Technology (MIT), MA, USA
Radu Grosu at CPS, Technische Universität Wien (TU Wien), Austria
(* Indicates authors with equal contribution)

A pre-print of this paper is available as:
Axel Brunnbauer, Luigi Berducci, Andreas Brandstätter, Mathias Lechner, Ramin M. Hasani, Daniela Rus, and Radu Grosu. „Model-based versus Model-free Deep Reinforcement Learning for Autonomous Racing Cars“. In: *CoRR* abs/2103.04909 (2021). arXiv: 2103.04909

Abstract

World models learn behaviors in a latent imagination space to enhance the sample-efficiency of deep Reinforcement Learning (RL) algorithms. While learning world models for high-dimensional observations (e.g., pixel inputs) has become practicable on standard RL benchmarks and some games, their effectiveness in real-world robotics applications has not been explored. In this paper, we investigate how such agents generalize to real-world autonomous vehicle control tasks, where advanced model-free deep RL algorithms fail. In particular, we set up a series of time-lap tasks for an F1TENTH racing robot, equipped with a high-dimensional LiDAR sensor, on a set of test tracks with a gradual increase in their complexity. In this continuous-control setting, we show that model-based agents capable of learning in imagination substantially outperform model-free agents with respect to performance, sample efficiency, successful task completion, and generalization. Moreover, we show that the generalization ability of model-based agents strongly depends on the choice of their *observation model*. We provide extensive empirical evidence for the effectiveness of world models provided with long enough memory horizons in sim2real tasks.

5.1 Introduction

Deploying deep Reinforcement Learning (RL) agents in the real world is difficult. This is because they require running a significantly large amount of episodes to obtain reasonable performance [SB18]. This performance is only tractable in simulation environments. Subsequently, the agents should also overcome the challenges of transferring learned dynamics from simulation to the real world.

In RL settings, it was shown that by learning representations of the state-space model from high-dimensional observations and using them as a predictive model to train policies in imagination, we gain performance, sample efficiency, and robustness [Haf20]. This world model algorithm [HS18] was called Dreamer, and it demonstrated great performance in learning long-horizon visual control tasks and Atari games [Haf21].

Providing a model of the world (state transition probability) to the agents (even though in simulation) improves sample efficiency and, in some cases, the performance of a deep RL agent [HS18; Haf19; Haf20; Haf21]. However, there are still open research questions to be investigated: For instance, how would world model compartments improve the sim2real performance of RL agents in real-world applications? Where is the boundary between the generalization capability of model-based algorithms such as Dreamer, compared to advanced model-free agents such as soft actor-critic [Haa18], distributional models [Bar18], and policy gradient [Sch17; Abd18], in sim2real applications?

In this paper, we aim to provide answers to the above questions. We construct a series of time-lap autonomous racing tasks with varying degrees of complexity for an F1TENTH robot. The task is to learn to drive autonomously directly from high-dimensional LiDAR inputs to successfully finish a lap without collisions.

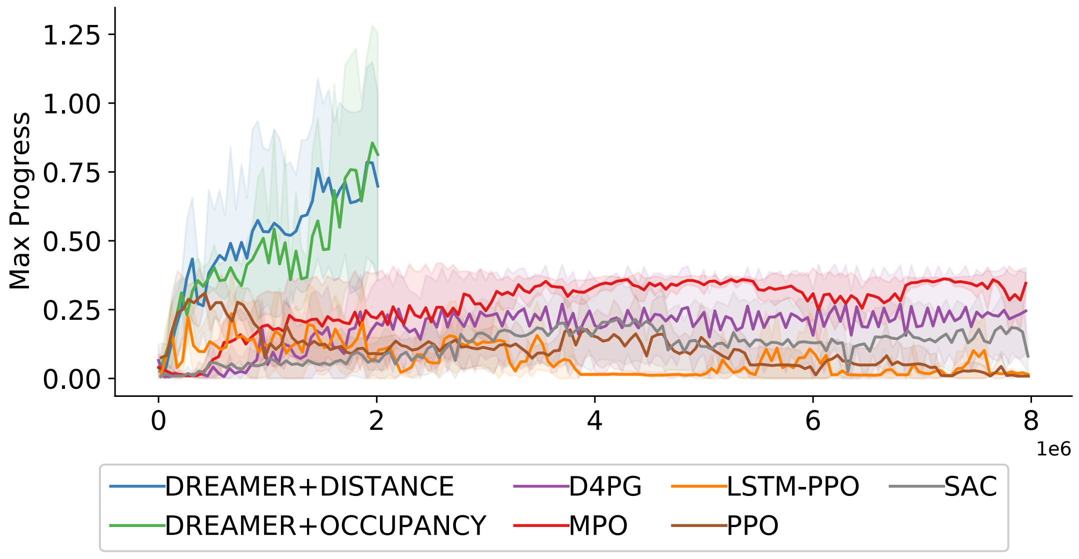


Figure 5.1: Model-based deep-RL (Dreamer) solves autonomous racing in complex tracks where all advanced model-free methods fail.

We observe that as the complexity of the map increases, model-free agents get stuck in similar local minima and cannot learn to complete the task. Contrarily, Dreamer can learn a proper state transition model, and given enough imagination horizon, solves time-lap tasks of arbitrary complexity (See Fig. 5.1). Moreover, we discover that model-based techniques demonstrate desirable transferability: Dreamer agents trained on a single map can generalize well to test tracks they have never seen before. In summary, our **main contributions** are as follows:

- We demonstrate the effectiveness of advanced model-based deep RL compared to model-free agents in the real-world application of autonomous racing.
- We show the transferability of advanced model-based deep RL agents to the real-world applications where model-free agents fail.
- We empirically show that the learning performance and generalization ability of Dreamer in sim2real applications highly depends on the choice of the observation model, its resulting state transition probability model, and its imagination horizon.

5.2 Problem Definition

We set out to design deep RL models that are able to learn to autonomously complete time-lap racing tasks. Considering the real-world conditions, we formalize the problem as a Partially-Observable Markov Decision Process (POMDP).

Notation and terminology.
A POMDP is defined as a tuple $(S, A, \Omega, \mathcal{O}, \mathcal{T}, \mathcal{R})$, where S, A, Ω are the sets of states, actions and observations, respectively. $\mathcal{O}$ and $\mathcal{T}$ denote stochastic observation and transition functions, and $\mathcal{R}$ is a deterministic reward function. The transition function $\mathcal{T}$ models the system dynamics and includes its uncertainty. It is defined as a stochastic function $\mathcal{T} : S \times A \times S \to [0, 1]$ which returns the transition probability between two states by applying actions. The observation function $\mathcal{O}$ models the system's perception, including its uncertainty. Its stochastic formulation $\mathcal{O} : S \times \Omega \to [0, 1]$ returns the probability of perceiving an observation in a given state. The reward function is deterministic $\mathcal{R} : S \times A \times S \to \mathbb{R}$, which returns the credit assigned to a transition. We discuss our reward shaping subsequently.

Racing-agent setup.
In autonomous racing, the car drives along a race track by controlling its actuators with continuous actions $\boldsymbol{a} = (F, \alpha)^T$, where F is the motor force applied to reach a fixed target velocity, and α is the steering angle. In this scenario, the observations are range measurements obtained by a Light Detection and Ranging (LiDAR) sensor with 1080 range measurements evenly distributed over a 270° field of view (see Fig. 5.2).

How do we aim to solve this racing objective?
Traditional control techniques model the state space as a set of continuous variables such as the car's pose, velocity, and acceleration. These quantities are usually estimated from observations by dedicated perception and filtering modules. State-space planning commonly uses sophisticated dynamics models for cars. This demands non-trivial system identification techniques. Contrarily, we aim to replace the entirety of such modules by learning an abstract state representation and transition model in a self-supervised manner by building upon recent advances of model-based RL [Haf20].

5.3 Related Work

We include works that are closely related to our contributions.

Model-free RL in robot control.
Robot control has been a playground for RL algorithms for decades [SB18; Has20]. Model-free algorithms [Sch17; Haa18; Bar18] achieved state-of-the-art results in various continuous control tasks, but their inherent sample inefficiency [Yu18] makes it challenging to apply them in real-world settings. Off-policy [Mni13; Haa18] methods reduce the sample complexity by reusing old experience to some degree. However, they might induce errors when deployed in a closed-loop with the real-world environment. Thus, model-free approaches are often trained in simulation before being deployed on real

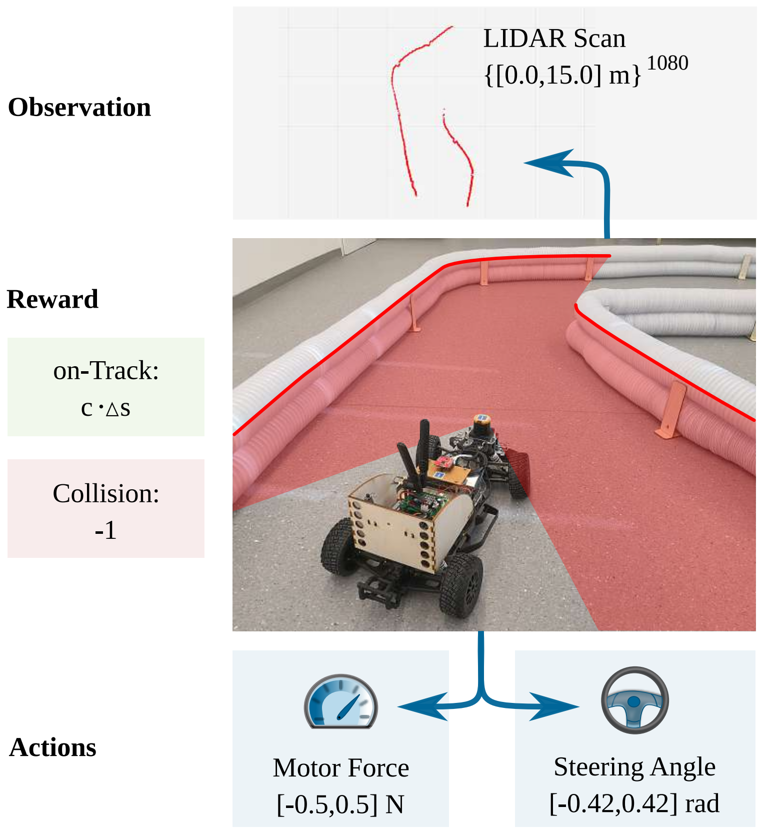


Figure 5.2: Racing-agent setup: observations, actions and reward.

robots. This leads to difficulties when reproducing learned behaviors in the real setting [Zhu20; ZQW20]. To overcome simulation mismatches, some approaches rely on domain randomization and subsequent fine-tuning. In order to learn more robust policies [Pen18], other works refined the physics simulator [Tan18; KMB20] or directly trained the policy on real-world robots [Haa19; Sin19].

Model-based RL in robot control.
Model-based approaches, on the other hand, can leverage their learned dynamics model by either planning or generating new training experience [Sut91]. Problems arise when no sufficiently accurate dynamics model can be learned from data. An early representative of model-based RL algorithms is PILCO [DR11] which learned a Gaussian Process from the system's states. Recent works on world models [HS18; Haf19; Haf20; Sch21] proved the feasibility of learning a dynamics model for POMDPs by using noisy high-dimensional observations instead of accurate states. They achieve better than state-of-the-art results on various simulated control tasks while being significantly more sample efficient. However, to the best of our knowledge, their application and challenges in real-world testbeds have been less attended, which we will explore in the present work.

Optimal control for autonomous racing.
Autonomous racing has been an active field of research in the control community. Traditional control does not solve the problem end-to-end but divides it into independent sub-problems: perception, planning, and control. Recent success in the context of Formula Student [Kab19b; Kab19a; And20] has been achieved by Model Predictive Control (MPC) and by engineering the perception pipeline using sensor fusion of LiDAR and RGB cameras. Several approaches plan an optimal trajectory [VT05; RNH15; TC13; Váz20]. However, these techniques usually require detailed global information (e.g., map) and an accurate dynamics model, not available in a POMDP setting, as in our case. Here, we use a fully automated process that learns an approximated dynamics model in an unsupervised fashion by only having access to LiDAR observations.

RL for autonomous racing.
A large body of research made use of camera images and adopt model-free methods [Jar18; RMD07; Ken19]. In [Fuc20], the authors used Soft Actor-Critic (SAC) to control a racing car in simulation by feeding the controller with state information and ad-hoc features about the road curvature. To deal with the complexity of continuous action spaces, a common technique is to discretize the domain, but this approach is not scalable. Another recurrent trend consists of combining MPC and deep RL [BB20b; Wil17]. However, MPC requires to efficiently perform sampling through the model to scale with respect to the time constraints. Conversely, the adoption of a policy network is more efficient and suitable for an online setting.

Imitation Learning.
Imitation learning is the process of learning observation-to-action mappings from supervised data [Sch99]. It allows for behavior cloning by data aggregation (Dagger) [RGB11] via supervised learning modes [Vor21; Lec20b] or by inverse RL [NR00]. Imitation learning has been further adapted to imperfect [Wu19] and incomplete demonstrations [SM19; Lec21]. In the context of end-to-end control of autonomous vehicles, learning from expert demonstrations is the dominant choice of algorithm [Lec20a; Lec19], because other RL methods would require efficient simulation platforms to achieve desirable performance [Ami20]. For autonomous racing, as we already have a sample efficient simulator available, we settle to use model-based RL algorithms over imitation learning.

5.4 Adapting Dreamer for Autonomous Racing

In this section, we revisit Dreamer [Haf20] and discuss its merits for autonomous racing. We present two observation models that we utilize to learn the latent-dynamics model. We then formulate a specialized reward function.

Introduction.
Dreamer is a model-based deep-RL algorithm that has recently achieved state-of-the-art performance in simulated standard RL benchmarks [Haf20]. Dreamer learns a model of the system dynamics from high-dimensional observations in an unsupervised fashion and uses it to produce latent-state sequences to train an agent using an actor-critic

algorithm [Haf20]. Dreamer implements the dynamics model as a Recurrent State-Space Model (RSSM) with both stochastic and deterministic components [Haf19]. It consists of four components, which are all implemented as deep neural networks [Haf20]:

Representation model:	$p_\theta(s_t \vert s_{t-1}, a_{t-1}, o_t)$	(5.1)
Observation model:	$q_\theta(o_t \vert s_t)$	(5.2)
Reward model:	$q_\theta(r_t \vert s_t)$	(5.3)
Transition model:	$q_\theta(s_t \vert s_{t-1}, a_{t-1})$	(5.4)

All components are jointly optimized to maximize the variational lower-bound as follows, where $D_{\mathrm{KL}}(P \,\|\, Q)$ is the Kullback–Leibler divergence of distributions P and Q:

$$\begin{aligned} &\mathcal{J}_{\mathcal{O}}^t = \ln q(o_t|s_t) \qquad \mathcal{J}_{\mathcal{R}}^t = \ln q(r_t|s_t) \\ &\mathcal{J}_{\mathcal{D}}^t = -\beta D_{\mathrm{KL}}\left(p(s_t|s_{t-1}, a_{t-1}, o_t) \,\|\, q(s_t|s_{t-1}, a_{t-1})\right) \\ &\mathcal{J}_{\mathcal{REC}} = \mathbb{E}_p\left[\sum_t \mathcal{J}_{\mathcal{O}}^t + \mathcal{J}_{\mathcal{R}}^t + \mathcal{J}_{\mathcal{D}}^t\right] \end{aligned} \tag{5.5}$$

In contrast to [Haf20], our representation model uses a Multilayer Perceptron (MLP) to encode the observations instead of convolutional layers. The reason for this design decision is that LiDAR scans are not as high-dimensional as RGB images, and further compression does not show better performance in our empirical evaluation.

Observation model.
This model reconstructs observations from latent states and is used only in the training phase to generate learning signals for the representation model. In this work, we propose two observation models for autonomous racing. One simply reconstructs the LiDAR scan, which is analogous to the original model, and the other is based on reconstructing occupancy maps. In Figure 5.3 we show snapshots of the simulation, the associated LiDAR observation, and reconstructions sampled from these models.

The *Distance reconstruction* model, implemented by MLPs, predicts a Gaussian distribution over the distance measurement of each LiDAR ray from latent states. We call it Dreamer-Distance. Here, the observation is both the representation model's input and the observation model's output.

Our novel *Occupancy reconstruction* model attempts to predict an occupancy grid map of the agent's surrounding based on its current state. Specifically, the model generates the parameters for a multivariate Bernoulli distribution which, for every pixel, models the probability of being occupied. The observation model is implemented using transposed convolutions to construct a 2D grayscale image and is trained by providing patches of the true occupancy grid map. We call the resulting agent Dreamer-Occupancy.

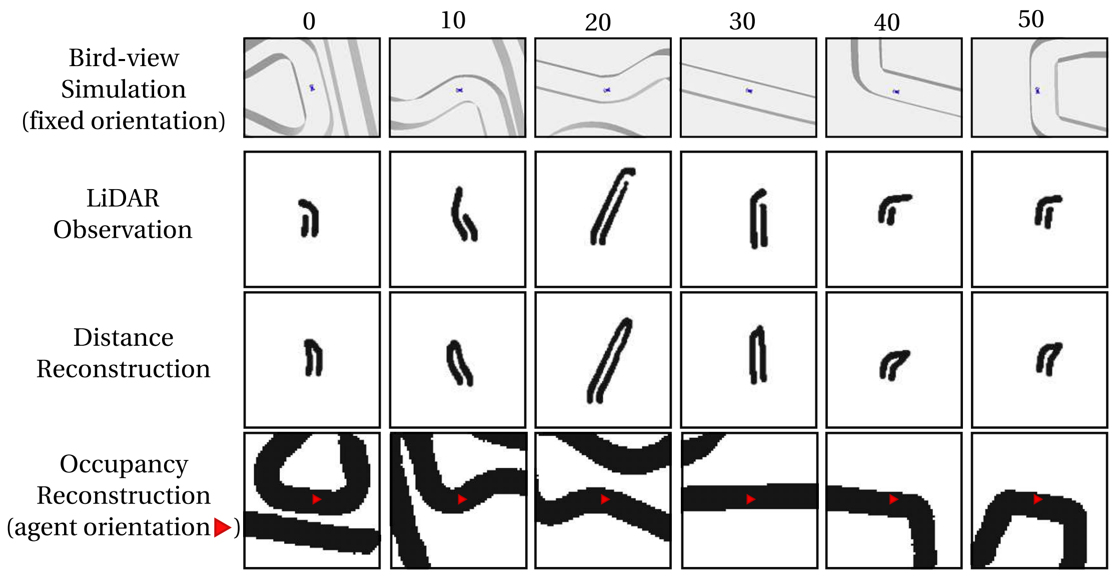

0
10
20
30
40
50
Bird-view Simulation (fixed orientation)
LiDAR Observation
Distance Reconstruction
Occupancy Reconstruction (agent orientation ▶)

Figure 5.3: Observation models and their reconstruction methods. Observations are in agent coordinates. **Row 1:** bird-view of the racecar in simulation, **Row 2:** LiDAR scan in racecar coordinates, **Row 3:** reconstructed LiDAR scan, **Row 4:** reconstructed local occupancy map

Actor-critic on latent imagination.
Having first learned a world model from observations, Dreamer then learns a policy purely on latent state-action sequences [Haf20]. This approach allows Dreamer to efficiently produce thousands of training sequences for RL without direct environment interaction, which results in more data-efficient learning. Starting from these imagined sequences, Dreamer uses an actor-critic algorithm to train the agent. The action model (policy) aims to predict the best action while the value model estimates the value for each latent state:

$$\text{Action model:} \qquad q_\phi(a_t|s_t) \tag{5.6}$$

$$\text{Value model:} \qquad q_\psi(v_t|s_t) \tag{5.7}$$

To trade off bias and variance in value estimation, Dreamer uses an exponentially weighted sum over different horizons [Haf20]. Moreover, the training process leverages the availability of a learned dynamics model by back-propagating through it. This leads to more effective gradient updates.

Reward shaping.
The reward design is critical for learning a policy. Sparse rewards over long episodes makes the learning problem challenging. Starting from a simple approach that rewards agents only when finished the race, we gradually refine the reward to provide a high-density learning signal. The reward signal we propose for this task is defined as

$$c * |p_t - p_{t-1}| = c * \Delta p_t, \tag{5.8}$$

where p_t denotes the progress that has been made on the track at time t and c is a constant scalar. When colliding with the wall or other objects, the agent receives a penalty and the episode terminates (see also Figure 5.2). The progress is computed by a distance transform applied to the gridmaps for each track. This yields a normalized progress estimate for each pixel on the map at a resolution of 5cm per pixel. Therefore, the progress value can be easily queried from these maps given the current pose of the car. In Figure 5.4 (bottom) we show the precomputed progress maps for each of our tracks. The progress value ranges from 0 (light) to 1 (dark), where a progress of 1 corresponds to one full lap.

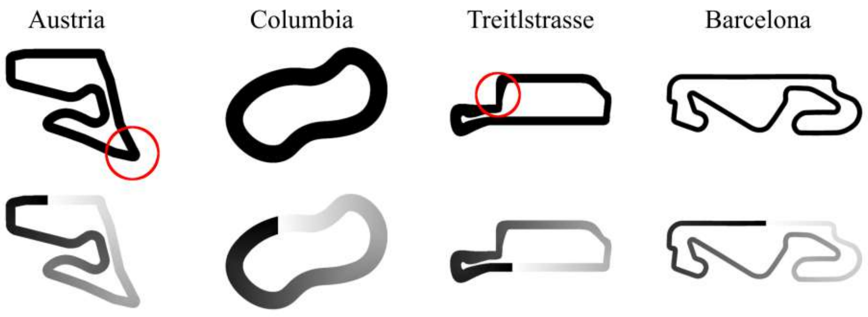


Figure 5.4: Top-view of the race tracks and relative Progress Maps adopted for reward design. Austria (length = $79.45m$), Columbia ($61.2m$), Treitlstrasse ($51.65m$) are used for training, Barcelona ($201m$) is used only for evaluation. Red circles indicate difficult parts of the track. The red circles show where all model-free agents fail to improve their performance.

Track	**Min. Width**	**Length**	**Min. Radius**
Austria	1.86m	79.45m	**2.78m**
Columbia	3.53m	61.20m	7.68m
Treitlstrasse	0.89m	51.65m	3.55m
Barcelona	1.86m	201.00m	2.98m

Table 5.1: Track characteristics.

5.5 Experiments

Here, we describe our experiments and evaluate the performance of model-free versus model-based RL algorithms.

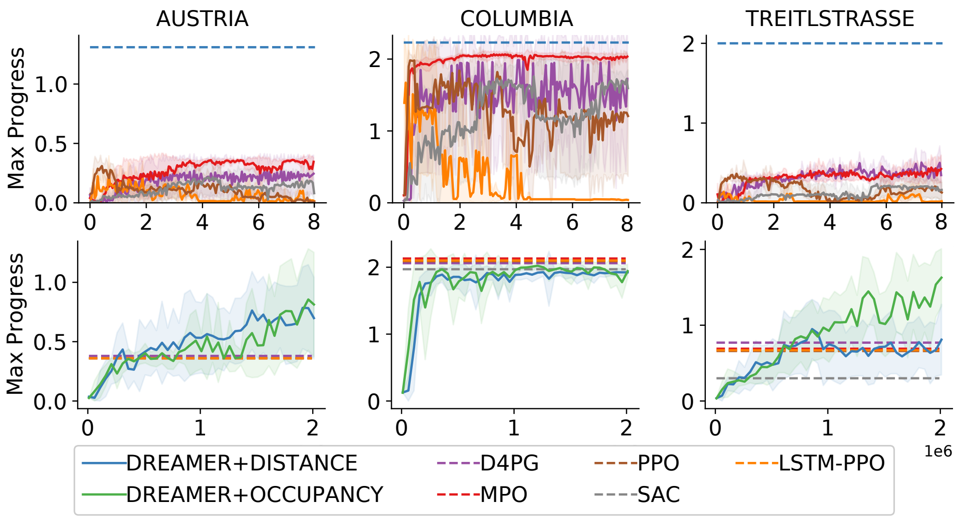
5.5.1 Experimental Setup

For our experiments, we train the agents in simulation before we transfer them to real cars. In the following, we provide an overview of the simulation and real-world setup as well as the training process.

Simulation environment.
To simulate a car model, we use the open-source physics engine PyBullet [CB19]. The car model is a rigid body system based on the URDF model in [BB20a]. Our training environment can simulate a broad set of sensory inputs, such as LiDAR sensors, RGB cameras, and odometry. The model is actuated by applying force to the steering and acceleration joints.

Training pipeline.
During training in simulation, we place the agent randomly on the track at the beginning of an episode. Each training episode has a maximum length of 2 000 timesteps, resulting in 20 seconds of real-time experience. Agents learn to directly maximize the progress covered in a small, predefined time interval. Observations and actions are normalized. To account for latency experienced during testing and to increase the effectiveness of actions, we repeat each action multiple times. We regularly evaluate the agent by placing it on a fixed starting position for each track and let it run for at most 4 000 timesteps (i.e., 40 seconds) and average the maximum progress reached over five consecutive trials. Dreamer is trained for 2 million timesteps. The model-free baselines are trained for 8 million timesteps.

Hardware setup.
The hardware platform is the F1TENTH race series [OKe20]. It consists of an off-the-shelf model race car chassis, with a Traxxas Velineon 3351R brushless DC electric motor, which is driven by a VESC 6 MkIV Electronic Speed Controller (ESC). The laser range measurements are produced by a Hokuyo UST-10LX LiDAR sensor. On-board computation and control tasks are performed on an NVIDIA Jetson TX2. The on-board system runs Ubuntu 18.04 as the base operating system and hosts the core services for the Robot Operating System (ROS) stack [Sta]. To run Dreamer agents on our hardware, we use Docker. A ROS interface node is used to translate observation and action messages. The motor force commands are processed by integration to get the desired speed values. We added an adaptive low pass filter for the steering commands to protect the servo from high frequent steering operations.



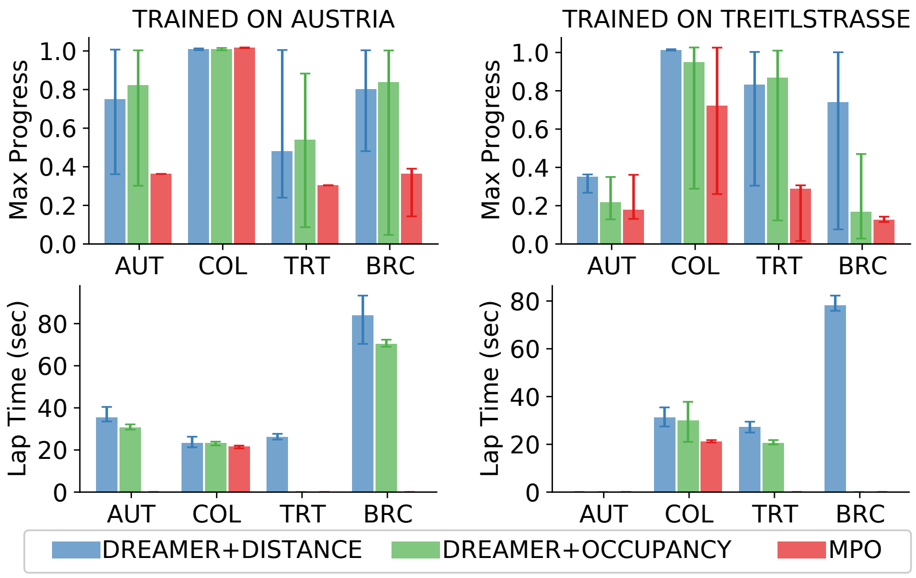


Figure 5.5: Top: Learning curves of model-free methods (*top row*) over $8M$ steps and Dreamers (*bottom row*) over $2M$ steps. The dashed lines report the maximum performance obtained by the other algorithms as baselines. Performance averages over 5 runs. **Bottom:** Maximum progress and lap time of trained models over different tracks in simulation. The bars show the result averaged over 10 episodes on each track. The delimiters show the minimum and maximum achieved. For Lap-Time results, we consider the best episode that finished one full lap.

Track difficulty.
To compare the difficulty of the tracks, we classified them according to several characteristics (as also discussed in [Bra08] and [LL01]). In particular, we measure the minimal track width, the track length (shortest path), and its minimum curve radius. For curves, we measure the radius of the largest circle that tangentially touches the outside of the curve and also touches the inside of the curve. In Table 5.1 we summarize the characteristics of the tracks.

5.5.2 Experimental Evaluation

In this section, we present the baseline algorithms and discuss their performance in various racing scenarios. We conduct three different experiments: I) Evaluation of the learning curves of Dreamer and model-free algorithms in simulation, II) Evaluation of the generalization ability by testing the trained models in simulation, and III) Evaluation of the transferability on our real testing platform.

Model-free baselines.
We compare the performance of the Dreamer agent against the following advanced model-free baselines: D4PG, an enhanced version of DDPG [Bar18], MPO, a stable off-policy algorithm [Abd18], SAC, an off-policy actor-critic algorithm with less sensitivity to hyperparameters [Haa19], PPO, an on-policy algorithm [Sch17], and PPO-LSTM, the recurrent version of PPO using long short-term memory [HS97]. PPO-LSTM is chosen as a baseline since policies built by recurrent networks demonstrate remarkable performance in learning to control [Has21; Has19; LH20; Has22]. We tuned the hyperparameters for each baseline algorithm with Optuna [Aki19].

Learning performance and sample efficiency.
In Figure 5.5 (left) we show the learning curves of the model-free and model-based algorithms in three different tracks. Except for the simple track Columbia, all advanced model-free methods fail to perform one lap in more complex maps successfully. On the other hand, Dreamer efficiently learns to complete the tasks regardless of the degrees of complexity of the tracks.

As shown in Figure 5.5, the performance of the Dreamer agents equipped with distance and occupancy reconstruction models are comparable in Austria and Columbia. However, the experiments on Treitlstrasse show better performance with occupancy-map reconstruction, with which the agent can almost complete two full laps over the evaluation-time window. This experiment suggests that the occupancy-map reconstruction speeds up the training process. However, it biases the learning process on the track on which it was trained. This affects the generalization capabilities of the policy, as illustrated in the following experiment.

Performance on unseen tracks.
In this experiment, we evaluate the cross-track generalizability of the learned policies and demonstrate the domain-adaptation skills of Dreamer. We trained polices on a single, fixed track, i.e., Austria (AUT) and Treitlstrasse (TRT) and reported their evaluation

performance on other unseen tracks: Columbia (COL) and Barcelona (BRC). We compare Dreamer to the best-performing model-free policy, MPO, and to a Follow-the-gap (FTG) agent, which is a LiDAR-based reactive method that was successfully applied in past F1TENTH competitions [SG12].

Considering the learning methods shown in Figure 5.5 (right, top row), all the algorithms can complete the simple track, COL. However, only Dreamer can generalize the racing task and transfer the learned skills to other complex tracks. Generally, policies trained in AUT achieve better performance as they can complete at least one lap in each other track. Conversely, even though TRT contains challenging turns, the trained agents perform poorly on unseen complex tracks (AUT). The reason might be that most turns in TRT have the same direction, and consequently, the trained policy cannot generalize to altered scenarios.

Moreover, we observed that the Dreamer-Occupancy agent shows lower generalization skills compared to Dreamer-Distance. The learned policy results in a more aggressive strategy, presenting a lower lap-time as shown in Figure 5.5 (right, bottom row). However, the car often gets dangerously close to the track walls. This results in unsafe behavior that increases the probability of collisions. Comparing the performance of Dreamer with the adaptive FTG, we observe similar lap-times on all the tracks. However, the predictable behavior of FTG results in a more stable controller. This is not surprising considering that FTG is a programmed algorithm, and its parameters have been carefully tuned to drive on the most challenging track, AUT. In conclusion, reconstructing the occupancy area allows the agent *overfit* to the track it was trained on. This speeds up learning, but hurts robustness to domain changes and thus worsens generalization.

Influence of imagination horizon.
Figure 5.6 shows that as we increase the imagination horizon for Dreamer, it performs better. While this increase helps generate long-term trajectories in latent space as training data, we observe that a further increase leads to a drop in performance in some of the experiments. We suspect that this drop is caused by compounding model errors for longer horizons.

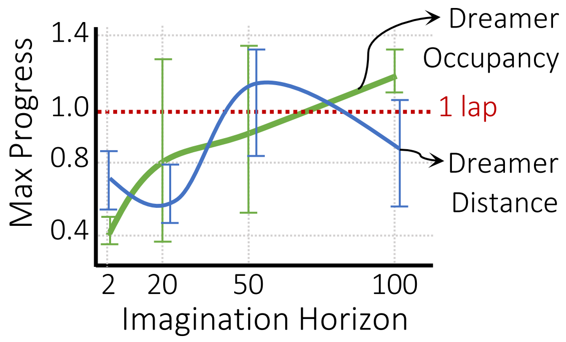


Figure 5.6: Dreamer's performance vs. imagination horizon. Batch length =50, and action repeat = 4. n=5.

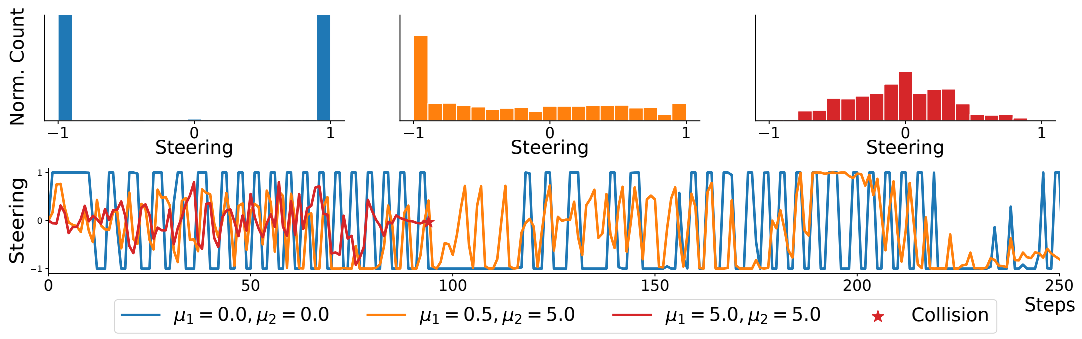


Figure 5.7: Impact of action regularization on TRT. The first row reports the steering command's distributions under various regularization weights. The second row reports the continuous command over a single simulation snapshot. n=5

Action Regularization.
Additionally, we observed that the driving behavior of the car resembles an on-off control law, resulting in unnecessarily curvy trajectories. While this type of behavior is expected for acceleration control [Lib11], it is counterproductive to perform unnecessary steering commands. We experimented with different regularization techniques for continuous control to minimize the steering effort and enforce temporal smoothness, similar to the approaches in [Mys21] and [Sey21]. The resulting policy showed smoother steering commands (see Figure 5.7) but did not show superior performance compared to the one trained without regularization. The objective for the regularized action model is defined as

$$\mathbb{E}_{p_\theta, q_\phi}\left[\sum_{t=0}^{H} \gamma^t (V_\lambda(s_t) - \mu_1 ||a_t||_2^2 - \mu_2 ||a_t - a_{t-1}||_2^2)\right],$$

where $V_\lambda(s_t)$ denotes the original value estimation term, μ_1 and μ_2 are the control effort and the temporal smoothing coefficients, respectively.

Sim2Real transfer.
In this experiment, we evaluate Dreamer policies with respect to their sim2real transferability. We test trained dreamers deployed on the car in a physical test track. A video[1] demonstrating the driving performance of the Dreamer agent is supplied with the submission. It presents the Dreamer agent completing a full lap in TRT in the forward direction. Then, to evaluate its generalization capabilities, we tested the agent in reversed direction. We observe that even if the agent was not trained in the reversed direction and not directly on the TRT track, the agent can complete a successful lap. Finally, we placed two obstacles after the most challenging turn and ran the agent in this configuration. The Dreamer agents can complete the lap and securely avoid obstacles.

[1]The video can be viewed at: `https://youtu.be/8ofWVLArZJQ`

5.6 Conclusions

We show that Dreamer, a model-based deep RL algorithm, outperforms several other model-free RL algorithms in simulation. Furthermore, we empirically demonstrate that Dreamer is able to successfully transfer the policy that it learned in simulation to a real-world test environment without the use of explicit domain randomization techniques. Ultimately, we show how observation models and model horizon affect generalization and domain adaptation of learned policies and that model-based agents can enable robust autonomy in real-world settings.

Why Dreamer?
Dreamer is a comparably sample-efficient, high-performance deep RL algorithm. Its learned state space model can not only be used for policy learning, but also for trajectory planning approaches [Haf19].

How does the sim2real gap influence Dreamer's performance?
The discrepancy between simulation and reality is and remains one of the main challenges in RL. However, Dreamer's latent state-space model is robust enough to transfer the learned policy to the real world. This result is key for the deployment of RL algorithms in the real world.

What are the limitations of this approach?
We observed that our learned policies often resemble bang-bang control [Sey21], which is often not desirable in real-world robotics applications. To avoid this, the objective function of the learning problem has to be carefully designed. Our experiments mitigated its emergence by regularizing the original actor's loss with action penalties to discourage sharp changes between extreme values. This method relaxed the bang-bang behavior observed in the obtained policies. Furthermore, the lack of structure in the latent space of the world model makes it hard to interpret and is subject to ongoing research. All code, data and appendix are available at: `https://github.com/CPS-TUWien/racing_dreamer`.

5.7 Acknowledgments

Luigi Berducci was supported by the Doctoral College Resilient Embedded Systems. Mathias Lechner was supported in part by the ERC-2020-AdG 101020093 and the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). Ramin Hasani and Daniela Rus were supported by The Boeing Company and the Office of Naval Research (ONR) Grant N00014-18-1-2830. Radu Grosu was partially supported by the Horizon-2020 ECSEL Project grant No. 783163 (iDev40) and Axel Brunnbauer by FFG Project ADEX.

© 2022 IEEE. Reprinted, with permission, from Axel Brunnbauer*, Luigi Berducci*, Andreas Brandstätter*, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. „Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing“. In: *2022 International Conference on Robotics and Automation (ICRA)*. (* indicates equal contribution). 2022, pp. 7513–7520. doi: `10.1109/ICRA46639.2022.9811650`.

5.8 Author contribution details

For this paper, the co-first authors declare to have contributed in the way, as described hereafter.

Axel Brunnbauer:
Responsible for the conceptualization and adaptation of the model-based RL algorithm to the racing task and the implementation of the observation reconstruction models for LiDAR scans and RGB images. Conceptualized the progress-based objective function. Designed and implemented the multi-agent racecar simulation environment. Implemented and evaluated baseline algorithms in the performance experiments. Conducted experiments to evaluate the sensitivity to the model horizon for the model-based approach.

Luigi Berducci:
Responsible for the conceptualization and design of the models in the model-based RL algorithm, in particular the novel reconstruction model based on occupancy-map for racing task. Design and implementation of the regularized actor loss function, including temporal and spatial regularization terms to make the actor amenable to sim-to-real transfer. Conducted experiments to assess the training performance of the resulting model-based RL algorithm. Conducted offline evaluation of the proposed model-based RL algorithm and model-free baselines over unseen tracks.

Andreas Brandstätter:
Responsible for the hardware setup, including all race track infrastructure with track-barriers, network and camera setup, to conduct the hardware experiments. Building and operating the race car hardware platform, consisting of chassis, sensors, actuators, and embedded computing platform. All software on the model race car, required to operate the car and to execute the same inference node, as in simulation: Linux operating system, drivers, and the ROS stack. Conducting the hardware experiments in the lecture hall, recording and evaluating the associated results to demonstrate the sim2real capability of the proposed RL algorithm.

5.9 Autonomous Racing in the Context of Antagonistic MAS

This section was not published in [Bru22], but comprises additional reasoning and further description about this research in the context of antagonistic Multi-agent Systems (MAS).

Our work is based on the hardware platform of the F1TENTH race series [OKe20]. Therefore, we also refer to this race series to analyze the various levels of complexity and involvement of opposing agents, representing various types of MAS scenarios. In principle, this also applies to other similar race series, such as *Indy Autonomous Challenge* [Ind24] or *Roborace* [FIA24]. We classify the abilities of an agent in the following levels, where one might build upon another. In this classification, (A) and (B) can be considered basic abilities, and (C) to (E) are abilities for actual MAS.

(A) A single agent is operating on the race track with a static map. There are no other agents and no obstacles, which are not represented on the map.

  (A.1) The agent shall be able to continue a single lap without crashing into the race track barriers. This can be considered a minimum requirement for any functional agent.

  (A.2) The agent continues multiple consecutive laps without crashing into the race track barriers. This is a metric of the reliability of an agent. Note that in simulation, if there is no noise, this might be of limited significance, but on real hardware, even small perturbations and sensor noise might make a difference between multiple laps.

  (A.3) Laps should be completed as fast as possible. This metric might be the most obvious one for any racing scenario. Over multiple laps, it might be relevant to assess only the fastest lap, but alternatively also to consider the full distribution of all lap times.

(B) Avoid static obstacles, which are not represented on the map. This ensures the agent is actually able to react to its environment, despite only relying on map-based pre-programmed information.

(C) Evading a single opponent that behaves rationally w.r.t. to the racing scenario. This means it drives in the defined direction of the race track and follows any reasonable race line. So, the agent can expect the opponent to not exhibit deliberate blocking, wiggling, or abrupt stopping.

  (C.1) The speed of the opponent is considerably lower than the agent's own speed.

  (C.2) The opponent is of similar speed as the agent, or even faster.

(D) Evade an arbitrary number of opponents that behave rationally w.r.t. to the racing scenario. Opponents are assumed to be independent of each other, derive their actions independently, and do not share a common collaboration tactic.

(E) Evade one or more overly aggressive opponents. These might exhibit erratic or malicious behavior w.r.t. to the racing scenario. This includes deliberate blocking, wiggling, or even abrupt stopping. We still can assume the absence of wrong-way drivers, as this contradicts their own objective of completing laps in the defined racing direction. However, in most racing series, such overly aggressive behavior is explicitly forbidden.

In our work, we successfully demonstrate that our method is able to solve (A.1) on maps AUT, TRT, COL, and BRC. The other model-free methods, that we compare in our work, fail to successfully complete a single lap on some of these maps. When testing simulation experiments with multiple laps (A.2), we did not notice any significant problems and, therefore, conclude that our trained agent is also capable of finishing a high number of consecutive laps. In the hardware experiments, we performed successful experiments with multiple consecutive laps. However, as the focus of this work is not on this aspect, we did not run extensive experiments focusing on this metric. Regarding achieving fast lap times (A.3), there is definitely a need for further research to optimize our trained policies for that aspect. We compare our method with a FTG agent and got comparable lap times in simulation and hardware experiments. FTG is an engineered reactive method that is well-known and regularly applied in F1TENTH competitions [SG12] by a multitude of teams. However, there exist more advanced methods, such as MPC-based controllers [Bec23], resulting in significantly faster lap times. In our hardware experiments, we successfully demonstrate the ability to avoid static obstacles (B) by placing boxes on the track. Such an example of an obstacle avoidance maneuver is also included in the accompanying video[1] (in Figure 5.8 there is a snapshot of the video showing the relevant scene with obstacles).

While participating in a number of international F1TENTH races since 2020 with our team *Scuderia Segfault* [Res24], we carefully observed various racing strategies and algorithms employed by different teams. We notice that the vast majority of algorithms in this race series do not actively consider other agents but rather treat them as bare obstacles to avoid. This leads to our claim that in these cases, it should rather be considered as "passing by at another agent", than calling it "overtaking". This is especially true for the case where the opponent car is considerably slower than the overtaking one (C.1). Even though not directly demonstrated, we, therefore, reason that based on the results for obstacle avoidance (B), our proposed method would also be able to deal with slower opponents on the track. In cases where the opponent is of comparable speed or even faster (C.2) or when there are more opponents simultaneously on track (D), it needs further analysis to draw a final conclusion, but we consider it likely that our method is also capable of these scenarios. As aggressive opponents are prohibited by the rules of the race series, we do not further consider these scenarios (E).

[1]The video can be viewed at: `https://youtu.be/8ofWVLArZJQ`

As our method's observations are solely the LiDAR scans without any predefined trajectories, it exhibits a fully reactive behavior. In Table 5.2 we compare the abilities of our method with a FTG based reactive agent. Conclusions of FTG based agent are drawn on observations from participating in a number of international F1TENTH races between 2020 and 2024. Even though not primarily developed and trained for MAS scenarios, our method is still applicable for the aforementioned classes (C) to (D) to at least some extent. Further work could focus on testing and verifying such a trained policy on those scenarios and also developing improved policies targeting these MAS abilities.

	Our method	**FTG based agent**	**Remarks**
Basic ability (A.1)	✓ demonstrated	✓ yes	
Basic ability (A.2)	✓ demonstrated	✓ yes	
Basic ability (A.3)	∼	∼	there exist superior controllers
Basic ability (B)	✓ demonstrated	✓ yes	
MAS ability (C.1)	✓ by reasoning	✓ yes	
MAS ability (C.2)	∼ likely	∼ partially	
MAS ability (D)	∼ likely	∼ partially	
MAS ability (E)	–	–	prohibited by the rules

Table 5.2: Comparison of agent's abilities. Conclusions of FTG based agent are drawn on observations from participating in a number of international F1TENTH races between the years 2020 and 2024.

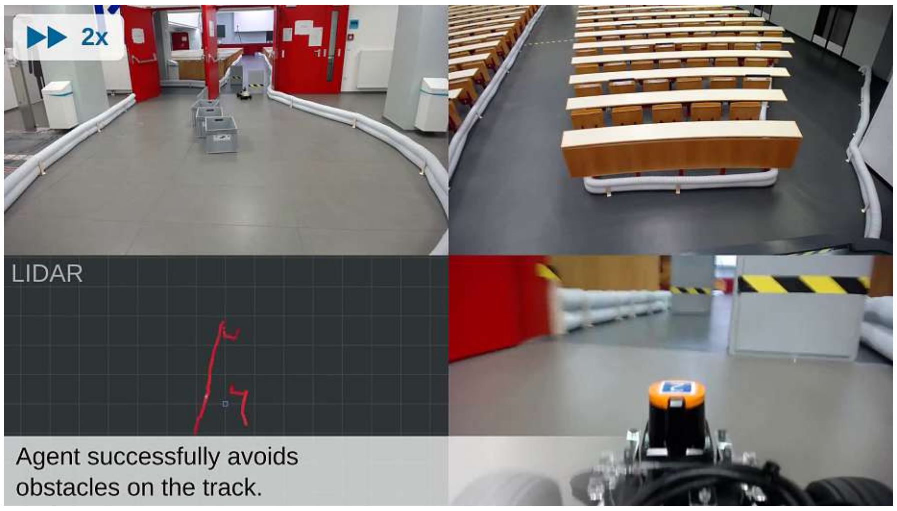


Figure 5.8: Snapshot of the video[1] accompanying the paper. We demonstrate avoidance of static obstacles using two boxes that are placed on the racetrack.

CHAPTER 6

Summary and Conclusions

In this work, we introduce methods to coordinate and control robotic agents in both antagonistic and collaborative Multi-agent Systems (MAS). For each of these scenarios, we use a three-pronged approach consisting of theoretical analysis and reasoning, implementation and simulation, and hardware experiments.

For the autonomous racing task, we use machine learning, more specifically Reinforcement Learning (RL), to drive a Partially-Observable Markov Decision Process (POMDP) agent in a timed racing scenario. We show that the model-based deep RL algorithm *Dreamer* outperforms a number of other model-free RL algorithms in simulation for this task. Using an actual model race car, we empirically demonstrate that this control method is able to successfully transfer the learned policy from simulation to a real-world test environment. We also show how observation models and model horizon affect generalization and domain adaptation of learned policies and that model-based agents can enable robust autonomy in real-world settings. For developing robot control methods, the sim2real transfer gap is an important issue. This is especially challenging in RL, when using simulated environments to train a policy. We can demonstrate that in our scenario, the learned policy is robust enough to be transferred to the real world. Despite these interesting findings and observations, we notice that the ultimate performance of our control method, w.r.t. to the relevant metric for racing, which is lap time, still lags behind traditional control methods by a large margin. Developing competitive control methods for racing in this scenario, therefore, requires additional research.

Our collaborative MAS consists of a group of drones, more specifically quadcopters, that are tasked to form and maintain a flock formation. We first introduce the concept of *Spatial Predictive Control (SPC)*, and demonstrate its utility on the drone flocking problem in simulation and using real hardware drones. SPC is fully distributed and is based only on the position of the individual drone itself and on those of neighboring drones. This information is used to compute the gradient of the local cost function and to perform a spatial prediction for the best next action. In contrast to Model

Predictive Control (MPC), SPC has a low computational cost and does not require a dynamic model of the plant. We give an experimental evaluation of SPC on the drone flocking problem using a physics engine with a detailed drone model. These results demonstrate SPC's ability to form and maintain a flock, avoid obstacles, and move the flock to multiple target locations. Hardware experiments demonstrate SPC's robustness against a potential sim-to-real transfer gap and its capability to perform properly in the presence of significant sensor noise and the extra latency of positional and control signals. We also experimentally compare SPC with a related approach based on a Potential Field Controller (PFC) and show the superiority of our proposed method. SPC is a general technique for designing *middle-level controllers* sandwiched between high-level planners and Positional Low-Level Controllers (PLLCs) that often come integrated with the hardware. In this research, we limit the observations to only provide positions of neighboring agents but do not allow measurement or information exchange of velocities, acceleration, and/or actions. However, there might exist scenarios, where even positional observations of agents are not possible. This leads to a follow-up work which is also part of this thesis.

We introduce *Distributed Distance-based Control (DDC)* as an MAS-formation-control approach, which is fully distributed, and solely based on scalar distance measurements and local position estimation. To validate DDC, we perform simulation experiments using a physics engine with a detailed drone model and also compare it to a variant of a controller with the same observations. Our DDC clearly shows superior performance in terms of all relevant quality metrics. We demonstrate DDC on drone-flocking with target-seeking using real hardware drones, thereby showing its fitness to form and maintain a flock, even though scalar-distance measurements are extremely limited observations. To the best of our knowledge, we are the first to demonstrate such a controller on aerial MAS and perform experiments with hardware drones. While DDC is able to satisfy the specified performance metrics, future work could further improve this controller and focus on ensuring that the flock reaches the target location by a given deadline and extending it with obstacle avoidance capabilities.

In this thesis, we deal with coordination and control for different types of MAS in confined environments. First, for each of these scenarios, we give a theoretical analysis and reasoning of the respective problem statement in order to propose an applicable control method or technique. Second, this control method is implemented as actual software and extensively validated using a suitable simulation environment. Last, the implementation is also tested on real hardware robotic MAS to ultimately demonstrate the usefulness and fitness of the proposed control method.

List of Figures

List of Tables

List of Acronyms

2D	two-dimensional
3D	three-dimensional
AUV	Autonomous Underwater Vehicle
CPS	Cyber-Physical Systems
DDC	Distributed Distance-based Control
ESC	Electronic Speed Controller
FTG	Follow-the-gap
GNSS	Global-Navigation Satellite System
GPS	Global Positioning System
IIR	Infinite Impulse Response
IMU	Inertial Measurement Unit
LED	Light-Emitting Diode
LiDAR	Light Detection and Ranging
MAS	Multi-agent Systems
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perceptron
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
PFC	Potential Field Controller

PID	Proportional-Integral-Derivative
PLLC	Positional Low-Level Controller
POMDP	Partially-Observable Markov Decision Process
RGB	red, green and blue
RL	Reinforcement Learning
ROS	Robot Operating System
RSSM	Recurrent State-Space Model
SAC	Soft Actor-Critic
SAR	Search-and-Rescue
SPC	Spatial Predictive Control
UAV	Unmanned Aerial Vehicle
UWB	Ultra-Wideband
VIO	Visual Inertial Odometry

Bibliography

[Abd18] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. „Maximum a Posteriori Policy Optimisation“. In: *International Conference on Learning Representations*. 2018. URL: `https://openreview.net/forum?id=S1ANxQW0b`.

[ADR19] Kaveh Azadeh, René De Koster, and Debjit Roy. „Robotized and Automated Warehouse Systems: Review and Recent Developments“. In: *Transportation Science* 53.4 (2019), pp. 917–945. DOI: `10.1287/trsc.2018.0873`.

[Agn20] Abhijeet Agnihotri, Matthew O'Kelly, Rahul Mangharam, and Houssam Abbas. „Teaching Autonomous Systems at 1/10th-scale: Design of the F1/10 Racecar, Simulators and Curriculum“. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 657–663. ISBN: 9781450367936. DOI: `10.1145/3328778.3366796`.

[Aki19] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: `1907.10902 [cs.LG]`.

[Ami20] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. „Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation“. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1143–1150. DOI: `10.1109/LRA.2020.2966414`.

[And20] L. Andresen, A. Brandemuehl, A. Honger, B. Kuan, N. Vödisch, H. Blum, V. Reijgwart, L. Bernreiter, L. Schaupp, J. J. Chung, M. Burki, M. R. Oswald, R. Siegwart, and A. Gawel. „Accurate Mapping and Planning for Autonomous Racing“. In: *2020 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4743–4749. DOI: `10.1109/IROS45743.2020.9341702`.

[Ara16] Miguel Aranda, Gonzalo Lopez-Nicolas, Carlos Sagues, and Michael M. Zavlanos. „Distributed Formation Stabilization Using Relative Position Measurements in Local Coordinates“. In: *IEEE Transactions on Automatic Control* 61.12 (Dec. 2016), pp. 3925–3935. ISSN: 0018-9286, 1558-2523. DOI: `10.1109/TAC.2016.2527719`.

[Att] Alessandro Attanasi, Andrea Cavagna, Lorenzo Del Castello, Irene Giardina, Asja Jelic, Stefania Melillo, Leonardo Parisi, Oliver Pohl, Edward Shen, and Massimiliano Viale. „Emergence of collective changes in travel direction of starling flocks from individual birds' fluctuations". In: *Journal of the Royal Society* 12 (108). DOI: 10.1098/rsif.2015.0319.

[AY11] Brian D. O. Anderson and Changbin Yu. „Range-only sensing for formation shape control and easy sensor network localization". In: *2011 Chinese Control and Decision Conference (CCDC)*. 2011 23rd Chinese Control and Decision Conference (CCDC). Mianyang, China: IEEE, May 2011, pp. 3310–3315. ISBN: 978-1-4244-8737-0. DOI: 10.1109/CCDC.2011.5968829.

[AYS14] Saleh Alaliyat, Harald Yndestad, and Filippo Sanfilippo. „Optimisation Of Boids Swarm Model Based On Genetic Algorithm And Particle Swarm Optimisation Algorithm (Comparative Study)". In: May 2014. DOI: 10.7148/2014-0643.

[Bal08] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic. „Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study". In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1232–1237. DOI: 10.1073/pnas.0711437105.

[Bar18] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. „Distributed Distributional Deterministic Policy Gradients". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SyZipzbCb.

[Bar20] Rhaian J. F. Barros, Jorge L. P. Silva Filho, João V. S. Neto, and Tiago P. Nascimento. „An Open-Design Warehouse Mobile Robot". In: *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. 2020, pp. 1–6. DOI: 10.1109/LARS/SBR/WRE51543.2020.9307137.

[BB20a] Varundev Suresh Babu and Madhur Behl. „f1tenth.dev - An Open-source ROS based F1/10 Autonomous Racing Simulator". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. 2020, pp. 1614–1620. DOI: 10.1109/CASE48305.2020.9216949.

[BB20b] G. Bellegarda and K. Byl. „An Online Training Method for Augmenting MPC with Deep Reinforcement Learning". In: *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020, pp. 5453–5459. DOI: 10.1109/IROS45743.2020.9341021.

[Bec23] Jonathan Becker, Nadine Imholz, Luca Schwarzenbach, Edoardo Ghignone, Nicolas Baumann, and Michele Magno. „Model- and Acceleration-based Pursuit Controller for High-Performance Autonomous Racing“. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 5276–5283. DOI: 10.1109/ICRA48891.2023.10161472.

[Bet22] Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. „Autonomous vehicles on the edge: A survey on autonomous vehicle racing“. In: *IEEE Open J. Intell. Transp. Syst.* 3 (2022), pp. 458–488. DOI: 10.1109/ojits.2022.3181510.

[Bit21] Bitcraze. *Loco Positioning system*. 2021. URL: https://www.bitcraze.io/documentation/system/positioning/loco-positioning-system/.

[Bla21] Black-i Robotics. *Landshark UGV*. 2021. URL: https://www.blackirobotics.com/landshark-ugv/ (visited on 09/14/2023).

[BLM08] Jur van den Berg, Ming Lin, and Dinesh Manocha. „Reciprocal Velocity Obstacles for real-time multi-agent navigation“. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489.

[Bra08] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. „Race driver model“. In: *Computers & Structures* 86.13 (2008). Structural Optimization, pp. 1503–1516. ISSN: 0045-7949. DOI: 10.1016/j.compstruc.2007.04.028.

[Bra22a] Andreas Brandstätter. *Research report: Coordination of a Flock of Drones using Ephemeral Environment Learning*. 2022. URL: https://www.marshallplan.at/images/All-Papers/mp-2022/Brandst%C3%A4tter_Andreas.PDF.

[Bra22b] Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. *Multi-Agent Spatial Predictive Control with Application to Drone Flocking (Extended Version)*. 2022. DOI: 10.48550/ARXIV.2203.16960.

[Bra22c] Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Towards Drone Flocking Using Relative Distance Measurements“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2022, pp. 97–109. ISBN: 978-3-031-19759-8. DOI: 10.1007/978-3-031-19759-8_7.

[Bra23] Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Multi-Agent Spatial Predictive Control with Application to Drone Flocking“. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 1221–1227. DOI: 10.1109/ICRA48891.2023.10160617.

[Bra24] Andreas Brandstätter, Scott A. Smolka, Scott D. Stoller, Ashish Tiwari, and Radu Grosu. „Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements“. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 12193–12200. DOI: 10.1109/ICRA57147.2024.10610147.

[Bru21] Axel Brunnbauer, Luigi Berducci, Andreas Brandstätter, Mathias Lechner, Ramin M. Hasani, Daniela Rus, and Radu Grosu. „Model-based versus Model-free Deep Reinforcement Learning for Autonomous Racing Cars“. In: *CoRR* abs/2103.04909 (2021). arXiv: 2103.04909.

[Bru22] Axel Brunnbauer*, Luigi Berducci*, Andreas Brandstätter*, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. „Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing“. In: *2022 International Conference on Robotics and Automation (ICRA)*. (* indicates equal contribution). 2022, pp. 7513–7520. DOI: 10.1109/ICRA46639.2022.9811650.

[BS17] Daniel Brandtner and Martin Saska. „Coherent swarming of unmanned micro aerial vehicles with minimum computational and communication requirements“. In: *2017 European Conference on Mobile Robots (ECMR)*. 2017 European Conference on Mobile Robots (ECMR). Paris: IEEE, Sept. 2017, pp. 1–6. ISBN: 978-1-5386-1096-1. DOI: 10.1109/ECMR.2017.8098702.

[BS18] Andrew Boggio-Dandry and Tolga Soyata. „Perpetual Flight for UAV Drone Swarms Using Continuous Energy Replenishment“. In: *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 2018, pp. 478–484. DOI: 10.1109/UEMCON.2018.8796684.

[BS20] Reza Babazadeh and Rastko Selmic. „Distance-Based Multiagent Formation Control With Energy Constraints Using SDRE“. In: *IEEE Transactions on Aerospace and Electronic Systems* 56.1 (2020), pp. 41–56. DOI: 10.1109/TAES.2019.2910361.

[Câm14] Daniel Câmara. „Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios“. In: *2014 IEEE Conference on Antenna Measurements Applications (CAMA)*. 2014, pp. 1–4. DOI: 10.1109/CAMA.2014.7003421.

[Cao21] Zhiqiang Cao, Ran Liu, Chau Yuen, Achala Athukorala, Benny Kai Kiat Ng, Muraleetharan Mathanraj, and U-Xuan Tan. „Relative Localization of Mobile Robots with Multiple Ultra-WideBand Ranging Measurements“. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Prague, Czech Republic: IEEE, Sept. 27, 2021, pp. 5857–5863. ISBN: 978-1-66541-714-3. DOI: 10.1109/IROS51168.2021.9636017.

[Čap20] Karel Čapek. *R.U.R. (Rossum's Universal Robots)*. Prague, 1920. URL: https://gutenberg.org/ebooks/59112.

[CB19] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning.* `http://pybullet.org`. 2019.

[CCW12] Giuseppe C. Calafiore, Luca Carlone, and Mingzhu Wei. „A Distributed Technique for Localization of Agent Formations From Relative Range Measurements“. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42.5 (Sept. 2012), pp. 1065–1076. ISSN: 1083-4427, 1558-2426. DOI: `10.1109/TSMCA.2012.2185045`.

[Cha18] Anusna Chakraborty, Kevin Brink, Rajnikant Sharma, and Laith Sahawneh. „Relative Pose Estimation using Range-only Measurements with Large Initial Uncertainty“. In: *2018 Annual American Control Conference (ACC)*. 2018 Annual American Control Conference (ACC). Milwaukee, WI, USA: IEEE, June 2018, pp. 5055–5061. ISBN: 978-1-5386-5428-6. DOI: `10.23919/ACC.2018.8431743`.

[Chu18a] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. „A Survey on Aerial Swarm Robotics“. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 837–855. DOI: `10.1109/TRO.2018.2857475`.

[Chu18b] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. „A Survey on Aerial Swarm Robotics“. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 837–855. DOI: `10.1109/TRO.2018.2857475`.

[CLD24] Sijia Chen, Yuzhu Li, and Wei Dong. „High-Performance Relative Localization Based on Key-Node Seeking Considering Aerial Drags Using Range and Odometry Measurements“. In: *IEEE Transactions on Industrial Electronics* 71.6 (June 2024), pp. 6021–6031. ISSN: 0278-0046, 1557-9948. DOI: `10.1109/TIE.2023.3292851`.

[Cos21] Charles Champagne Cossette, Mohammed Shalaby, David Saussie, James Richard Forbes, and Jerome Le Ny. „Relative Position Estimation Between Two UWB Devices With IMUs“. In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 4313–4320. ISSN: 2377-3766, 2377-3774. DOI: `10.1109/LRA.2021.3067640`.

[Cou92] R. Craig Coulter. *Implementation of the Pure Pursuit Path Tracking Algorithm.* Tech. rep. CMU-RI-TR-92-01. Pittsburgh, PA: Carnegie Mellon University, Jan. 1992. URL: `https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf`.

[CS07a] Felipe Cucker and Steve Smale. „Emergent Behavior in Flocks“. In: *IEEE Transactions on Automatic Control* 52.5 (2007), pp. 852–862. DOI: `10.1109/TAC.2007.895842`.

[CS07b] Felipe Cucker and Steve Smale. „On the mathematics of emergence“. In: *Japanese Journal of Mathematics* 2.1 (Mar. 2007), pp. 197–227. ISSN: 1861-3624. DOI: `10.1007/s11537-007-0647-x`.

[CS19] Yang Cai and Yuan Shen. „An Integrated Localization and Control Framework for Multi-Agent Formation“. In: *IEEE Transactions on Signal Processing* 67.7 (Apr. 2019), pp. 1941–1956. ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2019.2897968.

[CYA09] Ming Cao, Changbin Yu, and Brian D. O. Anderson. „Coordination with the leader in a robotic team without active communication“. In: *2009 17th Mediterranean Conference on Control and Automation*. 2009 17th Mediterranean Conference on Control and Automation (MED). Thessaloniki, Greece: IEEE, June 2009, pp. 252–257. ISBN: 978-1-4244-4684-1. DOI: 10.1109/MED.2009.5164548.

[CYA11] Ming Cao, Changbin Yu, and Brian D.O. Anderson. „Formation control using range-only measurements“. In: *Automatica* 47.4 (Apr. 2011), pp. 776–781. ISSN: 00051098. DOI: 10.1016/j.automatica.2011.01.067.

[DR11] Marc Peter Deisenroth and Carl Edward Rasmussen. „PILCO: A Model-Based and Data-Efficient Approach to Policy Search“. In: *Proceedings of the 28th International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 465–472. ISBN: 9781450306195. URL: https://dl.acm.org/doi/10.5555/3104482.3104541.

[Dur24] Yuri Durodié, Bryan Convens, Gaoyuan Liu, Thomas Decoster, Adrian Munteanu, and Bram Vanderborght. „Where are You? Unscented Particle Filter for Single Range Relative Pose Estimation in Unobservable Motion Using UWB and VIO“. In: *IEEE Robotics and Automation Letters* 9.12 (Dec. 2024), pp. 11754–11761. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2024.3495592.

[DW08] Raffaello D'Andrea and Peter Wurman. „Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities“. In: *2008 IEEE International Conference on Technologies for Practical Robot Applications*. 2008, pp. 80–83. DOI: 10.1109/TEPRA.2008.4686677.

[Ede07] Amnon H. Eden. „Three Paradigms of Computer Science“. In: *Minds Mach.* 17.2 (July 2007), pp. 135–167. ISSN: 0924-6495. DOI: 10.1007/s11023-007-9060-8.

[Edu12] Brazilian Ministry of Education. *Qualis conference ranking (2012)*. https://www.gov.br/capes/pt-br. 2012.

[Edu23] Computing Research & Education. *CORE2023 Conference Ranks*. https://portal.core.edu.au/conf-ranks/. 2023.

[ER10] Jonathan Eversham and Virginie F. Ruiz. „Parameter analysis of Reynolds flocking model“. In: *2010 IEEE 9th International Conference on Cybernetic Intelligent Systems*. 2010, pp. 1–7. DOI: 10.1109/UKRICIS.2010.5898089.

[Fan86] Bertrand T. Fang. „Trilateration and extension to Global Positioning System navigation“. In: *Journal of Guidance, Control, and Dynamics* 9.6 (Nov. 1986), pp. 715–717. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.20169.

[Fer17] Andre Filipe Goncalves Ferreira, Duarte Manuel Azevedo Fernandes, Andre Paulo Catarino, and Joao L. Monteiro. „Localization and Positioning Systems for Emergency Responders: A Survey“. In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2836–2870. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2703620.

[FIA24] FIA - Federation Internationale de l'Automobile. *Roborace Car Revealed*. Dec. 3, 2024. URL: https://www.fiaformulae.com/en/news/6049.

[Flo14] M. Florek, M. Huba, F. Duchoň, J. Šovčík, and M. Kajan. „Comparing approaches to quadrocopter control“. In: *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*. 2014, pp. 1–6. DOI: 10.1109/RAAD.2014.7002241.

[Fox99] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. „Monte Carlo localization: efficient position estimation for mobile robots“. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. AAAI '99/IAAI '99. Orlando, Florida, USA: American Association for Artificial Intelligence, 1999, pp. 343–349. ISBN: 0262511061. URL: https://www.ri.cmu.edu/pub_files/pub1/fox_dieter_1999_1/fox_dieter_1999_1.pdf.

[Fuc20] Florian Fuchs, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Duerr. „Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning“. In: *arXiv preprint arXiv:2008.07971* (2020). DOI: 10.1109/LRA.2021.3064284.

[Gar17] Gonzalo A. Garcia, A. Ram Kim, Ethan Jackson, Shawn S. Keshmiri, and Daksh Shukla. „Modeling and flight control of a commercial nano quadrotor“. In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 524–532. DOI: 10.1109/ICUAS.2017.7991439.

[Gel15] Aviram Gelblum, Itai Pinkoviezky, Ehud Fonio, Abhijit Ghosh, Nir Gov, and Ofer Feinerman. „Ant groups optimally amplify the effect of transiently informed individuals“. In: *Nature Communications* 6.1 (July 28, 2015), p. 7729. ISSN: 2041-1723. DOI: 10.1038/ncomms8729.

[Gen21] General Dynamics. *Bluefin(R)-12 with Integrated Survey Package*. 2021. URL: https://gdmissionsystems.com/-/media/General-Dynamics/Maritime-and-Strategic-Systems/Bluefin/PDF/Bluefin-12-UUV-Datasheet.ashx.

[Geo13] Alexander V. Georgiev, Amanda C. E. Klimczuk, Daniel M. Traficonte, and Dario Maestripieri. „When Violence Pays: A Cost-Benefit Analysis of Aggressive Behavior in Animals and Humans“. In: *Evolutionary Psychology* 11.3 (July 1, 2013), p. 147470491301100. ISSN: 1474-7049, 1474-7049. DOI: 10.1177/147470491301100313.

[Gie17] Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. „Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering“. In: *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2017, pp. 37–42. DOI: 10.1109/MMAR.2017.8046794.

[GLX20] Kexin Guo, Xiuxian Li, and Lihua Xie. „Ultra-Wideband and Odometry-Based Cooperative Relative Localization With Application to Multi-UAV Formation Control“. In: *IEEE Transactions on Cybernetics* 50.6 (June 2020), pp. 2590–2603. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2019.2905570.

[Goo17] Anne E. Goodenough, Natasha Little, William S. Carpenter, and Adam G. Hart. „Birds of a feather flock together: Insights into starling murmuration behaviour revealed using citizen science“. In: *PLOS ONE* 12.6 (June 19, 2017). Ed. by Charlotte K. Hemelrijk, e0179277. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0179277.

[Guo24] Lin Guo, Ran Liu, Zhongyuan Deng, and Yufeng Xiao. „Reconfigurable Multi-Robot Formation via Ultra-Wideband Ranging in Unknown Environments“. In: *IEEE Control Systems Letters* 8 (2024), pp. 321–326. ISSN: 2475-1456. DOI: 10.1109/LCSYS.2024.3373825.

[Haa18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. „Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor“. In: *Int. Conf. on Machine Learning*. 2018, pp. 1861–1870. URL: https://openreview.net/forum?id=HJjvxl-Cb.

[Haa19] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. „Learning to Walk Via Deep Reinforcement Learning“. In: *Proceedings of Robotics: Science and Systems*. FreiburgimBreisgau, Germany, June 2019. DOI: 10.15607/RSS.2019.XV.011.

[Haf19] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. „Learning Latent Dynamics for Planning from Pixels“. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, July 2019, pp. 2555–2565. URL: https://proceedings.mlr.press/v97/hafner19a.html.

[Haf20] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. *Dream to Control: Learning Behaviors by Latent Imagination*. 2020. arXiv: 1912.01603 [cs.LG].

[Haf21] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. „Mastering Atari with Discrete World Models“. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=0oabwyZbOu.

[Has19] Ramin Hasani, Alexander Amini, Mathias Lechner, Felix Naser, Radu Grosu, and Daniela Rus. „Response Characterization for Auditing Cell Dynamics in Long Short-term Memory Networks“. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8851954.

[Has20] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. „A Natural Lottery Ticket Winner: Reinforcement Learning with Ordinary Neural Circuits“. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 4082–4093. URL: https://proceedings.mlr.press/v119/hasani20a.html.

[Has21] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. „Liquid Time-constant Networks“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9 (May 2021), pp. 7657–7666. DOI: 10.1609/aaai.v35i9.16936.

[Has22] Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. „Closed-form continuous-time neural networks“. In: *Nature Machine Intelligence*. Vol. 4. 11. Springer Science and Business Media LLC, Nov. 2022, pp. 992–1003. DOI: 10.1038/s42256-022-00556-7.

[He13] Fenghua He, Ye Wang, Yu Yao, Long Wang, and Weishan Chen. „Distributed formation control of mobile autonomous agents using relative position measurements“. In: *IET Control Theory & Applications* 7.11 (July 2013), pp. 1540–1552. ISSN: 1751-8652, 1751-8652. DOI: 10.1049/iet-cta.2012.1034.

[Hep21] Zhang Hepeng, Di Jian, Yan Han, Wang Xinghu, and Ji Haibo. „Varying-coefficient Based Distributed Formation Control of Multiple UAVs“. In: *2021 40th Chinese Control Conference (CCC)*. 2021, pp. 5524–5528. DOI: 10.23919/CCC52363.2021.9549453.

[Hör12] Horst Hörtner, Matthew Gardiner, Roland Haring, Christopher Lindinger, and Florian Berger. „Spaxels, pixels in space“. In: *Proceedings of the International Conference on Signal Processing and Multimedia Applications and Wireless Information Networks and Systems*. 2012, pp. 19–24. URL: https://www.scitepress.org/PublishedPapers/2012/41264/41264.pdf.

[Hor24] Jiří Horyna, Vít Krátký, Václav Pritzl, Tomáš Báča, Eliseo Ferrante, and Martin Saska. „Fast Swarming of UAVs in GNSS-Denied Feature-Poor Environments Without Explicit Communication“. In: *IEEE Robotics and Automation Letters* 9.6 (June 2024), pp. 5284–5291. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2024.3390596.

[HS18] David Ha and Jürgen Schmidhuber. „Recurrent World Models Facilitate Policy Evolution“. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[Hua21] Jie Huang, Guoqing Tian, Jiancheng Zhang, and Yutao Chen. „On Unmanned Aerial Vehicles Light Show Systems: Algorithms, Software and Hardware“. In: *Applied Sciences* 11.16 (Aug. 21, 2021), p. 7687. ISSN: 2076-3417. DOI: 10.3390/app11167687.

[Ind24] Indy Autonomous Challenge. *Indy Autonomous Challenge*. Dec. 3, 2024. URL: https://www.indyautonomouschallenge.com/.

[JAH20] Bomin Jiang, Brian D. O. Anderson, and Hatem Hmam. „3-D Relative Localization of Mobile Systems Using Distance-Only Measurements via Semidefinite Optimization“. In: *IEEE Transactions on Aerospace and Electronic Systems* 56.3 (June 2020), pp. 1903–1916. ISSN: 0018-9251, 1557-9603, 2371-9877. DOI: 10.1109/TAES.2019.2935926.

[Jar18] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi. „End-to-End Race Driving with Deep Reinforcement Learning“. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2070–2075. DOI: 10.1109/ICRA.2018.8460934.

[JDA17] Bomin Jiang, Mohammad Deghat, and Brian D. O. Anderson. „Simultaneous Velocity and Position Estimation via Distance-only Measurements with Application to Multi-Agent System Control“. In: *IEEE Transactions on Automatic Control* 62.2 (Feb. 2017), pp. 869–875. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2558040.

[Jin20] Wei Jing, Di Deng, Yan Wu, and Kenji Shimada. „Multi-UAV Coverage Path Planning for the Inspection of Large and Complex Structures“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, Oct. 24, 2020, pp. 1480–1486. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341089.

[Kab19a] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. „Learning-based Model Predictive Control for Autonomous Racing“. en. In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 3363–3370. ISSN: 2377-3766. DOI: 10.3929/ethz-b-000351561.

[Kab19b] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, Ankit Dhall, Eugenio Chisari, Napat Karnchanachari, Sonja Brits, Manuel Dangel, Inkyu Sa, Renaud Dubé, Abel Gawel, Mark Pfeiffer, Alexander Liniger, John Lygeros, and Roland Siegwart. *AMZ Driverless: The Full Autonomous Racing System*. 2019. arXiv: `1905.05150 [cs.RO]`.

[Kan14] Sung-Mo Kang, Myoung-Chul Park, Byung-Hun Lee, and Hyo-Sung Ahn. „Distance-based formation control with a single moving leader“. In: *2014 American Control Conference*. 2014 American Control Conference - ACC 2014. Portland, OR, USA: IEEE, June 2014, pp. 305–310. DOI: `10.1109/ACC.2014.6858587`.

[Ken19] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah. „Learning to Drive in a Day“. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8248–8254. DOI: `10.1109/ICRA.2019.8793742`.

[KH04] N. Koenig and A. Howard. „Design and use paradigms for Gazebo, an open-source multi-robot simulator“. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: `10.1109/IROS.2004.1389727`.

[KMB20] M. Kaspar, J. D. Muñoz Osorio, and J. Bock. „Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4383–4388. DOI: `10.1109/IROS45743.2020.9341260`.

[KPA17] Sung-Mo Kang, Myoung-Chul Park, and Hyo-Sung Ahn. „Distance-Based Cycle-Free Persistent Formation: Global Convergence and Experimental Test With a Group of Quadcopters“. In: *IEEE Transactions on Industrial Electronics* 64.1 (Jan. 2017), pp. 380–389. ISSN: 0278-0046, 1557-9948. DOI: `10.1109/TIE.2016.2606585`.

[Lao18] Christos Laoudias, Adriano Moreira, Sunwoo Kim, Sangwoo Lee, Lauri Wirola, and Carlo Fischione. „A Survey of Enabling Technologies for Network Localization, Tracking, and Navigation“. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3607–3644. ISSN: 1553-877X, 2373-745X. DOI: `10.1109/COMST.2018.2855063`.

[LCH19] Yiming Liu, Mengxia Chen, and Hejiao Huang. „Multi-agent Pathfinding Based on Improved Cooperative A* in Kiva System“. In: *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*. 2019, pp. 633–638. DOI: `10.1109/ICCAR.2019.8813319`.

[Lec19] Mathias Lechner, Ramin Hasani, Manuel Zimmer, Thomas A. Henzinger, and Radu Grosu. „Designing Worm-inspired Neural Networks for Interpretable Robotic Control“. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 87–94. DOI: 10.1109/ICRA.2019.8793840.

[Lec20a] Mathias Lechner, Ramin Hasani, Alexander Amini, Thomas A Henzinger, Daniela Rus, and Radu Grosu. „Neural circuit policies enabling auditable autonomy“. In: *Nature Machine Intelligence* 2.10 (2020), pp. 642–652. DOI: 10.1038/s42256-020-00237-3.

[Lec20b] Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. „Gershgorin Loss Stabilizes the Recurrent Neural Network Compartment of an End-to-end Robot Learning Scheme“. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5446–5452. DOI: 10.1109/ICRA40945.2020.9196608.

[Lec21] Mathias Lechner, Ramin Hasani, Radu Grosu, Daniela Rus, and Thomas A. Henzinger. „Adversarial Training is Not Ready for Robot Learning“. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4140–4147. DOI: 10.1109/ICRA48506.2021.9561036.

[Lem21] Clément Lemardelé, Miquel Estrada, Laia Pagès, and Mónika Bachofner. „Potentialities of drones and ground autonomous delivery devices for last-mile logistics“. In: *Transportation Research Part E: Logistics and Transportation Review* 149 (2021), p. 102325. ISSN: 1366-5545. DOI: 10.1016/j.tre.2021.102325.

[LH20] Mathias Lechner and Ramin Hasani. *Learning Long-Term Dependencies in Irregularly-Sampled Time Series*. 2020. arXiv: 2006.04418 [cs.LG].

[Li20] Bai Li, Shaoshan Liu, Jie Tang, Jean-Luc Gaudiot, Liangliang Zhang, and Qi Kong. „Autonomous Last-Mile Delivery Vehicles in Complex Traffic Environments“. In: *Computer* 53.11 (2020), pp. 26–35. DOI: 10.1109/MC.2020.2970924.

[Lib11] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. USA: Princeton University Press, 2011. ISBN: 0691151873.

[Liu17] Ran Liu, Chau Yuen, Tri-Nhut Do, Dewei Jiao, Xiang Liu, and U-Xuan Tan. „Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information“. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, Singapore: IEEE, May 2017, pp. 5623–5629. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989660.

[Liu23] Fen Liu, Shenghai Yuan, Wei Meng, Rong Su, and Lihua Xie. „Non-cooperative Stochastic Target Encirclement by Anti-synchronization Control via Range-only Measurement“. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023 IEEE International Conference on Robotics and Automation (ICRA). London, United Kingdom: IEEE, May 29, 2023, pp. 5480–5485. ISBN: 9798350323658. DOI: 10.1109/ICRA48891.2023.10161054.

[LK18] Giuseppe Loianno and Vijay Kumar. „Cooperative Transportation Using Small Quadrotors Using Monocular Vision and Inertial Sensing“. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 680–687. DOI: 10.1109/LRA.2017.2778018.

[LL01] F. Lamiraux and J. -. Lammond. „Smooth motion planning for car-like vehicles“. In: *IEEE Transactions on Robotics and Automation* 17.4 (2001), pp. 498–501. DOI: 10.1109/70.954762.

[Loq21] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. „Learning high-speed flight in the wild“. In: *Science Robotics* 6.59 (2021), eabg5810. DOI: 10.1126/scirobotics.abg5810.

[LS70] P. B. S. Lissaman and Carl A. Shollenberger. „Formation Flight of Birds“. In: *Science* 168.3934 (May 22, 1970), pp. 1003–1005. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.168.3934.1003.

[Lu23] Junjie Lu, Hongming Shen, Lianrong Pan, Xuewei Zhang, Bailing Tian, and Qun Zong. „Autonomous Flight for Multi-UAV in GPS-Denied Environment“. In: *Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control*. Ed. by Zhang Ren, Mengyi Wang, and Yongzhao Hua. Vol. 934. Series Title: Lecture Notes in Electrical Engineering. Singapore: Springer Nature Singapore, 2023, pp. 892–901. DOI: 10.1007/978-981-19-3998-3_85.

[LXW18] Xuejing Lan, Wenbiao Xu, and Yun-Shan Wei. „Adaptive 3D Distance-Based Formation Control of Multiagent Systems with Unknown Leader Velocity and Coplanar Initial Positions“. In: *Complexity* 2018 (Sept. 6, 2018), pp. 1–9. ISSN: 1076-2787, 1099-0526. DOI: 10.1155/2018/1814653.

[Mar14] Samuel Martin, Antoine Girard, Arastoo Fazeli, and Ali Jadbabaie. „Multiagent Flocking Under General Communication Rule“. In: *IEEE Transactions on Control of Network Systems* 1.2 (2014), pp. 155–166. DOI: 10.1109/TCNS.2014.2316994.

[McG86] John J. McGlone. „Agonistic Behavior in Food Animals: Review of Research and Techniques“. In: *Journal of Animal Science* 62.4 (Apr. 1, 1986), pp. 1130–1139. ISSN: 0021-8812, 1525-3163. DOI: 10.2527/jas1986.6241130x.

[Meh18] Usama Mehmood, Nicola Paoletti, Dung Phan, Radu Grosu, Shan Lin, Scott D. Stoller, Ashish Tiwari, Junxing Yang, and Scott A. Smolka. „Declarative vs Rule-Based Control for Flocking Dynamics“. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. SAC '18. Pau, France: Association for Computing Machinery, 2018, pp. 816–823. ISBN: 9781450351911. DOI: 10.1145/3167132.3167222.

[MFK11] Nathan Michael, Jonathan Fink, and Vijay Kumar. „Cooperative manipulation and transportation with aerial robots“. In: *Autonomous Robots* 30.1 (2011), pp. 73–86. DOI: 10.1007/s10514-010-9205-0.

[Mia20] Sami Mian, Tyler Garrett, Alexander Glandon, Christopher Manderino, Swee Balachandran, Cesar A. Munoz, and Chester V. Dolph. „Autonomous Spacecraft Inspection with Free-Flying Drones“. In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 2020 IEEE/AIAA 39th Digital Avionics Systems Conference (DASC). San Antonio, TX, USA: IEEE, Oct. 11, 2020, pp. 1–9. ISBN: 978-1-72819-825-5. DOI: 10.1109/DASC50938.2020.9256569.

[Mic14] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. „Collaborative Mapping of an Earthquake Damaged Building via Ground and Aerial Robots“. In: *Field and Service Robotics: Results of the 8th International Conference*. Ed. by Kazuya Yoshida and Satoshi Tadokoro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 33–47. ISBN: 978-3-642-40686-7. DOI: 10.1007/978-3-642-40686-7_3.

[MK11] Daniel Mellinger and Vijay Kumar. „Minimum snap trajectory generation and control for quadrotors“. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[Mni13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[Moh18] Kartik Mohta, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, Ke Sun, Alex Zhu, Jeffrey Delmerico, Konstantinos Karydis, Nikolay Atanasov, Giuseppe Loianno, Davide Scaramuzza, Kostas Daniilidis, Camillo Jose Taylor, and Vijay Kumar. „Fast, autonomous flight in GPS-denied and cluttered environments“. In: *Journal of Field Robotics* 35.1 (Jan. 2018), pp. 101–120. ISSN: 15564959. DOI: 10.1002/rob.21774.

[Mou02] Michael C. Mountz. *Material handling system and method using mobile autonomous inventory trays and peer-to-peer communications*. US Patent US6950722B2. 2002. URL: https://patents.google.com/patent/US6950722B2/.

[MTH11] Nathan J. Mlot, Craig A. Tovey, and David L. Hu. „Fire ants self-assemble into waterproof rafts to survive floods“. In: *Proceedings of the National Academy of Sciences* 108.19 (May 10, 2011), pp. 7669–7673. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1016658108.

[Mys21] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. *Regularizing Action Policies for Smooth Control with Reinforcement Learning.* 2021. arXiv: 2012.06644 [cs.RO].

[NK22] Dharna Nar and Radhika Kotecha. „Optimal waypoint assignment for designing drone light show formations“. In: *Results in Control and Optimization* 9 (Dec. 2022), p. 100174. ISSN: 26667207. DOI: 10.1016/j.rico.2022.100174.

[NM14] R.R. Negenborn and J.M. Maestre. „Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities“. In: *IEEE Control Systems Magazine* 34.4 (2014), pp. 87–97. DOI: 10.1109/MCS.2014.2320397.

[NNX22] Thien Hoang Nguyen, Thien-Minh Nguyen, and Lihua Xie. „Flexible and Resource-Efficient Multi-Robot Collaborative Visual-Inertial-Range Localization“. In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 928–935. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3136286.

[NR00] Andrew Y. Ng and Stuart J. Russell. „Algorithms for Inverse Reinforcement Learning“. In: *Proceedings of the Seventeenth International Conference on Machine Learning.* ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1558607072.

[NX23] Thien Hoang Nguyen and Lihua Xie. „Relative Transformation Estimation Based on Fusion of Odometry and UWB Ranging Data“. In: *IEEE Transactions on Robotics* 39.4 (Aug. 2023), pp. 2861–2877. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2023.3264946.

[OA14] Kwang-Kyo Oh and Hyo-Sung Ahn. „Distance-based undirected formations of single-integrator and double-integrator modeled agents in n -dimensional space: DISTANCE-BASED UNDIRECTED FORMATIONS“. In: *International Journal of Robust and Nonlinear Control* 24.12 (Aug. 2014), pp. 1809–1820. ISSN: 10498923. DOI: 10.1002/rnc.2967.

[OKe20] Matthew O’Kelly, Hongrui Zheng, Achin Jain, Joseph Auckley, Kim Luong, and Rahul Mangharam. „TUNERCAR: A Superoptimization Toolchain for Autonomous Racing“. In: *2020 IEEE International Conference on Robotics and Automation (ICRA).* 2020, pp. 5356–5362. DOI: 10.1109/ICRA40945.2020.9197080.

[Olf06] R. Olfati-Saber. „Flocking for multi-agent dynamic systems: algorithms and theory“. In: *IEEE Transactions on Automatic Control* 51.3 (2006), pp. 401–420. DOI: 10.1109/TAC.2005.864190.

[OPA15] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. „A survey of multi-agent formation control". In: *Automatica* 53 (Mar. 2015), pp. 424–440. ISSN: 00051098. DOI: 10.1016/j.automatica.2014.10.022.

[Par21] Parrot. *Parrot for developers: Autonomous flight*. 2021. URL: https://developer.parrot.com/docs/airsdk/general/autonomous_flight.html.

[Pay98] Robert J.H. Payne. „Gradually escalating fights and displays: the cumulative assessment model". In: *Animal Behaviour* 56.3 (1998), pp. 651–662. ISSN: 0003-3472. DOI: https://doi.org/10.1006/anbe.1998.0835.

[Pen18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. „Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (May 2018). DOI: 10.1109/icra.2018.8460528.

[Pop59] Karl Popper. *The Logic of Scientific Discovery*. London: Routledge, 1959. ISBN: 978-0-415-27844-7.

[Por14] Steven J. Portugal, Tatjana Y. Hubel, Johannes Fritz, Stefanie Heese, Daniela Trobe, Bernhard Voelkl, Stephen Hailes, Alan M. Wilson, and James R. Usherwood. „Upwash exploitation and downwash avoidance by flap phasing in ibis formation flight". In: *Nature* 505.7483 (Jan. 16, 2014), pp. 399–402. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature12939.

[Poz21] Beniamino Pozzan, Giulia Michieletto, Angelo Cenedese, and Daniel Zelazo. „Heterogeneous Formation Control: a Bearing Rigidity Approach". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 6451–6456. DOI: 10.1109/CDC45484.2021.9683374.

[Qor21] Qorvo Inc. *DWM1000 – 3.5 - 6.5 GHz Ultra-Wideband (UWB) Transceiver Module*. 2021. URL: https://www.qorvo.com/products/d/da007948 (visited on 09/14/2023).

[Que20a] Jorge Peña Queralta, Carmen Martínez Almansa, Fabrizio Schiano, Dario Floreano, and Tomi Westerlund. „UWB-based System for UAV Localization in GNSS-Denied Environments: Characterization and Dataset". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4521–4528. DOI: 10.1109/IROS45743.2020.9341042.

[Que20b] Jorge Pena Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. „Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision". In: *IEEE Access* 8 (2020), pp. 191617–191643. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3030190.

[QY24] Xian Qiao and Chengpu Yu. „Real-time Relative Pose Estimation for Muti-UAV Systems Using Odometry and UWB Measurements“. In: *2024 43rd Chinese Control Conference (CCC)*. 2024 43rd Chinese Control Conference (CCC). Kunming, China: IEEE, July 28, 2024, pp. 4755–4760. ISBN: 978-988-758-158-1. DOI: 10.23919/CCC63176.2024.10662276.

[Raz21] Eric Razafimahazo, Pierre De Saqui-Sannes, Rob A. Vingerhoeds, Claude Baron, Julien Soulax, and Romain Mège. „Mastering Complexity for Indoor Inspection Drone Development“. In: *2021 IEEE International Symposium on Systems Engineering (ISSE)*. 2021, pp. 1–8. DOI: 10.1109/ISSE51541.2021.9582483.

[Res24] Research Unit Cyber-Physical Systems, TU Wien. *Scuderia Segfault*. Dec. 4, 2024. URL: https://scuderia-segfault.at/.

[Rey87] Craig W. Reynolds. „Flocks, Herds and Schools: A Distributed Behavioral Model“. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 25–34. ISBN: 0897912276. DOI: 10.1145/37401.37406.

[RGB11] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. „A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning“. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 627–635. URL: https://proceedings.mlr.press/v15/ross11a.html.

[Rin21] Bernhard Rinner, Christian Bettstetter, Hermann Hellwagner, and Stephan Weiss. „Multidrone Systems: More Than the Sum of the Parts“. In: *Computer* 54.5 (2021), pp. 34–43. DOI: 10.1109/MC.2021.3058441.

[RMD07] M. Riedmiller, M. Montemerlo, and H. Dahlkamp. „Learning to Drive a Real Car in 20 Minutes“. In: *2007 Frontiers in the Convergence of Bioscience and Information Technologies*. 2007, pp. 645–650. DOI: 10.1109/FBIT.2007.37.

[RNH15] A. Rucco, G. Notarstefano, and J. Hauser. „An Efficient Minimum-Time Trajectory Generation Strategy for Two-Track Car Vehicles“. In: *IEEE Trans on Control Systems Tech* 23.4 (2015), pp. 1505–1519. DOI: 10.1109/TCST.2014.2377777.

[SAI18] Giuseppe Silano, Emanuele Aucone, and Luigi Iannelli. „CrazyS: A Software-In-The-Loop Platform for the Crazyflie 2.0 Nano-Quadcopter“. In: *2018 26th Mediterranean Conference on Control and Automation (MED)*. 2018, pp. 1–6. DOI: 10.1109/MED.2018.8442759.

[Sas17] Martin Saska, Tomas Baca, Justin Thomas, Jan Chudoba, Libor Preucil, Tomas Krajnik, Jan Faigl, Giuseppe Loianno, and Vijay Kumar. „System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization“. In: *Autonomous Robots* 41.4 (Apr. 2017), pp. 919–944. ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-016-9567-z.

[SB18] Richard S Sutton and Andrew G Barto. *RL: An introduction.* MIT press, 2018.

[Sch09] Mac Schwager. „A Gradient Optimization Approach to Adaptive Multi-Robot Control“. PhD thesis. Massachusetts Institute of Technology, 2009. URL: https://dspace.mit.edu/handle/1721.1/55256.

[Sch17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms.* 2017. arXiv: 1707.06347 [cs.LG].

[Sch20] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. „Swarm Robotic Behaviors and Current Applications“. In: *Frontiers in Robotics and AI* 7 (Apr. 2, 2020), p. 36. ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00036.

[Sch21] Wilko Schwarting, Tim Seyde, Igor Gilitschenski, Lucas Liebenwein, Ryan Sander, Sertac Karaman, and Daniela Rus. „Deep Latent Competition: Learning to Race Using Visual Control Policies in Latent Space“. In: *arXiv preprint arXiv:2102.09812* (2021). URL: https://arxiv.org/abs/2102.09812.

[Sch99] Stefan Schaal. „Is imitation learning the route to humanoid robots?“ In: *Trends in Cognitive Sciences* 3.6 (1999), pp. 233–242. ISSN: 1364-6613. DOI: https://doi.org/10.1016/S1364-6613(99)01327-3.

[Sey21] Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. „Is bang-bang control all you need? solving continuous control with bernoulli policies“. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems.* NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393. URL: https://proceedings.neurips.cc/paper/2021/file/e46be61f0050f9cc3a98d5d2192cb0eb-Paper.pdf.

[SF51] J. P. Scott and Emil Fredericson. „The Causes of Fighting in Mice and Rats“. In: *Physiological Zoology* 24.4 (Oct. 1951), pp. 273–309. ISSN: 0031-935X. DOI: 10.1086/physzool.24.4.30152137.

[SFA09] Iman Shames, Barış Fidan, and Brian D.O. Anderson. „Minimization of the effect of noisy measurements on localization of multi-agent autonomous formations“. In: *Automatica* 45.4 (Apr. 2009), pp. 1058–1065. ISSN: 00051098. DOI: 10.1016/j.automatica.2008.11.018.

[SG12] Volkan Sezer and Metin Gokasan. „A novel obstacle avoidance algorithm: "Follow the Gap Method"". In: *Robotics and Autonomous Systems* 60.9 (2012), pp. 1123–1134. ISSN: 0921-8890. DOI: 10.1016/j.robot.2012.05.021.

[SG17] Fabrizio Schiano and Paolo Robuffo Giordano. „Bearing rigidity maintenance for formations of quadrotor UAVs". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1467–1474. DOI: 10.1109/ICRA.2017.7989175.

[Sha21] Mohammed Shalaby, Charles Champagne Cossette, James Richard Forbes, and Jerome Le Ny. „Relative Position Estimation in Multi-Agent Systems Using Attitude-Coupled Range Measurements". In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 4955–4961. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3067253.

[Sha22] Feng Shan, Haodong Huo, Jiaxin Zeng, Zengbao Li, Weiwei Wu, and Junzhou Luo. „Ultra-Wideband Swarm Ranging Protocol for Dynamic and Dense Networks". In: *IEEE/ACM Transactions on Networking* 30.6 (2022), pp. 2834–2848. DOI: 10.1109/TNET.2022.3186071.

[Sin19] Avi Singh, Larry Yang, Chelsea Finn, and Sergey Levine. „End-To-End Robotic Reinforcement Learning without Reward Engineering". In: *Proceedings of Robotics: Science and Systems*. 2019. DOI: 10.15607/RSS.2019.XV.073.

[SM19] Mingfei Sun and Xiaojuan Ma. „Adversarial Imitation Learning from Incomplete Demonstrations". In: *CoRR* abs/1905.12310 (2019). arXiv: 1905.12310.

[Son23] Yunlong Song, Angel Romero, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. „Reaching the limit in autonomous racing: Optimal control versus reinforcement learning". In: *Science Robotics* 8.82 (2023), eadg1462. DOI: 10.1126/scirobotics.adg1462.

[SS18] Raik Suttner and Zhiyong Sun. „Formation Shape Control Based on Distance Measurements Using Lie Bracket Approximations". In: *SIAM Journal on Control and Optimization* 56.6 (Jan. 2018), pp. 4405–4433. ISSN: 0363-0129, 1095-7138. DOI: 10.1137/18M117131X.

[SSF19] Enrica Soria, Fabrizio Schiano, and Dario Floreano. „The Influence of Limited Visual Sensing on the Reynolds Flocking Algorithm". In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 138–145. DOI: 10.1109/IRC.2019.00028.

[SSF21] Enrica Soria, Fabrizio Schiano, and Dario Floreano. „Predictive control of aerial swarms in cluttered environments". In: *Nature Machine Intelligence*. 2021. DOI: 10.1038/s42256-021-00341-y.

[SSF22] Enrica Soria, Fabrizio Schiano, and Dario Floreano. „Distributed Predictive Drone Swarms in Cluttered Environments". In: *IEEE Robotics and Automation Letters* 7.1 (2022), pp. 73–80. DOI: 10.1109/LRA.2021.3118091.

[ST08] Rajnikant Sharma and Clark Taylor. „Cooperative navigation of MAVs In GPS denied areas“. In: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2008). Seoul: IEEE, Aug. 2008, pp. 481–486. ISBN: 978-1-4244-2143-5. DOI: 10.1109/MFI.2008.4648041.

[Sta] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. URL: https://www.ros.org (visited on 09/14/2023).

[Stu09] Ethan Stump, Vijay Kumar, Ben Grocholsky, and Pedro M. Shiroma. „Control for Localization of Targets using Range-only Sensors“. In: *The International Journal of Robotics Research* 28.6 (June 2009), pp. 743–757. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364908098559.

[Sun07] Sunghwan Kim, Chang-Kyung Ryoo, Keeyoung Choi, and Choonbae Park. „Multi-vehicle formation using range-only measurement“. In: *2007 International Conference on Control, Automation and Systems*. 2007 International Conference on Control, Automation and Systems. Seoul, South Korea: IEEE, 2007, pp. 2104–2109. DOI: 10.1109/ICCAS.2007.4406677.

[Sut91] Richard S. Sutton. „Dyna, an Integrated Architecture for Learning, Planning, and Reacting“. In: *SIGART Bull.* 2.4 (July 1991), pp. 160–163. ISSN: 0163-5719. DOI: 10.1145/122344.122377.

[SZ 21] SZ DJI Technology Co., Ltd. *dji developer: Missions*. 2021. URL: https://developer.dji.com/document/aab56894-31c7-4f38-b12d-616d312965a6.

[Tan18] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. „Sim-to-Real: Learning Agile Locomotion For Quadruped Robots“. In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018. DOI: 10.15607/RSS.2018.XIV.010.

[Tau21] Taurob Technologies. *The Taurob Inspector*. 2021. URL: https://taurob.com/wp-content/uploads/2020/08/Technical-Product-Sheet.pdf (visited on 08/19/2024).

[TC13] Julian P Timings and David J Cole. „Minimum maneuver time calculation using convex optimization“. In: *Journal of Dynamic Systems, Measurement, and Control* 135.3 (2013). DOI: 10.1115/1.4023400.

[TJP03] Herbert G. Tanner, Ali Jadbabaie, and George J. Pappas. *Stability of Flocking Motion*. Tech. rep. University of Pennsylvania, 2003. URL: https://www.georgejpappas.org/papers/boids03.pdf.

[Vás18] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E. Eiben, and Tamás Vicsek. „Optimized flocking of autonomous drones in confined environments“. In: *Science Robotics* 3.20 (2018). DOI: 10.1126/scirobotics.aat3536.

[Váz20] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros. „Optimization-Based Hierarchical Motion Planning for Autonomous Racing“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2397–2403. DOI: 10.1109/IROS45743.2020.9341731.

[Vor21] Charles J Vorbach, Ramin Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. „Causal Navigation by Continuous-time Neural Networks“. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=ckVbQs5zD7_.

[VT05] E. Velenis and P. Tsiotras. „Minimum Time vs Maximum Exit Velocity Path Optimization During Cornering“. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.* Vol. 1. 2005, pp. 355–360. DOI: 10.1109/ISIE.2005.1528936.

[Wal18] Viktor Walter, Nicolas Staub, Martin Saska, and Antonio Franchi. „Mutual Localization of UAVs based on Blinking Ultraviolet Markers and 3D Time-Position Hough Transform“. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. 2018, pp. 298–303. DOI: 10.1109/COASE.2018.8560384.

[Wal19] V. Walter, N. Staub, A. Franchi, and M. Saska. „UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs“. In: *IEEE Robotics and Automation Letters* 4.3 (July 2019), pp. 2637–2644. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2901683.

[Wen22] Kai-Chun Weng, Shu-Ting Lin, Chen-Chi Hu, Ru-Tai Soong, and Ming-Te Chi. „Multi-view approach for drone light show“. In: *The Visual Computer* (Nov. 12, 2022). ISSN: 0178-2789, 1432-2315. DOI: 10.1007/s00371-022-02696-8.

[WG18] Li Wang and Qiao Guo. „Distance-based Formation Stabilization and Flocking Control for Distributed Multi-agent Systems“. In: *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2018 IEEE International Conference on Mechatronics and Automation (ICMA). Changchun: IEEE, Aug. 2018, pp. 1580–1585. DOI: 10.1109/ICMA.2018.8484412.

[Whe20] David O. Wheeler, Daniel P. Koch, James S. Jackson, Gary J. Ellingson, Paul W. Nyholm, Timothy W. McLain, and Randal W. Beard. „Relative navigation of autonomous GPS-degraded micro air vehicles“. In: *Autonomous Robots* 44.5 (May 2020), pp. 811–830. ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-019-09899-4.

[Wil17] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. „Information theoretic MPC for model-based reinforcement learning“. In: *2017 IEEE International*

Conference on Robotics and Automation (ICRA). 2017, pp. 1714–1721. DOI: 10.1109/ICRA.2017.7989202.

[WSF18] Viktor Walter, Martin Saska, and Antonio Franchi. „Fast Mutual Relative Localization of UAVs using Ultraviolet LED Markers“. In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2018, pp. 1217–1226. DOI: 10.1109/ICUAS.2018.8453331.

[Wu19] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. „Imitation Learning from Imperfect Demonstration“. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 6818–6827. URL: https://proceedings.mlr.press/v97/wu19a.html.

[WWP10] Baolin Wu, Danwei Wang, and Eng Kee Poh. „Decentralized control for satellite formation using local relative measurements only“. In: *IEEE ICCA 2010*. 2010, pp. 661–666. DOI: 10.1109/ICCA.2010.5524080.

[XL14] Hongli Xu and Guannan Li. „An Elasticity Inspired method for multi-AUVs formation control using range-only measurements“. In: *OCEANS 2014 - TAIPEI*. OCEANS 2014 - TAIPEI. Taipei, Taiwan: IEEE, Apr. 2014, pp. 1–6. DOI: 10.1109/OCEANS-TAIPEI.2014.6964448.

[Yu18] Yang Yu. „Towards Sample Efficient Reinforcement Learning“. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden: AAAI Press, 2018, pp. 5739–5743. ISBN: 9780999241127. DOI: 10.24963/ijcai.2018/820.

[ZCI21] Ornella Zerlenga, Vincenzo Cirillo, and Rosina Iaderosa. „Once Upon a Time there were Fireworks. The New Nocturnal Drones Light Shows“. In: *img journal* (Sept. 10, 2021). Artwork Size: 402-425 Pages Publisher: img journal, 402–425 Pages. DOI: 10.6092/ISSN.2724-2463/12628.

[Zel15] Daniel Zelazo, Antonio Franchi, Heinrich H. Bülthoff, and Paolo Robuffo Giordano. „Decentralized rigidity maintenance control with range measurements for multi-robot systems“. In: *The International Journal of Robotics Research* 34.1 (2015), pp. 105–128. DOI: 10.1177/0278364914546173.

[Zho22] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, and Fei Gao. „Swarm of micro flying robots in the wild“. In: *Science Robotics* 7.66 (2022), eabm5954. DOI: 10.1126/scirobotics.abm5954.

[Zhu20] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. „The Ingredients of Real World Robotic Reinforcement Learning“. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=rJe2syrtvS.

[Zie21] Thomas Ziegler, Marco Karrer, Patrik Schmuck, and Margarita Chli. „Distributed Formation Estimation Via Pairwise Distance Measurements“. In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021), pp. 3017–3024. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3062347.

[ZQW20] W. Zhao, J. P. Queralta, and T. Westerlund. „Sim-to-Real Transfer in Deep RL for Robotics: a Survey“. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.

[ZWG19] Ying Zou, Changyun Wen, and Mingyang Guan. „Distributed adaptive control for distance-based formation and flocking control of multi-agent systems“. In: *IET Control Theory & Applications* 13.6 (Apr. 2019), pp. 878–885. ISSN: 1751-8652, 1751-8652. DOI: 10.1049/iet-cta.2018.6001.