Agata Maria Barciś

Synchronizing and Swarming Robots

Spatio-Temporal Coordination in Multi-Robot Systems

DISSERTATION

submitted in fulfilment of the requirements for the degree of
Doktorin der Technischen Wissenschaften

Universität Klagenfurt
Fakultät für Technische Wissenschaften

Supervisors

Univ.-Prof. Dipl.-Ing. Dr. Christian Bettstetter
Universität Klagenfurt
Institut für Vernetzte und Eingebettete Systeme

Univ.-Prof. Dipl.-Ing. Dr. Wilfried Elmenreich
Universität Klagenfurt
Institut für Vernetzte und Eingebettete Systeme

Expert Reviewers

Univ.-Prof. Dipl.-Ing. Dr. Christian Bettstetter
Universität Klagenfurt
Institut für Vernetzte und Eingebettete Systeme

Prof. Dr.-Ing. Heiko Hamann
Universität zu Lübeck
Institut für Technische Informatik

Klagenfurt, 2021

Affidavit

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated,
- I have fully disclosed all assistance received from third parties during the process of writing the thesis, including any significant advice from supervisors,
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g. in footnotes),
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- when passing on copies of the academic thesis (e.g. in bound, printed or digital form), I will ensure that each copy is fully consistent with the submitted digital version.

I understand that the digital version of the academic thesis submitted will be used for the purpose of conducting a plagiarism assessment. I am aware that a declaration contrary to the facts will have legal consequences.

Agata Barciś, m.p. Klagenfurt, April 2021

Abstract

Constant advancement in robotics and ever-decreasing prices of hardware allow applications of multi-robot systems to become more and more popular. Whenever multiple robots collaborate, their actions need to be coordinated in time and space. Often, if this coordination needs to happen in a distributed manner, the solutions based on self-organization are employed. However, the research on self-organized spatial and temporal coordination remained disconnected until recently. In this thesis, we focus on applying these concepts in multi-robot systems. Specifically, we take into account constrained communication, movement safety, and the reproducibility of patterns. With these assumptions we first present separately our approaches to temporal and spatial coordination, which are afterward coupled in a unified *sandsbot* model. Our solution allows the emergence of various spatio-temporal patterns. At the same time, it works with a low update rate, is robust against delays and provides more control over the formed pattern. We analyze our model in simulation and present its feasibility in experiments featuring ground and aerial robots.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Christian Bettstetter, for his guidance, motivation, and introduction to the world of research. His passion for self-organization was truly inspiring and gave direction for this thesis.

I would also like to thank my co-supervisor, Prof. Wilfried Elmenreich, for his thorough feedback that helped to improve this thesis. Furthermore, I would like to thank Prof. Heiko Hamann, who kindly agreed to be the reviewer of this thesis.

All the discussions with KPK NAV members, among them Prof. Stephan Weiss, Prof. Hermann Hellwagner, and Prof. Bernhard Rinner, provided valuable ideas and feedback that helped to shape this research. I would like to thank Michał Barciś for our endless discussions that have led to many new ideas and for his continuous support. I am grateful to all my colleagues, especially Mahin Atiq, Petra Maždin, Roland Jung, and Samira Hayat, for the great memories and time we shared during lunch breaks and weekends. I would also like to thank Kornelia Lienbacher for the excellent organization and her priceless advice.

Last but not least, I would like to thank my family, especially my parents, for their unconditional support throughout my life.

Contents

1 Introduction

Robots working together need to coordinate their actions in time and space. Temporal coordination allows to organize the actions of agents in time, e.g., execute some task synchronously or, just the opposite, avoid using resources when other agents need them. It is often realized by coupling of the internal clocks (commonly called phases) of agents. Spatial coordination is focused on how the agents can interact in space, e.g., avoid collisions, move in formation, or gather around the point of interest. This means that the position of an agent needs to be influenced by the positions of other agents. If the phases and positions of agents affect each other, we speak of bi-directional coupling of temporal and spatial coordination. Even if both, temporal and spatial coordination, are based on very simple rules, their combination can cause complex behaviors to emerge. This thesis explores if and how the mutual coupling of temporal and spatial coordination can be realized in multi-robot systems. More specifically, we present how to employ this concept considering peculiarities of multi-robot systems, like constrained communication and movement.

1.1 Motivation

Multi-robot systems offer numerous advantages in comparison to usage of a single robot [4]. For example, the time of an inspection of a huge field can be greatly reduced. In a surveillance mission, some robots can be recharging, while the others are in the operation, guaranteeing a continuous monitoring of the area of interest. Moreover, if there are multiple robots with different capabilities (e.g., equipped with different sensors), few tasks can be performed in parallel, without the need to reconfigure a single robot. In some cases it is even impossible to perform a task with a single robot, for example when a heavy package needs to be transported, few robots might be able to collaboratively pick it up, whereas it will be too heavy for a single one. These and many more advantages encourage the rising interest in applications of multi-robot systems. The development of control and coordination strategies done in the recent years, together with

the constantly decreasing prices of hardware, make it possible to use such systems in reality.

For some applications it is necessary or desirable that more robots can join the system at any point and the success of the mission should not be at risk in case a single agent fails or its battery gets depleted. Obviously, requirements like that can be taken into account by planning algorithms. However, these can become too complex to work on the compute modules of robots, especially if the group of robots gets bigger. Therefore, the solutions based on self-organization gain interest in such cases. Thanks to the simple rules, often based on phenomena observed in nature, such algorithms are flexible, robust and scalable. They allow agents to join or leave the group at any point, without hindering the operation of others. Therefore, in this thesis we focus on self-organizing approaches to spatio-temporal coordination.

1.2 *Background*

The work described in this thesis was realized as a part of the Karl Popper Kolleg on Networked Autonomous Aerial Vehicles (KPK NAV). KPK NAV is a doctoral school associating researchers from four different research groups at the University of Klagenfurt. To foster collaborations within KPK NAV, a common, interdisciplinary

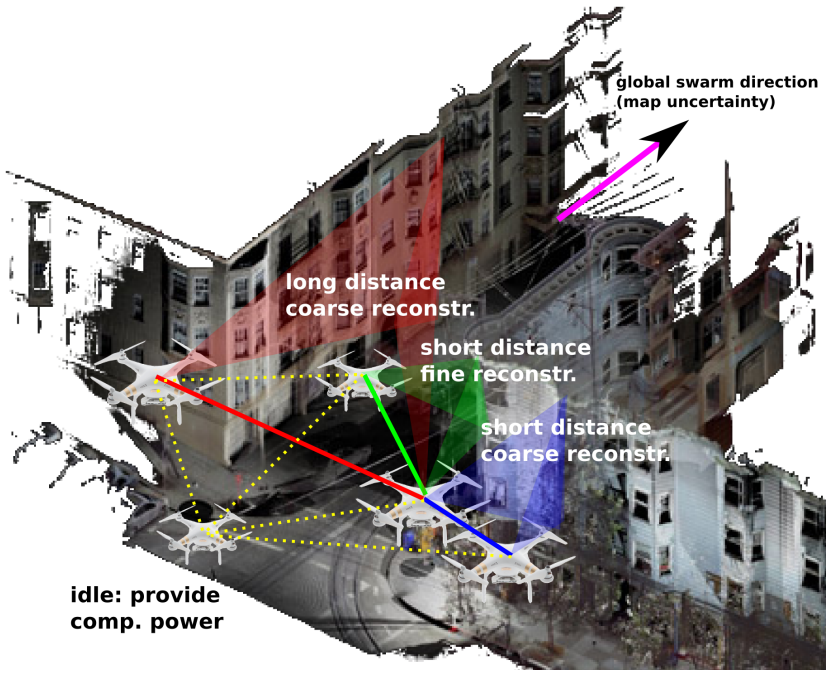


Figure 1.1: Outline of the common use case of KPK NAV. A group of UAVs is autonomously creating a 3D reconstruction of the environment. Source: KPK NAV proposal [5], modified with permission from a work by Scott Page Design (`http://www.scottpagedesign.com/`).

use case was defined: a group of *unmanned aerial vehicles* (UAVs) need to create a 3D model of a given area. The vision of this application is outlined in Figure 1.1. There are a few additional assumptions, but here we focus only on the two most important for this thesis:

1. Each UAV is equipped with a single camera. The agents need to collaborate in order to obtain a stereo vision by merging a pictures taken by two different cameras. Depending on the distance from the reconstructed surface and required precision of the model the UAVs need to adjust the *baseline* (i.e., the distance between cameras).

2. The environment in which the UAVs are operating is unknown and might contain *dynamic objects*.

To create a 3D model of a dynamic object, two UAVs need to take pictures exactly in the same moment. Therefore, this thesis was planned to solve the problem of synchronization of multiple UAVs and to provide a common time frame for them.

However, just after the KPK NAV was started, a novel mathematical model coupling synchronization with swarming was introduced by O'Keeffe, Hong, and Strogatz [6]. The spatio-temporal patterns presented in that work inspired us to rethink the potential of self-organization in the use case of KPK NAV. For example, combining coordination in time and space might allow the UAVs to move within the reconstructed area, all the time maintaining the constant baseline and synchronously taking pictures. Another possibility would be to gather UAVs around a dynamic object of interest, split them into few groups, each of these groups would take pictures synchronously, but at different moment than other groups. This approach would allow to create a sequence of 3D models, for example of a moving car. These findings motivated the research question answered in this thesis: how to couple temporal and spatial coordination in a multi-robot system.

1.3 Contributions

Even though the theoretical "sync and swarm" model [6] was an inspiration for this work and its straightforward implementation served as a proof-of-concept, such a solution did not work robustly in a robotic setting. Motivated by the possibility to apply a similar concept in practical missions, we propose a new model that allows emergence of similar patterns, but is suitable for multi-robot systems. In this section we present the main contributions of this thesis.

Sandsbots — Synchronizing and Swarming Robots

The main contribution of this work is the *sandsbot model* that couples temporal and spatial coordination and meets the requirements of multi-robot systems. Specifically, it is robust against delays and works even with low update rates, which helps to reduce communication load. Applying this model leads to the emergence of the patterns similar to the ones presented by O'Keeffe *et al.* [6]. At the same time it enables to directly control distribution of phases of agents, which helps to adjust the pattern to the mission specifics. To the best of our knowledge this is the first model coupling temporal and spatial coordination suitable for application in multi-robot systems. To achieve this we needed to present a few subsidiary contributions, described in the following paragraphs.

Temporal Coordination

The first building block of the sandsbot model is temporal coordination (Chapter 4). In this work we propose a novel model that utilizes discrete coupling and allows forming three temporal patterns: *synchronized*, *clustered* and *splay*. Thanks to the synchronization of clocks and discretization in both, time and phase, it is robust against delays and works even with a low frequency of updates.

Spatial Aggregation

The second building block needed for the sandsbot model is spatial coordination, specifically we propose a spatial aggregation model (Chapter 5). The time synchronization and discretization allows the agents to robustly form the pattern even if the messages are delayed. Moreover, we limit the speed of agents to account for the low update rate and guarantee stable and collision-free operation.

Bi-Directional Coupling

Finally, to merge these two building blocks into a unified sandsbot model, we propose coupling functions (Chapter 6). We split the interactions of phases and positions into two parts: attraction and repulsion. The spatio-temporal coupling functions specify how spatial proximity of other agents influences attraction and repulsion and, analogously, how other agents' phases influence attraction and repulsion in space.

Experimental Setup and Evaluation

To evaluate the proposed models, we prepared a software framework. It enables multiple modes of operation, including simulation and experiments featuring robots. This approach allows us to test exactly the same implementation in the simulation and on the robots. Furthermore, the framework supports two robotic platforms introduced for the work presented in this thesis: ground robots and small UAVs. This software framework is used in the analysis of the models throughout this thesis. The structure of the software framework is described in Appendix B.

1.4 Structure

The remainder of this thesis is structured as follows. In Chapter 2 we present a scientific background for our work, including the detailed description of sync and swarm model by O'Keeffe *et al.* [6]. Chapter 3 includes the description of the robotic platforms introduced and used in the course of this work and our proof-of-concept of implementing the sync and swarm model in a multi-robot system. Especially the last two sections of this chapter are important: Section 3.3 presents our findings from the proof-of-concept and definition of the requirements for the model suitable for multi-robot systems, and Section 3.4 contains the definitions used in the introduced models and is crucial for understanding the rest of this thesis. In Chapters 4 and 5 we introduce the models for temporal coordination and spatial aggregation, respectively. Finally, in Chapter 6 we show how to couple these models and which patterns emerge as a result. Chapter 7 summarizes this work. In this summary we recall the defined requirements for the model suited for multi-robot systems and show that the sandsbot model fulfills them.

Additionally, this thesis contains three appendices. In Appendix A we present supplementary videos prepared in the course of this thesis. Appendix B contains description of the software framework prepared and used for this thesis and example configurations utilized in experiments. In Appendix C we provide some additional results for sandsbot model.

Parts of this thesis have been published [1]–[3] with contributions of the co-authors: Prof. Christian Bettstetter and Michał Barciś. Our publications that were used in this thesis are gathered in a separate list in the references.

2 *Self-Organization in Temporal and Spatial Coordination*

In this chapter we review the state-of-the-art. We focus on distributed, self-organizing approaches to temporal and spatial coordination and how these two parts can be combined to bring further benefits. As self-organizing solutions are often based on the phenomena observed in nature, for each of these parts we first describe some examples of the behaviors that can be observed in nature. Later we present some theoretical models of these behaviors and finally, describe work utilizing them in technical systems.

2.1 Temporal Coordination: Coupled Oscillators

Synchronization can be observed in many situations in nature and every-day life. Some prominent examples include synchronized insects, like flashing fireflies [7] or crickets chirping in unison [8]. These kind of behaviors is not limited to big groups of individuals. Also physiological processes in our bodies make use of synchronization: heart controlled by pacemaker cells that provide a rhythmic beat [9] or sleeping schedule governed by circadian rhythm that adjusts to day-night cycle of the Earth [10]. The synchronization is visible not only inside our organisms, but also in society: the audience clapping in unison [11] or synchronized menstrual cycle of women that spend a lot of time together [12].

In many cases synchronization is actually undesirable. In nature prey animals tend to have a life cycle duration that helps them to desynchronize from the life cycles of predators [13]. Neurons in our brains are normally desynchronized and fire sequentially, synchronization of a big group of them may lead to an epileptic seizure [14]. After a few accidents when bridges collapsed because of marching troops, soldiers are not allowed to march in unison over bridges. Similar phenomenon occurred after opening of the Millennium Bridge in London when synchronized movement of people caused lateral vibrations of the bridge [15].

A comprehensive summary of these and other examples of synchronization in nature and its modeling is given by Strogatz in an overview book [16].

Theoretical Models

Wiener [17] was probably the first one to mathematically study synchronization. Even though his work turned out to be incorrect [16] and was not continued, it inspired other scientists to pursue a similar path and try to describe this phenomenon. Winfree [18] presented a very general model aiming at describing synchronization of all biological oscillators. He described the phase coupling between oscillators with two functions of phase: *influence functions* specifying how the oscillator influences the other ones and *sensitivity function* showing how the oscillator reacts to the received signals. His work was applicable to very broad set of behaviors, from discrete (pulse) phase coupling of flashing fireflies to continuous influence of pheromones in the menstrual cycle. Because Winfree's model was so generic it was impossible to solve it analytically for different sets of functions. Therefore he used computer simulations to verify the result and concluded that the oscillators are synchronizing for various pairs of influence and sensitivity functions.

Kuramoto [19] simplified Winfree's model focusing on the continuous coupling. He formulated one of the best known mathematical models of synchronization. Thanks to a clever choice of the coupling function he was able to solve the model analytically. Additionally, Kuramoto introduced order parameter to quantify a level of synchronization. The values of the order parameter vary from 0, meaning the oscillators are in a *balanced state*, to 1, if they are perfectly *synchronized*.

In the meantime researchers also studied pulse coupling of oscillators. Buck and Buck [20] analyzed a mechanism that allows fireflies to synchronize. Similar behavior was modeled by Peskin [21] for synchronization of heart pacemaker cells. Their work laid the ground for mathematical formulation of pulse coupled oscillator model by Mirollo and Strogatz [22]. Along with proposing a new model, Mirollo and Strogatz [22] also proved that a system governed by this model almost always synchronizes.

When the coupling between oscillators is negative, their phases repel each other. Although already Winfree [18] took into account the negative coupling in his model, this phenomenon is not as broadly studied as the positive coupling. There is mainly some work on the coexistence of positively and negatively coupled oscillators and the conditions under which the system can reach synchrony [23], [24].

In some cases, researchers focus not only on synchronization, but more general temporal coordination of oscillators and propose algorithms to achieve splay [25]–[27] or cluster state [28], [29]. Choe *et al.* [30] proposed a method to achieve both, splay and cluster states, and analyzed their stability. Often splay and clustered states are considered to be unwanted equilibria of oscillator systems, so their stability is analyzed for continuous [31] and pulse coupling [32], [33]. Nevertheless, these states can prove beneficial in robotic systems.

Realizations in Technical Systems

In technical systems it is often needed to coordinate big groups of agents in time. For example, nodes in sensor networks need to take measurements at the same time [34], or communication collisions need to be avoided by scheduling messages to be sent in *anti-phase* [35]. Similarly to the coupled oscillators in nature, in technical systems they might be realized using various communication interfaces, for example light [36], sound [37] or radio [38].

Researchers in the field of temporal coordination in technical systems focus on other topics than in the theoretical field. The examples include communication latency [39], different network topologies [40], multi-hop communication [41]. The problems that researchers try to tackle differ also based on the choice of communication interface. For radio communication Brandner *et al.* [38] showed that the synchronization can be improved if it is implemented in the physical layer. For robots emitting light pulses Perez-Diaz *et al.* [36] showed how the synchronization is influenced by the camera field of view and movement speed.

Also in technical applications splay and cluster states hold interest. Degesys *et al.* [35] proposed an algorithm based on firefly synchronization to achieve splay state and applied it to implement time division multiple access (TDMA) for wireless sensor networks. Xia *et al.* [42] developed two methods for phase clustering in multi-agent systems.

2.2 *Spatial Coordination: Swarming*

Self-organized spatial coordination can be observed in the behavior of many organisms. For many animals survival in a group is much easier or makes very complex tasks achievable. An interesting example is building of nests by honeybees, wasps, ants and termites [43]. In this process very complex structures are created, which would not be achievable by an individual. Ants also show collective behavior while foraging for food. They mark most promising paths, creating

kind of "motorways" leading to food source and managing them without any congestions [44].

Foraging for food is also one of the reasons why fish school and birds flock [45]. It is more probable that at least one member of a group spots the food source, than that an individual finds it alone. The similar logic holds for predators, it is enough if one agent spots danger and the whole swarm can avoid it. Some predators can also get confused and take the whole swarm as a big creature, thus running away from their potential prey. These reasons make functioning of small animals in the swarm safer [46].

Finally, swimming and flying in formation makes movement more efficient and the tired leaders might be exchanged by some other agents and get opportunity to rest. An example are migrating geese flying in V formation [47]. The similar behavior can be observed in groups of bicyclist taking turns on the lead, while others save energy riding behind [48].

Interestingly, collective behavior is not always a reflection of the care for well-being of the group, but might be caused by selfish behavior of many individuals. For example crickets that lack some nutrients try to hunt other cricket. This results in an orderly march of the whole population, in which each cricket tries to eat the one in front of it and avoid being eaten by the one behind [49].

Theoretical Models

The swarm behaviors observed in nature have inspired many researchers to describe them formally and create mathematical models trying to mimic animals [50]. The theoretical models for self-organized spatial coordination are often based on very simple, local interactions between agents. One of the most known examples of such a model is the flocking model created by Reynolds [51]. The agents called *boids* follow only three simple rules:

1. Avoid collisions with other agents.
2. Try to stay close to other agents.
3. Align velocity with your neighbors.

The simulations following these rules revealed a behavior very similar to flocking birds or schooling fish. When the simulated flock interacts with the environment, the proper balance between these rules allows the flock to split in order to avoid an obstacle or predator and merge again afterwards. A mathematical model for self-organized flocking was proposed by Vicsek *et al.* [52]. They consider particles moving at constant speed that align their velocities.

Another problem related to spatial coordination of a swarm is formation control [53]. The approaches employ different methods, for example potential fields [54], leader-follower [55] or virtual structure [56].

Finally, the spatial coordination behavior that is the most important for this work is swarm aggregation, also called self-assembly or simply swarming in some sources. Berlinger *et al.* [57] used a swarm of fish-inspired robots that used emitting and sensing of blue light to implement different collective behaviors, including swarm aggregation. The theoretical models of swarm aggregation are based on two counteracting forces: attraction that keeps the agents together and repulsion that allows agents to avoid collisions [58], [59]. This problem is a bit simpler than flocking: the forces are fulfilling the first two rules by Reynolds and in this case the third one (velocity alignment) is unnecessary. The attraction and repulsion forces create an *artificial potential field*, which is used to control motion of agents.

Many researchers focus on studying stability of spatial coordination approaches. Gazi and Passino [60] and Fetecau *et al.* [61] analyzed stability of swarm aggregation based on potential fields. Tanner *et al.* [62] provided conditions that need to be fulfilled by the potential function to guarantee stability of the flock.

Realizations in Technical Systems

Self-organized spatial coordination can be used in multitude of applications, for example bird-like flocking is a robust way to move a big group of robots from one place to the other. Another example might be forming a pattern around a point of interest to observe it from each side. Therefore, the theoretical models are more and more often implemented in multi-robot systems. In these real-world implementations new problems need to be considered, like connectivity or movement constraints [63]. The book by Hamann [64] gives an overview of different approaches to achieve self-organized behaviors in robotics.

Turgut *et al.* [65] presented a self-organized flock of mobile robots. Their solution is based on heading alignment and proximal control, that allows avoiding collisions with other agents and obstacles. Ferrante *et al.* [66] extended this work and presented a method that allows flocking of mobile robots, but removed the requirement of explicit heading alignment. Hauert *et al.* [63] implemented a flock of fixed-wings drones based on Reynolds' rules. They estimated maximum turning angle of the flock based on connectivity and speed of agents. Vásárhelyi *et al.* [67] focused on the solution for flocking of aerial robots in confined spaces. They implemented a mathematical

model for flying robots proposed by Virágh *et al.* [68], taking into account communication capabilities, delays, perturbations and obstacles. All of these solutions were successfully implemented and tested in experiments featuring a swarm of robots.

A prominent example of robotic swarms is self-assembly of over a thousand robots realized by Rubenstein *et al.* [69]. The swarm, following only few simple rules, can recreate a shape that can be provided as a picture.

2.3 *Sync and Swarm: Swarmalators*

Although temporal and spatial coordination have a lot in common, the research considering these two phenomena remained disconnected for a long time. O'Keeffe *et al.* [6] created a model which bi-directionally couples two models: synchronization based on the Kuramoto model [19] and swarming based on potential fields. Two parameters are used to control how strong is the coupling: one specifies how phase similarity influences spatial attraction, the other describes influence of spatial proximity on synchronization. Depending on their values, model yields five different patterns (depicted in Figure 2.1). The particles acting according to this model are called *swarmalators* (a neologism for "swarming oscillators"). The model is described in details in Section 2.3.

In the followup work, O'Keeffe *et al.* [70] added a third parameter to the model, which controls how phase similarity influences repulsion in space. They analyzed smaller populations of swarmalators and ring states that emerge in this setup. These results were used to estimate the number of clusters formed in one of the previously introduced patterns (splintered phase wave, depicted in Figure 2.1d).

One of the patterns found by O'Keeffe *et al.* (active phase wave, presented in Figure 2.1e), was initially identified to appear only if the spatial proximity influences phase synchronization negatively [6]. Hong [71] showed that it can also appear if this influence is positive.

Lizarraga *et al.* [72] obtained new patterns in a swarmalator system with an additional, external force.

Jiménez-Morales [73] studied a swarmalator system with short range interactions (inspired by natural systems). They discovered six different collective behaviors, some of which did not appear in the systems governed by the original swarmalator model [6].

Swarmalator Model

The swarmalator model [6] is especially important for this thesis and we use similar concepts in the further parts, thus we present it here

in detail.

A swarmalator k has a position $\mathbf{x}_k \in \mathbb{R}^2$ in the two-dimensional space and an internal phase $\Phi_k \in [0, 2\pi)$. Its natural frequency of oscillation is ω_k. The position difference between two agents k and j is $\mathbf{x}_{jk} = \mathbf{x}_k - \mathbf{x}_j$ and their phase difference is $\Phi_{jk} = \Phi_k - \Phi_j$. The behavior of agent k in a system of N agents is modeled by two differential equations [6] shown in Box 2.1.

$$\dot{\mathbf{x}}_k = \frac{1}{N} \sum_{j \neq k}^{N} \left[\mathbf{I}_1(\mathbf{x}_{jk}) F_1(\Phi_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk}) \right]$$

$$\dot{\Phi}_k = \omega_k + \frac{K}{N} \sum_{j \neq k}^{N} \Gamma_1(\Phi_{jk}) \Lambda_1(\mathbf{x}_{jk})$$

Box 2.1: Mutually dependent movement and phase dynamics.

The function $\mathbf{I}_1$ describes the spatial attraction between two agents and F_1 determines how this attraction is influenced by their phase similarity. $\mathbf{I}_2$ describes the spatial repulsion between two agents. In a similar way, the synchronization is defined by two functions: Γ_1 is the attraction of the phases and Λ_1 is its dependency on the proximity of two agents. For swarmalators [6], the functions presented in Box 2.2 are used.

$$\mathbf{I}_1(\mathbf{x}_{jk}) = \frac{\mathbf{x}_{jk}}{\left\| \mathbf{x}_{jk} \right\|}$$

$$\mathbf{I}_2(\mathbf{x}_{jk}) = \frac{\mathbf{x}_{jk}}{\left(\left\| \mathbf{x}_{jk} \right\| - d \right)^2}$$

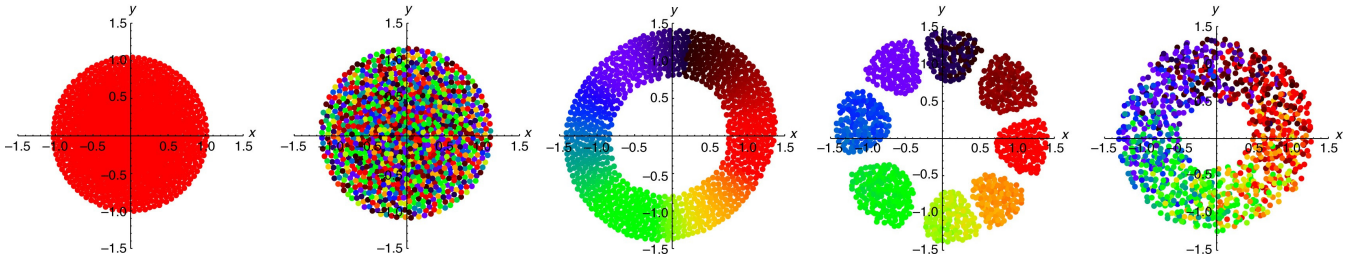
$$F_1(\Phi_{jk}) = 1 + J_1 \cos \Phi_{jk}$$

$$\Gamma_1(\Phi_{jk}) = \sin \Phi_{jk}$$

$$\Lambda_1(\mathbf{x}_{jk}) = \frac{1}{\left\| \mathbf{x}_{jk} \right\|}$$

Box 2.2: Interaction functions used in swarmalator model [6].

This system has two parameters: J_1 and K. The parameter $J_1 \in [-1, 1]$ determines how strong the influence of phase similarity on spatial attraction is. If J_1 is positive, an agent wants to be close to agents with the same phase. If J_1 is negative, an agent is attracted by agents with the opposite phase. The parameter $K \in \mathbb{R}$ determines how strongly



(a) Static sync (b) Static async (c) Static phase wave (d) Splintered phase wave (e) Active phase wave

Figure 2.1: Patterns obtained from sync and swarm model by O'Keeffe *et al.* [6]. Source: [6] (CC BY 4.0).

coupled the phases of two agents are. Positive values of this coupling strength tend to lower the phase difference between agents. Negative values tend to increase it.

Applying this model with different sets of parameters to a group of agents yields, after some time, one of the five different patterns (depicted in Figure 2.1):

Static sync The agents synchronize their phases and gather evenly distributed on a disk (Figure 2.1a).

Static async The agents are gathered on a disk, but their phases are evenly distributed and agents with similar phases tend to spread in space (Figure 2.1b).

Static phase wave The agents form an annulus and position themselves in a way that sorts their phases (Figure 2.1c). In a special case of this pattern, called ring phase wave [70] the agents are positioned on a circle.

Splintered phase wave The agents split in space into clusters with similar phases. The clusters are positioned on a ring. The agents keep moving within their clusters while their phases slightly change (Figure 2.1d).

Active phase wave The agents form an annulus, similar to the static phase wave, but their phases keep changing while they travel around the annulus to get close to the ones having a similar phase (Figure 2.1e).

The formed pattern depends on the choice of parameters J_1 and K. The details of this relationship are presented in Figure 2.2. From the plot we can see that static sync appears when $K > 0$ and static phase wave forms for $K = 0$ and $J_1 > 0$. When $K < 0$ and $J_1 < 0$ static async emerges. Perhaps the most interesting part of this plot is its second quadrant, which is split between three patterns. The simulation runs are marked with dots. The border between static

async and active phase wave (black line) is approximated, whereas the boundary between active phase wave and splintered phase wave is designated only based on the obtained simulation results and marked by connecting the dots with red dashed line.

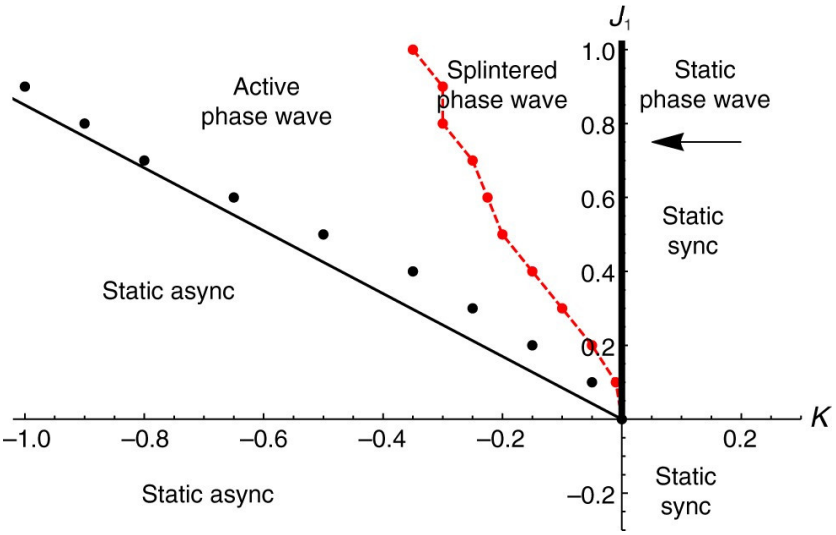


Figure 2.2: Relationship between values of parameters and pattern type. Source: [6] (CC BY 4.0).

Swarmalators in Nature

O'Keeffe *et al.* [6] identified some possible swarmalator systems in nature. One of these systems is a group of Japanese tree frogs. In the mating season, they tend to move away from the ones that are chorusing in sync with them and de-synchronize their chorusing from their neighbors [74]. Another candidate for a natural swarmalator are myxobacteria. Igoshin *et al.* [75] modeled the internal state of myxobacteria as an oscillator and confirmed in the experiments that this phase depends on spatial density of agents. The first confirmed system that employs coupling between the internal state and movement dynamics was found by Hrabec *et al.* [76]. They analyzed magnetic domain walls and found out that their internal state (polarization angle of the internal magnetic field) gets synchronized, which affects velocity.

Realizations in Technical Systems

O'Keeffe and Bettstetter [77] identified some possible technical systems in which "sync and swarm" concept could prove beneficial. The examples of such systems include autonomous transport systems and self-configuring antenna arrays. To the best of our knowledge our preliminary work on this topic [1] (also described in this thesis) is the

first that directly applies the idea of bi-directional coupling of temporal and spatial coordination in a technical system. Nevertheless, in many cases temporal and spatial coordination somehow benefit from each other, but the coupling between them appears only in one direction.

For example, synchronization can be exploited to control swarm behavior. Hartbauer *et al.* [78] proposed a swarm of synchronized agents that uses the emitted pulses for navigation. Robots close to the goal increase their oscillation frequency so that the other agents follow their signals. Christensen *et al.* [79] proposed a method for detecting faulty agents in a swarm. The robots periodically emit light. If the light on any agent is not detected in time, that agent is considered to be broken and its task needs to be taken over. A method proposed by Bezzo *et al.* [80] uses synchronization to detect changes in the network topology, which was shown to maintain the formation of robots.

The mechanisms of synchronization of coupled oscillators can be applied to stabilize collective motion of multiple robots [81]. Motivated by the application for underwater vehicles, Sepulchre *et al.* [82] presented a method based on the Kuramoto model [19] that allows stabilization of parallel and circular motion of self-propelled particles. Gao *et al.* [83] solved the same problem but based on pulse coupling. Depending on the desired behavior, they modify the phase response of the agents. Such a solution is easier to apply if communication is restricted to discrete instants of time.

3 Preparatory Work and Definitions

In this chapter we present technical background, assumptions and definitions important for the rest of this thesis. Parts of this chapter are taken from our published work [1], [3]. This chapter is structured as follows. In Section 3.1 we describe the robotic platforms introduced and used in the course of work on this thesis. Later we present our preliminary work on robots that sync and swarm in Section 3.2. Section 3.3 presents the conclusions drawn from the preliminary work and defines the requirements towards the model suitable for multi-robot systems. Section 3.4 is especially important for understanding the rest of this thesis. It contains definitions and common elements of the models introduced in the following chapters.

3.1 Robot Platforms

During the course of the KPK NAV project we needed to carry out experiments for our research and additionally prepare a few demonstrations. In order to achieve this, we prepared several robotic platforms. Each of them is tailored for a specific task and enables testing various aspects of swarming behaviors. In the following sections the platforms used in this thesis are described.

Balboa Ground Robots

Our first task was to create a platform for swarm robotics. The goal was to create a fleet of around twenty small, ground robots that can be operated in a small space. Further requirements included cost (around 200 € per robot) and low effort of assembly. In terms of computational power we wanted the platform to be capable of running *Robot Operating System* (ROS) and performing more complex tasks, like localization and basic computer vision. As at that time we did not know what exactly the robots will be used for, we needed a possibility of hardware extensions.

Most off-the-shelf platforms for swarm robotics are either too big and expensive (e.g., Turtlebot[1]) or lack computing power that

[1] https://www.turtlebot.com/

Figure 3.1: Pololu Balboa robots.

we needed (e.g., Kilobots [84] and Spiderinos [85]). We identified a few platforms that can be adjusted to fulfill these requirements and eventually selected the Balboa robots from Pololu[2]. A group of these robots is shown in Figure 3.1. One of their advantages for our project was good state estimation capability due to an integrated inertial measurement unit (IMU) and quadrature encoders connected to the motors. Each robot was extended with a RaspberryPi 3B+ to provide additional computing power.

[2] https://www.pololu.com/docs/0J70

The robots are using the wireless communication interface built in the RaspberryPi. Although the wireless modules on RaspberryPi 3B+ support IEEE 802.11ac, we use by default IEEE 802.11bg for the sake of compatibility with other devices. The modules are configured to work in an ad-hoc mode, which allows us to keep the system fully distributed and robust against failure or loss of some agents.

The official software library for the low-level controller (LLC) provided by the vendor has been tuned to our needs. We changed the communication interface between the computing board and the LLC to a serial connection using the UART (universal asynchronous receiver-transmitter) protocol. This enables us to remotely reprogram the microcontroller and implement a protocol for robot control and readout of measurements. We have tuned the controller responsible for balancing to make it work with additional load. All firmware modifications are published in the GitLab repository[3]. Finally, to

[3] https://gitlab.aau.at/aau-nav/development/balboa_llc_aws

enable visualization of the phase of an agent, we attached a strip of Neopixel RGBW LEDs (light-emitting diodes) to each robot.

For the purpose of this work robots need to know their positions in a global reference frame. Outdoors they could acquire their position with GPS, but as Balboas operate indoors, whenever global position is needed they use a *motion capture system* (OptiTrack). Therefore, each robots have four motion capture markers, in a unique configuration, attached to its bumpers. To assure that the solutions tested indoors are also applicable in the outdoor setting the robots use the motion capture system in the same way as they would use GPS, i.e., each robot acquires only its own position from the system, and, if needed, transmits the position to other robots.

The first field test of Balboa robots was a demonstration of firefly synchronization (see Video A.1 in Appendix A). We used their swinging as an additional visualization of phases. In this setting their self-balancing behavior was especially attractive.

Crazyflie Quadrocopter

The Crazyflies (Figure 3.2) were initially chosen only for demonstration purposes (see Video A.2 in Appendix A). The goal was to showcase our research during opening of the indoor flying area — Drone Hall Klagenfurt[4]. The platform is easily configurable with attachable decks. For our purposes we used two decks: one to mount a motion capture marker and the other one with an LED ring, both of them are visible in Figure 3.2. To facilitate work with a big group of drones in a motion capture system we used the Crazyswarm project[5].

[4] https://medium.com/dronehub/drone-hall-klagenfurt-9f478a15c5d

[5] https://github.com/USC-ACTLab/crazyswarm

The small size of Crazyflies and their relatively low price enabled us to prepare a demonstration featuring multiple drones in a confined space. On the other hand, minimalistic design and very low weight leads to limited computational capabilities and battery life. Usually, Crazyflies are used in offline experiments — following precomputed paths or performing some simple tasks on-board. In contrary, we wanted to present a live demonstration, with the paths computed online. Due to the limited capabilities of these drones in terms of inter-agent communication, we decided to additionally use a base station in our setup. The base station was connected to the OptiTrack system and was broadcasting the positions of Crazyflies to them. Additionally, it was running the sync and swarm model for each agent and sending a new state to it: phase to update the color of LEDs and position where it should move.

Even though this setup does not require inter-agent communication, we used it also afterwards in our work. It enabled us to test the implementations of the models introduced in this work (Section 6.5)

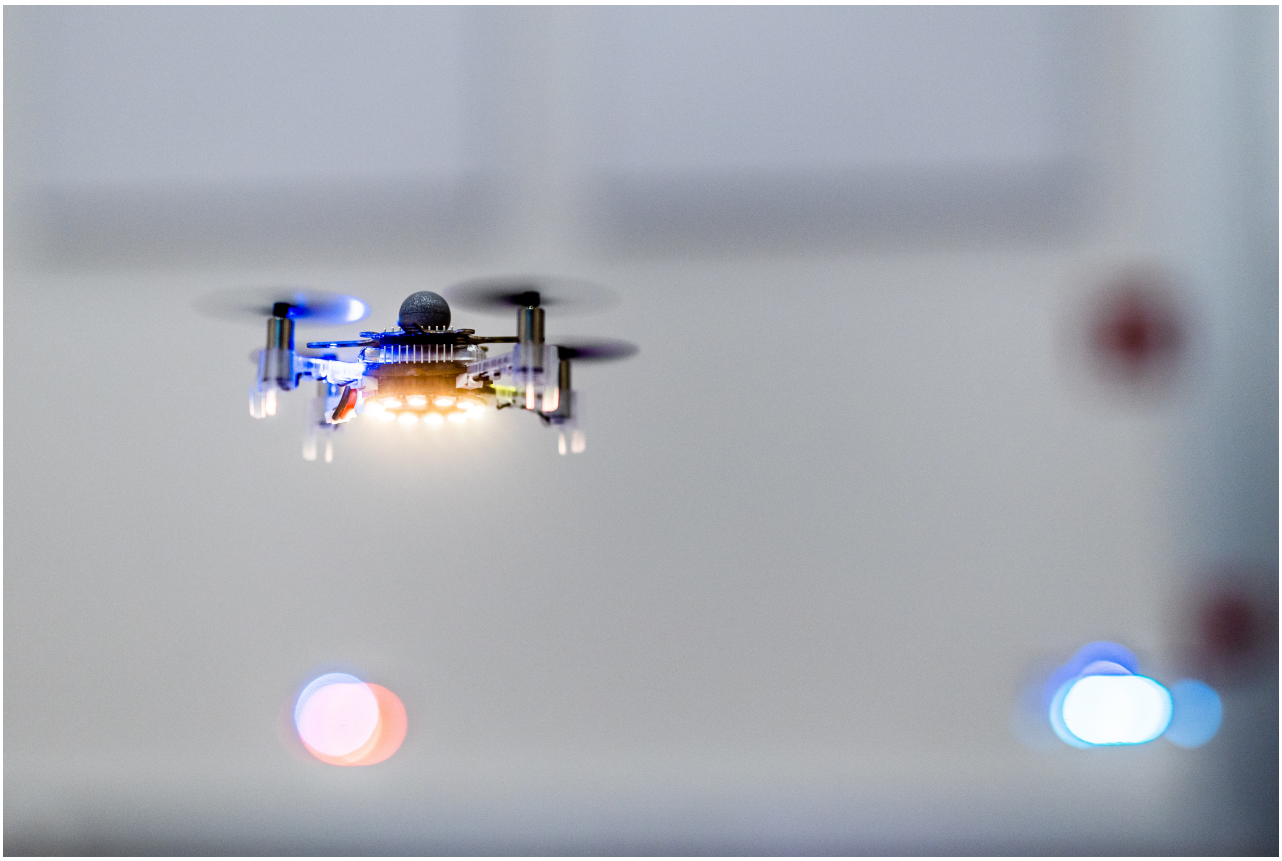


Figure 3.2: Crazyflie UAV. The UAV has a 130 mm diagonal (motor to motor) and weights 27 g without payload. Source: picture provided by the University of Klagenfurt.

with realistic movement dynamics of Crazyflies, but without communication constraints. This allowed to split the work on the real-world implementation into two steps, making it easier to switch from the simulation to experiments.

TwinSCIENCE Quadrocopter

To fulfill the goal of the KPK NAV project — perform a 3D reconstruction with a fleet of drones, we needed bigger, more capable drones. Unfortunately, the UAV platform previously used at the University of Klagenfurt became discontinued, therefore we looked for a new platform suitable for our work. We decided to use a custom drone built, according to our specification, by twins GmbH[6] — TwinSCIENCE quadrocopter[7] (Figure 3.3).

The TwinSCIENCE uses Holybro Pixhawk 4[8], which is a widely used flight controller. It is compatible with a variety of popular software like PX4[9], Ardupilot[10], QGroundControl[11] or MAVROS[12]. This UAV was planned to be used in various projects with different re-

[6] https://www.twins.co.at/en/english/

[7] Detailed specification of the TwinSCIENCE UAV: https://medium.com/dronehub/introducing-the-twinscience-v1-7b56b0c592db.

[8] https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk4.html

[9] https://px4.io/

[10] https://ardupilot.org/

[11] http://qgroundcontrol.com/

[12] http://wiki.ros.org/mavros

Figure 3.3: TwinSCIENCE UAV.

quirements. To ensure this possibility the platform should be easy to modify. Therefore, the drone has plenty of unoccupied space and mounting holes to attach additional hardware. By default the drone is equipped with a RaspberryPi 3B+, which allows us to use the same configuration, software and communication interface as on Balboa robots. There is also a possibility to exchange the companion board to a more powerful one for more demanding tasks. Finally, a relatively small size makes it possible to use the drone outdoors with GPS positioning as well as indoors with the motion capture system.

3.2 *Swarmalatorbots*

As a first step towards a solution for robots that couples temporal and spatial coordination we decided to use the swarmalator model [6] and apply it on robots as directly as possible. However, in order to do this we needed to introduce some modifications. First, we adjusted the parts that were necessary to use the mathematical

model in a robotic setting. Then, we added the possibility to align the orientations of the agents. In this section we describe the introduced modifications and the results that we obtained. We call robots acting according to the obtained model *swarmalatorbots* (a neologism for "swarmalator robots").

Modifications Due to Physical Constraints

The Balboa robots have the following physical limitations:

Movement constraints: A swarmalator in the original model can move freely in all directions. In contrast, the robot has non-holonomic constraints: it can only turn around its center or move in the direction it is facing, either forward or backward. It is therefore impossible to execute the velocity behavior as proposed in the swarmalator model (see Box 2.1). Instead, we interpret it as the *desired velocity*, denoted by $\mathbf{v}_k^{\mathrm{d}}$ for robot k. Additionally, we introduce a new state variable α_k for the orientation of the robot. The function $\mathcal{H}$ describes how the robot's orientation is influenced by its desired velocity. Finally, the velocity $\dot{\mathbf{x}}_k$ is the component of the desired velocity in the direction that the robot is facing. All equations are given in Box 3.1.

Speed limit: The speed of the robot is limited by its drivetrain. The linear and angular velocities obtained from the equations need to be limited appropriately.

Collision avoidance: A swarmalator is treated as a point-shaped agent whereas robots occupy a certain area. To prevent robots from colliding with each other, we define a circular safety area around each robot. The repulsion function ($\mathbf{I}_2$) is redefined using the distance between these safety areas, denoted by $\left(\left\|\mathbf{x}_{jk}\right\| - d\right)$. This ensures that the robots will repel each other if they get too close.

Alignment of Orientations

We also introduce an extension to the swarmalator model in which the orientations α can be aligned. The function $\mathcal{A}$ describes how the orientations of two agents are attracted. The function $\mathcal{G}$ determines how their spatial proximity influences this attraction. The attraction is controlled by the coefficient

$$\lambda = \min\left\{1, \frac{\left\|\mathbf{v}_k^{\mathrm{d}}\right\|}{PC}\right\}, \tag{3.1}$$

with the parameter $P \in \mathbb{R}_0^+$ defining the attraction strength and the constant $C \in \mathbb{R}^+$ depending on the movement specifics. If $P = 0$, the

agent will always turn to the direction of its desired movement. As P increases, the influence from other agents increases and finally, for $P \to \infty$, the agent will align only with its neighbors and never try to turn in the direction of the desired velocity.

The product PC can be seen as a threshold velocity above which an agent does not align with other agents but only with its desired velocity. A particular value is $P = 1$ because the threshold velocity becomes equal to C. In our experiments, C is set to be the maximum velocity of the robots. If the model tries to make an agent move faster than possible, this agent will only turn in the desired direction and remain unaffected by the orientations of other agents. As soon as the desired velocity drops and is achievable, the orientation will start to be influenced by others. At the full stop, only the orientation attraction will be considered.

Swarmalatorbot Model

$$\mathbf{v}_k^{\mathrm{d}} = \frac{1}{N}\sum_{j\neq k}^{N}\Big[\mathbf{I}_1(\mathbf{x}_{jk})F_1(\Phi_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk})\Big]$$

$$\dot{\alpha}_k = (1-\lambda)\frac{1}{N}\sum_{j\neq k}^{N}\mathcal{A}(\alpha_{jk})\mathcal{G}(\mathbf{x}_{jk}) + \lambda\mathcal{H}(\alpha_k, \mathbf{v}_k^{\mathrm{d}})$$

$$\dot{\mathbf{x}}_k = \mathbf{v}_k^{\mathrm{d}}\cos\alpha_k$$

$$\dot{\Phi}_k = \omega_k + \frac{K}{N}\sum_{j\neq k}^{N}\Gamma_1(\Phi_{jk})\Lambda_1(\mathbf{x}_{jk})$$

Box 3.1: Swarmalatorbot model.

The overall model is given in Box 3.1. We use $\mathbf{I}_1$, F_1, Γ_1 and Λ_1 from the swarmalator model described earlier. Orientation attraction is chosen to be the same as phase attraction:

$$\mathcal{A}(\alpha_{jk}) = \sin\alpha_{jk}, \tag{3.2}$$

$$\mathcal{G}(\mathbf{x}_{jk}) = \frac{1}{\left\|\mathbf{x}_{jk}\right\|}. \tag{3.3}$$

Furthermore, we use:

$$\mathbf{I}_2(\mathbf{x}_{jk}) = \frac{\mathbf{x}_{jk}}{\left(\left\|\mathbf{x}_{jk}\right\| - d\right)^2}, \tag{3.4}$$

$$\mathcal{H}(\alpha_k, \mathbf{v}_k^{\mathrm{d}}) = \sin\Big(\angle(\mathbf{v}_k^{\mathrm{d}}) - \alpha_k\Big). \tag{3.5}$$

Implementation in ROS 2

We implemented the swarmalatorbots in ROS 2 (Bouncy Bolson release) using eProsima Fast RTPS (Real Time Publish Subscribe) as a communication middleware. ROS 2 has been chosen despite its early development stage due to its distributed nature and configurable communication middleware. Multihop communication is provided by Babel routing protocol.

The original swarmalator model assumes that the interactions between oscillators take place continuously. Robots however can exchange only a limited number of messages at discrete points in time. Our swarmalatorbots periodically broadcast their states (i.e., messages containing their identifier, phase and position) with a configurable frequency. Each swarmalatorbot gathers the received messages and stores the last received update for every robot. The new control variables are calculated from the model (Box 3.1) by each swarmalatorbot when it receives information about its position. Between two published states, control variables might be updated multiple times. The implementation of the swarmalatorbot can run both in the simulation environment and on the robots in the experiment. For the simulation of more than 20 agents the message passing is substituted with intra-process communication to speed up computations. In order to make the development fast and the results easy to present, we have implemented a module for *visualization*. It listens to messages passed between swarmalatorbots and renders images. It enables us to visualize both the simulation and the real robots using the same software.

Resulting Space-Time Patterns

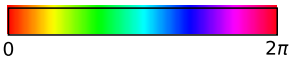
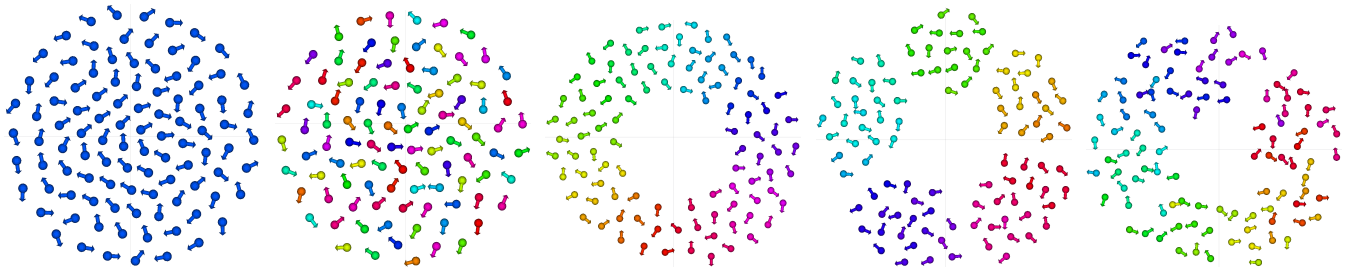


Figure 3.4: Color map used in plots.

We now study the behavior of N swarmalatorbots via ROS 2 simulations and experiments with our robotic platform. After initial placement, each agent acts as a swarmalator and interacts with all other agents. As time progresses, each agent changes its state in terms of position $\mathbf{x}$, orientation α and phase Φ. After some time, a snapshot of the resulting pattern is taken. The orientation of an agent is visualized by an arrow and its phase by a color. All tests are made with the natural frequency equal to 0 ($\forall k$: $\omega_k = 0$). The color map used here and throughout the thesis, whenever phase is visualized, is presented in Figure 3.4.

Simulation Results

Simulations are conducted for $N = 100$ agents, whose initial placement is sampled from a uniform random distribution in a square



(a) Static sync $(J_1, K, P) = (0.1, 1, 0)$.
(b) Static async $(J_1, K, P) = (0.1, -1, 0)$.
(c) Static phase wave $(J_1, K, P) = (1, 0, 0)$.
(d) Splintered phase wave $(J_1, K, P) = (1, -0.1, 0)$.
(e) Active phase wave $(J_1, K, P) = (1, -0.3, 0)$.

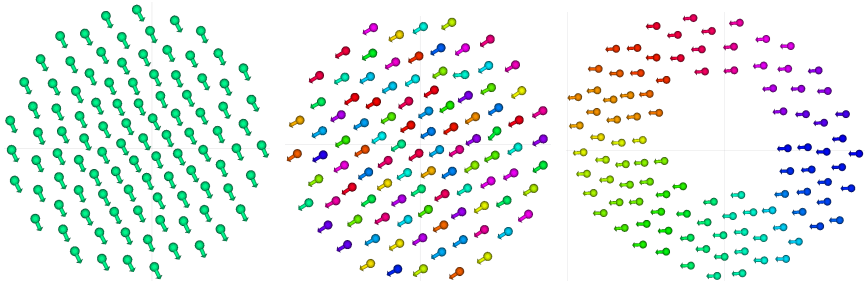
Figure 3.5: Patterns formed by swarmalators with movement constraints ($N = 100$ agents, time step $dt = 0.05$).

area with $x, y \in [-1, 1]$ length units. As a first step, we simulate the original swarmalator model for mutual validation. All five patterns emerging from the swarmalator model [6] are qualitatively reproduced. Next, we simulate the model adjusted to robots in order to check whether the modifications influence the capability to form patterns. Figure 3.5 shows some examples of patterns achieved without alignment of orientations ($P = 0$). We conclude that it is possible to obtain all the patterns in the swarmalatorbot model.

We observed that — even with static patterns — the swarmalatorbots are not fully stationary but their positions slightly oscillate. This behavior can occur due to the discretization in time of the model and because the simulation is distributed, i.e. the communication between agents is asynchronous and may suffer from message delay and loss. These phenomena can lead to disturbances in the patterns. Robustness of the model with discrete coupling and realistic communication conditions is one of the main issues addressed in this thesis.

The static patterns also emerge if we include orientation alignment ($P > 0$). Examples are presented in Figure 3.6. The two non-stationary patterns cannot, in general, be formed with our orientation alignment.

We observed that the introduction of movement constraints typ-



(a) Static sync $(J_1, K, P) = (0.1, 1, 0.1)$.
(b) Static async $(J_1, K, P) = (0.1, -1, 0.1)$.
(c) Static phase wave $(J_1, K, P) = (1, 0, 0.1)$.

Figure 3.6: Static patterns formed by swarmalators with movement constraints and orientation alignment ($N = 100$, $dt = 0.05$).

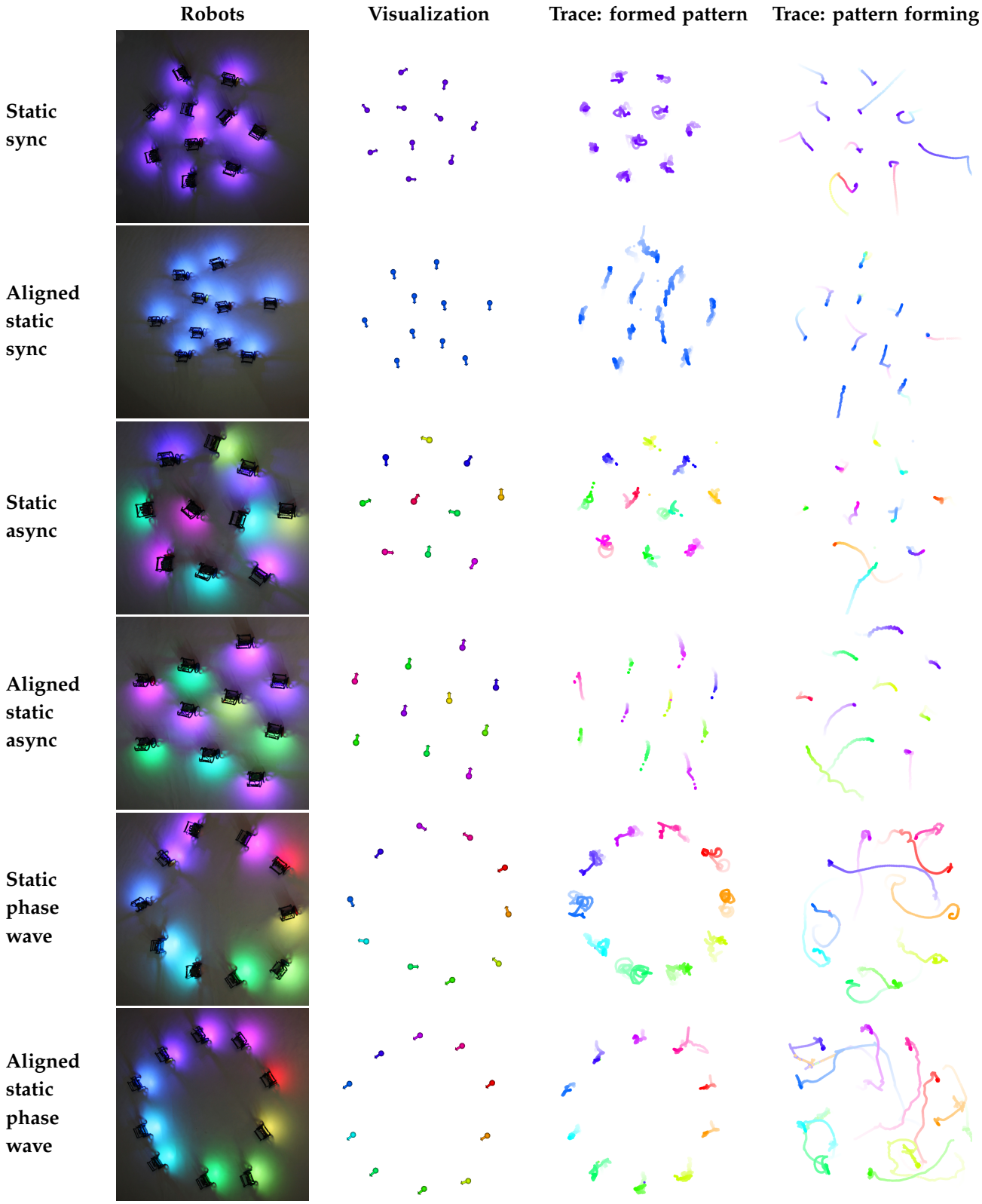
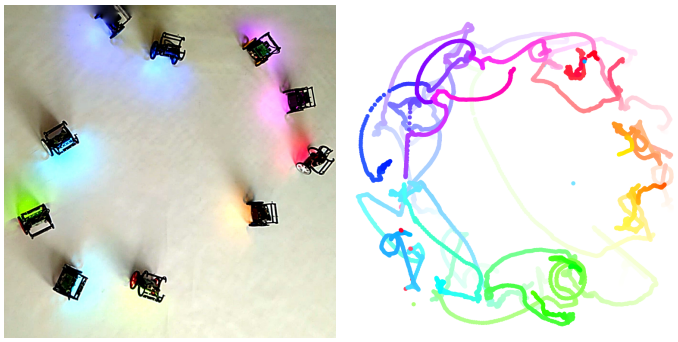


Figure 3.7: Stationary patterns formed by swarmalatorbots.



(a) (b)

Figure 3.8: Active phase wave formed by swarmalatorbots.

ically slows down the formation of the patterns. This also means that orientation alignment leads to even slower convergence. Once the pattern stabilizes, the variance in position is slightly higher with orientation alignment, as the correction of disturbances occurs slower.

Results With Robotic Prototype

Experiments are made with $N = 10$ robots running the model implemented in ROS 2. Our experiments are presented in Video A.3 (see Appendix A). Example results are depicted in Figures 3.7 and 3.8. All patterns except the splintered phase wave are reproduced. The splintered phase wave is not visible with our setup due to the small number of agents. For all stationary patterns (Figure 3.7), both versions with and without orientation alignment are created. Despite the small number of agents, the formed patterns are clearly visible and can be mapped to the different pattern types. Figure 3.7 contains four figures for each stationary state: the bird's eye view of the robots forming the pattern (*Robots*), snapshot from the visualization tool (*Visualization*), trace with the formed pattern, showing that the state is stationary (*Trace: formed pattern*) and the trace of pattern forming (*Trace: pattern forming*). Older samples are visualized with a lighter color. The active phase wave formed by robots is presented in Figure 3.8; the trace shows the movement of robots after the pattern has formed.

Disturbances similar to the ones in the simulations can be observed with the real robots. They become even more visible in the experiments because of the variable communication characteristics as well as the inertia and imperfect state estimation of robots.

3.3 *Lessons Learned*

Although the swarmalatorbot model can act as a proof-of-concept that temporal and spatial coordination can be mutually coupled in

a multi-robot system, during the course of the preliminary work we identified some problems that make this model unsuitable for robots. Therefore, we decided to propose a new model that addresses these problems. We call it *sandsbot* model (a neologism for "synchronizing and swarming robot"). Here we present a short summary of the problems found in the swarmalatorbot model with the comment how are they addressed in the sandsbot model and references to detailed descriptions.

Coupling

In the swarmalator model continuous and instant coupling is assumed. Meaning that the current states (phases and positions) of other agents are known at each point in time without any delay. In practice none of these assumptions hold: the agents can communicate only at discrete moments and the communication process always leads to some delays. In the swarmalatorbot model we used straightforward discretization — the model was evaluated periodically with the highest frequency that could have been handled by our communication interface. In our experiments we noticed that such an approach combined with communication delays leads to oscillations of positions of agents, which in turn destabilizes static states. In the remainder of this thesis we overcome the problems with communication by allowing agents to communicate at discrete moments in a synchronized manner. The details of this approach are presented in Section 3.4.

Furthermore, the very low frequency of communication can cause agents to collide. Therefore, we dynamically limit their speed based on the frequency of updates. This method is described in Chapter 5.

Reproducability of Patterns

In the swarmalator model the final shape of the pattern depends on the initial conditions. This is especially visible in two cases:

1. In static phase wave there are no phase interactions between agents. This means that they are just sorting their initial phases. This state would look like static sync if swarmalators start with synchronized phases, instead of uniformly distributed. In a robotic system with 10 to 20 agents drawing initial phases from a uniform distribution may lead to asymmetric formation, because there is a high chance that with such a small sample the obtained phases are not evenly distributed.

2. The exact number of clusters in splintered phase wave is unknown. It can be roughly estimated using heuristic algorithms [70],

but even then the result is a range instead of a precise number. Furthermore, the number of clusters typically is too high compared to the number of agents we are considering. Consequently, this state cannot be observed in a small robotic swarm.

For robotic applications it is necessary to be able to control the pattern independently of the initial conditions, so the same pattern can form at any stage of the mission. Additionally, the possibility to control the number of clusters allows to tailor the patterns for the specific task. Therefore, in this work we introduce the temporal coordination model (Chapter 4) and later couple it with spatial aggregation (from Chapter 5) to achieve "controllable" version of the sync and swarm model, i.e., in which the same pattern will form regardless of the initial conditions and with the specified number of clusters.

Size and Movement of Agents

Swarmalators are treated as point-shaped freely moving particles. Such approach guarantees only that the agents cannot move to exactly the same place, whereas it does not preserve any minimal distance between agents. This means that the agents in a multi-robot system could still be attracted and move towards each other even if this would cause a collision. We initially solved this problem in the swarmalatorbot model by treating robots like disks of a given size and calculating repulsion between these disks instead of their centers. In the sandsbot model we further improve it by constraining the minimal distance that should be kept between agents. The details of the collision-free movement are described in Section 3.4 and Chapter 5. Additionally, robots are not able to perform unrestricted movement. The movement capabilities of robots were already taken into account in the swarmalatorbot model and are later used in the experiments with the sandsbot model.

Global Interactions

The swarmalator model assumes global knowledge of the states of all other agents. In the robotic system it might not always be achievable — some messages might be dropped or robots can be out of range. We have done some initial analysis of limited range of interactions of swarmalatorbots. For a subset of initial conditions the pattern was not fully converging, e.g., multiple synchronized clusters were formed in static sync or band-type pattern was formed in static phase wave. It was also confirmed by Lee *et al.* [86] who described how the patterns formed by swarmalators are deformed when the interaction

range is limited. These variations of the original states are undesirable in robotic systems. Therefore, in this work we assume that robots are working in proximity and can always exchange messages. Nevertheless, our experiments show that the system can recover from a limited message drop (see Section 6.5).

3.4 *Model Tailored for Robots*

In the remainder of this work we present a new model tailored for robotic systems — sandsbot model — solving the problems described above. Here we present the basic concepts needed for sandsbots to work.

Definitions

Each sandsbot is modeled as an agent $k \in \{1, \ldots, N\}$. It has an internal oscillator, whose value is represented by a phase $\Phi_k(t) \in [0, 2\pi)$ evolving over time t. The time index t is skipped if not needed. The phase difference between k and another agent j with phase Φ_j is denoted by Φ_{jk}. The spatial position of an agent is $\mathbf{x}_k \in \mathbb{R}^2$. The position difference vector from k to j located at $\mathbf{x}_j$ is $\mathbf{x}_{jk} = \mathbf{x}_j - \mathbf{x}_k$ and the Euclidean distance between them is $\left\|\mathbf{x}_{jk}\right\|$.

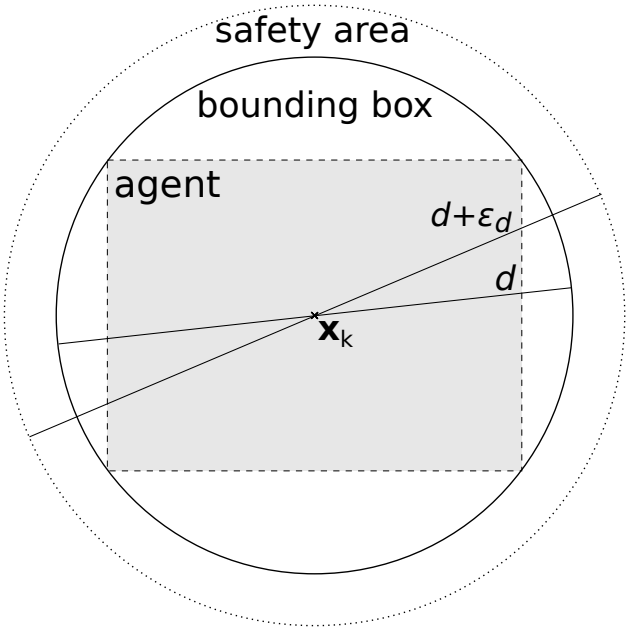


Figure 3.9: Safety area of agent k.

To guarantee collision-free operation, we use a *safety area* around each agent (presented in Figure 3.9). This area comprises a circular *bounding box* with diameter d and an additional *safety margin* ϵ_d

defining the minimum distance that needs to be kept between the bounding boxes of agents at all times. The safety area of agent k is thus a circle of diameter $d + \epsilon_d$ centered at $\mathbf{x}_k$.

Time-Discrete Interactions

Each agent must repeatedly share its state (phase and position) with other agents. In real-world systems, these state updates can occur only at discrete points in time, and each message experiences processing and propagation delays. Moreover, the fact that the phase Φ_k is constantly changing and, if the agent is moving, the position $\mathbf{x}_k$ is changing, a simple sharing of the state would lead to the situation that the up-to-date state is unknown to the receiver at the time of reception. To tackle these issues, we split the phase into two components: a discrete phase θ_k and an oscillatory part ϕ_k, formally $\Phi_k = \phi_k + \theta_k$.

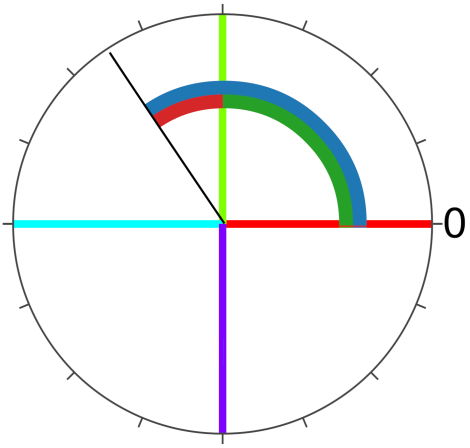


Figure 3.10: Phase represented as a rotating unit vector. Phase Φ_k is marked with blue arc, discrete phase θ_k with green arc and oscillatory part ϕ_k with red arc. Four lines in different colors mark the possible values of discrete phase.

An oscillator is often represented as a rotating unit vector. We use this representation to visualize the split of the phase in Figure 3.10. The vector is marked with a black line and the phase values are marked with colored arcs: blue shows phase Φ_k, green — discrete phase θ_k and red — oscillatory part ϕ_k. The possible values of θ are marked with four lines of different colors. The phase evolution in time is presented in Figure 3.11. The discrete phase θ_k is exchanged between agents rather than the overall continuous phase Φ_k. The value of θ_k remains constant throughout a period T, so the message can be sent and received by other agents before the value changes. To operate on integer-valued phases, we introduce the *phase level* $\hat{\theta}_k \in \{0, 1, \ldots, L-1\}$ with L being the number of phase levels. The discrete phase is then defined as $\theta_k = \frac{\hat{\theta}_k}{L} 2\pi$.

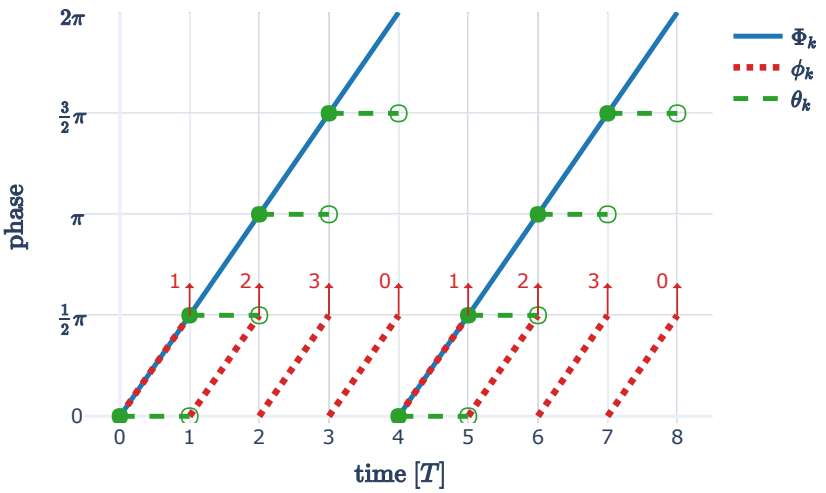


Figure 3.11: Phase of an agent Φ_k with its components θ_k and ϕ_k over normalized time. Arrows show the moment a signal is emitted; the numbers next to them represent the phase level.

To maintain consistency, sandsbots operate in synchrony. For this purpose, we use the oscillatory part $\phi_k \in \left[0, \frac{2\pi}{L}\right)$ as the *internal clock*, which oscillates with period T and is synchronized throughout the whole system. Such synchronization can be achieved by an established technique. In our implementation we tested two methods: firefly synchronization [22] and chrony[13] (which realizes Network Time Protocol (NTP)). Both approaches successfully synchronized the internal clocks. Each oscillation cycle is split into three parts:

[13] `https://chrony.tuxfamily.org/`

- Whenever $\phi_k = 0$, the agent updates its state, computes the *predicted* state (phase level and predicted position) for the end of the oscillation cycle, and sends a message containing this information.
- For ϕ_k from 0 to a threshold $\phi_U \in \left[0, \frac{2\pi}{L}\right)$, the agent gathers messages containing states of the other ones.
- For ϕ_k from ϕ_U to the maximum value $\frac{2\pi}{L}$, the agent calculates the new update of its state based on the received predicted states and its own predicted state.

The threshold ϕ_U can be chosen depending on the expected delays and the computational effort of calculating a new update (and thus the time needed). High values ensure that even delayed messages are taken into account but leaves less time to calculate the update; low values leave more time for the calculations with the risk of missing some delayed messages. The predicted position is calculated based on the current position, period length, and calculated velocity.

An agent can change its phase immediately but its position only slowly by setting a new velocity. Thus, the output of the sandsbot model (state update) consists of a new phase level and velocity.

Our model assumes that messages suffer from delays and takes these delays into account. If a message arrives before ϕ_k reaches the threshold ϕ_U, the agent uses the up-to-date data to calculate the next update (as messages contain a prediction, not the past state). Messages that arrive after reaching the threshold are dropped. This means that delays shorter than $T\phi_U / \frac{2\pi}{L}$ do not influence the performance. In our experiments, the period length T and threshold ϕ_U are chosen based on the measured latency in the network to ensure that the vast majority of messages will be delivered before the threshold ϕ_U is reached.

4 *Temporal Coordination*

In this chapter we introduce a temporal coordination model. It was designed as a building block of the sandsbot model, but it can also be used as a standalone model to coordinate agents in time. Parts of this chapter are taken from our published work [2], [3].

Multi-agent systems often require agents to perform actions in a simultaneous way. For instance, a swarm of robots performing cooperative stereo vision needs to be synchronized to achieve good depth resolution. Centralized approaches are not always applicable, which is why distributed algorithms for mutual synchronization have been developed. Examples include extensions and modifications of well-known models for self-synchronizing coupled oscillators, such as the Kuramoto model [19] and the Mirollo-Strogatz model [22]. However, in some cases, synchronization to the same oscillator phase is not required and even undesirable, but the agents should be in *anti-phase*. For example, during a surveillance mission, aerial robots should fly to a charging station at different times to avoid congestion. This problem is often solved by planning algorithms, although self-organizing solutions offer advantages in terms of robustness, flexibility, and adaptability (e.g., agents can join and leave the mission at any time).

In some applications, an even more complex temporal behavior than anti-phase holds interest. For example, two aerial robots cooperatively delivering a package should simultaneously take off but avoid flying into the warehouse at the time when another group is inside. Time coordination can also be used indirectly to manage the behavior of agents. For instance, splitting of an aerial swarm flying in formation into phase-dependent clusters could facilitate obstacle avoidance [87].

In conclusion, there is a need for coordination models that enable the distributed control of timing and the creation of temporal patterns. In this chapter we propose and evaluate such a model. Our approach is based on the well-known concept of coupled oscillators but also takes into account the practical particularities for implemen-

tation in technical systems, such as signaling overhead and delays.

Robots are often connected with a wireless network that is used by multiple programs running on each robot. Temporal coordination is only one of these programs. An efficient temporal coordination method should therefore lead to a low signaling overhead. Models based on permanent information exchange (e.g., utilizing continuous phase coupling) are not applicable in practice for such systems.

Communication always introduces delay between the instants when the message was sent and received. Whenever a robot receives a phase transmitted by another oscillator, it will actually receive an outdated value from a moment in the past. In addition, the more devices that share the same medium, the higher the communication delays and usually jitter (variation of delays). Hence, a temporal coordination model robust against such variable delays would facilitate the operation of large groups of robots.

Related Work

Coupled oscillators are extensively studied from the perspective of theory (e.g., [31], [88]) and engineering (e.g., sensor networks [40], radio communications [38], and robotics [36]). The ultimate goal in these papers is always to achieve synchronization, i.e., to align the phases of all oscillators. The opposite — repulsive coupling of oscillators — is rarely studied. There is some work on this topic, albeit it is mostly theoretical and focuses on the coexistence of repulsive and attractive coupling (see [23], [24]). There is also work on applications, e.g., repulsive coupling between clusters of nodes in a network [89].

A generalization of oscillator coupling allowing the formation of splay or clustered patterns has been analyzed in theory (see [28], [90]) and sometimes with potential applications in technology (e.g., multi-agent systems [42] and networks with different topologies [30]). Clustered and splay patterns are often considered as unwanted equilibria; therefore their stability is extensively examined for both continuous coupling [31] and pulse coupling [27], [32], [33].

The task of phase discretization has also been considered in the literature, although the models are either limited to an arbitrary number of phase levels [91] or they focus on stochastic interactions between oscillators [92]. Additionally, these approaches only regard continuous coupling.

Coupled oscillators are also employed in distributed control algorithms, where not only synchronization but also splay and clustered patterns hold interest [82], [93], [94]. Many approaches focus on multi-agent systems, they take into account connectivity (see [95], [96]) or propose discrete time models to permit the exchange of mes-

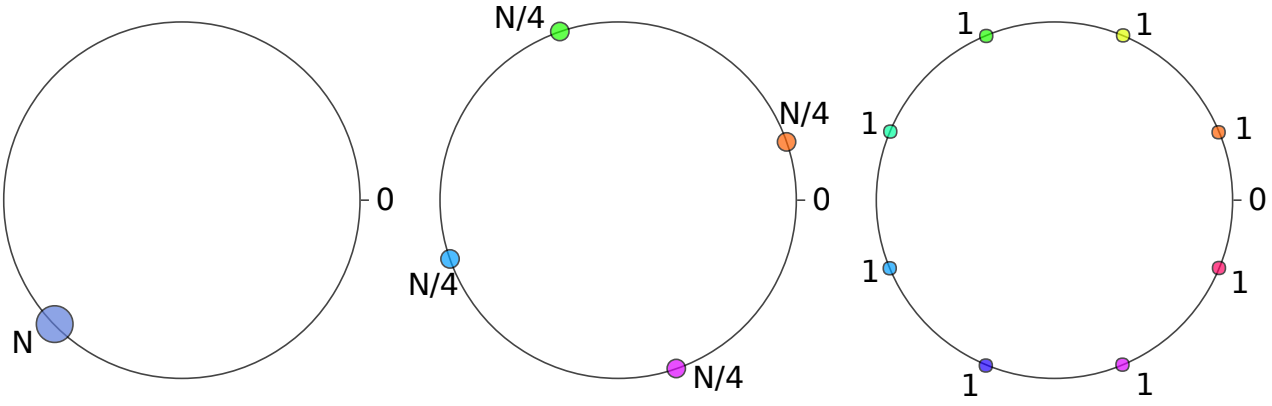
sages [97]. There is also some work undertaken to cope with delayed messages [98].

Contributions

To the best of our knowledge, there is no coupled oscillator model that enables the formation of synchronized, splay, and clustered patterns while also being robust to delays. We contribute to this topic by introducing a unified temporal coordination model that is discrete in both time and phase and enables us to obtain synchronized, splay, and clustered patterns. Furthermore, we present a way to put the system off unwanted equilibria, we verify the model in simulation, and finally demonstrate its feasibility in real-world experiments using small mobile robots.

The remainder of this chapter is structured as follows. Section 4.1 introduces the temporal patterns with example applications of each of them. Section 4.2 describes the temporal coordination model. Section 4.3 contains the simulation-based analysis. Section 4.4 presents experiments with robots, and finally Section 4.5 concludes.

4.1 Temporal Patterns



(a) Synchronized pattern.

(b) Clustered pattern.

(c) Splay pattern.

Figure 4.1: Schematic representation of temporal patterns.

Different applications require different temporal arrangements. We are interested in three of them: the *synchronized* pattern (Figure 4.1a), with all oscillators having the same phase level; the *splay* pattern (Figure 4.1c), with oscillators evenly spaced in the phase domain; and the *clustered* pattern (Figure 4.1b), with oscillators forming evenly-spaced clusters of equal size in the phase domain. Note that even though in these patterns phase levels ($\hat{\theta}$) may differ, internal clocks

(ϕ) are not influenced and remain synchronized.

For a given number of oscillators N, the three patterns can be described in a unified way by defining the number of clusters M that oscillators should form: $M = 1$ for the synchronized pattern, $M = N$ for the splay pattern, and $1 < M < N$ for the clustered pattern We consider only patterns with clusters of equal size, i.e., $M \mid N$ (M divides N). Furthermore, we assume that $M \mid L$ (recall L is the number of phase levels), which allows maintaining equal distances between clusters. We call these patterns *M-cluster patterns*.

The proposed model can be generalized, whereby each oscillator can maintain multiple phase levels with one internal clock. The multiple phase levels of a single oscillator should not influence each other but the corresponding levels of other oscillators. With this approach, a single oscillator can be a part of multiple patterns at the same time. There should always be only one internal clock, so all signals will be sent at the same time, and thus they can be merged in one message.

Let us outline potential applications, one for each pattern:

Synchronized patterns can be used for agents that act simultaneously. For example, robots take pictures at the same moment to assure that the objects are at the same place in all of the pictures, even if the scene is dynamic. Later the pictures can be used for image stitching or 3D reconstruction.

Splay patterns can be used for agents that perform asynchronous actions. For example, robots send measurements in bundles to a base station in different time slots to avoid interference. Each robot transmits when its phase level is equal to 0.

Clustered patterns can be used to split agents into groups performing actions asynchronously with other groups. For example, based on the phase levels, robots split into M groups (those with the same level are in one group). They can only exchange messages (e.g., pictures or measurements) when their phase level is 0 to avoid inter-group interference.

4.2 Model

Order Parameters

The phase θ_k of an agent k can be written as a complex number $e^{i\theta_k}$. A freely running oscillator is thus modeled as rotating vector of unit length.

A measure of synchrony is the complex *order parameter* [19], presented in Box 4.1.

$$z = \frac{1}{N}\sum_{k=1}^{N} e^{i\theta_k}$$
$$r = |z|$$

Box 4.1: Synchronization order parameter.

The magnitude r reveals the state in which the system is: It reaches the maximum value ($r = 1$) in a synchronized state, i.e., when all oscillators have equal phases. When oscillators achieve a *balanced state*, their phase centroid coincides with the point $(0, 0)$ on the complex plane, and thus r reaches its minimal value ($r = 0$).

Splay and clustered patterns are special cases of a balanced state. In order to distinguish them, we use the order parameters of higher phase harmonics [93] z_m presented in Box 4.2, with $m \in \mathbb{N}$ and $m \leq M$. It can also be thought of as the m-th moment of the N

$$z_m = \frac{1}{Nm}\sum_{k=1}^{N} e^{im\theta_k}$$
$$r_m = |z_m|$$

Box 4.2: Complex order parameter of higher phase harmonics.

phases. The magnitude r_m is the radius of the centroid of the m-th harmonic of phases. Similar to the first order parameter, if $r_m = \frac{1}{m}$, the m-th phase harmonic is synchronized. In the balanced state, its centroid is $r_m = 0$. In order to achieve the desired number of clusters, the first $M - 1$ harmonics must be in a balanced state and the M-th harmonic must be in synchrony ($r_M = \frac{1}{M}$). To give some examples, synchronized agents have $r_1 = 1$, a two-cluster pattern yields $r_1 = 0$ and $r_2 = \frac{1}{2}$, and a three-agent splay pattern has $r_1 = r_2 = 0$ and $r_3 = \frac{1}{3}$.

State Potential

Based on the order parameter, we define the potential U_m of the m-th phase harmonic similarly as in [82]:

$$U_m = \frac{N}{2} K_m r_m^2, \tag{4.1}$$

where K_m is the coupling strength of the m-th harmonic.

If $K_m < 0$ the potential reaches its minimum if the phases of the m-th harmonic are synchronized. In contrast, if $K_m > 0$ the phases of the m-th harmonic need to be balanced for U_m to be minimal. To have the first $M-1$ harmonics balanced and the M-th harmonic synchronized, we set $K_m > 0$ for $m \in \{1, \ldots, M-1\}$ and $K_M < 0$.

The sum of potentials of M harmonics describes the potential of the M-cluster pattern [82]:

$$U^{(M)} = \sum_{m=1}^{M} U_m. \tag{4.2}$$

which reaches its global minimum if each U_m is minimized. For the given coupling strengths, this minimum corresponds to the M-cluster pattern [82].

Continuous Phase Coupling

In a system of agents with continuous phases, the overall potential can be minimized with gradient control [82]:

$$\dot{\Phi}_k = \omega_k - \frac{1}{N} \frac{\partial U^{(M)}}{\partial \Phi_k}, \tag{4.3}$$

where ω_k is the agent's natural frequency of oscillations. The derivative can be defined as the sum of phase coupling functions $\Gamma(\Phi_{jk})$ between the agents [82]:

$$\frac{\partial U^{(M)}}{\partial \Phi_k} = \sum_{j=1}^{N} \Gamma(\Phi_{jk}) \,. \tag{4.4}$$

This results in a phase coupling function being a linear combination of the continuous couplings, similar to the Kuramoto model [19], for the first M phase harmonics [82]:

$$\Gamma(\Phi_{jk}) = \sum_{m=1}^{M} \frac{K_m}{m} \sin\Big(m\Phi_{jk}\Big) \,. \tag{4.5}$$

From Continuous to Discrete Phase Coupling

The model presented until now is valid for continuous phase and continuous time. For discrete phase θ_k and discrete time, we propose a *phase interaction function*:

$$\Psi_k(\theta_k) = \frac{1}{N} \sum_{j=1}^{N} \Gamma(\theta_{jk}) \,, \tag{4.6}$$

which uses the same phase coupling function $\Gamma(\cdot)$ but now evaluated for discrete phase values.

We split the phase coupling function into two additive parts based on the value of K_m:

$$\Gamma(\theta_{jk}) = \Gamma_1(\theta_{jk}) + \Gamma_2(\theta_{jk}), \tag{4.7}$$

$$\Gamma_1(\theta_{jk}) = \sum_{m \in \mathcal{M}_1} \frac{K_m}{m} \sin\Big(m\theta_{jk}\Big), \tag{4.8}$$

$$\Gamma_2(\theta_{jk}) = \sum_{m \in \mathcal{M}_2} \frac{K_m}{m} \sin\Big(m\theta_{jk}\Big), \tag{4.9}$$

with $\mathcal{M}_1 = \{m \mid K_m \leq 0\}$ and $\mathcal{M}_2 = \{m \mid K_m > 0\}$, which corresponds to $\mathcal{M}_1 = \{M\}$ and $\mathcal{M}_2 = \{m \mid 1 \leq m < M\}$ for the M-cluster pattern. Γ_1 can be thought of as the *phase attraction function* that tries to synchronize harmonics belonging to $\mathcal{M}_1$, whereas Γ_2 is the *phase repulsion function* that balances the other phase harmonics. The split into attraction and repulsion gives the phase coupling a similar structure as the position coupling used for spatial coordination (Equation (5.5)) and enables us to combine the two models (Chapter 6).

The discrete temporal coordination operates on the integer phase levels. The phase interactions are usually too minor for an agent to immediately jump to the next phase level, especially if the system is close to reaching the desired pattern. Thus, each agent integrates these interactions over time and keeps them as *phase level correction* $\delta\hat{\theta}_k$. Whenever the internal clock resets ($\phi_k = 0$) the phase is updated. The phase level $\hat{\theta}$ is always incremented by 1 and the phase level correction rounded towards 0 is applied:

$$\hat{\theta}_k[t+1] = \Big(\hat{\theta}_k[t] + 1 + \operatorname{sgn}(\delta\hat{\theta}_k[t])\, \big\lfloor \big|\delta\hat{\theta}_k[t]\big| \big\rfloor\Big) \bmod L\,. \tag{4.10}$$

If the rounded phase level correction was non-zero (i.e., phase level correction was large enough to change the phase by a full level), it is reset ($\delta\hat{\theta}_k[t] = 0$).

Unwanted Equilibria

A continuous system with the coupling function Γ defined in Equation (4.5) has multiple equilibrium points. Only a subset of them is stable and corresponds to a desired M-cluster pattern [82]. Other equilibria are unstable in systems with a continuous phase. For instance, a synchronized pattern is an equilibrium for splay and clustered patterns, as in this pattern all phase differences are 0, implying $\forall k, j\colon \Gamma(\theta_{jk}) = 0$. Note that stable equilibria points coinciding with discrete phase levels exist if $M \mid L$. Hence, it is possible to achieve an M-cluster pattern in a discretized system.

The phase discretization strongly increases the probability of the system being in one of the equilibria. For example, consider a system

with $L = 4$ phase levels and $N = 2$ designated to reach synchrony. If the oscillators start with random phase levels, there is a 25% chance that they will start exactly in anti-phase ($\hat{\theta}_k = \hat{\theta}_j + \frac{L}{2}$), which is an equilibrium for the synchronized pattern.

The goal of the discretization of the temporal coordination is to prevent the influence of disruptions (e.g., communication delays, oscillators imperfections or noise). If the system is in equilibrium, all interactions cancel out. As a result, there is no influence that can unbalance the system. Thus, in order to break the unwanted clusters and equilibria, we introduce noise η and *energy* of state E.

If two oscillators have the same phase level, they are equally influenced by others and will thus always change their phase levels in the same moment, even if they should be separated to achieve the desired pattern. The role of noise is to slightly variate interactions between oscillators. In the described example, noise can cause the oscillators to change their phase levels in different time steps.

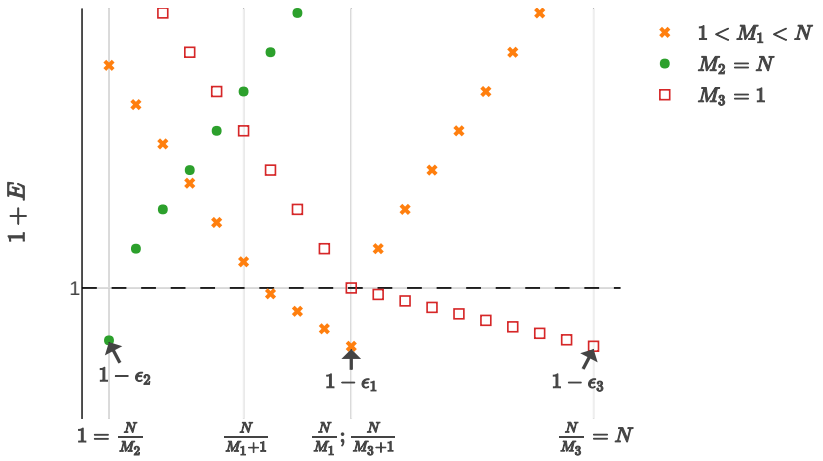


Figure 4.2: Shape of energy function for different sizes of clusters. $-\epsilon_k$ denotes minimal energy.

The energy of the state has two roles: stabilizing the expected pattern and breaking unwanted clusters or equilibria. First, it suppresses phase correction (originating from both oscillators interactions and noise) of the oscillators that create clusters of size N_C meeting the condition $\frac{N}{M+1} < N_C \leq \frac{N}{M}$. Accordingly, oscillators that form clusters of proper size (or almost proper, if there are still too few oscillators in the cluster) are more reluctant to change their phase levels. Therefore, noise and some small deviations of the system state from the desired one will not cause them to change their phase levels. Second, if the cluster is too large, the energy amplifies the phase correction, causing the cluster to break. If the system is in an unstable equilibrium, only noise influences phase correction. In this case,

energy amplifies the noise, making it easier for oscillators to leave this state. Additionally, oscillators in significantly too small clusters ($N_C \leq \frac{N}{M+1}$) are more "eager" to change their phase level and "look for" a larger cluster. This property is not crucial for achieving the desired pattern but speeds up this process. Examples of energy functions for different cluster sizes are presented in Figure 4.2.

The energy is used to either suppress or amplify the previous phase correction. The modified equation of the phase update considers noise and energy is thus:

$$\delta\hat{\theta}_k[t] = (1+E)\cdot\delta\hat{\theta}_k[t-1] + \eta + \frac{1}{N}\sum_{j=1}^{N}\Gamma\left(\theta_{jk}[t]\right). \tag{4.11}$$

Constraints

In order to assure that energy and noise have only a limited impact on the system state, we introduce the following constraints:

Energy: Phase correction suppression should never be too strong. Even minimal interactions with other oscillators should eventually lead to changing the phase level. We assume that the phase level of an oscillator is incorrect if it is influenced by at least minimal non-zero force of at least $0.5N$ other agents. In other words, the largest correction loss, caused by energy, occurs when correction approaches 1. Even at this point, the loss should be lower than half of the minimal non-zero influence. Thus, we can formulate the energy constraint as:

$$(1-\epsilon) + \tfrac{1}{2}\min_{\substack{\theta_{jk},\\ \Gamma(\theta_{jk})\neq 0}}\left|\Gamma(\theta_{jk})\right| > 1, \tag{4.12}$$

where ϵ is *energy margin*, and $-\epsilon$ is the minimum value of energy, as marked in Figure 4.2.

Noise: If the oscillator is in the desired state (size of its cluster is proper), the energy should suppress phase correction sufficiently strongly to prevail noise. As a result, noise cannot cause an oscillator to change its phase level if it is not subject to other influences and the cluster is of proper size. We can formulate it as follows:

$$(1-\epsilon) + \eta < 1. \tag{4.13}$$

Number of phase levels: We consider only the number of phase levels, such that $M \mid L$. Moreover, the condition $L \geq 3M$ needs to be fulfilled. Otherwise, for $L = M$ and $L = 2M$, the coupling of the M-th harmonic is only sampled at the points where its value is 0.

Based on the constraints from Equations (4.12) and (4.13) we can calculate the minimal energy ϵ and the maximal value of a random noise. Moreover, we can determine the number of phase levels needed to form the desired M-cluster pattern.

To summarize, we present the overall model in Box 4.3.

$$\Psi_k(\theta_k) = \frac{1}{N}\sum_{\substack{j=1\\ j\neq k}}^{N} \Gamma_1(\theta_{jk}) + \Gamma_2(\theta_{jk})$$

$$\Gamma_1(\theta_{jk}) = \sum_{m\in\mathcal{M}_1} \frac{K_m}{m}\sin\Big(m\theta_{jk}\Big)$$

$$\Gamma_2(\theta_{jk}) = \sum_{m\in\mathcal{M}_2} \frac{K_m}{m}\sin\Big(m\theta_{jk}\Big)$$

$$\delta\hat{\theta}_k[t] = (1+E)\delta\hat{\theta}_k[t-1] + \eta - \Psi_k\left(\theta_k[t]\right)$$

$$\hat{\theta}_k[t+1] = \Big(\hat{\theta}_k[t] + 1 + \mathrm{sgn}(\delta\hat{\theta}_k[t]) \cdot \big\lfloor\big|\delta\hat{\theta}_k[t]\big|\big\rfloor\Big) \mod L$$

$$\delta\hat{\theta}_k[t] = 0 \text{ if phase was corrected}$$

Box 4.3: Temporal coordination model.

4.3 *Simulation-Based Analysis*

The temporal coordination model is tested with a simulation implemented in *Python 3*. We focus on the evaluation of temporal pattern formation with a discretized model.

All oscillators start with uniformly distributed random phases Φ (both components ϕ and θ are random) and without any phase correction ($\delta\hat{\theta} = 0$). During our tests, the coupling strength $K = 0.25$ proved to be the most versatile for different parameter sets; therefore this value is used in all presented results. Based on this value, we calculate $K_m = \frac{L}{2\pi}\cdot K$ for $m < M$ and $K_M = -0.1\cdot\frac{L}{2\pi}\cdot K$.

A scaling factor $\frac{L}{2\pi}$ is used to speed up the convergence. It assures that if the influence is very high (e.g., $\sin\Big(\hat{\theta}_{jk}\Big) = 1$), the attraction will be sufficiently high for the oscillator to change its level by more than 1. It is especially useful if the number of levels is high. For example, if $L = 1000$, $M = 1$, $N = 2$ and in the initial state $\hat{\theta}_1 = \hat{\theta}_2 + 250$, each oscillator will have to change its phase level by 125. If the $\max_{\hat{\theta}_{jk}} \Gamma(\hat{\theta}_{jk}) = 1$, they would need at least 125 oscillation cycles to meet.

Phenomena related to communication and imperfections of oscillators are not taken into account. Agents have full connectivity without

delays or packet losses. This leads to the full knowledge about phase levels of other oscillators and perfect synchronization of the internal clocks.

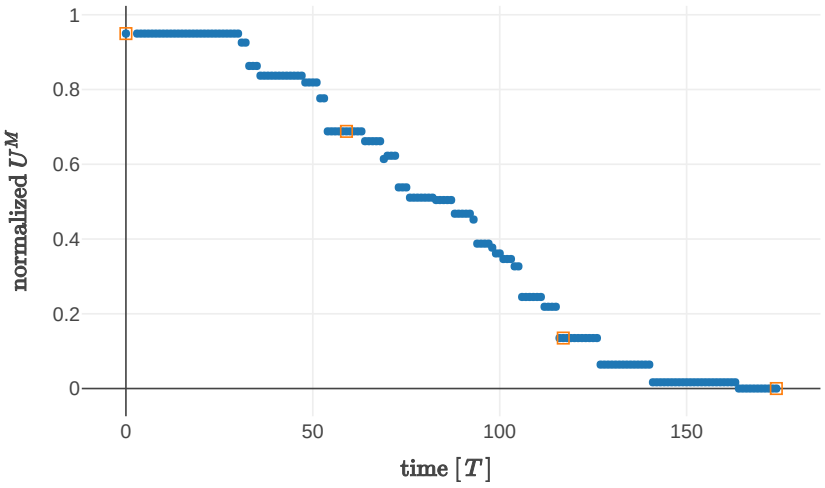
Results

For each experiment we present a plot showing how the potential $U^{(M)}$ converges over time. We shift its value by $\frac{K_M N}{2M^2}$ and normalize it. Thus, it always converges to 0 and has a maximum value of 1, independent of the parameters. Additionally, we show the phase plots at the beginning, in the middle, and at the end of the simulation. The values of the potential in these moments are marked with squares in the potential plot. In each phase plot, the possible phase levels are numbered on the angular axis and marked with the lines of corresponding color. The clusters are marked with circles filled with the color of their level. The size of the circles and the numbers next to them indicate the size of clusters.

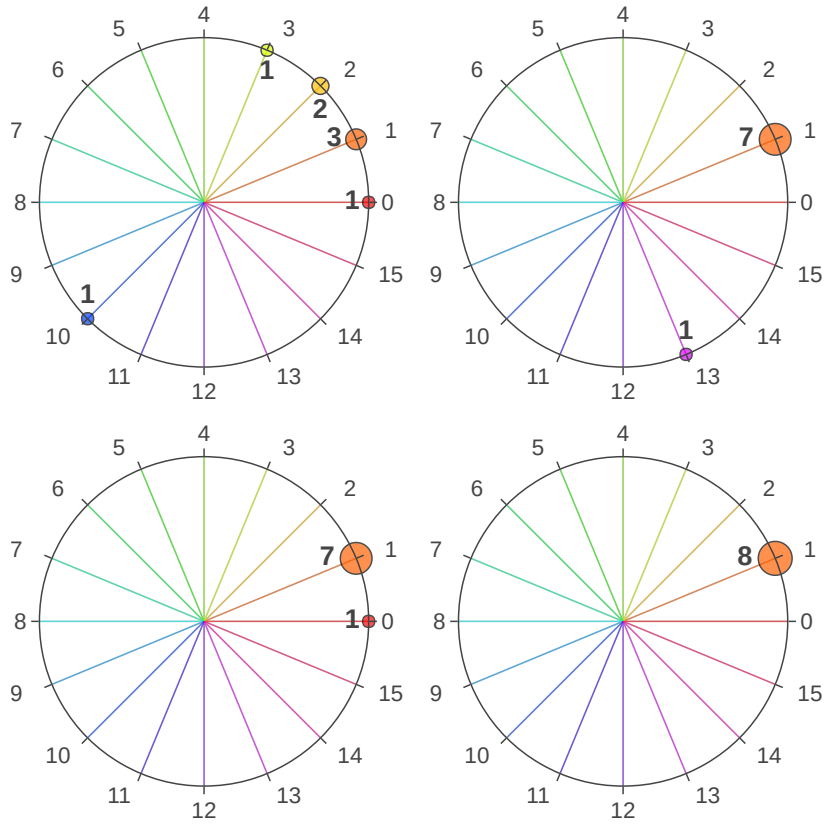
Figures 4.3 to 4.6 show the simulation results for different parameters, presenting M-cluster patterns for $M \in \{1, 2, 4, 8\}$. We notice that for $M = 1$ it takes a long time until the first change in the potential appears. This is caused by the initial distribution of phase levels, which is close to the balanced state. In this case, the interactions between the oscillators are weak, therefore more oscillation cycles are needed to change this state. This behavior cannot be observed for $M > 1$ because higher harmonics of phase levels are not balanced in the initial state. For all patterns, the closer the system gets to the desired pattern, the less dynamic the changes, as the interactions between oscillators become weaker when the system approaches the stable state.

The gradient control used in the continuous model would cause the potential to decrease. In some of the potential plots (Figures 4.4a and 4.5a), occasional growth of potential can be observed. This can be caused by energy and noise that breaks an unwanted cluster or equilibrium state. In this case, the change in one oscillator's phase level is caused entirely by noise. Another reason for a potential growth can be the discretization of the model. For example, a local minimum of the k-th dimension of continuous potential can be located between two phase levels. In this case, the k-th oscillator changes its phase level, moving in the direction of the local minimum, which in fact causes it to jump to the other side of the minimum.

In both scenarios, the temporary increase of the potential can either cause the system to find an optimal way toward the global minimum of potential or return to the previous state if the last change was not desired.

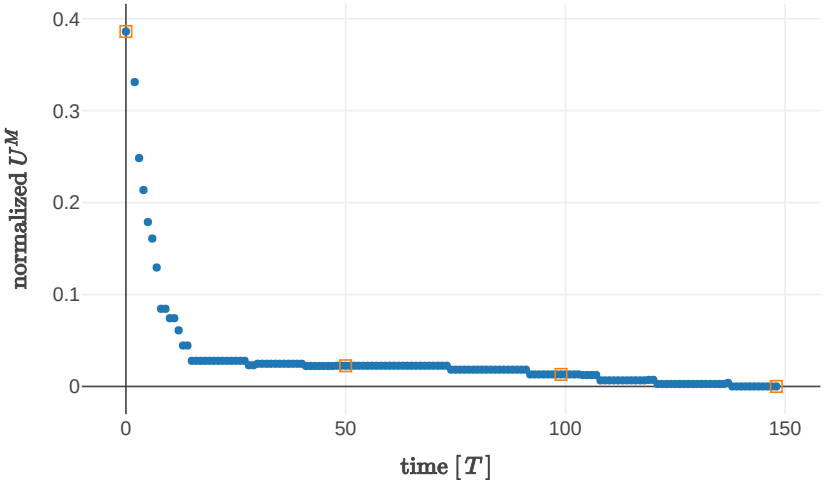


(a) Potential convergence

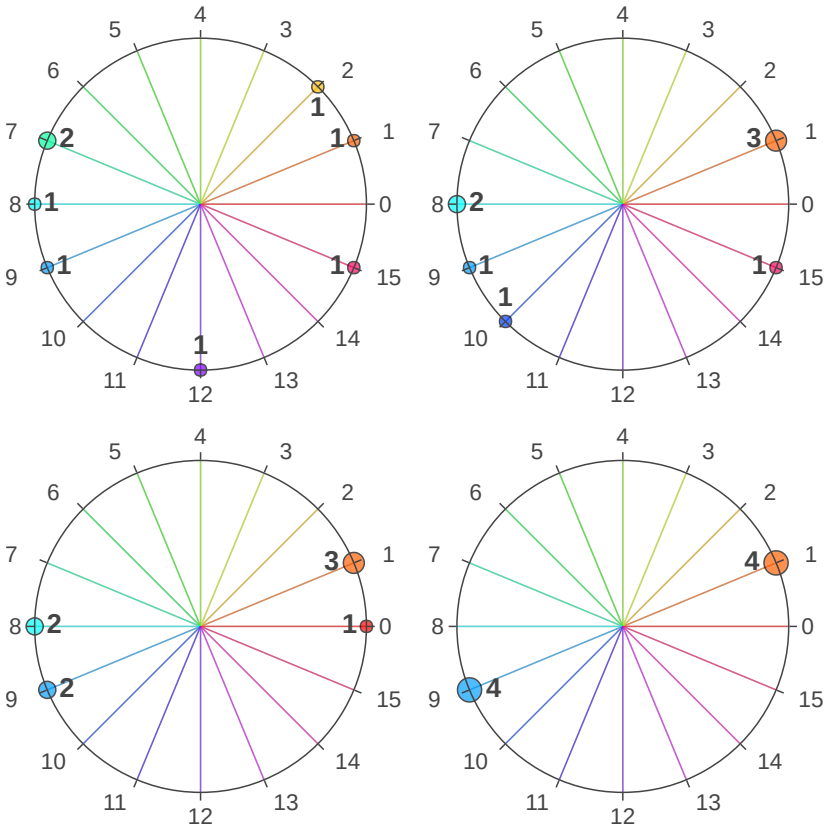


(b) Phase plot

Figure 4.3: The process of forming a synchronized pattern — M-cluster pattern for the following parameters: $N = 8$, $L = 16$, $M = 1$. The agents start with random phases drawn from the uniform distribution. The plots depict how the potential is converging and the phase levels of agents at a few selected moments, including initial state and formed pattern.

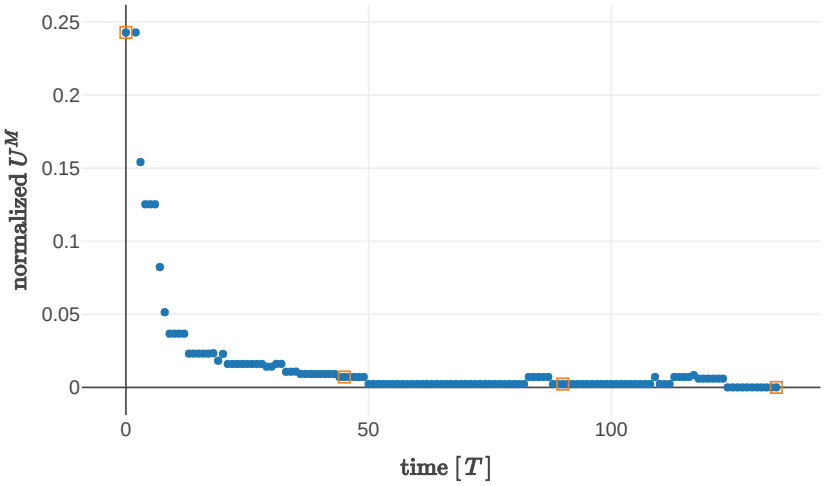


(a) Potential convergence

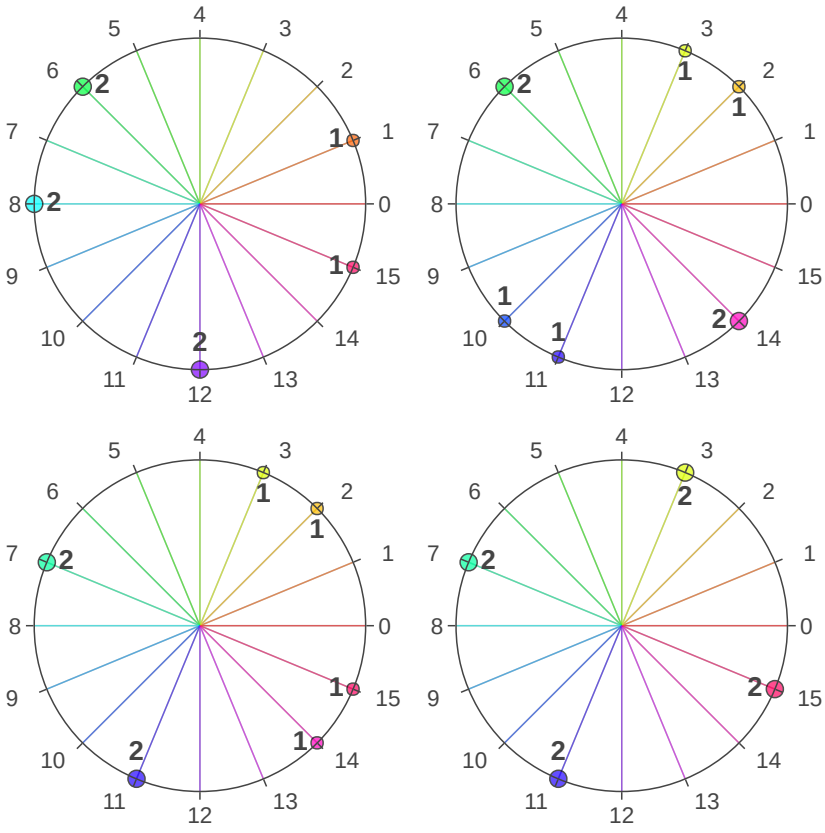


(b) Phase plot

Figure 4.4: The process of forming a clustered pattern — M-cluster pattern for the following parameters: $N = 8$, $L = 16$, $M = 2$. The agents start with random phases drawn from the uniform distribution. The plots depict how the potential is converging and the phase levels of agents at a few selected moments, including initial state and formed pattern.

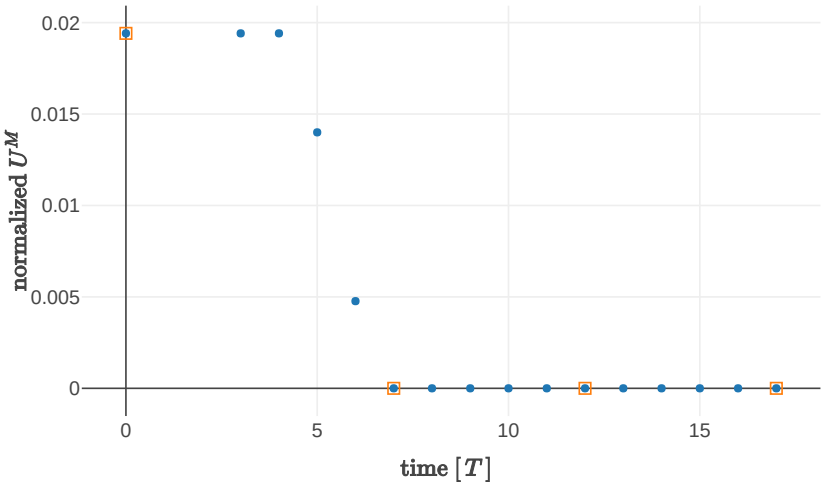


(a) Potential convergence

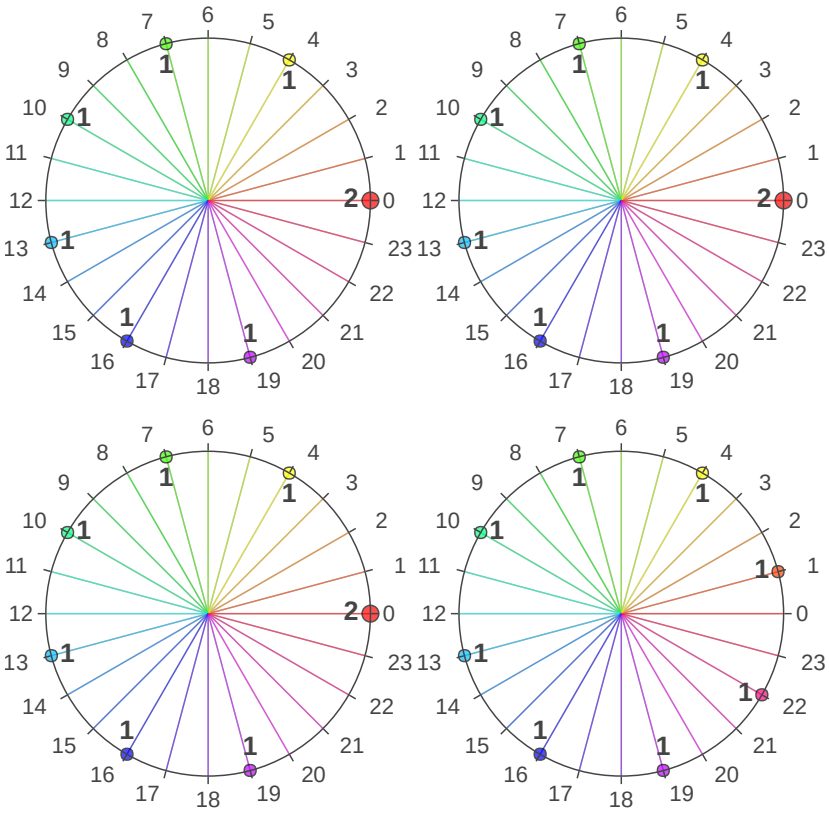


(b) Phase plot

Figure 4.5: The process of forming a clustered pattern — M-cluster pattern for the following parameters: $N = 12$, $L = 16$, $M = 4$. The agents start with random phases drawn from the uniform distribution. The plots depict how the potential is converging and the phase levels of agents at a few selected moments, including initial state and formed pattern.



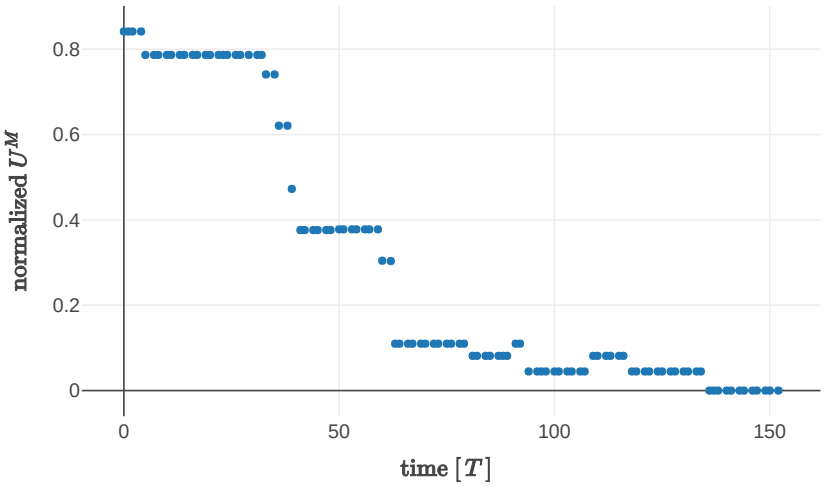
(a) Potential convergence



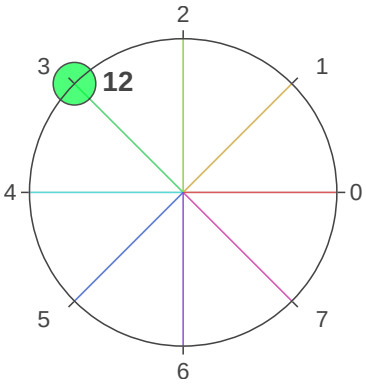
(b) Phase plot

Figure 4.6: The process of forming a splay pattern — M-cluster pattern for the following parameters: $N = 8$, $L = 24$, $M = 8$. The agents start with random phases drawn from the uniform distribution. The plots depict how the potential is converging and the phase levels of agents at a few selected moments, including initial state and formed pattern.

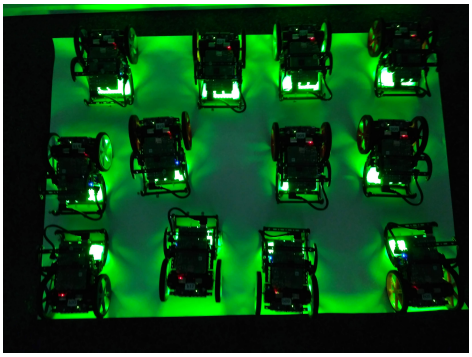
4.4 *Experiments With Robots*



(a) Potential convergence



(b) Phase plot

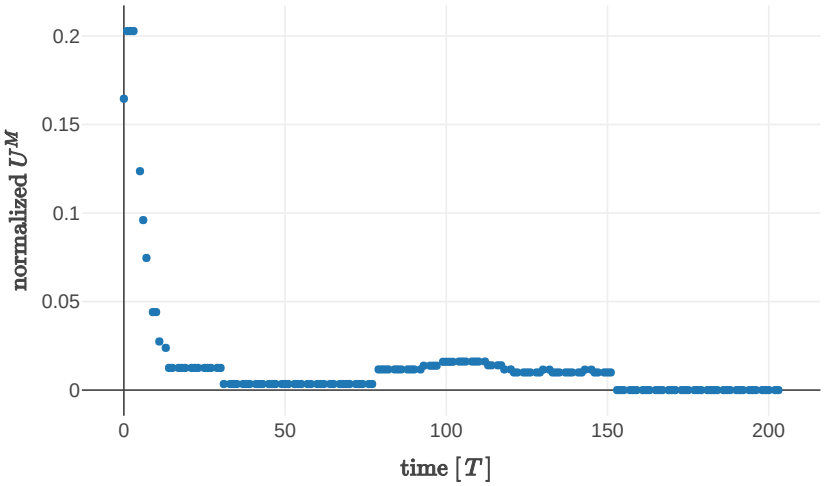


(c) Picture of robots

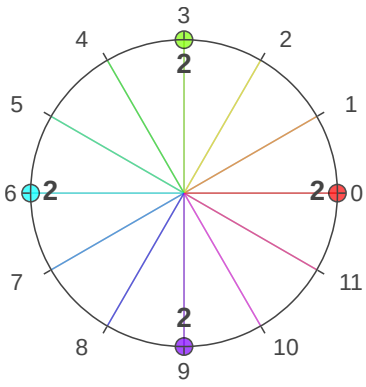
Figure 4.7: Experiment results: The process of forming a synchronized pattern by robots. It is achieved as an M-cluster pattern for the following parameters: $N = 12$, $L = 8$, $M = 1$. The robots start with random phases drawn from the uniform distribution. The plot depicts how the potential is converging. The phase plot and picture present the state of the system in a formed pattern.

For the evaluation of the temporal coordination model we are using Balboa robots. Similarly to our other experiments, robots communicate via an IEEE 802.11bg network operating in ad-hoc mode, which enables them to join and leave the group without any infrastructure. From the software perspective, the communication is realized in ROS 2 framework (Bouncy Bolson release) using the Data Distribution Service (DDS) communication standard, which applies a Real Time Publish Subscribe (RTPS) protocol (we use eProsima Fast RTPS). The robots are configured to communicate in the best-effort mode with multicast enabled to reduce communication load.

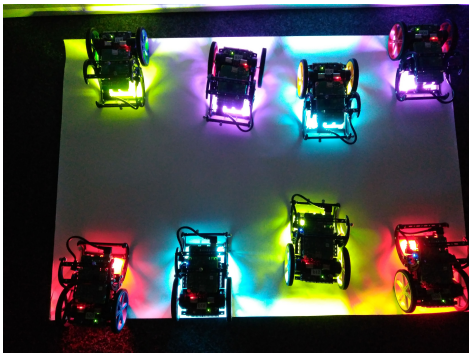
The robots use the described communication only to exchange their phase levels. A robot receives messages from all robots (its own messages are filtered out, and thus not taken into account, neither for internal clock synchronization nor for time coordination). The



(a) Potential convergence



(b) Phase plot



(c) Picture of robots

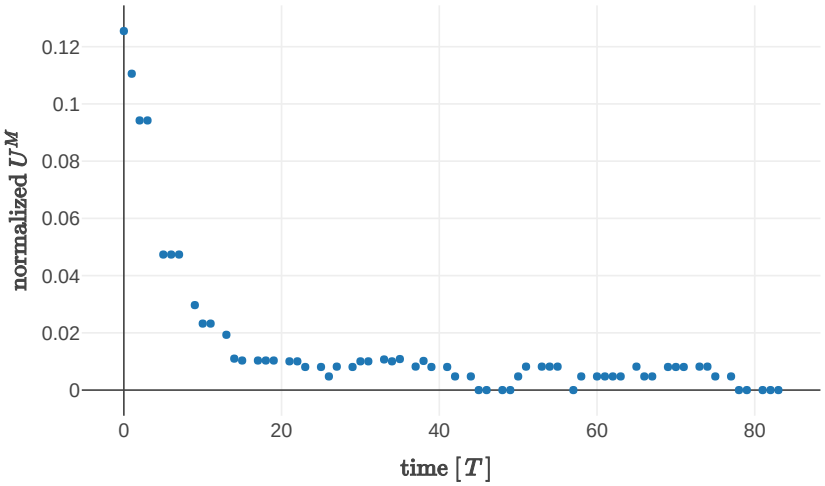
Figure 4.8: Experiment results: The process of forming a clustered pattern by robots. It is achieved as an M-cluster pattern for the following parameters: $N = 8$, $L = 12$, $M = 4$. The robots start with random phases drawn from the uniform distribution. The plots depict how the potential is converging. The phase plot and picture present the state of the system in a formed pattern.

signaling effort depends on the natural frequency of oscillations. In our experiments, we use $\omega = \frac{1}{L}\mathrm{s}^{-1}$. This means that each robot sends only one message per second.

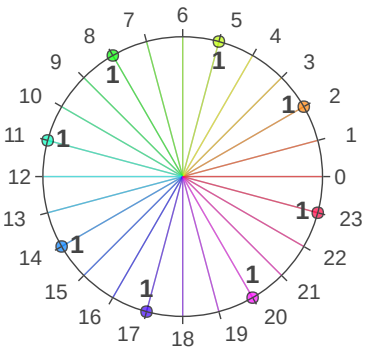
Moreover, the robots know neither the total number of agents nor the number of agents with the same phase level. For an update of phase correction and energy, the number of the messages received during the last oscillation cycle was assumed to be the number of oscillators and the data contained in these messages was used to count the number of agents with the same phase level. Therefore, if many messages are delayed or missing, it can have an impact on the convergence and stability of the desired pattern.

Results

The experimental results are shown in Figures 4.7 to 4.9. They are presented in a similar way as the simulation results aside from the



(a) Potential convergence



(b) Phase plot

(c) Picture of robots

Figure 4.9: Experiment results: The process of forming a splay pattern by robots. It is achieved as an M-cluster pattern for the following parameters: $N = 8$, $L = 24$, $M = 8$. The robots start with random phases drawn from the uniform distribution. The plots depict how the potential is converging. The phase plot and picture present the state of the system in a formed pattern.

fact that the phase plot is only shown for the final state, and a snapshot of the robots with lights representing their phase level is given for each pattern.

Again, we observe an occasional increase of the potential, now due to packet losses and previously described reasons. Although short disruptions of potential occur, the system recovers and converges to the desired pattern.

During the experiment, we measure the accuracy of internal clock synchronization, defined as a maximum difference between their phases. Although synchronization of the internal clocks is not the focus of this work, we use its poor precision to present the link quality and show that the time coordination model does not require precise synchronization. The results show that the accuracy sometimes drops to 0.04 s, which suggests that the link quality varies over time and significant delays occur. Despite these disturbances, our temporal coordination model leads to the desired patterns.

4.5 Chapter Summary

In this chapter we have introduced a distributed and adaptive model for temporal coordination in multi-agent systems. It allows three temporal patterns to emerge: synchronized, splay, and clustered. The model was tested with mobile robots, showing that time and phase discretization enables the system to converge to a desired pattern even in the presence of delays. These results confirm that the temporal coordination model is a good candidate to be used as a standalone temporal coordination method in multi-robot systems and to be applied as a part of a discrete, delay-robust sync and swarm model. In Chapter 6 we present how we have realized such an application.

5 Spatial Aggregation

In this chapter we focus on the second building block of the sandsbot model — spatial aggregation. Even though both components are equally important in the final model, this chapter is not as extensive as the previous one. The main reason for this is that the research on the problem of spatial aggregation is much more mature than on creating various temporal patterns. Hence, this model is based on state-of-the-art approaches, modified to our needs. For these reasons, it was not analyzed extensively as a standalone model, rather as a part of the sandsbot model. Parts of this chapter are taken from our published work [3].

Multiple robots working together almost always need to be coordinated in space. There are various scenarios in which this is required, to name a few: two robots carrying a big package have to keep a constant distance to each other; robots can increase efficiency of movement by moving in a formation (this is especially helpful for swimming and flying robots, as their motion resistance is high); or during exploration of an unknown terrain, robots should prioritize visiting places that are not yet explored by some other agent. Irrespective of the performed task, agents usually need to avoid collisions with each other.

There are various centralized planning algorithms that solve such problems. However, such solutions often lack scalability, thus are not efficient for bigger groups of robots. Moreover, they introduce a single point of failure, which makes them unsuited for multi-robot systems in which a failure of a single agent should not hinder the performance of the whole group. In contrast, distributed solutions based on self-organization are flexible, scalable and robust. They allow agents to freely join and leave the group.

Related Work

Researchers of multi-robot systems often try to employ self-organization in different aspects of spatial coordination. Examples include swarm aggregation [99], flocking [67], and pattern formation [53]. A widespread

technique in this context is to use the theory of potential fields. One of the reasons why this approach became popular is that it can be used to fulfill different tasks, like navigation [100], formation control [101], [102], and swarm aggregation [103], the last being most relevant to this work. Gazi [58] provided a formulation of the artificial potential for swarm aggregation and a controller that considers the dynamics of agents.

The stability of swarm aggregation based on potential fields was analyzed by Fetecau *et al.* [61] and Gazi *et al.* [60]. Tanner *et al.* [62] proposed conditions that need to be fulfilled by the potential to guarantee stable flocking based on potential fields.

Another approach to swarm aggregation was presented by Kernbach *et al.* [104]. They investigated aggregation of honeybee swarm in specific conditions and implemented a similar behavior on robots. Their solution allowed robots to aggregate in the warmest place of the testing area.

Contributions

In this chapter we focus on adapting the spatial aggregation based on potential fields to the practical system with constrained communication.

First, we adjust the potential field method to consider physical properties of agents (based on Figure 3.9). To achieve this, we define the potential that takes into account size of a bounding box of an agent. Later we introduce two constraints to limit the speed of robots. One is based on the physical capabilities of a robot. The other, more important, assures that the safety areas of the agents will never overlap.

Second, we reduce communication load caused by the model. In most cases, the potential and its gradient need to be computed continuously or at least at a high rate. This computation is based on positions of other agents. Sometimes these positions can be measured directly, but often some information needs to be communicated. For instance, in the experiments we performed each robot was broadcasting its estimated position to other robots. Such an approach requires agents to often communicate with each other, which might cause significant traffic in the network. However, simple reduction of the update rate might destabilize the swarm. Therefore, we use work by Armijo [105] to constraint the maximum speed of robots for the given update rate. This allows agents to form the pattern independently on the update rate. Moreover, thanks to the time-discrete interactions introduced in Section 3.4, the model is robust against communication delays.

The remainder of this chapter is structured as follows. In Section 5.1 we present the spatial aggregation model with time discrete interactions (as introduced in Section 3.4). Section 5.2 presents order parameter used to quantify the formed shape. The theoretical analysis of the model foundations together with simulation results and experimental test are described in Section 5.3. Section 5.4 concludes.

5.1 Model

Potential Field and Gradient Descent

The locations of the agents and their dynamics is controlled by an aggregation model based on potential fields. The potential created by each agent represents two counteracting forces: attraction and repulsion. Attraction helps agents to gather, whereas repulsion preserves spacing between them.

We base our solution on the potential used for swarm aggregation in a two-dimensional space by Fetecau *et al.* [61]:

$$\mathcal{V}_j(\mathbf{x}) = \frac{\left\|\mathbf{x}_j - \mathbf{x}\right\|^2}{2} - \frac{1}{2\pi}\ln\left(\left\|\mathbf{x}_j - \mathbf{x}\right\|\right), \tag{5.1}$$

and modify it to our needs. We introduce two scaling factors η_{attr} and η_{rep} to adjust the pattern size. Furthermore, in the repulsion part of the potential, we take into account the distance between bounding boxes to assure that the repulsion goes to infinity if agents are close to collision (bounding boxes are almost tangent). This approach guarantees that even if an agent is attracted by some other agents, it will always prioritize avoiding collision with one agent over moving close to the other agents. After these modifications, the potential at position $\mathbf{x}$ generated by the agent j used in our approach is:

$$\mathcal{V}_j(\mathbf{x}) = \eta_{\text{attr}}\frac{\left\|\mathbf{x}_j - \mathbf{x}\right\|^2}{2} - \eta_{\text{rep}}\ln\left(\left\|\mathbf{x}_j - \mathbf{x}\right\| - d\right), \tag{5.2}$$

for non-overlapping bounding boxes (i.e., $\left\|\mathbf{x}_j - \mathbf{x}\right\| - d > 0$, recall d is a diameter of the bounding box of an agent), and undefined if the bounding boxes overlap. The plot of $\mathcal{V}_j(\mathbf{x})(\mathbf{x}_k)$ is presented in Figure 5.1, where the black dashed line marks the minimum distance for which the bounding boxes are not overlapping. The potential of agent k is defined as the average of the potentials created by other agents:

$$V_k = \frac{1}{N}\sum_{\substack{j=1 \\ j\neq k}}^{N}\mathcal{V}_j(\mathbf{x}_k). \tag{5.3}$$

Similar to temporal coordination, each agent aims at minimizing its potential. In order to find this minimum we use the gradient de-

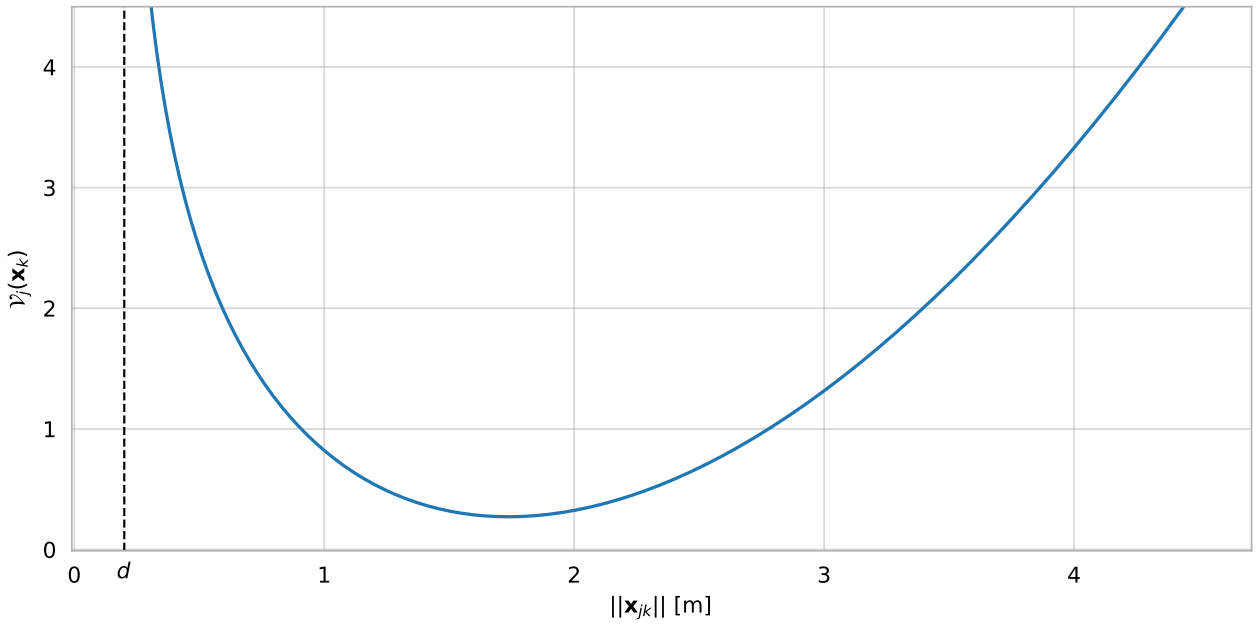


Figure 5.1: Potential at position $\mathbf{x}_k$ generated by the agent j.

scent method. We call the output of this method *demanded velocity*, as opposed to actual velocity, which also takes into account additional constraints. The demanded velocity of agent k is defined as $\mathbf{v}_k^{\mathrm{d}} = \boldsymbol{\nabla} V_k$, which for Equation (5.3) yields:

$$\mathbf{v}_k^{\mathrm{d}} = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \boldsymbol{\nabla} \mathcal{V}_j(\mathbf{x}_k)\,, \tag{5.4}$$

where the summands are

$$\boldsymbol{\nabla} \mathcal{V}_j(\mathbf{x}_k) = \mathbf{I}_1(\mathbf{x}_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk}) \quad \text{with} \tag{5.5}$$

$$\mathbf{I}_1(\mathbf{x}_{jk}) = \eta_{\mathrm{attr}}\, \mathbf{x}_{jk}, \tag{5.6}$$

$$\mathbf{I}_2(\mathbf{x}_{jk}) = \frac{\eta_{\mathrm{rep}}}{\left\|\mathbf{x}_{jk}\right\| \left(\left\|\mathbf{x}_{jk}\right\| - d\right)}\, \mathbf{x}_{jk} \tag{5.7}$$

describing how strong agent k is attracted or repelled by the other agents.

Figure 5.2 depicts an example potential field generated by agents 2 and 3 ($0.5(\mathcal{V}_2(\mathbf{x}) + \mathcal{V}_3(\mathbf{x}))$) in a formed pattern. It is visible that the applied control strategy allowed agent 1 to reach the minimum of the potential. There is another minimum of the potential on the right side of the picture, but the gradient descent strategy steered the agent 1 towards the closest minimum and allowed it to stay there. During the run, agents 2 and 3 also adjusted their positions. As a result,

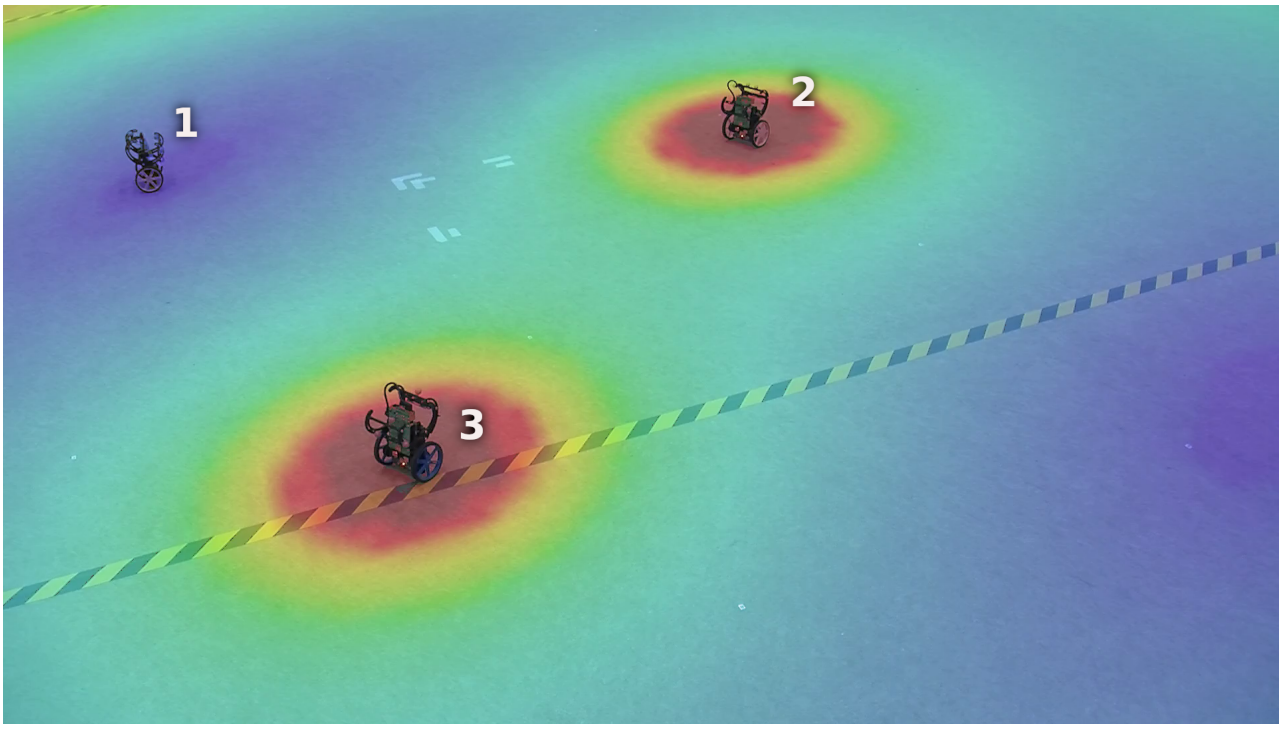


Figure 5.2: Visualization of the potential field generated by robots 2 and 3 in the formed pattern. The lowest potential is marked with blue color and the highest with red, places with undefined potential close to robots 2 and 3 are dark red.

the potential was changing while they were moving. Our approach allowed robot 1 to adjust to this change and always move towards the current minimum until it was reached and the whole pattern stabilized.

Consideration of Robot Constraints

The maximum speed that the robot can achieve is limited by two factors: its physical capabilities and safety constraints. We denote this maximum speed as $v_k^{\max}$. The major physical constraint is the maximum possible speed $v_k^{\max R}$ that can be obtained with the given hardware (e.g., it depends on engines and gearbox). This value depends on the robot platform in use.

The safety constraint guarantees the safe, collision-free operation of robots. It is formulated as

$$v_k^{\max S} = \frac{\min_{j \neq k} \left(\left\| \mathbf{x}_{jk} \right\| \right) - d - \epsilon_d}{4T}. \tag{5.8}$$

This constraint ensures that during one period each agent can move by at most a quarter of the gap between its own safety area and the one of the closest neighbor. The distance between safety areas of each pair of agents will change by not more than half, which should

not only guarantee that the agents never collide but also that their bounding boxes do not come closer than ϵ_d. When agents get very close to each other, the repulsion between them is very high. Such a strong interaction between two agents might result with both of them moving at their maximum speed to move away from each other. It can be especially risky if one of these agents is surrounded by multiple other ones, as this might lead to a collision. The safety constraint additionally slows down agents operating in a close proximity, thus preventing rapid reactions. As a result of the constraints, the maximum speed is $v_k^{\max} = \min(v_k^{\max R}, v_k^{\max S})$. Based on this, the velocity of agent k is

$$\mathbf{v}_k = \min\left(\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, v_k^{\max}\right) \cdot \frac{\mathbf{v}_k^{\mathrm{d}}}{\left\|\mathbf{v}_k^{\mathrm{d}}\right\|}, \tag{5.9}$$

and the position change during one oscillation cycle is

$$\mathbf{x}_k[t] = \mathbf{x}_k[t-1] + T\,\mathbf{v}_k[t-1]. \tag{5.10}$$

Mitigation of Physical Oscillations

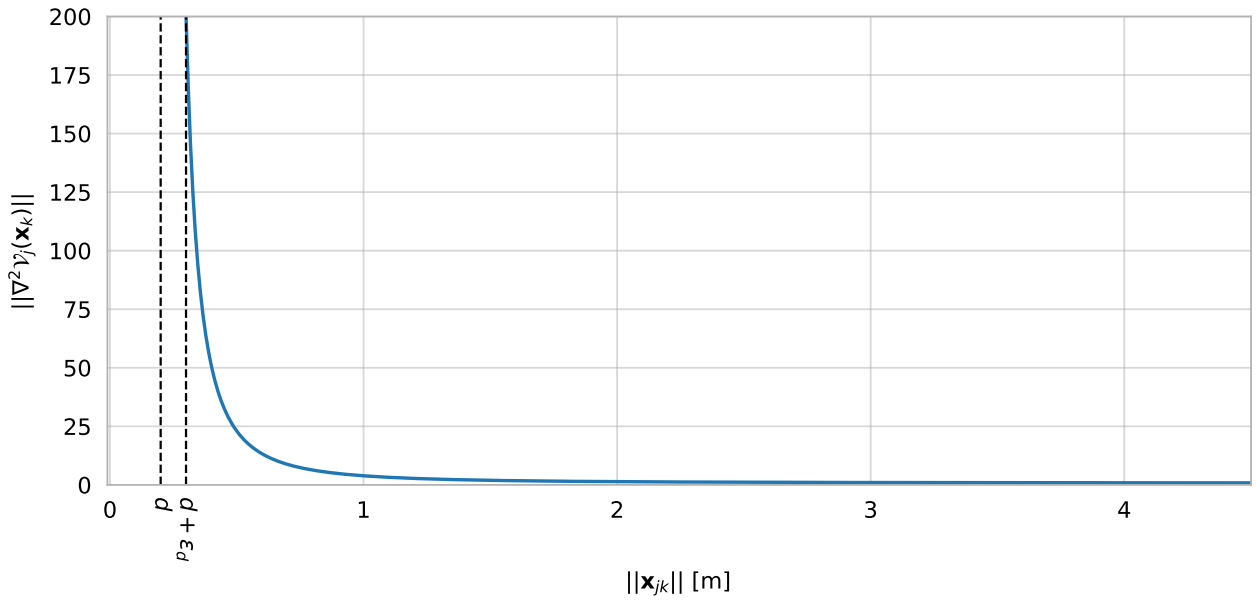


Figure 5.3: The plot of $\left\|\nabla^2 \mathcal{V}_j(\mathbf{x}_k)\right\|$. The black dashed line labeled d marks the asymptote of this function, whereas the other line with label $d + \epsilon_d$ marks the left bound of the domain.

The simulation of such a model shows an unwanted behavior: agents oscillate around their positions. To compensate this phenomenon, we modify the maximum speed based on a theorem introduced by Armijo [105], which states: If the gradient of function f is Lipschitz continuous, gradient descent converges for step size

$s \leq \frac{1}{2\lambda}$. We say ∇f is Lipschitz continuous, when

$$\forall x, y \in \mathcal{D}_f \colon \|\nabla f(x) - \nabla f(y)\| < \lambda \|x - y\|, \tag{5.11}$$

where $\mathcal{D}_f$ is a domain of f and λ is called Lipschitz constant. Hence, to avoid spatial oscillations, we introduce an additional term $s\left\|\mathbf{v}_k^{\mathrm{d}}\right\|$ in the calculations of the velocity (Equation (5.9)) to limit the speed of each agent, if needed:

$$\mathbf{v}_k = \min\left(\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, s\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, v_k^{\max}\right) \cdot \frac{\mathbf{v}_k^{\mathrm{d}}}{\left\|\mathbf{v}_k^{\mathrm{d}}\right\|}, \tag{5.12}$$

where

$$s = \frac{1}{2\lambda T} \tag{5.13}$$

is a step size calculated based on period length and the Armijo's theorem. This modification limits the speed to guarantee that the step size is short enough for the gradient descent to converge.

By limiting the speed due to the safety constraint (Equation (5.8)), we guarantee that the minimum distance constraint

$$\forall k,\, j \quad \left\|\mathbf{x}_{jk}\right\| > d + \epsilon_d \tag{5.14}$$

is met. Hence, we can limit the domain of V_k. Note that the value of potential generated by agent j goes to infinity when $\left\|\mathbf{x}_{jk}\right\| \to d$ (see its plot in Figure 5.3). However, $\left\|\nabla^2 \mathcal{V}_j(\mathbf{x}_k)\right\|$ is bounded above on the domain (for $\left\|\mathbf{x}_{jk}\right\| > d + \epsilon_d$). Hence, we can conclude that $\left\|\nabla^2 V_k\right\|$ is bounded above and the gradient is Lipschitz continuous. Based on the used parameters we calculated the Lipschitz constant $\lambda = \eta_{\mathrm{attr}} - \eta_{\mathrm{rep}}/\epsilon_d^2$.

This approach enables us to slow down the agent if the demanded velocity would lead to oscillations. To achieve faster convergence, the Lipschitz constant can be computed for the gradient limited only to the neighborhood of the current position, with the assumption that the agents can move in any direction at their maximum speed.

The obtained model is summarized in Box 5.1.

5.2 *Order Parameter*

We are interested in three types of spatial arrangement: circle, disk and annulus. To distinguish them formally, we introduce the *swarming order parameter* $V \in \mathbb{R}_0^+$, presented in Box 5.2. It is calculated based on the variance of distances of the agents from the centroid of the group (denoted as σ_d). The distance of the agent k from the centroid is marked as d_k. This value is normalized by $\sigma_{d'(R)}$, denoting the variance of the distance from the center of a disk with

$$
\begin{aligned}
\mathbf{v}_k^{\mathrm{d}} &= \frac{1}{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \boldsymbol{\nabla} \mathcal{V}_j(\mathbf{x}_k) \\
\boldsymbol{\nabla} \mathcal{V}_j(\mathbf{x}_k) &= \mathbf{I}_1(\mathbf{x}_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk}) \\
\mathbf{I}_1(\mathbf{x}_{jk}) &= \eta_{\mathrm{attr}}\, \mathbf{x}_{jk} \\
\mathbf{I}_2(\mathbf{x}_{jk}) &= \frac{\eta_{\mathrm{rep}}}{\left\|\mathbf{x}_{jk}\right\| \left(\left\|\mathbf{x}_{jk}\right\| - d\right)}\, \mathbf{x}_{jk} \\
\mathbf{v}_k &= \min\left(\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, s\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, v_k^{\max}\right) \cdot \frac{\mathbf{v}_k^{\mathrm{d}}}{\left\|\mathbf{v}_k^{\mathrm{d}}\right\|} \\
\mathbf{x}_k[t] &= \mathbf{x}_k[t-1] + T\, \mathbf{v}_k[t-1]
\end{aligned}
$$

Box 5.1: Spatial aggregation model.

radius R to a point chosen uniformly at random from this disk. We set $R = \max_k(d_k)$.

$$
V = \frac{\sigma_d}{\sigma_{d'(R)}} = \frac{1}{N} \sum_{k=1}^{N} \frac{(d_k - \mu_d)^2}{\sigma_{d'(R)}}, \tag{5.15}
$$

Box 5.2: Swarming order parameter V.

We observe the following: Agents placed on a circle yield $V = 0$, agents on an annulus have $0 < V < 1$, and agents distributed on a disk lead to $V > 1$. If the disk is fully occupied (i.e., the agents are tightly packed, minimum distances are not preserved), V is equal to 1, but in practice it is higher due to the minimum distance constraint (from Equation (5.14)).

5.3 *Analysis*

The analysis of this model is based mainly on theoretical considerations, confirmed by simulation-based analysis of our technique to mitigate physical oscillations and qualitative experimental results. Nevertheless, its strong theoretical basis lets us assume that the model works and can be successfully used in the sandsbot model. Below we present our reasoning and results from simulations and tests with Balboa robots.

Theoretical Considerations

First we will show that the model based on potential (from Equation (5.2)) works and then that introduced modifications do not break it. Attraction $\mathbf{I}_1$ and repulsion $\mathbf{I}_2$ should fulfill the following two conditions for the aggregation to work: attraction should be stronger than repulsion at large distances, otherwise the agents will disperse, repulsion should dominate at short distances to avoid collisions [61]. In our case, the gradient (Equation (5.5)) is monotone and the forces balance each other at a *neutral distance* x_0 (this distance coincides with the minimum of the potential in Figure 5.1). For greater distances attraction is bigger than repulsion, for smaller the opposite is true. Moreover, the neutral distance must be greater than safety area:

$$x_0 > d + \epsilon_d, \tag{5.16}$$

this condition guarantees that the agents repel each other before their safety areas overlap. To check if the condition from Equation (5.16) is met, we need to find the neutral distance.

$$\eta_{\text{attr}} x_0 = \frac{\eta_{\text{rep}}}{x_0 - d}, \text{ equivalent to}$$
$$x_0^2 - d x_0 - \frac{\eta_{\text{rep}}}{\eta_{\text{attr}}} = 0,$$

which yields two solutions, but only one belongs to the domain (distance must be greater than 0):

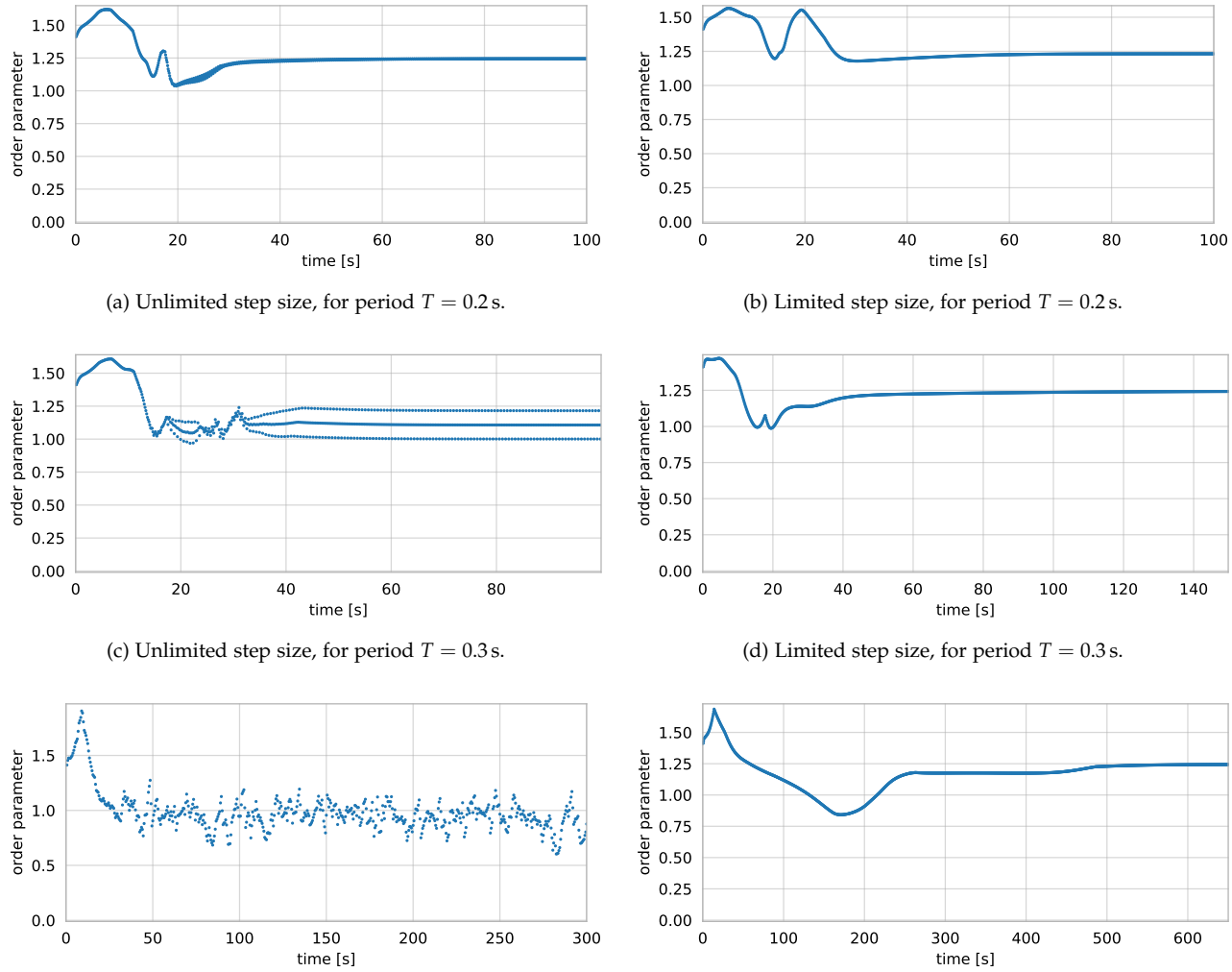
$$x_0 = \frac{d + \sqrt{d^2 + 4\frac{\eta_{\text{rep}}}{\eta_{\text{attr}}}}}{2}. \tag{5.17}$$

Of course, the value of neutral distance depends on the parameters. We use $d = 0.2\,\text{m}$, $\eta_{\text{attr}} = 0.75$, $\eta_{\text{rep}} = 2$ and $\epsilon_d = 0.1\,\text{m}$. For these values $x_0 \approx 1.73\,\text{m}$ and $d + \epsilon_d = 0.3\,\text{m}$, which means that the condition $x_0 > d + \epsilon_d$ holds and the aggregation is possible with this potential.

Furthermore, the theorem by Armijo [105] specifies that in order to guarantee the convergence gradient descent cannot progress by more than $\left|\frac{1}{2\lambda}\boldsymbol{\nabla} f\right|$ in each update. In our model the gradient descent progress (position change) is equal to $T\mathbf{v}_k$ (see Equation (5.10)). From the theorem we can conclude that the following condition must hold:

$$T\mathbf{v}_k \leq \frac{1}{2\lambda}\mathbf{v}_k^{\text{d}}, \text{ which can be simplified to:}$$
$$\mathbf{v}_k \leq s\mathbf{v}_k^{\text{d}}.$$

Note that $s\mathbf{v}_k^{\text{d}}$ is the second term of the minimum in Equation (5.12), which guarantees that the condition is always true. Such limitation



(a) Unlimited step size, for period $T = 0.2\,\mathrm{s}$.

(b) Limited step size, for period $T = 0.2\,\mathrm{s}$.

(c) Unlimited step size, for period $T = 0.3\,\mathrm{s}$.

(d) Limited step size, for period $T = 0.3\,\mathrm{s}$.

(e) Unlimited step size, for period $T = 1\,\mathrm{s}$.

(f) Limited step size, for period $T = 1\,\mathrm{s}$.

Figure 5.4: The simulation results for the spatial aggregation in two cases: the left column shows the order parameter without limiting the step size, the right column presents analogous results but with limited the step size (thus also speed).

slows down the convergence, but also guarantees that the pattern will form.

We showed that the model based on potential (Equation (5.2)) works and is not hindered by introduced modifications. Therefore, we can conclude that the spatial aggregation model always converges.

Simulation-Based Analysis

Figure 5.4 presents the simulation-based comparison of swarm aggregation in two cases. Left column presents plots without additional speed limitation (with $s = 1$). Right column shows the results of swarm aggregation with our technique to mitigate physical oscil-

lations (with step size s from Equation (5.13)). The plots show how the value of order parameter V changes over time. When the pattern is successfully formed, the value stabilizes at $V > 1$. In each of the simulation runs the agents start from the positions drawn from the uniform distribution, with the same random seed. This means that each simulation is started with the same initial conditions. We use the same parameters as in our other simulations in this thesis: $d = 0.2\,\mathrm{m}$, $\epsilon_d = 0.1\,\mathrm{m}$, $v^{\mathrm{maxR}} = 0.2\,\mathrm{m/s}$ and varying period lengths, specified for each plot separately.

Figure 5.4 shows the results of three different scenarios: with short, medium and long period ($T = 0.2\,\mathrm{s}$, $T = 0.3\,\mathrm{s}$, $T = 1\,\mathrm{s}$ respectively). Each of them is run without and with speed limitation. For the short period length ($T = 0.2\,\mathrm{s}$) both solutions converge to a stable pattern (see Figures 5.4a and 5.4b). There is a slight variation visible in the case without limitation of the step size from 20 s to 30 s, but it does not influence the final result. The agents need around 50 s to successfully aggregate in this case and a bit longer (around 70 s) with the limited step size.

With the medium period length ($T = 0.3\,\mathrm{s}$) and without limiting the step size, the agents aggregate, but slightly oscillate around their positions. It is visible as three lines after 50 s in Figure 5.4c, which correspond to different positioning of agents. The agents that have the speed limited based on the step size (Figure 5.4d) successfully converge, but they need more time than with shorter period — around 100 s.

Finally, with long period length ($T = 1\,\mathrm{s}$) the agents do not aggregate if the step size is not limited. In Figure 5.4e the value of the order parameter constantly changes. Moreover, the safety distances between agents are not preserved and collisions can occur. In contrary, the limitation of the step size allows the agents to successfully converge even with long period length (see Figure 5.4f). The aggregation takes longer than for shorter periods(around 500 s), but eventually the value of order parameter stabilizes.

To summarize, our technique to limit the step size (and thus also speed) increases the convergence time, but allows the agents to always aggregate. Without limiting the speed the physical oscillations can occur or even the whole pattern can destabilize.

Tests With Robots

We performed multiple experiments featuring Balboa robots to validate our implementation. The setup used was similar to our other experiments: robots communicate via an IEEE 802.11bg network in ad-hoc mode and use ROS 2 framework (Eloquent Elusor release

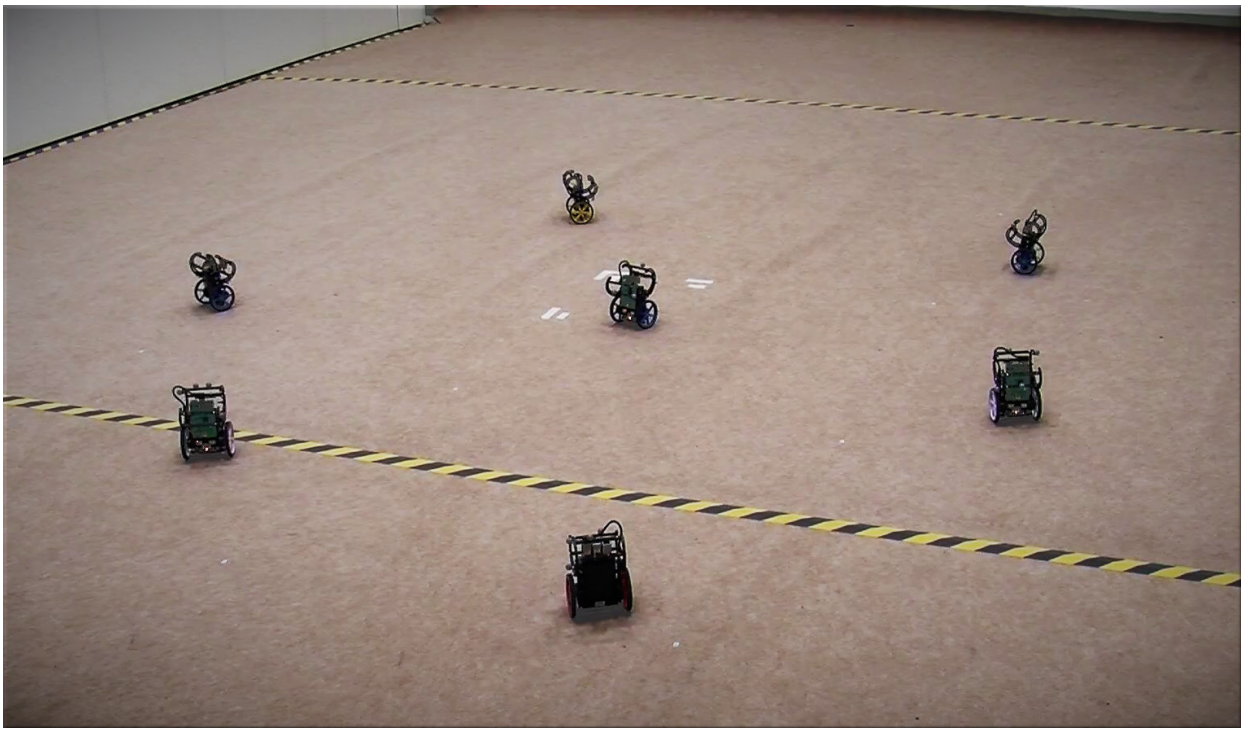


Figure 5.5: Balboa robots running spatial aggregation model.

with eProsima Fast RTPS). The communication was done in best-effort mode with enabled multicast. The robots use modification of the model related to their movement constraints as introduced in Section 3.2.

The robots start from arbitrary positions. We use the following parameters: $d = 0.2\,\mathrm{m}$, $\epsilon_d = 0.1\,\mathrm{m}$, $v^{\mathrm{maxR}} = 0.2\,\mathrm{m/s}$, $\eta_{\mathrm{attr}} = 0.75$, $\eta_{\mathrm{rep}} = 2$, and $T = 0.125\,\mathrm{s}$, so each robot sends eight messages per second. The result of one of the experiments is presented in Figure 5.5. It shows robots successfully aggregating and forming a disk.

5.4 *Chapter Summary*

In this chapter we have introduced the spatial aggregation model that allows robots to avoid collisions and keep safe distances to each other. It takes into account the physical limitations of agents and constrained communication. Thanks to exchanging predicted positions instead of the current ones, agents can successfully and safely aggregate even with a very low update rate. The provided analysis shows that the introduced constraints do not prevent aggregation, however they can slow it down. This result is confirmed by a test featuring Balboa robots. In the next chapter we show how the spatial aggregation model can be coupled with temporal coordination (Chapter 4) into a unified model.

6 *Sandsbots: Synchronizing and Swarming Robots*

In this chapter we finally utilize the two previously introduced models for temporal coordination (Chapter 4) and spatial aggregation (Chapter 5) to create a unified sandsbot model. We show that the emerging spatio-temporal patterns are similar to the ones obtained with the swarmalator model [6]. However, sandsbots form the desired pattern regardless of the initial conditions and the model allows to directly control more aspects of the patterns. The exact differences are described in Section 6.3. Parts of this chapter are taken from our published work [3].

The key motivation of our research is the possibility to apply the spatio-temporal patterns formed by swarmalators [6] in multi-robot systems. For example, robots could not only take photos of a point of interest simultaneously but also form a spatial pattern around this point and take sequence of photos sorted by the viewing angle in a self-organizing manner. However, such applications pose new challenges to the model, such as collision avoidance, discrete coupling (due to the need of communication) and movement constraints. Hence, the direct application of the swarmalator model [6] in a robotic system is not possible.

Even though our preliminary work (Section 3.2) served as a proof of concept that robots can act as swarmalators, it still suffered from multiple problems described in Section 3.3. In this chapter we introduce a new model that overcomes those problems.

Contributions

The main contribution presented in this chapter is the model suitable for multi-robot systems that allows to form emergent space-time patterns. It consists of two mutually coupled parts: temporal coordination (Chapter 4) and spatial aggregation (Chapter 5) with time-discrete interactions between agents. The proposed model displays the following properties:

- Has low communication requirements and guarantees robust behavior in the presence of communication delays and with a very low frequency of interactions between agents (i.e., once each few seconds).
- Adapts the maximum speed of agents to the communication rate, allowing the agents to avoid collisions and successfully form a pattern.

We verify the sandsbot model in both simulations and experiments featuring small mobile ground robots and aerial robots (drones), thus providing a proof of concept for robotic swarmalators. To the best of our knowledge, this is the first "sync and swarm solution" robustly working in robot systems and creating emerging space-time patterns.

The remainder of this chapter is structured as follows. Section 6.1 introduces the sandsbot model. Section 6.2 describes an order parameter needed to quantitatively distinguish spatio-temporal patterns. Section 6.3 shows the emerged patterns. Sections 6.4 and 6.5 present a simulation-based analysis and an experimental proof of concept. Section 6.6 concludes. Additionally to the results presented in this chapter, Appendix C contains calculations of the radius of the ring phase wave pattern.

6.1 Model

We extend the models from Chapters 4 and 5 to propose a time-discrete solution in which temporal and spatial coordination are mutually coupled. The overall result is given in Box 6.1, we use it with the interaction functions specified in Box 6.2.

Influence of Phase on Spatial Coordination

First we introduce how phases influence the position interactions. The demanded velocity of this phase-influenced aggregation model has the following form (similar to the formulation proposed by O'Keeffe *et al.* [70]):

$$\mathbf{v}_k^{\mathrm{d}} = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \mathbf{I}_1(\mathbf{x}_{jk}) F_1(\theta_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk}) F_2(\theta_{jk}), \tag{6.1}$$

where functions $F_1(\theta_{jk}) = 1 + J_1 \cos\left(\theta_{jk}\right)$ and $F_2(\theta_{jk}) = 1 - J_2 \cos\left(\theta_{jk}\right)$ describe how the phase similarity of agents influences their spatial attraction and repulsion, respectively, with parameters J_1 and J_2 defining the strength of these influences. The attraction of agents with similar phases is stronger than in the aggregation model if $J_1 > 0$,

it remains unchanged for $J_1 = 0$, and it is weaker if $J_1 < 0$. We set $J_2 = 0$ to prevent collisions. Using a high J_2 value could cause agents with similar phases not to repel but rather attract each other, no matter how close they are, which could lead to collisions. Even though we do not use J_2 in this work we keep it in the model due to its potential to extend abilities of sandsbots. If it is used carefully, in a safe range that does not hinder collision-free operation, it might even allow emergence of new patterns.

Since the phases now influence the positions, the previously calculated step size (Equation (5.13)) does not guarantee oscillation-free convergence. The reason being that, once a phase changes, attraction and thus velocity might change significantly. There are two options to compensate for this behavior: using more phase levels ($L \gg M$) or keeping the same number of phase levels and changing the step size. We use the step size

$$s = \frac{1}{2\lambda T} \cdot G\big(\hat{U}^{(M)}\big), \tag{6.2}$$

where $\hat{U}^{(M)} \in [0,1]$ is a normalized value of the potential, which is 1 if $U^{(M)}$ is maximal and converges to 0 when agents' phases reach the desired pattern. The function $G(\cdot)$ defines how strong should be the influence of phase potential on the step size of the spatial aggregation. In this work we use:

$$G\big(\hat{U}^{(M)}\big) = 1 - \sqrt[p]{\hat{U}^{(M)}}\,, \tag{6.3}$$

which allows us to control which behavior has priority, temporal or spatial coordination. High values of p will make agents move slowly when their phases have not formed the correct pattern yet. Low values will make agents move dynamically even before the temporal pattern has converged. We use $p = M$, which means: the more clusters should be formed, the less dynamic the movements are before the temporal pattern has converged.

Influence of Position on Temporal Coordination

The phase interactions are modified in a similar way. We enable the distance between agents to influence coupling of each phase harmonic:

$$\Psi_k(\theta_k) = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \Gamma_1(\theta_{jk}) \Lambda_1(\mathbf{x}_{jk}) + \Gamma_2(\theta_{jk}) \Lambda_2(\mathbf{x}_{jk}), \tag{6.4}$$

where the functions Λ_1 and Λ_2 define how the distance between agents influences their phase attraction and repulsion, respectively.

We use the following form of these functions:

$$\Lambda_1(\mathbf{x}_{jk}) = 1 + P\frac{\epsilon_d + d}{\left\|\mathbf{x}_{jk}\right\|}; \tag{6.5}$$

$$\Lambda_2(\mathbf{x}_{jk}) = 1 - P\frac{\epsilon_d + d}{\left\|\mathbf{x}_{jk}\right\|}. \tag{6.6}$$

The parameter $P \in [-1, 1]$ quantifies the strength of influence of distance between agents on their phase coupling. For positive P, when agents move closer to each other, their phase attraction is amplified and their phase repulsion is weakened. If $P = 0$, the positions do not influence the phases. The influence of spatial proximity is strongest at the shortest distance between agents ($\epsilon_d + d$) and decreases with increasing distance.

$$\mathbf{v}_k^{\mathrm{d}} = \frac{1}{N}\sum_{\substack{j=1\\ j\neq k}}^{N} \mathbf{I}_1(\mathbf{x}_{jk})F_1(\theta_{jk}) - \mathbf{I}_2(\mathbf{x}_{jk})F_2(\theta_{jk})$$

$$\mathbf{v}_k = \frac{\mathbf{v}_k^{\mathrm{d}}}{\left\|\mathbf{v}_k^{\mathrm{d}}\right\|} \cdot \min\left(\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, s\left\|\mathbf{v}_k^{\mathrm{d}}\right\|, v_k^{\max}\right)$$

$$\mathbf{x}_k[t+1] = \mathbf{x}_k[t] + \min(s, T)\mathbf{v}_k$$

$$\Psi_k(\theta_k) = \frac{1}{N}\sum_{\substack{j=1\\ j\neq k}}^{N} \Gamma_1(\theta_{jk})\Lambda_1(\mathbf{x}_{jk}) + \Gamma_2(\theta_{jk})\Lambda_2(\mathbf{x}_{jk})$$

$$\delta\hat{\theta}_k[t] = (1+E)\delta\hat{\theta}_k[t-1] + \eta - \Psi_k\left(\theta_k[t]\right)$$

$$\hat{\theta}_k[t+1] = \left(\hat{\theta}_k[t] + 1 + \mathrm{sgn}(\delta\hat{\theta}_k[t]) \cdot \left\lfloor\left|\delta\hat{\theta}_k[t]\right|\right\rfloor\right) \mod L$$

$$\delta\hat{\theta}_k[t] = 0 \text{ if phase was corrected}$$

Box 6.1: Sandsbots model.

6.2 *Order Parameter*

To express the correlation between the phases θ_k and the angular positions γ_k of the agents, we use the order parameter S introduced in Box 6.3 [6]. Its value varies from 0 (no correlation) to 1 (perfect correlation). The angular position is $\gamma_k = \arctan(y_k/x_k)$ with the coordinates x_k and y_k with respect to the centroid of the whole swarm. Perfect correlation occurs in the static phase wave.

$$
\begin{aligned}
\mathbf{I}_1(\mathbf{x}_{jk}) &= \eta_{\text{attr}}\mathbf{x}_{jk} \\
\mathbf{I}_2(\mathbf{x}_{jk}) &= \eta_{\text{rep}}\frac{\mathbf{x}_{jk}}{\left\|\mathbf{x}_{jk}\right\|\left(\left\|\mathbf{x}_{jk}\right\| - d\right)} \\
F_1(\mathbf{x}_{jk}) &= 1 + J_1\cos\left(\theta_{jk}\right) \\
F_2(\mathbf{x}_{jk}) &= 1 - J_2\cos\left(\theta_{jk}\right) \\
\Gamma_1(\theta_{jk}) &= \sum_{m\in\mathcal{M}_1}\frac{K_m}{m}\sin\left(m\theta_{jk}\right) \\
\Gamma_2(\theta_{jk}) &= \sum_{m\in\mathcal{M}_2}\frac{K_m}{m}\sin\left(m\theta_{jk}\right) \\
\Lambda_1(\mathbf{x}_{jk}) &= 1 + P\frac{\epsilon_d + d}{\left\|\mathbf{x}_{jk}\right\|} \\
\Lambda_2(\mathbf{x}_{jk}) &= 1 - P\frac{\epsilon_d + d}{\left\|\mathbf{x}_{jk}\right\|}
\end{aligned}
$$

Box 6.2: Interaction functions.

$$
\begin{aligned}
S &= \max(S_+, S_-) \\
S_\pm &= \frac{1}{N}\left|\sum_{k=1}^{N} e^{\mathrm{i}(\gamma_k \pm \theta_k)}\right|
\end{aligned}
$$

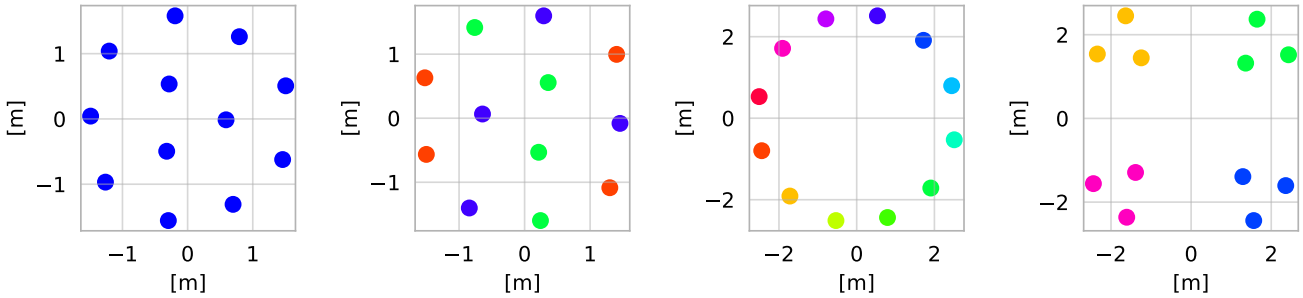
Box 6.3: Correlation between phases and angular positions.

6.3 *Patterns*

We now show that the sandsbot model is able to reproduce the patterns of the continuous sync and swarm model [6]. Some patterns are modified to guarantee they will emerge regardless of the initial conditions (static phase wave) or control their properties (static async, splintered phase wave). Such an approach allows to adjust the pattern for a specific task that needs to be executed. We show the position and phase of each agent, where the phase is indicated by a color.

Static Sync

The *static sync* pattern (Figure 6.1a) is created almost without any modifications in comparison to the one obtained from swarmalator



(a) Static sync: $J_1 = 0.1$, $K = 1$, $M = 1$, $P = 0.1$.

(b) Static async: $J_1 = -1$, $K = 0.2$, $M = 3$, $P = -0.03$.

(c) Static phase wave: $J_1 = 1.4$, $K = 1$, $M = N = 12$, $P = 0.1$.

(d) Splintered phase wave: $J_1 = 1.5$, $K = 1$, $M = 4$, $P = 0.1$.

Figure 6.1: Spatio-temporal patterns and parameter values for which they were obtained.

model [6]. All agents synchronize their phases and gather evenly distributed on a disk. This state is formed if $J_1 > -1$, $K > 0$, $M = 1$, and $P \geq 0$.

Static Async

The *static async* pattern distributes the agents on a disk as well, but their phases are now evenly distributed and similar phases spread in space. With the parameters of the swarmalator model [6] for this state ($K < 0$, $J_1 < 0$) and $M = 1$, the discrete phase levels make the obtained pattern unstable with phases constantly changing back and forth. Thus, we introduce a controlled version of this pattern (Figure 6.1b) in which we specify the number of clusters M to be created ($1 < M < N$) and set $K > 0$, $J_1 < 0$, and $P < 0$. Agents with different phases attract each other stronger in space and try to form clusters. However, phase interaction between agents is stronger if they are more distant from each other. Agents form M clusters in the phase domain and each phase cluster spreads on a disk.

Static Phase Wave

The *static phase wave* (Figure 6.1c) formed by our model looks similar to the one obtained from the swarmalator model [6]. The agents form an annulus and sort their uniformly distributed phases. Sandsbots can form this pattern regardless of their initial conditions, contrary to the swarmalator model [6], in which phase coupling does not exist in this pattern. This pattern appears for $K > 0$, $J_1 > 0$, $M = N$ and $P > 0$. In a special case of this pattern (called *ring phase wave* [70]) the agents are placed on a circle. In Appendix C we present a method to calculate the radius of the ring phase wave in our model.

Splintered Phase Wave

The *splintered phase wave* in the swarmalator model [6] splits the agents in space into clusters with similar phases. The clusters are positioned on a ring. The number of clusters formed depends on the initial conditions, and the agents keep changing phases slightly and move within their clusters. Similarly to static async, we propose a controlled version, in which we specify the number of clusters and set $K > 0$, $J_1 > 0$, and $P > 0$. Agents cluster in the phase domain, are placed on an annulus, and split into clusters in space based on their phase. After this pattern is formed, the positions and phases remain unchanged.

Active Phase Wave

Agents in the *active phase wave* form a ring and keep changing their phase while they travel around the ring to get close to the ones having a similar phase. This pattern is achieved for $K < 0$, $J_1 > 0$, $M = 1$, and $P > 0$. As we do not control the number of clusters directly here, $\hat{U}^{(M)}$ never converges to 0. Therefore, to speed up the formation, the influence of the phase potential on the step size s might be disabled ($G = 1$). Although we focus on stationary patterns, we reproduce this pattern for the sake of completeness, to show that it is possible to obtain similar behavior with discrete phase, but do not analyze it further.

Values of Order Parameters

To quantitatively classify the behavior obtained by each of the models we use order parameters (Boxes 4.1, 5.2 and 6.3). A combination of them allows us to formally distinguish the spatio-temporal patters. Here we provide values that identify each of the static patterns:

Static sync: The static sync pattern is the only one in which the agents are synchronized, therefore the synchronization order parameter $r = 1$ is sufficient to identify it. Nevertheless, we know that the agents should form a disk, therefore swarming order parameter $V > 1$. There should be no correlation between phase and position ($S = 0$).

Static async: In the static async pattern the phases of the agents should be in a balanced state ($r = 0$). Similarly to static sync, the agents form a disk, hence $V > 1$. The correlation between phase and position (S) should be low, but due to using only limited number of phase clusters, it does not necessarily converge to 0.

Static phase wave: Static phase wave pattern also requires the phases to be in a balanced state ($r = 0$). However, the agents should form an annulus, in which $V < 1$. Moreover, if $V = 0$ the agents form a circle, which is a special case of this pattern — ring phase wave. The main characteristic of this pattern is the full correlation between phase and angular position ($S = 1$).

Splintered phase wave: Similarly to the previous two patterns phases should be in a balanced state ($r = 0$). The agents form an annulus, so the swarming order parameter $0 < V < 1$. The correlation between phases and angular position should be high, but not full ($0 < S < 1$).

6.4 Simulation-based Analysis

The sandsbot model is analyzed in more detail with a simulation implemented in *Python 3*. We first study how to form patterns with the discrete model in perfect conditions. The imperfections of robots and issues associated with communication are not taken into account. Agents are connected without delays and packet loss. This leads to full knowledge about phase levels and positions of other agents and perfect synchronization of the clocks. The agents can move freely in space, with the only constraint being their maximum speed. Real-world issues related to communication, movement constraints, and hardware imperfections are addressed later in our proof-of-concept with robotic platforms (Section 6.5).

All agents start with random phase levels and random positions in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ square both drawn from the uniform distributions. The initial positions are redrawn until they meet the minimum distance constraint (Equation (5.14)). The clocks ϕ_k are synchronized and $\delta\hat{\theta}_k = 0$. Simulations are run with $T = 0.125\,\mathrm{s}$, $\epsilon_d = 0.1\,\mathrm{m}$, $d = 0.2\,\mathrm{m}$, and $v^{\mathrm{maxR}} = 0.2\,\mathrm{m/s}$, where this choice of values is motivated by capabilities of robots and aims at simulating a setup similar to the experimental one.

Results

Order Parameters

The plots in Figures 6.2 to 6.5 show for each of the static patterns how the three order parameters evolve over time. In each plot, three instants are marked (dashed black lines) for which the corresponding space-time patterns are shown below. These instants are chosen to depict the starting condition, process of pattern formation, and the final pattern.

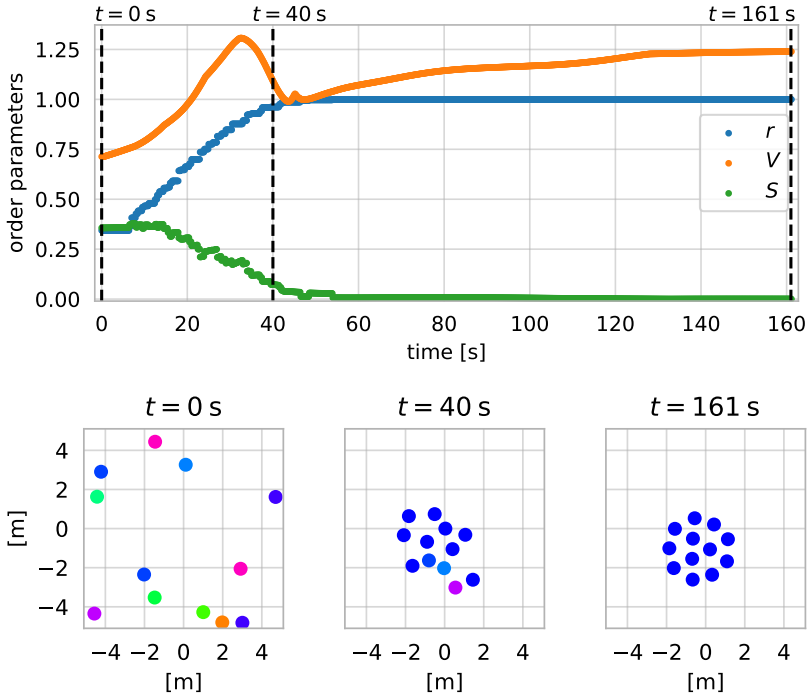


Figure 6.2: Convergence of order parameters and snapshots of the system showing the process of forming of static sync.

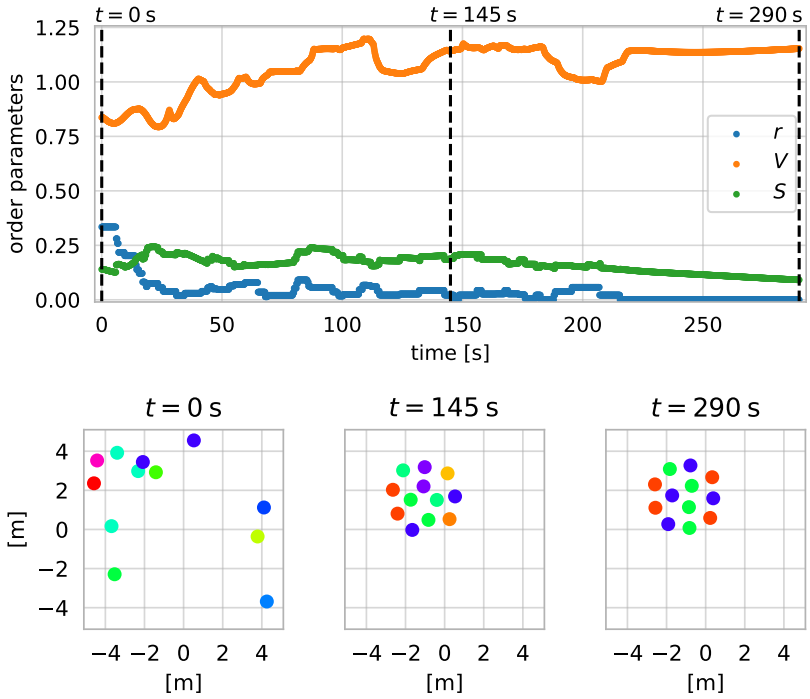


Figure 6.3: Convergence of order parameters and snapshots of the system showing the process of forming of static async.

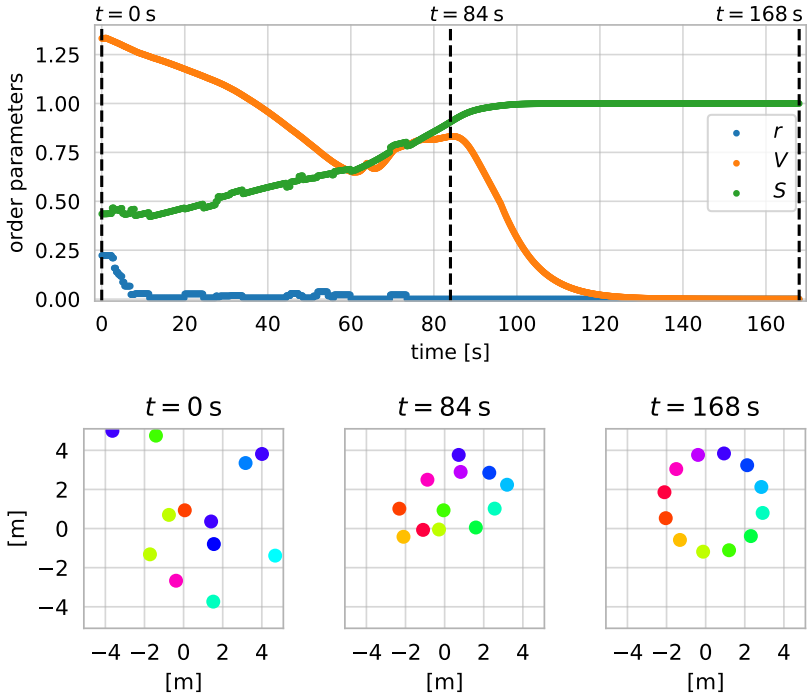


Figure 6.4: Convergence of order parameters and snapshots of the system showing the process of forming of static phase wave.

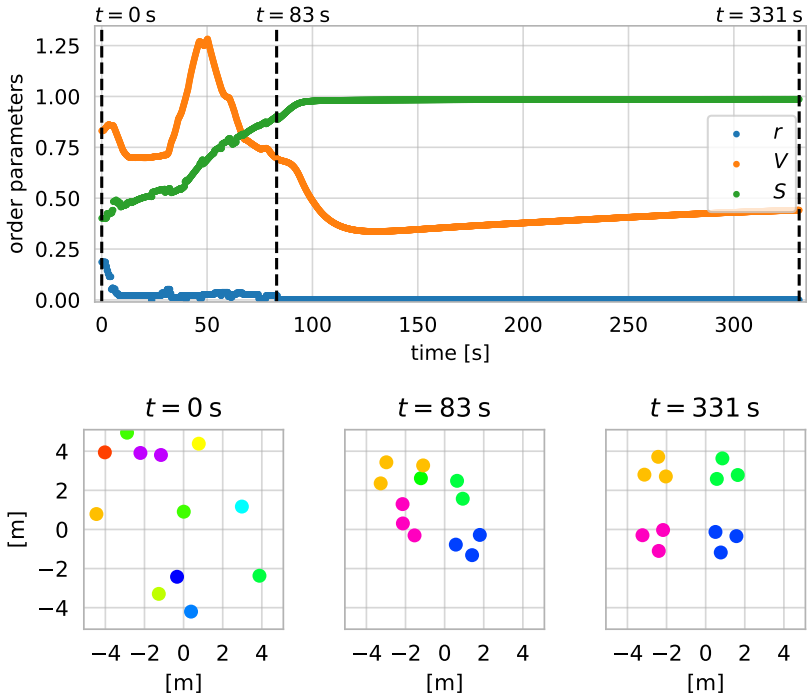


Figure 6.5: Convergence of order parameters and snapshots of the system showing the process of forming of splintered phase wave.

The plots confirm that for all patterns the order parameters converge to the values described in Section 6.3. This shows that the introduced combination of order parameters can serve as a tool to distinguish the patterns. The splintered phase wave (Figure 6.5) forms very slowly. However, a state similar to the final pattern is formed much earlier. Around $t = 130\,\mathrm{s}$, agents already form the correct temporal pattern and the clusters are placed on a ring. After that, the interactions between agents are minimal and the clusters slowly rotate to reach their final positions.

Impact of Period Length on Pattern Convergence

We now analyze the impact of the period length T and thus the frequency of message exchange on the convergence time and capabilities of the model. Sandsbots are compared to swarmalators with interaction functions used by O'Keeffe *et al.* [70]. To ensure fair comparison, we add collision avoidance and maximum speed to the swarmalator model, as done in the swarmalatorbot model (see Section 3.2).

A comparison is possible only for the static sync pattern as it is the sole pattern emerging identically in both models. The static async pattern and splintered phase wave with sandsbots differ significantly from their theoretical counterparts. Whereas the static phase wave now involves phase interactions and forms regardless of the initial conditions although it might converge slower.

We observed that sandsbots successfully form the static sync pattern even with very long periods ($T = 5\,\mathrm{s}$), but forming takes significantly longer compared to short periods ($T = 0.1\,\mathrm{s}$). In contrast, with swarmalators (continuous in nature), increasing the period leads to destabilization and physical oscillations ($T = 1\,\mathrm{s}$) and eventually prevents forming the pattern all together ($T = 5\,\mathrm{s}$).

It is assumed that the pattern is formed when all agents move slower than $1\,\mathrm{mm/s}$. The relationship between T and convergence time is presented in Figure 6.6. The swarmalator model with $T = 0.1\,\mathrm{s}$ (as used in the simulations in [6]) serves as a baseline. It can be observed that for very short periods (in the order of 0.1 s) the convergence time is similar to the one obtained with the swarmalator model and grows for increasing period length. If the pattern needs to be formed fast but at the same time the network load should be minimized, the following technique can be used: start with a short period (in the order of 0.1 s) to form the pattern and then increase the period to reduce the communication load but keep the pattern stable.

Figure 6.6 also shows the number of exchanged messages for different period lengths. An interesting observation is that for very

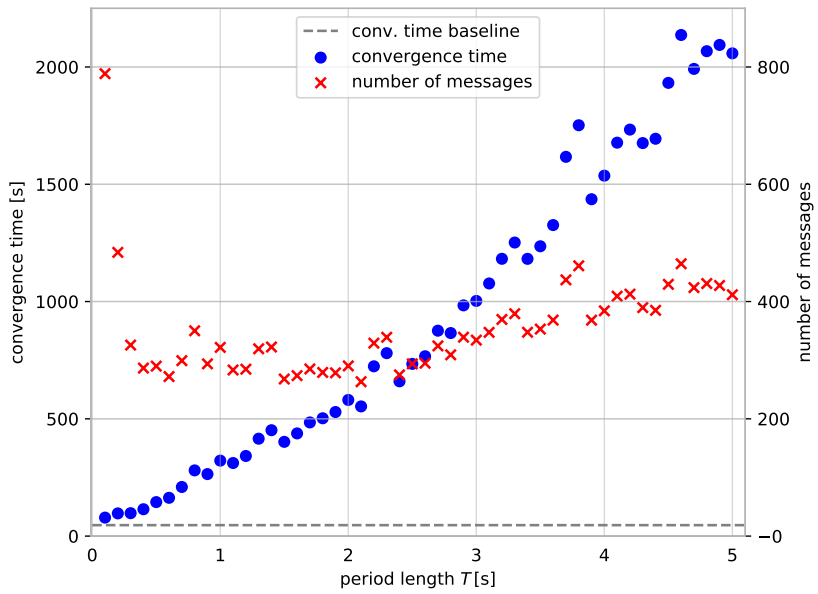


Figure 6.6: Relationship between period length and convergence time and the number of exchanged messages for static sync pattern.

short periods (0.1 s and 0.2 s) the number of exchanged messages required to form the pattern is much higher than for longer periods. This happens because the robots could travel safely with their maximum speed for a time longer than the period, but they are forced to exchange data anyway. It means that the pattern formation for such short periods is inefficient in terms of the number of exchanged messages. For longer periods, the number of messages grows slowly with the period length due to the adaptive speed limit (Equation (5.12)).

6.5 *Experimental Results*

Finally, we validate the feasibility of the sandsbots model with two robot platforms (both are described in Section 3.1). The recordings of our experiments are available in Video A.4 (see Appendix A). First, we use Crazyflie drones to check whether the simple model of robot dynamics is sufficient and whether the discrete model works correctly even with imperfect estimation of future positions. Second, we use Pololu Balboa self-balancing robots to see how the model behaves under realistic communication conditions with non-deterministic delays and message drops. We describe the setup for each platform and the results achieved in practical experiments. In each experiment, robots start from arbitrary positions with random phases.

Figure 6.7: *Static sync*: Crazyflies forming the pattern. Source: picture provided by the University of Klagenfurt.

Crazyflies

Eight Crazyflies are controlled by a server, which acquires the positions of all of them from an Optitrack motion capture system. This guarantees that full information is available with minimal latency to calculate state updates for each agent, thus allowing us to omit potential communication problems. At the same time, the movement dynamics of the agents is realistic. At the beginning of each oscillation cycle (i.e., whenever $\phi_k = 0$) the server transmits new velocity and color to visualize the phase on a ring of RGBW LEDs attached to each robot. This experiment is run with the following parameters: $T = 0.5\,\mathrm{s}$, $\epsilon_d = 0.1\,\mathrm{m}$, $d = 0.3\,\mathrm{m}$ and $v^{\mathrm{maxR}} = 0.2\,\mathrm{m/s}$. Because of a constrained communication interface we use $\omega = \frac{2}{L}\,\mathrm{s}^{-1}$, which means that the velocity and color are updated twice per second.

All patterns are successfully formed by the Crazyflies. An exam-

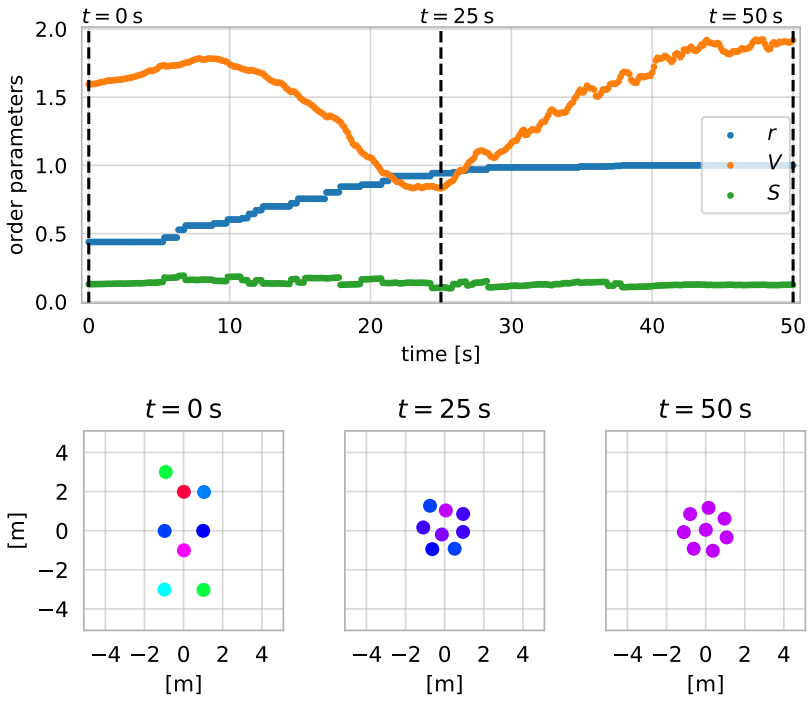


Figure 6.8: *Static sync*: Convergence of order parameters and snapshots of the system showing the process of pattern forming with Crazyflies.

ple is presented in Figure 6.7. Similar to the results of simulations, we studied how the order parameters converge for this pattern. In Figure 6.8 it is visible that the convergence of the swarming order parameter is not as smooth as in the simulation. It is caused by the imperfect predictions of positions related to movement dynamics and disturbances. Nevertheless, the values of order parameters stabilize and the pattern is successfully formed.

Balboas

For further evaluation — taking into account both movement dynamics and realistic communication conditions — we use a robot swarm platform based on Pololu Balboa robots. As in the other experiments, robots communicate via IEEE 802.11bg in ad-hoc mode, which enables them to join and leave the group without any infrastructure or single point of failure. From the software perspective, the communication is realized in ROS 2 (Eloquent Elusor release) using the Data Distribution Service (DDS) communication standard, which applies a Real-Time Publish Subscribe (RTPS) protocol (we use eProsima Fast RTPS). The robots communicate in best-effort mode with multicast enabled to reduce communication load.

The robots need to know their positions. Outdoors they could

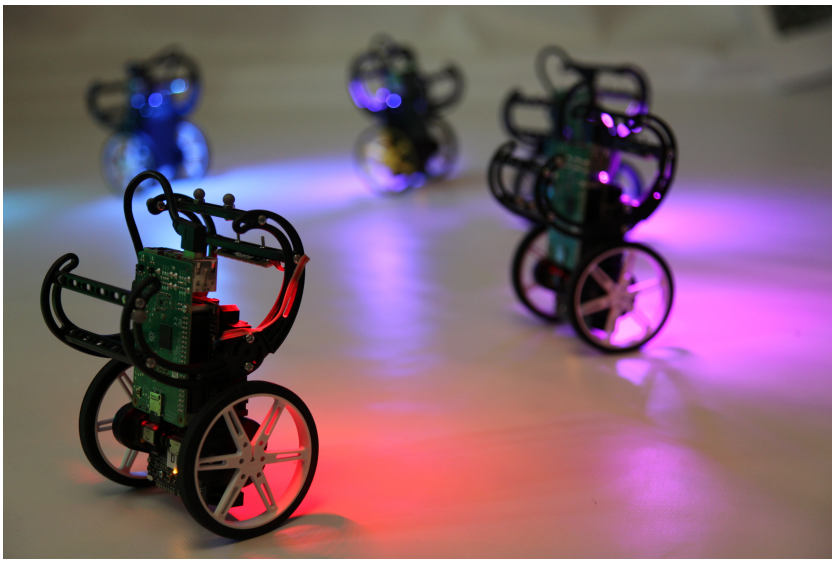


Figure 6.9: *Static phase wave*: Balboa robots forming the pattern.

use a satellite-based positioning system, but as our demonstrator operates indoors, we utilize an Optitrack motion capture system. To ensure that the demonstrator will be easily transferable to real-world applications (including outdoors), the robots use the motion capture system in the same way as they would use an outdoor solution: each robot acquires only its own position and ignores messages sent to other agents. Each robot visualizes its phase level by the hue of the color of an LED strip attached to the bumpers.

The wireless channel is only used to exchange the states (phase levels and positions), where each robot receives messages from all others. The signaling effort depends on the natural frequency of oscillations. In our experiments, each robot sends eight messages per second (we use $\omega = \frac{8}{L}\,\mathrm{s}^{-1}$).

The total number of agents and the number of agents with the same phase level are not explicitly provided to the robots. For an update of phase correction and velocity, the number of messages received during the last oscillation cycle is assumed to be the number of agents. Therefore, if the messages are significantly delayed or dropped, it can have an impact on the convergence and stability of the desired pattern.

The same parameters as in the simulation are used: $T = 0.125\,\mathrm{s}$, $\epsilon_d = 0.1\,\mathrm{m}$, $d = 0.2\,\mathrm{m}$, and $v^{\mathrm{maxR}} = 0.2\,\mathrm{m/s}$.

The Balboas acting as sandsbots are able to successfully form the patterns. A snapshot of robots forming the static phase wave is shown in Figure 6.9. The convergence of order parameters for this pattern is presented in Figure 6.10. The pattern might get disturbed

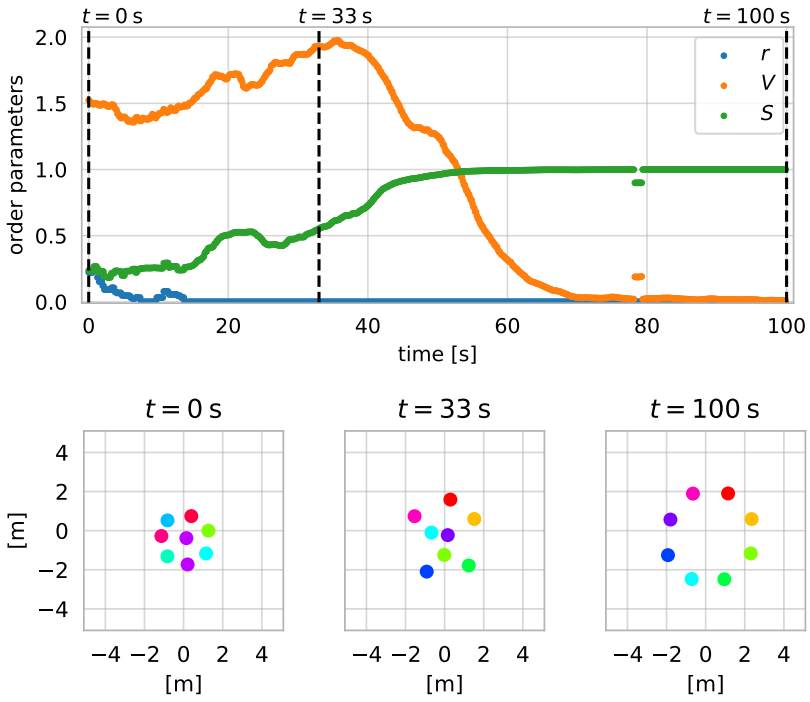


Figure 6.10: *Static phase wave*: Convergence of order parameters and snapshots of the system showing the process of pattern forming with Balboa robots.

due to the non-deterministic communication delays and message drops (as happened at 78 s), but it quickly recovers.

6.6 Chapter Summary

In this chapter we showed how the two previously introduced models for temporal coordination and spatial aggregation are coupled in a unified sandsbot model. Together they create a time-discrete model suited for multi-robot systems that successfully form spatio-temporal patterns of swarmalator theory [6]. It features controlled versions of static async and splintered phase wave with a specified number of clusters. Moreover, the static phase wave forms regardless of the initial conditions. These modifications enable us to create predictable patterns which can be employed in various missions. Sending synchronized periodic updates makes the system robust when messages get delayed.

7 Conclusions

In this thesis we started from the purely theoretical, mathematical model of swarmalators (Section 2.3) with the goal to apply a similar concept in multi-robot systems. Our preliminary work — swarmalatorbot model — directly implemented the swarmalator model and took into account movement constraints of robots. Even though it was enough to serve as a proof of concept for robots that sync and swarm, it still suffered from some problems related to properties of multi-robot systems. To summarize our work, we recall these problems of the swarmalator model [6] from the perspective of multi-robot systems (indicated in Section 3.3) and show that the sandsbot model indeed solves these problems.

7.1 Sandsbot Model for Multi-Robot Systems

In Table 7.1 we outline the main requirements of multi-robot systems towards the sync and swarm solution and compare their fulfillment by the three models: swarmalator, swarmalatorbot and sandsbot. The following sections provide details regarding these aspects.

requirement of multi-robot systems	swarmalator	swarmalatorbot	sandsbot
collision avoidance	✗	✓[+]	✓
movement constraints	✗	✓	✓
reproducible patterns	✗	✗	✓
discrete coupling	✗	✓[+]	✓
robust against delays	✗	✗	✓

[+] Does not work if the update rate is too low.

Table 7.1: Summary of properties of three models: swarmalator, swarmalatorbot and sandsbot.

Collision Avoidance

In the swarmalator model by O'Keeffe *et al.* [6] the agents are considered to be point-shaped particles. Due to their infinitely small size,

the possible collisions are not taken into account. Nevertheless, the repulsion force guarantees that two swarmalators cannot meet at exactly the same location.

In multi-robot systems such an assumption cannot be applied, as the centers of robots can still be quite far from each other at the moment of collision. Therefore, already in the swarmalatorbot model we took into account the size of the robots (specifically their bounding boxes). The repulsion between robots is calculated based on the distance between the bounding boxes, not the centers. This approach assured that the robots will be repelled from each other if they get very close. However, the decreased update rate of communication with other agents could have still led to collisions.

In the spatial aggregation used by the sandsbot model collision avoidance is also based on the distance between bounding boxes, however it is included in a slightly different way in the model. Moreover, we introduced a safety area around each agent (see Figure 3.9) and limited speed of the agents based on the size of their safety areas and the frequency of updates. This approach guarantees that the safety areas of agents will never overlap, protecting them from collisions, even if messages are exchanged at a very low rate.

Movement Constraints

The swarmalators in the mathematical model are able to freely move in space. This assumption is unrealistic in robotic systems, thus already in the swarmalatorbot model we needed to take into account physical capabilities of robots: maximum speed and movement constraints. In the sandsbot model we use the same approach, however the maximum speed can be further reduced based on safety features of the model.

Reproducibility of Patterns

The final shape of some patterns in the swarmalator model depends on the initial conditions. For example in the static phase wave the phase coupling does not exist, which means that the phases of agents need to be uniformly distributed since the beginning to form a rainbow-like pattern. In contrast, if the agents start with the synchronized phases, with the same parameters, static sync will emerge. Swarmalatorbot, as a direct implementation of the mathematical model, was facing the same problems. As a consequence, we were unable to reproduce splintered phase wave pattern on the robots, as the number of agents was too small in comparison to the number of clusters that would have been formed.

In the sandsbot model we use temporal coordination to directly

control the number of phase clusters. This allows us to have phase interactions present in each state. Thanks to this all the patterns can be formed independently on the initial conditions. For example, even if the phases of all the agents are synchronized, with the proper parameters, the static phase wave can emerge. The forming process might take longer, though. Nevertheless, the needed pattern can be formed at any stage of the mission, whenever it is useful.

Communication

Swarmalator model was based on the differential equations, which assumed that the coupling between agents is continuous and instant, i.e., the states of all agents are known at any point in time without delay. When the agents need to communicate, these assumptions do not hold: only a limited number of messages can be exchanged (coupling needs to be discrete) and the messages get delayed in the network. Therefore, the swarmalatorbot model needed to utilize discrete coupling. It was realized as a straightforward discretization of the swarmalator model and did not take into account delays. However, with high enough update rates it was able to create the same patterns as the continuous counterpart. Nevertheless, in some cases the delayed communication led to destabilization of the patterns. For example, for the non-zero natural frequency of oscillations, delays caused static phase wave pattern to rotate, i.e., robots were traveling around the formed circle.

To account for the properties of a realistic network, we take into account discrete coupling already in the design of the sandsbot model. Moreover, we assume that the messages are delayed. To compensate delays, all agents are synchronized and they update their own states and exchange the messages only at given points in time. Each message contains the estimated state of an agent from the time, when the updates will be applied. This means that the message is valid for longer time (until the update) and gets outdated only if the delay is very high (compared to the update frequency). For the networks with high delays the update frequency can be adjusted to guarantee robust convergence of the model. Even though we do not directly take into account message drop, our experiments confirmed that the model quickly recovers if some messages get lost.

7.2 Limitations

The main limitation of the proposed model is the utilization of the global knowledge. Similarly to swarmalator model by O'Keeffe *et al.* [6] the agents update their phases and positions based on the

states of all other agents. In the course of this work we did initial experiments to base the updates only on the local information (i.e., states of the neighbors), but with some initial conditions the patterns were not converging. To guarantee the convergence with any initial conditions and be able to predict what pattern will form, we decided to use the global knowledge. Nevertheless, the modifications, or new models that relax this requirement might help to further reduce the communication load and improve robustness of the model.

As a consequence of the requirement of the global knowledge, the communication used by agents should be reliable (i.e., all messages should be delivered). However, in our robotic experiments, we used unreliable communication, which resulted in some messages being dropped. We observed that such disturbances are destabilizing the pattern, but the system is quickly recovering.

Finally, we limit our implementation to networked systems, in which the agents need to communicate their states. In principle, it should be possible to obtain similar results if the robots are equipped with a sensor that measures the relative distance and phase of other agents. Such a solution could be based on vision or emitted sounds, but we did not investigate such possibilities.

7.3 Outlook

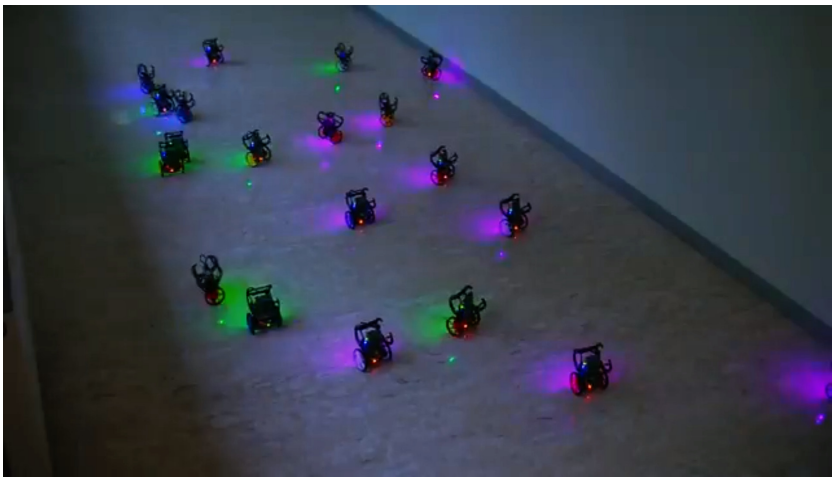
Future work on this topic might include direct consideration of message drop, which would increase robustness of the model in the real-world applications. Such modification can be made either in the model or in the implementation.

The other idea for future research might include different approaches to reducing the communication load required by sync and swarm behavior. For example, a different coupling temporal and spatial coordination might be developed that do not require global knowledge of the system.

Another path might include incorporating more complex movement dynamics of robots into the sandsbot model. It would also allow to limit the higher derivatives of position, not only velocity.

The other direction of continuing this research could be more application-oriented. It would include implementation of an interface that allows to execute practical missions by setting and configuring different patterns, depending on the actual needs. Such an interface might be very generic and support not only temporal, spatial and spatio-temporal patterns described in this thesis, but also some other swarm behaviors (e.g., collision avoidance or movement in formation). The aim of such work would be to allow the mission planner to perform high-level tasks in a self-organizing way.

A Supplementary Videos



https://youtu.be/BfPEWH4ASXE

Video A.1: Firefly Synchronization. This video presents a group of Balboa robots synchronizing with a firefly synchronization algorithm. This demonstration was also the first field test of Balboa robots.

https://youtu.be/Db6aiSa4soU

Video A.2: Drones that sync and swarm. This video presents Crazyflie UAVs running the swarmalatorbot model. The demonstration was prepared for the opening event of the drone hall at the University of Klagenfurt.
Video by: Prof. Christian Bettstetter.
Overlay picture provided by the University of Klagenfurt.

https://youtu.be/atLhROLzsFo

Video A.3: Robots that sync and swarm: A proof of concept in ROS 2. This video presents the ideas behind the swarmalatorbot model and the experiments featuring Balboa robots. It was prepared as a supplementary video for the publication [1].

https://youtu.be/YCZREhgCF84

Video A.4: Sandsbots: Robots that sync and swarm. The video presents experiments featuring Balboa robots and Crazyflie UAVs running the sandsbot model. It was prepared as a supplementary video for the publication [3].

B Software Framework for Sandsbots

During the course of the work presented in this thesis we needed to run numerous simulations and experiments on robots. After the initial implementation of swarmalatorbots we came up with some requirements for a more generic software framework, that would facilitate our work.

1. It should be possible to run the same code in the simulation and on the robots. Our observation was that it is much harder to test code directly on the robots because of some hardware issues, limited battery life and risk of crashes. The possibility to use the same code regardless of the environment allows for easier and faster testing and resolving of issues. Thanks to this the code running on the robots is already pre-tested and correct, while only the part related to interfacing the hardware needs to be integrated and can potentially cause problems.
2. The configurations should be easily exchangeable. For example, it should be possible to quickly change communication middleware or a type of robot.
3. The effort of adding new hardware interface (i.e., different type of robot) should be minimized.

These assumptions are realized in the implementation used throughout this thesis. We abstract a few type of components present in each system. In Section B.1 we describe the roles of these components.

B.1 Components

Controller

Controller is the main part of the framework. It can be thought of as a part of the whole system that joins everything together. The controller contains instances of all other components and runs the main loop of the program. It can handle a single agent or multiple

agents at once, which is especially useful when running a simulation. There are a few controllers implemented in the framework, but they differ mostly in the technical details and have the same principle of operation.

System State

System state keeps the information needed for operation of the agents that are being handled by one controller. For each of the agents it holds two types of data: *state* of an agent and *knowledge* of the states of all the other agents. State consists of the position and phase of an agent and other values that are needed to control an agent, e.g., velocity.

Knowledge contains the known states of all the other agents. There are two different implementations of it: *separate* and *shared*. In case of a separate one the knowledge of agents is independent, i.e., each agent can have a different information about the state of the whole system. The differences can occur if the messages are not broadcasted, i.e., addressed, or when some messages were not received properly by some agents. In contrast, in the shared knowledge all agents have access to the same information, which improves performance especially for bigger simulations.

Logic

Logic is a core of the behavior of agents — it includes the implementation of the models. The general idea behind this part is that there are separate logic implementations for *temporal* coordination and *spatial* aggregation. Besides the two models described in this thesis there are also some basic behaviors like firefly synchronization and aggregation without any modifications, or the spatial and temporal parts from the swarmalatorbot model. The implemented models can be freely combined to create a spatio-temporal behavior, an example of such combination are sandsbot and swarmalatorbot models. The output of the update function of logic is *state update*. State update contains the output of the model and its internal variables. For example, in the case of sandsbot model, it contains not only new phase and velocity, but also current value of phase level correction.

Communication

In our problem, the agents need to exchange information about their state with others. *Communication* is the part that allows them to do this. In a robotic setup we are using implementation based on ROS 2, however the rest of the system should still be independent of ROS 2.

For the simulation purposes it is often not needed and running ROS 2 decreases the performance. Therefore, we implemented "offline" communication, in which the state sent by an agent is automatically saved in the knowledge of the others.

Visualization and Robots

The *visualization* component is responsible for displaying the state of the system. In almost all cases we are using *live* visualization, which plots positions and phases of all agents to have the overview of the system state and see if the model is working properly. We have also implemented *hardware interface* visualization of a robot that gets a new state (including velocity) and its task is to "display" these values by setting LED color and new speed of engines on actual robots. In other words, the framework treats robots as a visualization of the results of the model.

Position Feedback

Position feedback is the implementation of the interface to the external source of position, for example a motion capture system or GPS. For the simulation purposes we implemented a mock of a motion capture system that uses the information about the past position and velocity to calculate current position. Such an approach helps to keep a similar structure of the whole system to the one from the real experiments or even run mock experiments with robots but without the source of global position.

C Radius of the Ring Phase Wave

In this appendix we present a method to calculate radius of ring phase wave. At the beginning we list formulas (Section C.1) and assumptions (Section C.2) that are useful in the latter part. The calculations are performed in two steps: First, we repeat the computations of the radius of the ring phase wave performed by O'Keeffe *et al.* [70] (Section C.3). Next, we perform similar computations for the sandsbot model (Section C.4) and propose a numerical method to approximate it (Section C.5).

C.1 Useful Formulas

$$\cos^2(\alpha) = \frac{\cos(2\alpha) + 1}{2} \tag{C.1}$$

$$\sum_{k=1}^{n} \cos(k\alpha) = \frac{\sin\frac{n\alpha}{2}}{\sin\frac{\alpha}{2}} \cos\frac{(n+1)\alpha}{2} \tag{C.2}$$

From Equations (C.1) and (C.2):

$$\sum_{k=1}^{n} \cos^2(k\alpha) = \frac{1}{2}\sum_{k=1}^{n} \cos(2k\alpha) + \frac{n}{2} = \frac{1}{2}\frac{\sin(n\alpha)}{\sin\alpha} \cos\left((n+1)\alpha\right) + \frac{n}{2} \tag{C.3}$$

C.2 Assumptions

Similarly to O'Keeffe *et al.* [70], we describe position and phase of agent k in the ring phase wave pattern with:

$$\mathbf{x}_k = R\cos\left(\frac{2\pi k}{N}\right)\hat{x} + R\sin\left(\frac{2\pi k}{N}\right)\hat{y} \tag{C.4}$$

$$\theta_k = \frac{2\pi k}{N} + C \tag{C.5}$$

Furthermore, we notice that agents are stationary, so step size and speed constraints do not influence the result. The only parameters specific to sandsbots that influence the size of the pattern are: attraction coefficient η_{attr}, repulsion coefficient η_{rep} and size of the bounding box d.

C.3 Calculations for the Swarmalator Model

In this section we reproduce the results obtained by O'Keeffe *et al.* [70], where the detailed computations were not provided, only the assumptions and results.

Based on Equations (C.4) and (C.5) we can calculate the distance between agents k and j:

$$\begin{aligned}\left\|\mathbf{x}_{jk}\right\| &= \left\|R\left(\cos\left(\frac{2\pi j}{N}\right)-\cos\left(\frac{2\pi k}{N}\right)\right)\hat{x}+R\left(\sin\left(\frac{2\pi j}{N}\right)-\sin\left(\frac{2\pi k}{N}\right)\right)\hat{y}\right\| = \\ &= R\left\|\left(-2\sin\left(\frac{2\pi(j+k)}{2N}\right)\sin\left(\frac{2\pi(j-k)}{2N}\right)\right)\hat{x}+\left(2\sin\left(\frac{2\pi(j-k)}{2N}\right)\cos\left(\frac{2\pi(j+k)}{2N}\right)\right)\hat{y}\right\| = \\ &= R\sqrt{4\sin^2\left(\frac{2\pi(j+k)}{2N}\right)\sin^2\left(\frac{2\pi(j-k)}{2N}\right)+4\sin^2\left(\frac{2\pi(j-k)}{2N}\right)\cos^2\left(\frac{2\pi(j+k)}{2N}\right)} = \\ &= 2R\sqrt{\sin^2\left(\frac{2\pi(j-k)}{2N}\right)\left(\cos^2\left(\frac{2\pi(j+k)}{2N}\right)+\sin^2\left(\frac{2\pi(j+k)}{2N}\right)\right)} = \\ &= 2R\sin\left(\frac{2\pi(j-k)}{2N}\right)\end{aligned} \tag{C.6}$$

In the ring phase wave agents are stationary, so the following relation is true for each k:

$$0=\dot{\mathbf{x}}_k=\frac{1}{N}\sum_{\substack{j=1\\ j\neq k}}^{N}\left(1+J_1\cos\left(\theta_{jk}\right)\right)\mathbf{x}_{jk}-\left(1-J_2\cos\left(\theta_{jk}\right)\right)\frac{\mathbf{x}_{jk}}{\left\|\mathbf{x}_{jk}\right\|^2} \tag{C.7}$$

Without the loss of generality, the rest of computations are performed for $k=N$.

$$\begin{aligned}0=\dot{\mathbf{x}}_N &= \frac{1}{N}\sum_{j=1}^{N-1}\left(1+J_1\cos\left(\theta_{jN}\right)\right)\mathbf{x}_{jN}-\left(1-J_2\cos\left(\theta_{jN}\right)\right)\frac{\mathbf{x}_{jN}}{\left\|\mathbf{x}_{jN}\right\|^2} = \\ &= \frac{1}{N}\sum_{j=1}^{N-1}\left(1+J_1\cos\left(\frac{2\pi j}{N}\right)\right)\left(R\left(\cos\left(\frac{2\pi j}{N}\right)-1\right)\hat{x}+R\sin\left(\frac{2\pi j}{N}\right)\hat{y}\right)- \\ &- \left(1-J_2\cos\left(\theta_{jN}\right)\right)\frac{\left(R\left(\cos\left(\frac{2\pi j}{N}\right)-1\right)\hat{x}+R\sin\left(\frac{2\pi j}{N}\right)\hat{y}\right)}{4R^2\sin^2\left(\frac{2\pi j}{2N}\right)}\end{aligned} \tag{C.8}$$

We observe that the formation is symmetrical with respect to axis X. Because agent N is placed on this axis, the $\hat{y}$ components of forces acting on it will cancel out. Therefore, in the rest of the computation

we focus only on $\hat{x}$ components. By splitting attraction and repulsion forces in separate sums, moving repulsion force to the other side of the equation and simplifying we get:

$$\sum_{j=1}^{N-1}\left(1+J_1\cos\left(\frac{2\pi j}{N}\right)\right)\left(\cos\left(\frac{2\pi j}{N}\right)-1\right)=\sum_{j=1}^{N-1}\left(1-J_2\cos\left(\theta_{jN}\right)\right)\frac{\cos\left(\frac{2\pi j}{N}\right)-1}{4R^2\sin^2\left(\frac{2\pi(j)}{2N}\right)} \tag{C.9}$$

Let us first consider the left side of Equation (C.9):

$$\begin{aligned}A&=\sum_{j=1}^{N-1}\left(1+J_1\cos\left(\frac{2\pi j}{N}\right)\right)\left(\cos\left(\frac{2\pi j}{N}\right)-1\right)\\&=\sum_{j=1}^{N-1}\left(\cos\left(\frac{2\pi j}{N}\right)+J_1\cos^2\left(\frac{2\pi j}{N}\right)-1-J_1\cos\left(\frac{2\pi j}{N}\right)\right)\\&=(1-J_1)\sum_{j=1}^{N-1}\cos\left(\frac{2\pi j}{N}\right)+J_1\sum_{j=1}^{N-1}\cos^2\left(\frac{2\pi j}{N}\right)-(N-1)\end{aligned} \tag{C.10}$$

Using Equations (C.2) and (C.3) we compute:

$$\begin{aligned}\sum_{j=1}^{N-1}\cos\left(\frac{2\pi j}{N}\right)&=\frac{\sin\frac{(N-1)2\pi}{2N}}{\sin\frac{2\pi}{2N}}\cos\frac{N2\pi}{2N}=\\&=\frac{\sin\left(\pi-\frac{2\pi}{2N}\right)}{\sin\frac{2\pi}{2N}}\cos(\pi)=1\cdot(-1)=-1\end{aligned} \tag{C.11}$$

$$\begin{aligned}\sum_{j=1}^{N-1}\cos^2\left(\frac{2\pi j}{N}\right)&=\frac{1}{2}(N-1)+\frac{1}{2}\frac{\sin\frac{(N-1)2\pi}{N}}{\sin\frac{2\pi}{N}}\cos\frac{N2\pi}{N}=\\&=\frac{1}{2}(N-1)+\frac{1}{2}\cdot(-1)\cdot1=\frac{1}{2}(N-1)-\frac{1}{2}\end{aligned} \tag{C.12}$$

Substituting these results to Equation (C.10), we obtain:

$$\begin{aligned}A&=(1-J_1)(-1)+J_1\left(\frac{1}{2}(N-1)-\frac{1}{2}\right)-(N-1)=\\&=N\left(\frac{1}{2}J_1-1\right)\end{aligned} \tag{C.13}$$

Now, let us consider right side of the equation Equation (C.9):

$$\begin{aligned}B&=\frac{1}{4R^2}\sum_{j=1}^{N-1}\left(1-J_2\cos\left(\frac{2\pi j}{N}\right)\right)\frac{\cos\left(\frac{2\pi j}{N}\right)-1}{\sin^2\left(\frac{2\pi(j)}{2N}\right)}\\&=\frac{1}{4R^2}\sum_{j=1}^{N-1}\left(1-J_2\cos\left(\frac{2\pi j}{N}\right)\right)\frac{1-\sin^2\left(\frac{2\pi j}{2N}\right)-1}{\sin^2\left(\frac{2\pi(j)}{2N}\right)}\\&=\frac{1}{4R^2}(-2)\sum_{j=1}^{N-1}1-J_2\cos\left(\frac{2\pi j}{N}\right)\end{aligned} \tag{C.14}$$

Using the result of Equation (C.2) in Equation (C.14), we obtain:

$$B = \frac{1}{4R^2} \cdot (-2)(N-1) + 2J_2 \cdot (-1) = \frac{1}{4R^2} \cdot (-2)(N-1+J_2) \quad \text{(C.15)}$$

Next, we use Equations (C.13) and (C.15) to calculate R:

$$\begin{aligned} N \cdot (\frac{1}{2}J_1 - 1) &= \frac{1}{4R^2} \cdot (-2)(N-1+J_2) \\ R^2 &= \frac{(-2)(N-1+J_2)}{4N \cdot (\frac{1}{2}J_1 - 1)} \\ R^2 &= \frac{N-1+J_2}{N \cdot (2-J_1)} \\ R &= \sqrt{\frac{N-1+J_2}{N \cdot (2-J_1)}} \end{aligned} \quad \text{(C.16)}$$

C.4 Calculations for the Sandsbot Model

We start the computations in the same way as in the previous section. Using the same description of positions and phases we get similar formula:

$$\begin{aligned} &\sum_{j=1}^{N-1} \eta_{\text{attr}} \left(1 + J_1 \cos\left(\frac{2\pi j}{N}\right)\right) \left(\cos\left(\frac{2\pi j}{N}\right) - 1\right) = \\ &= \sum_{j=1}^{N-1} \eta_{\text{rep}} \left(1 - J_2 \cos\left(\frac{2\pi j}{N}\right)\right) \frac{\cos\left(\frac{2\pi j}{N}\right) - 1}{4R \sin\left(\frac{2\pi j}{2N}\right) \left(R \sin\left(\frac{2\pi j}{2N}\right) - d\right)} \end{aligned} \quad \text{(C.17)}$$

We observe that the left side of the equation is similar to Equation (C.10), after the same transformations we obtain the result:

$$A = \eta_{\text{attr}} N \left(\frac{1}{2}J_1 - 1\right) \quad \text{(C.18)}$$

For the right side of the equation, the computations are different. We substitute $d \approx R\frac{d}{\tilde{R}}$, where $\tilde{R}$ is the approximated radius (for details see Section C.5).

$$\begin{aligned} B &= \sum_{j=1}^{N-1} \eta_{\text{rep}} \left(1 - J_2 \cos\left(\frac{2\pi j}{N}\right)\right) \frac{\cos\left(\frac{2\pi j}{N}\right) - 1}{4R \sin\left(\frac{2\pi j}{2N}\right) \left(R \sin\left(\frac{2\pi j}{2N}\right) - d\right)} \approx \\ &\approx \sum_{j=1}^{N-1} \eta_{\text{rep}} \left(1 - J_2 \cos\left(\frac{2\pi j}{N}\right)\right) \frac{1 - 2\sin^2(\frac{2\pi j}{2N}) - 1}{4R \sin\left(\frac{2\pi j}{2N}\right) \left(R \sin\left(\frac{2\pi j}{2N}\right) - R\frac{d}{\tilde{R}}\right)} = \\ &= \frac{\eta_{\text{rep}}}{4R^2} \sum_{j=1}^{N-1} \left(1 - J_2 \cos\left(\frac{2\pi j}{N}\right)\right) \frac{-2 \sin\left(\frac{2\pi j}{2N}\right)}{\sin\left(\frac{2\pi j}{2N}\right) - \frac{d}{\tilde{R}}} \end{aligned} \quad \text{(C.19)}$$

Combining the results of Equations (C.18) and (C.19) yields the following formula for R:

$$R = \sqrt{\frac{\eta_{\text{rep}}}{\eta_{\text{attr}}} \frac{\sum_{j=1}^{N-1} \left(1 - J_2 \cos\left(\frac{2\pi j}{N}\right)\right) \frac{\sin\left(\frac{2\pi j}{2N}\right)}{\sin\left(\frac{2\pi j}{2N}\right) - \frac{d}{\tilde{R}}}}{N(2 - J_1)}} \tag{C.20}$$

Note that the result depends on $\tilde{R}$. In the next section we show how to obtain the value of this approximation.

C.5 Numerical Method

We substitute Equations (C.4) and (C.5) into the model (Box 6.1). Due to the modification of the model that assures collision avoidance, we were not able to analytically compute R. Therefore, for given N, J_1, J_2, η_{attr} and η_{rep}, we use the following numerical procedure:

1. To get the first approximation of R, we set $d = 0$ and calculate R from the following formula:

$$\tilde{R} = \sqrt{\frac{\eta_{\text{rep}}}{\eta_{\text{attr}}} \cdot \frac{N - 1 + J_2}{N \cdot (2 - J_1)}} \tag{C.21}$$

2. Until the approximation of R is satisfactory, repeat:
   (a) Substitute $d \approx R\frac{d}{\tilde{R}}$.
   (b) Compute new approximation with the following formula:

$$\tilde{R} = \sqrt{\frac{\eta_{\text{rep}}}{\eta_{\text{attr}}} \cdot \sum_{j=1}^{N-1} \frac{2 \sin\left(\frac{2\pi j}{2N}\right)}{\sin\left(\frac{2\pi j}{2N}\right) - \frac{d}{\tilde{R}}} \cdot \frac{1}{2N\,(2 - J_1)}} \tag{C.22}$$

List of Figures

Bibliography

Own Publications

[1] A. Barciś, M. Barciś, and C. Bettstetter, "Robots that sync and swarm: A proof of concept in ROS 2," in *Proc. IEEE Int'l Symp. on Multi-Robot and Multi-Agent Systems (MRS)*, 2019. DOI: 10.1109/MRS.2019.8901095.

[2] A. Barciś and C. Bettstetter, "Beyond sync: Distributed temporal coordination and its implementation in a multi-robot system," in *Proc. IEEE Int'l Conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, 2019. DOI: 10.1109/SASO.2019.00020.

[3] ——, "Sandsbots: Robots that sync and swarm," *IEEE Access*, vol. 8, pp. 218 752–218 764, 2020. DOI: 10.1109/ACCESS.2020.3041393.

Other Publications

[4] G. Skorobogatov, C. Barrado, and E. Salamí, "Multiple UAV systems: A survey," *Unmanned Systems*, vol. 08, pp. 149–169, Nov. 2019. DOI: 10.1142/S2301385020500090.

[5] S. Weiss, C. Bettstetter, H. Hellwagner, and B. Rinner, "Proposal of Karl Popper Kolleg on Networked Autonomous Aerial Vehicles," Aug. 2016.

[6] K. P. O'Keeffe, H. Hong, and S. H. Strogatz, "Oscillators that sync and swarm," *Nature Communications*, vol. 8, Nov. 2017. DOI: 10.1038/s41467-017-01190-3.

[7] H. M. Smith, "Synchronous flashing of fireflies," *Science*, vol. 82, pp. 151–152, Aug. 1935. DOI: 10.1126/science.82.2120.151.

[8] T. J. Walker, "Acoustic synchrony: Two mechanisms in the snowy tree cricket," *Science*, vol. 166, pp. 891–894, Nov. 1969. DOI: 10.1126/science.166.3907.891.

[9] R. L. DeHaan and R. Hirakow, "Synchronization of pulsation rates in isolated cardiac myocytes," *Experimental Cell Research*, vol. 70, pp. 214–220, Jan. 1972. DOI: 10.1016/0014-4827(72)90199-1.

[10] M. C. Moore-Ede, F. M. Sulzman, and C. A. Fuller, *The Clocks That Time Us: Physiology of the Circadian Timing System*. Harvard University Press, 1982, 448 pp.

[11] Z. Néda, E. Ravasz, Y. Brechet, T. Vicsek, and A.-L. Barabási, "The sound of many hands clapping," *Nature*, vol. 403, pp. 849–850, 6772 Feb. 2000. DOI: 10.1038/35002660.

[12] M. K. Mcclintock, "Menstrual synchrony and suppression," *Nature*, vol. 229, pp. 244–245, 5282 Jan. 1971. DOI: 10.1038/229244a0.

[13] E. Goles, O. Schulz, and M. Markus, "Prime number selection of cycles in a predator-prey model," *Complexity*, vol. 6, pp. 33–38, Jun. 2001. DOI: 10.1002/cplx.1040.

[14] V. G. Shakkottai and C. Lomen-Hoerth, "Nervous System Disorders," in *Pathophysiology of Disease: An Introduction to Clinical Medicine*, G. D. Hammer and S. J. McPhee, Eds., 8th ed., New York, NY: McGraw-Hill Education, 2019.

[15] P. Dallard, A. J. Fitzpatrick, A. Flint, S. Le Bourva, A. Low, R. M. Ridsdill Smith, and M. Willford, "The London Millennium Footbridge," *The Structural Engineer*, vol. 79, pp. 17–33, Nov. 2001.

[16] S. H. Strogatz, *Sync: How Order Emerges from Chaos in the Universe, Nature, and Daily Life*. New York, NY: Hachette Books, Apr. 2004, 352 pp.

[17] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*. MIT Press, 1965, 244 pp.

[18] A. T. Winfree, "Biological rhythms and the behavior of populations of coupled oscillators," *Journal of Theoretical Biology*, vol. 16, pp. 15–42, Jul. 1967. DOI: 10.1016/0022-5193(67)90051-3.

[19] Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence*. Mineola, NY, USA: Dover Publications, Aug. 2003.

[20] J. Buck and E. Buck, "Mechanism of rhythmic synchronous flashing of fireflies: Fireflies of Southeast Asia may use anticipatory time-measuring in synchronizing their flashing," *Science*, vol. 159, pp. 1319–1327, Mar. 1968. DOI: 10.1126/science.159.3821.1319.

[21] C. S. Peskin, *Mathematical Aspects of Heart Physiology*. Courant Institute of Mathematical Sciences, 1975, 290 pp.

[22] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, pp. 1645–1662, Dec. 1990. DOI: 10.1137/0150098.

[23] H. Hong and S. H. Strogatz, "Kuramoto model of coupled oscillators with positive and negative coupling parameters: An example of conformist and contrarian oscillators," *Physical Review Letters*, vol. 106, Feb. 2011. DOI: 10.1103/PhysRevLett.106.054102.

[24] ——, "Mean-field behavior in coupled oscillators with attractive and repulsive interactions," *Physical Review E*, vol. 85, May 2012. DOI: 10.1103/PhysRevE.85.056210.

[25] R. Pagliari, Y.-W. P. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 564–575, May 2010. DOI: 10.1109/JSAC.2010.100506.

[26] S. Phillips and R. G. Sanfelice, "Robust asymptotic stability of desynchronization in impulse-coupled oscillators," *IEEE Transactions on Control of Network Systems*, vol. 3, pp. 127–136, Jun. 2016. DOI: 10.1109/TCNS.2015.2428308.

[27] F. Ferrante and Y. Wang, "Robust almost global splay state stabilization of pulse coupled oscillators," *IEEE Transactions on Automatic Control*, vol. 62, pp. 3083–3090, Jun. 2017. DOI: 10.1109/TAC.2017.2677740.

[28] K. Okuda, "Variety and generality of clustering in globally coupled oscillators," *Physica D: Nonlinear Phenomena*, vol. 63, pp. 424–436, Mar. 1993. DOI: 10.1016/0167-2789(93)90121-G.

[29] C. Favaretto, A. Cenedese, and F. Pasqualetti, "Cluster synchronization in networks of Kuramoto oscillators," *IFAC-PapersOnLine*, vol. 50, pp. 2433–2438, Jul. 2017. DOI: 10.1016/j.ifacol.2017.08.405.

[30] C.-U. Choe, T. Dahms, P. Hövel, and E. Schöll, "Controlling synchrony by delay coupling in networks: From in-phase to splay and cluster states," *Physical Review E*, vol. 81, Feb. 2010. DOI: 10.1103/PhysRevE.81.025205.

[31] S. H. Strogatz, "From Kuramoto to Crawford: Exploring the onset of synchronization in populations of coupled oscillators," *Physica D: Nonlinear Phenomena*, vol. 143, pp. 1–20, Sep. 2000. DOI: 10.1016/S0167-2789(00)00094-4.

[32] R. Zillmer, R. Livi, A. Politi, and A. Torcini, "Stability of the splay state in pulse-coupled networks," *Physical Review E*, vol. 76, Oct. 2007. DOI: 10.1103/PhysRevE.76.046102.

[33] B. Chen, J. R. Engelbrecht, and R. Mirollo, "Cluster synchronization in networks of identical oscillators with α-function pulse coupling," *Physical Review E*, vol. 95, Feb. 2017. DOI: 10.1103/PhysRevE.95.022207.

[34] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 3, pp. 281–323, May 2005. DOI: 10.1016/j.adhoc.2005.01.002.

[35] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks," in *Proc. Int'l Conf. on Information Processing in Sensor Networks (IPSN)*, Apr. 2007, pp. 11–20. DOI: 10.1145/1236360.1236363.

[36] F. Perez-Diaz, R. Zillmer, and R. Groß, "Firefly-inspired synchronization in swarms of mobile agents," in *Proc. Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015. DOI: 10.5555/2772879.2772917.

[37] V. Trianni and S. Nolfi, "Self-organizing sync in a robotic swarm: A dynamical system view," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 722–741, Aug. 2009. DOI: 10.1109/TEVC.2009.2015577.

[38] G. Brandner, U. Schilcher, and C. Bettstetter, "Firefly synchronization with phase rate equalization and its experimental analysis in wireless systems," *Computer Networks*, vol. 97, pp. 74–87, Mar. 2016. DOI: 10.1016/j.comnet.2016.01.001.

[39] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proc. Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, 2005, pp. 142–153. DOI: 10.1145/1098918.1098934.

[40] Y.-W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 1085–1099, May 2005. DOI: 10.1109/JSAC.2005.845418.

[41] J. Degesys and R. Nagpal, "Towards desynchronization of multi-hop topologies," in *Proc. IEEE Int'l Conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, 2008. DOI: 10.1109/SASO.2008.70.

[42] W. Xia and M. Cao, "Cluster synchronization algorithms," in *Proc. American Control Conference (ACC)*, 2010. DOI: 10.1109/ACC.2010.5531433.

[43] E. Invernizzi and G. D. Ruxton, "Deconstructing collective building in social insects: Implications for ecological adaptation and evolution," *Insectes Sociaux*, vol. 66, pp. 507–518, Nov. 2019. DOI: 10.1007/s00040-019-00719-7.

[44] I. D. Couzin and N. R. Franks, "Self-organized lane formation and optimized traffic flow in army ants," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, pp. 139–146, Jan. 2003. DOI: 10.1098/rspb.2002.2210.

[45] D. H. Morse, "Ecological aspects of some mixed-species foraging flocks of birds," *Ecological Monographs*, vol. 40, pp. 119–168, 1970. DOI: 10.2307/1942443.

[46] A. E. Magurran, "The adaptive significance of schooling as an anti-predator defence in fish," *Annales Zoologici Fennici*, vol. 27, pp. 51–66, 1990.

[47] C. Cutts and J. Speakman, "Energy savings in formation flight of pink-footed geese," *Journal of Experimental Biology*, vol. 189, pp. 251–261, Apr. 1994.

[48] H. Trenchard, A. Richardson, E. Ratamero, and M. Perc, "Collective behavior and the identification of phases in bicycle pelotons," *Physica A: Statistical Mechanics and its Applications*, vol. 405, pp. 92–103, Jul. 2014. DOI: 10.1016/j.physa.2014.03.002.

[49] S. Bazazi, C. C. Ioannou, S. J. Simpson, G. A. Sword, C. J. Torney, P. D. Lorch, and I. D. Couzin, "The social context of cannibalism in migratory bands of the Mormon cricket," *PLOS ONE*, vol. 5, Dec. 2010. DOI: 10.1371/journal.pone.0015118.

[50] E. Bonabeau, M. Dorigo, and G. Theraulaz, *From Natural to Artificial Swarm Intelligence*. USA: Oxford University Press, Inc., 1999, 320 pp.

[51] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. Ann. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1987. DOI: 10.1145/37401.37406.

[52] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, pp. 1226–1229, Aug. 1995. DOI: 10.1103/PhysRevLett.75.1226.

[53] Y. Q. Chen and Z. Wang, "Formation control: A review and a new consideration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005. DOI: 10.1109/IROS.2005.1545539.

[54] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, pp. 171–194, May 1999. DOI: 10.1016/S0921-8890(99)00004-4.

[55] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 1998. DOI: 10.1109/ROBOT.1998.680621.

[56] K.-H. Tan and M. A. Lewis, "Virtual structures for high-precision cooperative mobile robotic control," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, 1996. DOI: 10.1109/IROS.1996.570643.

[57] F. Berlinger, M. Gauci, and R. Nagpal, "Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm," *Science Robotics*, vol. 6, Jan. 2021. DOI: 10.1126/scirobotics.abd8668.

[58] V. Gazi, "Swarm aggregations using artificial potentials and sliding-mode control," *IEEE Transactions on Robotics*, vol. 21, pp. 1208–1214, Dec. 2005. DOI: 10.1109/TRO.2005.853487.

[59] L. Wang, H. Shi, T. Chu, W. Zhang, and L. Zhang, "Aggregation of foraging swarms," in *Proc. Advances in Artificial Intelligence (AI)*, G. I. Webb and X. Yu, Eds., 2005. DOI: 10.1007/978-3-540-30549-1_66.

[60] V. Gazi and K. Passino, "Stability analysis of swarms," *IEEE Transactions on Automatic Control*, vol. 48, pp. 692–697, Apr. 2003. DOI: 10.1109/TAC.2003.809765.

[61] R. C. Fetecau, Y. Huang, and T. Kolokolnikov, "Swarm dynamics and equilibria for a nonlocal aggregation model," *Nonlinearity*, vol. 24, pp. 2681–2716, Aug. 2011. DOI: 10.1088/0951-7715/24/10/002.

[62] H. Tanner, A. Jadbabaie, and G. Pappas, "Stable flocking of mobile agents, part I: Fixed topology," in *Proc. Conf. on Decision and Control (CDC)*, 2004, pp. 2010–2015. DOI: 10.1109/CDC.2003.1272910.

[63] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2011. DOI: 10.1109/IROS.2011.6095129.

[64] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer International Publishing, 2018.

[65] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, pp. 97–120, Dec. 2008. DOI: 10.1007/s11721-008-0016-2.

[66] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. Pinciroli, and M. Dorigo, "Self-organized flocking with a mobile robot swarm: A novel motion control method," *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, vol. 20, pp. 460–477, Dec. 2012. DOI: 10.1177/1059712312462248.

[67] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, Jul. 2018. DOI: 10.1126/scirobotics.aat3536.

[68] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. H. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots," *Bioinspiration & Biomimetics*, vol. 9, May 2014. DOI: 10.1088/1748-3182/9/2/025012.

[69] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, pp. 795–799, Aug. 2014. DOI: 10.1126/science.1254295.

[70] K. P. O'Keeffe, J. H. M. Evers, and T. Kolokolnikov, "Ring states in swarmalator systems," *Physical Review E*, vol. 98, Aug. 2018. DOI: 10.1103/PhysRevE.98.022203.

[71] H. Hong, "Active phase wave in the system of swarmalators with attractive phase coupling," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, Oct. 2018. DOI: 10.1063/1.5039564.

[72] J. U. F. Lizarraga and M. A. M. de Aguiar, "Synchronization and spatial patterns in forced swarmalators," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, Dec. 2019. DOI: 10.1063/1.5141343.

[73] F. Jiménez-Morales, "Oscillatory behavior in a system of swarmalators with a short-range repulsive interaction," *Physical Review E*, vol. 101, Jun. 2020. DOI: 10.1103/PhysRevE.101.062202.

[74] I. Aihara, T. Mizumoto, T. Otsuka, H. Awano, K. Nagira, H. G. Okuno, and K. Aihara, "Spatio-temporal dynamics in collective frog choruses examined by mathematical modeling and field observations," *Scientific Reports*, vol. 4, Jan. 2014. DOI: 10.1038/srep03891.

[75] O. A. Igoshin, R. Welch, D. Kaiser, and G. Oster, "Waves and aggregation patterns in myxobacteria," *Proceedings of the National Academy of Sciences*, vol. 101, pp. 4256–4261, Mar. 2004. DOI: 10.1073/pnas.0400704101.

[76] A. Hrabec, V. Křižáková, S. Pizzini, J. Sampaio, A. Thiaville, S. Rohart, and J. Vogel, "Velocity Enhancement by Synchronization of Magnetic Domain Walls," *Physical Review Letters*, vol. 120, May 2018. DOI: 10.1103/PhysRevLett.120.227204.

[77] K. P. O'Keeffe and C. Bettstetter, "A review of swarmalators and their potential in bio-inspired computing," in *Proc. Micro- and Nanotechnology Sensors, Systems, and Appications*, 2019. DOI: 10.1117/12.2518682.

[78] M. Hartbauer and H. Römer, "A novel distributed swarm control strategy based on coupled signal oscillators," *Bioinspiration & Biomimetics*, vol. 2, pp. 42–55, 2007. DOI: 10.1088/1748-3182/2/3/002.

[79] A. L. Christensen, R. O'Grady, and M. Dorigo, "From fireflies to fault-tolerant swarms of robots," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 754–766, Aug. 2009. DOI: 10.1109/TEVC.2009.2017516.

[80] N. Bezzo, P. J. Cruz, F. Sorrentino, and R. Fierro, "Decentralized identification and control of networks of coupled mobile platforms through adaptive synchronization of chaos," *Physica D: Nonlinear Phenomena*, vol. 267, pp. 94–103, Jan. 2014. DOI: 10.1016/j.physd.2013.08.012.

[81] R. Sepulchre, D. Paley, and N. Leonard, "Collective motion and oscillator synchronization," in *Cooperative Control. Lecture Notes in Control and Information Science*, V. Kumar, N. Leonard, and A. S. Morse, Eds., vol. 309, Berlin, Heidelberg: Springer, 2005, pp. 189–205. DOI: 10.1007/978-3-540-31595-7_11.

[82] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Transactions on Automatic Control*, vol. 52, pp. 811–824, May 2007. DOI: 10.1109/TAC.2007.898077.

[83] H. Gao and Y. Wang, "A pulse-based integrated communication and control design for decentralized collective motion coordination," *IEEE Transactions on Automatic Control*, vol. 63, pp. 1858–1864, Jun. 2018. DOI: 10.1109/TAC.2017.2765279.

[84] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2012. DOI: 10.1109/ICRA.2012.6224638.

[85] M. Jdeed, S. Zhevzhyk, F. Steinkellner, and W. Elmenreich, "Spiderino - A low-cost robot for swarm research and educational purposes," in *Proc. Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2017. DOI: 10.1109/WISES.2017.7986929.

[86] H. K. Lee, K. Yeo, and H. Hong, "Collective steady-state patterns of swarmalators with finite-cutoff interaction distance," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, Mar. 2021. DOI: 10.1063/5.0038591.

[87] B. D. O. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Systems Magazine*, vol. 28, pp. 48–63, Dec. 2008. DOI: 10.1109/MCS.2008.929280.

[88] F. Dörfler and F. Bullo, "Synchronization in complex networks of phase oscillators: A survey," *Automatica*, vol. 50, pp. 1539–1564, Jun. 2014. DOI: 10.1016/j.automatica.2014.04.012.

[89] L.-x. Yang, X.-l. Lin, and J. Jiang, "Influences of adding negative couplings between cliques of Kuramoto-like oscillators," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, Jun. 2018. DOI: 10.1063/1.5017772.

[90] F. Ferrante and Y. Wang, "A hybrid systems approach to splay state stabilization of pulse coupled oscillators," in *Proc. IEEE Conference on Decision and Control (CDC)*, 2016. DOI: 10.1109/CDC.2016.7798520.

[91] T. Prager, B. Naundorf, and L. Schimansky-Geier, "Coupled three-state oscillators," *Physica A: Statistical Mechanics and its Applications*, vol. 325, pp. 176–185, Jul. 2003. DOI: 10.1016/S0378-4371(03)00196-1.

[92] D. J. Jörg, "Stochastic Kuramoto oscillators with discrete phase states," *Physical Review E*, vol. 96, Sep. 2017. DOI: 10.1103/PhysRevE.96.032201.

[93] D. A. Paley, N. E. Leonard, and R. Sepulchre, "Oscillator models and collective motion: Splay state stabilization of self-propelled particles," in *Proc. IEEE Conference on Decision and Control (CDC)*, 2005, pp. 3935–3940. DOI: 10.1109/CDC.2005.1582776.

[94] D. A. Paley, N. E. Leonard, R. Sepulchre, D. Grunbaum, and J. K. Parrish, "Oscillator models and collective motion," *IEEE Control Systems Magazine*, vol. 27, pp. 89–105, Aug. 2007. DOI: 10.1109/MCS.2007.384123.

[95] D. J. Klein, P. Lee, K. A. Morgansen, and T. Javidi, "Integration of communication and control using discrete time Kuramoto models for multivehicle coordination over broadcast networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 695–705, May 2008. DOI: 10.1109/JSAC.2008.080511.

[96] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion with limited communication," *IEEE Transactions on Automatic Control*, vol. 53, pp. 706–719, Apr. 2008. DOI: 10.1109/TAC.2008.919857.

[97] D. J. Klein and K. A. Morgansen, "Set stability of phase-coupled agents in discrete time," in *Proc. American Control Conference (ACC)*, 2008. DOI: 10.1109/ACC.2008.4586832.

[98] B. I. Triplett, D. J. Klein, and K. A. Morgansen, "Discrete time Kuramoto models with delay," in *Proc. Networked Embedded Sensing and Control (NESC): Workshop*, 2006. DOI: 10.1007/11533382_2.

[99] T. Schmickl and H. Hamann, "BEECLUST: A swarm algorithm derived from honeybees: Derivation of the algorithm, analysis by mathematical models, and implementation on a robot swarm," in *Bio-Inspired Computing and Networking*, Y. Xiao, Ed., Boca Raton, FL: CRC Press, Apr. 2016, pp. 95–137.

[100] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. Int'l Conf. on Robotics and Automation (ICRA)*, 1991, pp. 1398–1404. DOI: 10.1109/ROBOT.1991.131810.

[101] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2000. DOI: 10.1109/ROBOT.2000.844042.

[102] L. Barnes, M. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Proc. Mediterranean Conf. on Control and Automation (MED)*, 2007. DOI: 10.1109/MED.2007.4433724.

[103] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil, "Distributed, physics-based control of swarms of vehicles," *Autonomous Robots*, vol. 17, pp. 137–162, Sep. 2004. DOI: 10.1023/B:AURO.0000033970.96785.f2.

[104] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system," *Adaptive Behavior*, vol. 17, pp. 237–259, Jun. 2009. DOI: 10.1177/1059712309104966.

[105] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, pp. 1–3, Jan. 1966. DOI: 10.2140/pjm.1966.16.1.