Christian Wustrau

Building a Scalable Swarm Application Using Sphero Robots

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Building a Scalable Swarm Application Using Sphero Robots

Bachelor Thesis

Christian Wustrau

May 30, 2020

Supervisor: Prof. Dr. Sanaz Mostaghim

Advisor: Dr. Christoph Steup

Christian Wustrau: *Building a Scalable Swarm Application Using Sphero Robots*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2020.

Abstract

Swarm robotics as an approach to the coordination of multi-robot systems is guided in its design by swarm intelligence principles. This thesis aims to implement a swarm scenario using spherical robots, called Spheros which were developed by Orbotix. Using the Robot Operating System and an external robot localization framework, we design a scalable swarm of spherical robots to display aggregation behavior. We investigate the application of swarm algorithms to the robot platform and evaluate if global complex behavior can be achieved by using simple rules on relatively simple robots.

The experiments conducted in this thesis show that theoretical concepts of swarm intelligence can be applied to a swarm of spherical robots to achieve complex behavior. The number of individuals in our robot swarm can be scaled up but results convey that typical swarm robotic challenges aggravate in the process of doing so.

Acknowledgements

I would first like to thank my thesis supervisor Prof. Dr.-Ing. habil. Sanaz Mostaghim, for the continuous encouragement and inspiration she provided which allowed this paper to be my own work and steered me in the right the direction whenever I struggled.

I would also like to acknowledge Dr.-Ing. Christoph Steup as my advisor for this thesis. I am thankful and indebted to him for sharing his valuable expertise and providing guidance through the process of writing this thesis.

My sincere thanks goes to Palina Bartashevich for always offering her support and providing valuable comments on my thesis. I would also like to extend my thanks to Michael Preuß for his help in offering me the necessary resources to create my application.

Finally, I must express my gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study.

This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The Rolling Swarm project aims at creating a swarm of autonomously moving robots to investigate swarm intelligence behavior on robots in a fully autonomous mode. In this scope, our goal is to create a presentable swarm robotic application as a practical and straightforward tech demo that scales with the number of robots. To do this, we realize and evaluate a theoretical swarm model on a corresponding physical robot platform as a proof of concept.

The Rolling Swarm is one of the five robot platforms in the SwarmLab of the Chair of Computational Intelligence at the Otto von Guericke University Magdeburg and contains over 65 Sphero robots. The Sphero is a spherical differential drive robot that can be controlled via Bluetooth. The robots are well fitted for swarm robotic applications due to their straightforward control mechanisms and convenient functionalities and were, therefore, selected as our robot platform. They can be used to perform various collective behaviors such as exploration in unknown environments or geometric pattern formation.

The considered application is implemented as a swarm robotic scenario and not as a classic multi-agent scenario. Swarm robotics is a very specific approach in the field of multi-robot systems and will be covered in detail in Section 2.2. In the following, we shortly introduce the main differences between these two types.

By definition, multi-agent systems are multiple autonomous agents that communicate with each other when they need to achieve a task jointly. The autonomy and communication aspects are crucial here since every agent has only information about its immediate environment and may need to achieve its own goals and tasks. To perform a global task, the agents have to directly communicate and coordinate with other agents.

Multi-agent systems consist of entities (e.g., robots or humans) that are each specialized for a certain task. In these systems, each robot is capable of something possibly unique, and together they cooperate to solve more complex tasks that could not be solved or just with great additional effort for a single individual agent or a monolithic system [1].

In turn, swarm robotics is an approach to the coordination of multi-robot systems which consist of a large amount of relatively simple robots. Here, the desired collective behavior emerges from the local interaction between the robots and the interaction of robots with the environment.

Swarm robotics shows great flexibility and adaptability due to its homogeneity and scalability. In this regard, the considered Rolling Swarm as a robotic platform is homogeneous because every robot has the same capabilities as all other Spheros. Due to the limited battery life and a large number of robots, they have to be interchangeable to keep the system's performance. Such specific properties of our robot platform demand the scalability factor in our application. At the same time, multi-agent systems may achieve better performance on specialized tasks, but at the cost of their flexibility, reusability and scalability [47]. Based on the criteria mentioned above, swarm robotics is more suitable for our application and is chosen as the approach for the implementation of a scalable system using Spheros.

In the current state of the art, there also exist other swarm robotic platforms like the Kilobot, which is designed to enable programming and experimenting with collective behaviors in large-scale autonomous swarms. In the following, we highlight its main differences with the Sphero.

The Kilobot is a robot that is designed for swarm robotics research by the Self-Organizing Systems Research Group at Harvard University [19]. At the first sight, these robots seem to be better suited for swarm robotics applications as they are smaller and cheaper compared to the Sphero but there are essential drawbacks with them. Kilobots move using a slip-stick based locomotion powered by two sealed coin-shaped vibration motors. Due to this type of locomotion, Kilobots are unable to move precisely over long distances or an extended period of time. Additionally, they are not equipped with any real form of odometry [39]. The vibrating leg technology requires flat surfaces to move and prevents the robot from moving as fast as the Sphero can. Since Kilobots do not offer self-localization capability, due to a missing encoder/sensor

to trace the movement or dead reckoning [33], they do not seem suitable for our application.

Spheros are more practicable compared to Kilobots as they can move smoother, have a wide range of sensors including odometry, can be tracked by our external localization framework and can drive on different floors. As a result, we decided to use them as our robot platform for the swarm application as we find them generally better suited for the task of creating a swarm robotic application for aggregation. Chapter 2.2 provides more insight on the Sphero robot, the Kilobot and other swarm robots.

1.2 Goals

The goal of this thesis is to practically apply swarm intelligence algorithms to swarm robotics. That is to say, we want to apply a theoretical model of a swarm to a real robotic platform and create a dynamic tech demo. Therefore, this thesis aims to answer the question of whether it is possible to transfer the theoretical concepts of swarm aggregation to practice in form of a robot swarm.

Additionally, the tech demo should demonstrate the dynamics within the swarm and serve as a prototype to highlight the basic swarm robotics principles. The demo should catch the interest of the audience to convince them of the viability of swarm robotics. Besides, the robots should show more movement than in a typical aggregation scenario. The swarm scenario must, therefore, be expanded by either using an approach that takes into account the environmental influence or by implementing an additional leader-follower principle.

Finally, we determine to which extent we can scale the number of individuals in our swarm and whether they are able to exhibit complex behavior with simple rules. Moreover, we measure and evaluate the performance of the swarm in the real world in different scenarios. The performance of the swarm is based on the extent to which the swarm is aligned and has formed a circle.

1.3 Outline

The next chapter covers three main background topics: swarm intelligence, swarm robotics and software used in the context of this thesis. The swarm intelligence section provides basic theoretical background about swarm stability and movement. The swarm robotics section gives insight into different accomplishments in the field of swarm robotics, describes the Sphero robot in detail and compares it to a set of different swarm robots. The last section covers the software libraries used in our implementation including the robot operation system, navigation software and localization framework. Chapter 3 covers the general concept, structure and detailed node architecture of our proposed application. In Chapter 4, we report the metrics and scenarios for our experiments. Additionally, we present and evaluate the experiment results and discuss them in context with the previous chapter. The paper concludes with a discussion about future work and a summary in Chapter 5.

2 Background

As the proposed application is a software implementation for the swarm behavior of robots, the following chapter first covers the aspects of swarm intelligence, swarm movement and swarm stability. The next section covers the robots that were already introduced in Chapter 1 in detail and presents additional swarm robotic platforms and their applications. The last section wraps up this chapter with the robotic framework and software that was used to implement this application.

2.1 Swarm Intelligence

Swarm intelligence is the collective behavior of natural and artificial systems composed of many individuals having simple rules with the ability to interact locally. The source of inspiration is the shown complex behavior of natural organisms when they are in groups. Mainly ants, termites, bees and many other insect species with a complex social structure are often referenced as inspiration. The term swarm intelligence was introduced and used for the first time by Beni et al. [11] in the context of cellular robotic systems where they define robot intelligence and robot system intelligence in terms of the unpredictability of improbable behavior.

The individuals that form the swarm coordinate using decentralized control and self-organization. Essential and complex behaviors emerge from relatively simple interactions between the individual members. Emergent behaviors are also sometimes considered to be systems that are more complex than the sum of their parts. That is to say, the behavior of a system is not explicitly described by the behavior of the components of the system, and is, therefore, unexpected to a designer or observer.

The complex collective group behavior observed in a swarm emerges from interactions between individuals that exhibit simple behavior [3]. In general, an

emergent behavior can be described as a non-obvious side effect that shows up when combining different capabilities of individuals. Therefore, it is not necessary to invoke individual complexity to explain and achieve complex collective behavior as individuals can exhibit simple behaviors [12]. It is necessary to point out that emergent behaviors are very difficult to foresee until they manifest themselves and can either be advantageous or beneficial but also harmful or destructive. Nevertheless, the main focus of swarm intelligence are the positive or desired behaviors.

To simulate and understand the behavior of a swarm, the swarm and its dynamics have to be modeled. The basic swarming model consists of three simple steering behaviors and was first described by Reynolds [37]. It is proposed that individuals maneuver based on the positions and velocities of its nearby individuals according to the three rules of

Attraction (Cohesion) Individuals move towards other individuals or more precisely towards the average position of local flock-mates.

Repulsion (Separation) Individuals of the swarm steer to avoid crowding local flock-mates to avoid collisions with neighbors.

Alignment All swarm members steer towards the average heading of local flock-mates.

2.1.1 Swarm Stability

In control theory, a dynamic system is considered to be stable when its output is under control. In terms of swarm intelligence, the swarm can be seen as a dynamical system where individuals move on trajectories. For aggregating swarms, this means that the swarm is able to preserve its density or even desired formation as individuals should stabilize their relative distance towards each other. Stability is a desirable property of a swarm to enable task allocation and reliable coordinated motion.

Gazi and Passino [14] describe a theoretical continuous-time model for swarm aggregation in n-dimensional space which is used as the fundamental for the 2-dimensional model in our application. They have shown that a cohesive swarm is formed by individuals in a finite time. Additionally, only depending on the parameters of the swarm, they obtain an explicit bound on the swarm

size with their model. This paper [14] provides most of the theoretical basics for our application and is, therefore, explained further in detail here.

Considering a swarm of M individuals in n-dimensional Euclidean space they, model the individuals in the swarm as points and ignore their dimensions. Therefore, the position of a member i of the swarm is described as an n-dimensional vector $x^i \in \mathbb{R}^n$. Assuming no time delays and synchronous motion, the motion dynamics evolve in continuous time and the according equation of motion for an individual $i = 1, ..., M$ is given by them as:

$$\dot{x}^i = \sum_{j=1, j\neq i}^{M} g(x^i - x^j) \tag{2.1}$$

where $g(\cdot)$ is a function of attraction and repulsion for the distance between two individuals. It represents an artificial social potential function to describe individual interactions.

The issues of potential functions for swarm scenarios are the topic of further research. In fact, Gazi and Passino consider those issues in [13] and specify a general class of attraction/repulsion functions. There they define the general potential function for $y = x^i - x^j$ as:

$$g(y) = -y[g_a(\|y\|) - g_r(\|y\|)] \tag{2.2}$$

where g_a represents an attraction term and, likewise, g_r a repulsion term. The equation of motion denotes how to calculate the derivative of the position of each individual i which is the velocity in classical mechanics. This means the velocity, which is equivalent to the specification of direction and magnitude of motion of each member is calculated as the sum of the attraction and repulsion between this member and all other members of the swarm.

The considered attraction and repulsion function to consider is:

$$g(y) = -y(a - b \cdot e^{-\frac{\|y\|^2}{c}}) \tag{2.3}$$

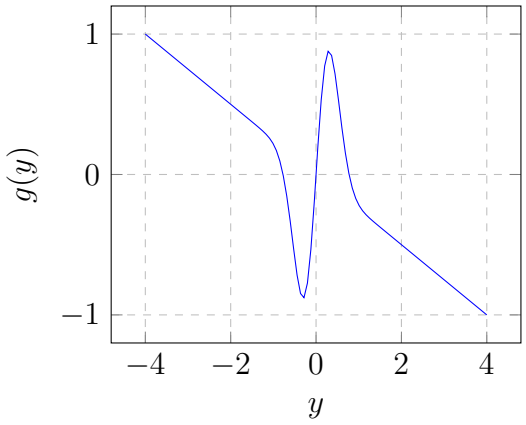


Figure 2.1: Attraction/repulsion function $g(\cdot)$ with $a = 0.25, b = 5$ and $c = 0.2$

The given conditions here are a, b, and c are positive constants, b $>$ a and $\|\cdot\|$ represents the Euclidean norm.

The function is evidently repulsive for small distances because the influence of the repulsion term $g_r = b \cdot e^{-\frac{\|y\|^2}{c}}$ grows exponentially with smaller distances due to the natural exponential function. For large distances, this function is attractive as repulsion diminishes and the attraction term $g_a = a$ grows linearly with the distance between individuals. This is generally consistent with aggregation behavior that can be observed in biological swarms [18] and the rules of attraction and repulsion defined by the swarm model in [37]. But the function is inconsistent with biological inspirations at large distances between individuals because attraction is not bounded to a maximum distance where animals are only attracted when they can sense each other which is bound to a finite range. Additionally, infinitesimally small ranges are inconsistent as g is not unbounded and approaches zero there.

By equating $y(a - be^{-\frac{\|y\|^2}{c}}) = 0$ the set points of this function can be calculated and are $y = 0$ and $\|y\| = \delta = \sqrt{c \ln(\frac{b}{a})}$. This distance δ is called comfortable distance and when individuals are at these distances they stop moving towards or away from each other, because the attraction and repulsion force between them reach a balance.

Individuals in a swarm will form a hyperball when they attract and repel themselves until they reach a comfortable distance towards all other individuals, where each member has the same distance to the center of the swarm.

The center of the swarm $\bar{x}$ can be computed as the average of the sum over the position x of each individual i:

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x^i \tag{2.4}$$

Gazi and Passino [14] prove that the center of the swarm is stationary because of the symmetry of $g(\cdot)$ with respect to the origin. In other words, "member i moves toward every other member j exactly the same amount as j moves toward i" [15]. In fact, this holds true when individuals are only attracted and repelled by other individuals and not influenced by other sources. As swarms are generally located inside a certain environment, they are additionally influenced by their surroundings. This leads to the effect that the center of the swarm is not stationary anymore and will move in the environment. The following chapter will cover more about environmental influence and the resulting swarm movement.

2.1.2 Swarm Movement

An interesting aspect of swarm behavior is the coordinated movement because in a real-world scenario the individuals should not only aggregate but perform a coordinated task. Therefore, we look further at swarm movement as an important behavior for coordinated control in this section. After the static aggregation is done, resulting in convergence, where each individual does not change its position anymore due to the balance of potential forces, there are two approaches to model the movement of swarms: an environmental profile and a leader-follower-principle.

A swarm model that takes attractants and repellents from the environment into consideration is introduced in [16]. An environmental profile σ called the "σ-profile" is addressed which affects all members of the swarm. The environmental profile is inspired by nature from chemical attractants and repellents where animals react to varying factors in the environment. This means individuals are influenced by other individuals through a local sensed potential and their surrounding environment, which can be understood as a constant acting force on them. The profile represents an artificial potential function that models the environment containing targets to be moved towards and obstacles or threats to be avoided in the context of a real-world scenario.

The according equation of motion of each individual results as an extension of Equation (2.1) by an additional attractant/repellent term as:

$$\dot{x}^i = -\nabla_{x^i}\sigma(x^i) + \sum_{j=1, j\neq i}^{M} g(x^i - x^j), \quad i = 1, ..., M \tag{2.5}$$

The $-\nabla_{x^i}\sigma(x^i)$ term represents the gradient of the profile and respectively the motion of the individuals according to the environmental profile of their environment. Individuals will move away from repellents and towards regions with higher attractant concentration additional to their distance-based attraction and repulsion towards other individuals of the swarm.

Until now we only considered the swarm to consist of homogeneous individuals but the question arises how the swarm would behave when some individuals were different from the rest. In case that the different individuals are not aggregating with the rest of the swarm anymore then this scenario can be seen as a type of leader-follower principle. The leader-follower principle specifies one or several leaders and several followers, where the classic approach is to have only one leader that has critical information for a task to perform. This could be a formation to build, where the leader specifies the reference trajectory [25] or the path to a target area [41]. The leader can be seen as an individual that only has an environmental profile without attraction or repulsion to other members of the swarm. The leader will move on a given path defined by the environment, ignore other individuals and the rest of the swarm will try to aggregate around him and lag behind his movement. As the leader usually performs a computationally expensive task, in our case the navigation in an environment, the followers implement a simple low-cost following approach and the swarm overall emerges a swarm movement behavior without implementing and performing the costly task for all individuals.

2.2 Swarm Robotics

Swarm robotics has emerged as the application of swarm intelligence to multi-robot systems [7]. It is a novel approach for the coordination of large numbers of robots to tackle a task that a single individual would be unable to accomplish [6]. A swarm of robots consists of a large number of redundant robots,

which act autonomously and show collective behavior due to interactions between robots and interactions of robots with the environment.

Compared to multi-agent systems, swarm intelligence systems are characterized by increased robustness, flexibility and scalability. Each of these characteristics is briefly discussed below:

Robustness With a homogeneous swarm that consists of a huge amount of redundant robots any loss or malfunction of an individual can be compensated by another one. Removing robots does not impact the overall performance of the swarm significantly [47]. The system should continuously operate and be prevented from performing the specified task even when a part of the system is destroyed, which is achieved by the decentralized coordination of simple individuals. Therefore, in a swarm robotic system individuals are simpler compared to a single complex system that could perform the same task, making the overall system less prone to failures [6].

Flexibility The overall task performed by the swarm is achieved by coordination of robots that are homogeneous and not essentially personalized to a given task [23]. Swarm robotic systems can offer solutions to different tasks by utilizing different coordination strategies, have the ability to generate modularized solutions to different tasks and can respond to the changes in the environment as there is no central coordination and no specialization of robots for certain tasks [47].

Scalability Interaction in a swarm is local which means actions from individuals do not affect the whole swarm. The swarm can deal with changes in the size of the population through reallocation of tasks [47] and is, therefore, able to operate under a wide range of group sizes. This is a useful and necessary property as with many robots in a swarm the chance that a robot will crash is significantly high and the group size will shrink over a certain period of time.

While swarm robotics offer great advantages, many challenges arise when working with a large number of robots at the same time. In swarm intelligence theory, individuals are usually considered in a simplified view and assumed to show flawless behavior. In practice, however, individuals and their behaviors are more complex, which leads to the following challenges for swarm robotics.

Locality Robots are limited in their area of perception since sensors only have specific ranges of detection. Furthermore, communication is also restricted and only possible over certain ranges. In other terms, robots can only operate in their local vicinity and everything outside their sensor and communication range is not perceived by them.

Communication The transmission of information takes time and energy. While in theory, individuals exchange information immediately, in practice, communication has a limited range and messages are received with a delay. Wireless technology is typically used to bridge long distances. However, wireless communication is subject to interference and can be obstructed by the environment.

Localization Swarm robots make decisions based on their knowledge about their own position. Therefore, the localization problem arises from the need for robots to determine their own position. Information about their own position and heading can be gathered through external or relative localization. Relative localization uses the robot's sensors to perform dead reckoning. Dead reckoning is prone to accumulating errors over time and requires precise and frequent sensor measurements. External localization requires an external system to estimate the pose of the robot. This method is usually time and location independent but is unable to keep track of the robot for short distances [17].

Aggregation For most behaviors in swarm robotics robots must be in some proximity of each other [45]. Aggregation is the precondition for many types of collective behavior and needs to be addressed according to the particular characteristics of the robotic system and of the environment in which it must take place [49].

Movement Control Robots need to move precisely to perform tasks effectively. But inaccuracies in robot movements are unavoidable and occur due to joint errors, kinematic errors, non-kinematic errors and sensor imperfections. Specific control systems are, therefore, required to compensate for the inaccuracy of the robot.

2.2.1 Robot Swarm Applications

There are many potential applications for swarm robotics. We present three approaches that are similar to our application as they tackle swarming and swarm formation.

Nouyan et al. [32] present a chain and a vector field based control mechanisms for distributed robot swarm path formation. They study how to control a swarm of robots to form a path between two locations, the nest and the pray, in a bounded arena, use a behavior-based architecture for their controller and the s-bot [30] as their robot platform.

In the chain-based approach a robot performs random walk until it finds a nest and from there on follows an existing chain, joins at the end of it or starts a new one. Robots at the end of the chain can leave it again to explore the environment and to form new chains in unexplored parts. When a chain member perceives the prey, it stays in the chain to stabilize it.

In the vector field-based approach, the robots have to search for the nest again. The first robot to find the nest activates a LED color pattern pointing towards the nest. Other robots that perceive this pattern will move away from the nest indicated by the direction of the pattern. Robots that sense only one LED-activated robot will join the structure and point towards the other robot. This resulting structure forms a vector field that globally leads to the nest. At each time step, a robot at the border of the vector field can leave it with a certain probability. Joining and leaving the vector field leads to an exploration of the environment similar to the chain approach. Moreover, robots perceiving the prey stay in the vector field to establish a stable path.

The work of Nouyan et al. [32] shows a similar swarm approach as each robot acts based on other robots in their environment and adjusts accordingly to build a formation. It is different from our application as they use another robot platform and different formation, where the nest is connected to prey while we aggregate only based on the position and alignment of other individuals in the swarm.

A well-known robot swarm that sparked public attention is the Intel Shooting Star drone [21] as they starred in the Superbowl halftime show in 2017 and 2019 and the PyeongChang Winter Olympics 2018. The Intel Shooting Star is a quadcopter with encased propellers, a soft frame made of flexible plastics and foam, built-in LED lights and weighs less than 330 grams.

They are mainly used for light shows for entertainment, where they fly in predefined formation. Intel offers specific software and animation interfaces to create such a light show in a matter of weeks. Their algorithms automate the animation process of an image by determining drone locations, the paths for each drone to fly and calculating the number of drones needed as well. Therefore, the drones are preprogrammed and only communicate wirelessly with a central computer and not with each other. The computer checks the GPS signal strength and battery level of each drone, assigns roles accordingly and tells each drone how to execute its flight routine. They utilize the Trinity Autopilot which is their commercial autopilot to navigate the drones in the air and to ensure they do not crash. Additionally, they implemented a "two layer geofence in the software [to] ensure that the drones are always in the bounding box implemented in [the] system." [5]

Compared to the Spheros the Shooting Star drones are able to build formations in 3-dimensional space. It is a more difficult task to solve than a hyperball aggregation and can be additionally seen as an extension to the next dimension as we are only considering 2-dimensional space in our application. Even though Intel's light shows consist of large numbers of redundant robots, the central computer calculates each path beforehand. In this way, the drones fly by themselves with autopilot and do not communicate or interact with each other. Therefore, they are classified more as a swarm of autonomous flying robots than a classical robot swarm.

Another application for a swarm of tiny drones is presented in [29], where a swarm of Crazyflie 2.0 [2] can explore unknown environments completely by themselves. This is especially useful in search-and-rescue scenarios, where it is too dangerous for humans to explore the environment. The main challenges for the drones here are flight navigation such as controlling the velocity and avoiding obstacles, detection of obstacles and collision avoidance with each other and obstacles. Each drone carries a wireless communication chip and then uses the signal strength between these chips to sense other drones in the environment.

McGuire et al. [29] proposed a navigation method that is a novel type of bug algorithm [26] where multiple robots fly of in different direction and deal with obstacles and each other on the fly instead of mapping the environment. After exploration for a certain amount of time when the batteries are running low, the robots fly back to a wireless beacon at the base station where they started.

This work shows the advantages of swarm robotics as a suitable approach for real-world tasks. In theory, a detailed map would be very advantageous as it would allow a robot to precisely navigate in the environment along an optimal path but the costs of making such a map are prohibitive. In practice, the proposed bug algorithm [29] is able to explore the environment even when this leads to less efficient paths. Furthermore, it can be implemented on tiny robots with very limited sensing and computation capabilities which reduces the cost significantly compared to a detailed map.

2.2.2 Swarm Robots

The introduction chapter already mentioned the Spheros as our robot platform and compared it to the Kilobot to justify them as suitable for this swarm application. In this chapter, we present achievements in the field of swarm robotics with Kilobots that tackle similar tasks to ours and their proposed swarm algorithms. Additionally, the Sphero robot, its capabilities and characteristics are described in detail in their respective section to provide more insight into our robot platform.

2.2.3 Kilobots

The Kilobot is one of the most well-known robot platform in the field of swarm robotics and the name of the robots refers to the platform being the first robot swarm to reach scale 1024 in size. The robot platform is widely used in research as it allows swarm algorithms to be tested on a large scale of real robots.

Rubenstein et al. [40] report a system that demonstrates programmable flexible, large-scale self-assembly of complex two-dimensional shapes with a Kilobot swarm. This means that the robots can cooperatively assemble into any 1-connected shape, specified by the user under some restrictions. Each robot has an image of the specific target shape and size and runs the fixed self-assembly algorithm. Robots only have access to the distance information to nearby neighbors but they can use this to collectively construct a coherent coordinate system. Using a distributed implementation of trilateration, robots locate themselves in the coordinate system and can become reference points for other robots.

Figure 2.2: Image of the Kilobot as described in [39]

Four user placed seed robots mark the position and orientation of the target shape in the environment and are the source of a gradient formation that propagates through the initial group. As robots can determine whether they are on an outer edge through comparison of gradient values between them and their neighbors, robots on the outer edge will follow these until they reach their seed. Through localization, edge-following and the given desired shape, a Kilobot determine whether it is already inside the specific target shape and if not it can follow the partially formed assemblies edge until it enters the shape. Continuing edge-following until inside the shape, robots stop when they are next to a stationary robot with the same gradient value or when they are about to exit the shape. The shape will be continuously filled as robots move and join the assembly.

Oh et al. [33] present self-organized collective formation tracking using Kilobots. The overall objectives are following and tracking a target, as well as maintaining the swarm formation. Due to Kilobot limitations in locomotion and communication, missing self-localization capabilities and directional sensing, new control algorithms are introduced. These algorithms obtain the movement direction indirectly through an objective function which is based on morphogen diffusion and network connectivity preservation to achieve collective object tracking and herding.

A distance-based attraction/repulsion controller gets the Kilobots to follow their neighbors based on their morphogen concentration and the corresponding

concentration values of the neighbors to stay in formation. This is combined with light source tracking as the robots have to move towards the target. They achieve this behavior only with local communication of the distance and morphogen gradient between robots.

2.2.4 Spheros

The Sphero SPRK+ robot was developed by the Sphero, Inc. (formerly Orbotix) and was released in 2016. The company produces consumer robotics and toys for educational use and started with the original Sphero 1.0 in 2011. However, previous work with the Spheros that was published focuses mainly on educational aspects [22], human-robot interaction [9] or augmented-reality applications [35]. Noteworthy is the work of Nistad [31] for presenting a camera-based navigation and control platform for the Sphero.

Figure 2.3: Sphero SPRK+ [46]

The SPRK+ has a clear, scratch-resistant, UV-coated, polycarbonate shell. The shell protects the internal mechanics from hard collisions and drops. Additionally, the robot is waterproof because the shell is sealed too. Inside the Sphero are three LEDs, two located at the center to light up the robot in a customizable color and the third one is located at the downside to indicate the current rotation of the ball. Additional weight is situated at the bottom to hold the core in a horizontal position to stabilize the Sphero. There are

two wheels inside of the Sphero shell that are differential driven by the motors which enable a max speed of 2m/s. To prevent erroneous behavior, there are two additional wheels inside which strengthen the bond of the motor-controlled wheels to the inner surface and to prevent the core from falling over. The robot can move only forward or rotate to a set angle relative to the start orientation and communicate wireless over Bluetooth Low Energy (BLE) at a maximum range of 30m. [46]

The Sphero has a Lithium Polymer battery that lasts for 1 hour. The robot can be charged on an inductive charging base with a USB charging cable. The robot has a height and width of 73mm and weighs 181g. Due to its size, the Sphero cannot overcome obstacles easily unless these are very small. But it can drive indoors or outdoors, on hard floors or low carpets. Moreover, it can drive through dirt, water, and paint.

Both a speed and a heading are required via command for the Sphero to roll along the provided heading with the given speed. The speed is capped at the maximum physically possible speed for the robot of 2 m/s. The heading follows the 360 degrees convention on a circle where 0 degrees is straight ahead, 90 degrees is to the right, 180 degrees is back and 270 degrees is to the left.

To be able to estimate the robot's condition and internal state, the Sphero is equipped with internal sensors. There are motor encoders to enable the generation of odometry data over time and an inertial measurement unit (IMU) that combines an accelerometer and a gyroscope. The Sphero has several built-in features, one of them is collision detection where each robot contains a collision detection algorithm. Using the high fidelity IMU data, the robot will send a message each time it detects a collision which can be used to ease navigation problems such as getting stuck or hitting a wall. Another feature is the so-called locator that provides real-time 2D position and velocity information about the robot via an implemented streaming protocol.

As already mentioned, the Sphero is a spherical mobile robot that is differential-driven. A differential drive robot consists of two wheels placed on a common axis powered by separate motors [8], where each wheel can independently be driven forward or backward. To perform a rolling motion, we can change the velocity of each wheel but the robot will rotate about the Instantaneous Center of Curvature (ICC). The ICC lies along the common axis of the left and right wheel. Varying relative velocities of the two wheels, changes the position of the ICC which leads to different vehicle trajectories. Because

the rate of rotation ω about the ICC must be the same for both wheels, the equation for the cross-radial speed is given by:

$$\omega(R + l/2) = v_r, \quad \omega(R - l/2) = v_l \tag{2.6}$$

where ω is the rotation rate, R the signed distance from the ICC to the midpoint between the wheels, l the distance between the centers of the wheels and v_l and v_r are the respective left and right velocities of the wheels along the ground .

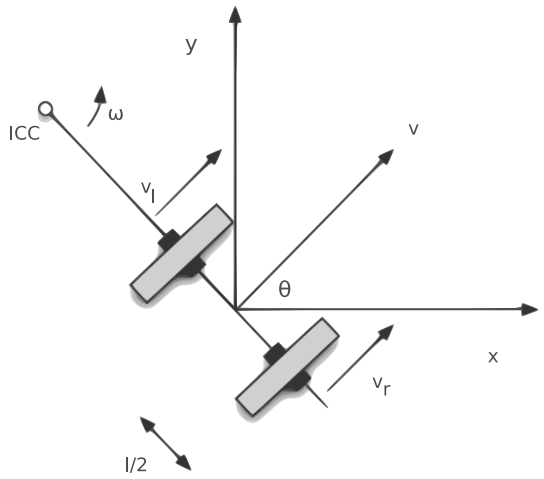


Figure 2.4: Differential Drive kinematics [8]

Solving for R and ω results in:

$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}, \quad \omega = \frac{v_r - v_l}{l} \tag{2.7}$$

If $v_l = v_r$, then R becomes infinite, there is effectively no rotation as $-\omega$ is zero and the robot moves forward, resulting in a linear motion in a straight line. When $v_l = -v_r$, then $R = 0$ and the robot rotates about the midpoint of the wheel axis which is in place rotation. For other values of v_l and v_r, we do not have movement in a straight line but rather a curved trajectory rotating about the ICC at distance R. One advantage of differential driven robots is the ability to rotate in place as this allows it to navigate in a narrow environment.

But they are also very sensitive to the relative velocity of the wheels. Since different velocities will not just slow down or speed up the robot but result in a different trajectory.

Our robot is not an ideal differential drive robot. The additional wheels that strengthen the bond of the motor-controlled wheels to the inner surface will not prevent errors while the robot drives. Moreover, those additional wheels add weight to balance and stabilize the robot while driving but do not compensate for the totter motion that occurs due to the robot's spherical shape. As our robot rolls with its shell on the surface and the wheels inside drive the shell via contact, the robot will slip heavily on the surface it moves on. This results in the effect that the robots own odometry, which estimates the traveled distance by measuring how much the wheels have turned, and the locator will not be accurate for long time periods. The unavoidable error in the position estimate will add up over time and become to complex to eliminate as there is no reference to the external world. These problems have to be considered when working with the Sphero as this can be categorized as the localization problem mentioned in Chapter 2.2.

2.2.5 Conclusion and Comparison

The robots presented in the previous section show all similarities with the Sphero in terms of their movement model, their swarm mechanism or their application, see Table 2.1 for comparison. Besides, all robots have certain advantages over the Sphero in various aspects. The purpose of this chapter is to summarize the key conclusions about these robotic platforms in a manner that explains the decision to choose the Sphero as our robotic platform.

The Kilobot was already described in detail in its own section. To summarize, the robot is designed to enable programming and experimenting with collective behaviors. The main disadvantage of the Kilobot, however, lies in its physical capabilities. Due to its small size and slip-stick based locomotion, the Kilobot is limited to flat surfaces and can only move at a percentage of the possible maximum speed of the Sphero.

The s-bot is physically very similar to the Sphero as both are differential-driven and are approximately the same size. The robot is even equipped with more sensors and features than the Sphero. But due to the fact that the s-bot is

custom-built and not commercially available, we would have to build all robots for the swarm ourselves.

As already mentioned before, the Intel Shooting Star drones are more a swarm of autonomous flying robots than a classical robot swarm. It remains unknown whether they are suitable for swarm robotics tasks at all, since they are controlled by an autopilot, do not communicate with each other and are also not available for purchase.

The Crazyflie, on the other hand, is commercially available and suitable for swarm robotics tasks. But the drone has no external monitoring options, no geofence support and the propellers are unsecured. For this reason, the drone is unsuitable for a tech demo since the audience would be at risk if the drones performed unsecured.

This shows that other robots also have considerable disadvantages that must be taken into account. The Sphero is a robust, durable robot that can be purchased was, therefore, chosen as the robotic platform for this application.

	Sphero	Kilobot	s-bot	Shooting Star	Crazyflie
movement model	differential	holonomic	differential	holonomic	holonomic
swarm mechanism	aggregation	aggregation	aggregation	none	cooperative mapping
application	self-assembly	self-assembly	path formation	display	search

Table 2.1: Comparison of presented robot platforms

2.3 Software

In order to address the challenges in swarm robotics presented in Chapter 2.2 a set of different software libraries are used in our application. The following sections cover the most important communication, control and localization frameworks.

2.3.1 ROS

The Robot Operating System (ROS) is a flexible open-source framework for writing robot software. It provides libraries, tools and conventions to create robust and complex robot behaviors [38]. Moreover, ROS "provides a structured communications layer above the host operating systems of a heterogeneous compute cluster" [36].

ROS is designed to support the creation of robot applications and adds value to most robotics projects. As ROS supports the reuse of code and organizes software in packages, which can be easily shared and distributed, there is no necessity to implement everything from scratch but rather choose which parts are useful to reuse and which parts need to be implemented or extended by oneself. The core base does not take much space and resources which allows the use of embedded computers and other platforms with less computing power. It is possible to control multiple robots with ROS, where each robot runs its own ROS system and communication between each other is still possible.

From the perspective of swarm robotics, ROS provides many independent control programs, because nodes can run independently, which form a virtually software swarm as a result where the overall program continues to run even when a single node crashes. The mentioned advantages of ROS for robotics projects, specifically swarm robotic applications, are the reason we chose ROS as our robot software framework.

The fundamental concepts of ROS communication are nodes, messages and topics. A Node is an executable file that uses ROS to communicate with other nodes. Nodes are processes that perform computation and a robot control system will usually consist of many nodes. These Nodes register to the ROS Master Node which is a central node providing naming and registration services to all other nodes in the ROS system. It manages the topic subscription and makes them reachable and available for other Nodes. The major advantages

are that crashes are isolated to individual nodes and compared to monolithic systems the code complexity is reduced.

Nodes communicate with each other by transmitting messages. A message is a strictly typed data structure described by a simple, language-neutral interface definition language (IDL). The IDL uses short text files to describe fields of each message and can include arbitrarily nested structures and arrays. The IDL specifies only the syntax used to define the data types and interfaces.

Topics are named buses over which nodes exchange messages. A node sends a message by publishing it to a given topic. In general, nodes are not aware of with whom they are communicating. Nodes can subscribe to relevant topics to get data or publish to relevant topics to share data. A node that subscribes to a topic is called **subscriber** and a node that publishes to a topic is called **publisher**. There can be multiple publishers and subscribers to a topic and a node can publish and subscribe at the same time.

Figure 2.5: Basic structure of communication between nodes in ROS

Melonee Wise created the Sphero ROS package [50] which is a set of drivers, nodes, and description files for using the Sphero with ROS. As these drivers were written for a previous version of the robot that used Bluetooth, we had to revise the implementation to be compatible with the Sphero Sprk+ that communicates over Bluetooth Low Energy. Additionally, adaptions were necessary to integrate our robot localization framework [20] into our ROS environment. The integration of all parts is further explained in Chapter 3.

2.3.2 Navigation

The ability to navigate in an environment is essential for any robotic system especially when multiple robots are used at the same time. As we want to realize the swarm aggregation of robots in our application every robot must be able to move towards a given point in the environment. ROS provides the navigation package containing a 2D navigation stack that takes sensor information and the position of a goal to output safe velocity commands to reach the given goal [28].

The navigation requires information about established coordinate systems using the ROS tf package [48]. The tf package maintains relationships between coordinate frames and a transform tree defines all offsets between different frames in a tree structure. The offset defines the relationships between coordinate frames in terms of both translation and rotation. Different coordinate frames can be the map of the environment, the base of the robot or a sensor on top of a robot. In swarm robotics, multiple robots are operating at the same time and each of them has its own coordinate system which has to be taken care of with specific transformation from the point of reference to the respective coordinate frame. Additional configuration is necessary as the shape and general dynamics of the robot are taken into consideration by the navigation stack as well. For this reason, the implemented path planners, that try to accomplish the navigation task, have a set of parameters which strongly influence the performance. Further information on tuning the navigation can be found in [51].

The navigation stack is only meant for differential-driven or holonomic wheeled robots that can process ROS messages of type Twist. A Twist message expresses velocity in free space broken into a three-dimensional linear and three-dimensional angular vector. However, due to the fact that the navigation stack does two-dimensional path planning, only the x- and y-linear velocity and z-angular velocity are used. For nonholonomic robots, the y-linear velocity is omitted and always set to 0 as those robots cannot move sideways. Furthermore, a static map is required in order for a robot to be navigated, as this map enables self-localization in the environment and path planning to a specific location within the map.

A sensor is generally required to estimate the distance of a robot to its environment but the Sphero is not equipped with such a sensor. Due to the fact that the Spheros are moving in a known environment, it is sufficient enough to provide the known map of the environment to the navigation beforehand. Along with a valid transformation from the map coordinate frame to the coordinate frame of our robot, the navigation stack can be used to navigate the Spheros. Details about the calculation of the transformation between those two coordinate frames are covered in Chapter 3.2.

2.3.3 Camera Tracking

Since we simultaneously run multiple robots in our application, we need to be able to identify each of them separately. In addition, we need to be able to track the position of each Sphero in our environment over time. By observing the internal parameters in the odometry and using the locator, we could keep track of the Spheros position as a dead reckoning approach. But due to the fact that this approach has no external reference, it will end up with complex errors over time. Moreover, it is necessary to know the initial orientation of each robot in the environment because every Sphero has its own coordinate system which always starts with a heading of zero after the boot process.

For this reason, we use the robot localization framework introduced by Hoyer, Steup and Mostaghim [20] to realize identification, position tracking and orientation detection of multiple robots. The framework [20] uses convolutional neural networks (CNN) for localization and instance identification in a two-stage approach. Different robot types and multiple instances of a robot type can be tracked. In our case, the Sphero is a robot that the network is already trained on.

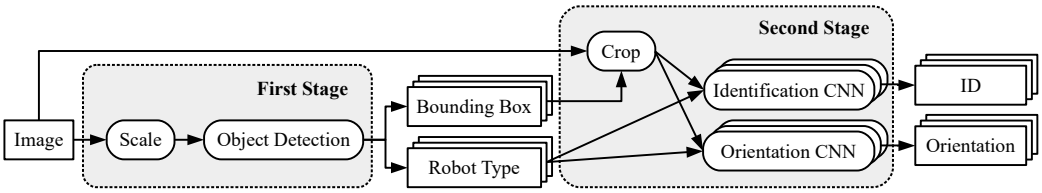


Figure 2.6: Architecture of the robot localization framework [20]

An external camera image of the environment is used as an input to the framework where the image is processed in two stages. In the first stage, the network does object detection on the downscaled image and outputs the according bounding boxes and robot types that were identified. Later, these are used by the second stage to determine the identifier and orientation of each detected robot. Depending on the estimated bounding boxes, the original camera image is cropped and fed into an orientation estimation and a single instance identification CNN. As each robot has different features for identification and orientation, corresponding neural networks are selected for the second stage. The merged output of both stages which contains bounding box, the robot

type, the instance identification, and the orientation of each robot are put out by the framework in the end.

The presented framework is convenient for our swarm robot application due to the fact that multiple robots of the same type can be tracked at the same time, where a constant stream of images is sufficient enough as an input. Furthermore, the framework reaches exceptionally low processing time with 20 ms on a GPU, because lightweight CNNs and a multi-resolution approach are used to provide fast inference speed.

The localization framework is mainly limited in its tracking capabilities by the first stage that uses a Single Shot Detector (SSD) [24] approach based on the MobileNetV2 [42]. The Single Shot Detector scales the camera image down to save computational power but Spheros only cover 25×25 pixel in the 1600×1200 camera image. Therefore, a single Sphero will merely be detected by one pixel in the downscaled image, resulting in poor accuracy. Furthermore, when multiple Spheros are too close to each other, the first stage is unable to create fitting bounding boxes, because they are not distinguishable anymore in the downscaled images and the robot LEDs will outshine each other.

3 Implementation

In this chapter, the implementation of our proposed swarm application is presented. First, the theoretical concept and swarm model are introduced and discussed. Secondly, the general architecture of our application and the ROS-nodes with their functionality are described. Finally, the integration and interaction of all components are summarized.

3.1 Concept

We model our swarm of M Sphero robots accordingly to the swarm model introduced in Chapter 2.1.1 but in two-dimensional space. We consider a simple arena as our environment and, therefore, there is no environmental profile for the members of the swarm. Individuals behave according to the rules of attraction, repulsion and alignment from Chapter 2.1. Consequently, the position of each Sphero i at time t is described as $x^i(t) \in \mathbb{R}^2$ and the resulting equation of position updates is:

$$x^i(t+1) = x^i(t) + \sum_{j=1, j\neq i}^{M} g(x^i(t) - x^j(t)) \tag{3.1}$$

In theory, position updates happen instantly and individuals move from their current position to their target position without any intermediate state. In practice, an immediate change of position is not possible. For this reason, only the target position can be updated and $x^i(t+1)$ is considered as the desired position for the next time step. The current position of individuals changes through their movement which is based on the navigation. Target position updates should not come as fast as possible because robots need a certain amount of time to reach their goal and the navigation has to calculate

a new path for each goal. Because of that, too frequent position updates would only disrupt the movement of robots and slow down the aggregation process of the swarm.

We assume dynamic time updates for the swarm which means that time t does not represent the real-time but serves more as an update counter for different position states which we call robot time. There is a dynamic period of time between states in robot time and $t + 1$ only indicates the next state after state t and not how much time passed between those states. There exists no connection between the robot time and the real-time due to the dynamic update steps in robot time. This discrete dynamic time model defines the position x^i of an individual i at any time $t + 1$ based on the current positions of all swarm members at time t. That is to say, the position of an individual at any time $t + 2$ can not be determined by applying the equation of position updates twice at time t. Since individuals will change their current position when going from state t to $t + 1$, the position of an individual can only be determined for the next state but not for the following states after that. Each new target position is calculated based on the current position of all members of the swarm which are tracked by the robot localization framework. Therefore, we do not aggregate any offset error that could be caused by delays between the real and the desired position of any robot. The target position is only relevant for the robot navigation system because a robot will move towards its target position.

We consider the already introduced attraction/repulsion function in Equation (2.3) and set $a = 0.25, b = 5$ and $c = 0.2$. The resulting comfortable distance between individuals is $\delta = 0.774$. The distance should not extend the size of our arena and leave enough space for the Spheros to move without getting too close where they would no longer be tracked by the localization framework. According to our model and position update equation, the swarm should show an aggregation behavior where a circle formation of the robots emerges over time. As proven by Gazi et al. [14] the center of the swarm is stationary which is the reason why all robots stop moving once they reach a comfortable distance to all other robots and do not have to change their position anymore.

This behavior is consistent with the rules of attraction and repulsion of the basic swarming model introduced in Chapter 2.1. To realize alignment for our swarm, we introduce the average heading of the swarm θ as the average heading θ of each member i of the swarm.

$$\theta(t) = \frac{1}{M} \sum_{i=1}^{M} \theta^i(t) \tag{3.2}$$

Accordingly to the position we define that each individual should align towards the average heading of the swarm, that is:

$$\theta^i(t+1) = \theta(t) \tag{3.3}$$

The heading of each individual is only a desired orientation that each robot should reach, because of the already mentioned delays in the real-world movement. At each time step, every individual of the swarm receives a position to reach, based on attraction and repulsion, and the desired heading to align with the swarm. This will lead the swarm to aggregate in an aligned circle of robots over time.

To enhance dynamic in the swarm we addressed the necessity to expand our scenario with an environmental profile or a leader-follower principle. As already mentioned in Chapter 2.1.2, the leader can be seen as an individual that only has an environmental profile without attraction or repulsion to other members of the swarm. Therefore, we extend our application by implementing a leader-follower principle that will indirectly use an environmental profile to guide the leader.

After the swarm aggregation, a leader is selected to drive a predetermined route indicated by the environment. Once the leader starts moving, all other members start to follow him as they are no longer in comfortable distance and try to aggregate around the leader. The leader is determined by a leadership selection which for our application is a trivial random selection. The leader selection can be replaced by any other selection mechanism such as dynamically changing the leader during movement. In case a leader robot fails, a new one is selected from the swarm because the swarm consists of redundant homogeneous individuals running the same algorithm. The overall aggregation behavior of the swarm is stable and not influenced by the failure of a single individual.

For the individual x^L that is selected as a leader, indicated with index L, the resulting equation of position updates is as follows:

$$x^L(t+1) = x^L(t) - \nabla_{x^L}\sigma(x^L) \tag{3.4}$$

where $-\nabla_{x^L}\sigma(x^L)$ represents the gradient of the environmental profile. This gradient will always point towards the next point on the predetermined route in our implementation.

Different aspects of our application are working together to overcome the swarm robotic challenges presented in Chapter 2.2. Communication is handled via Bluetooth Low Energy and the command and control protocol of the Sphero. The size of the environment does not exceed the communication range of the Sphero and the navigation stack is responsible for the robot's movement control. A combination of relative and external localization is used to address the localization problem. The internal sensors of the Sphero perform dead reckoning while the robot localization framework serves as an external reference. The introduced attraction and repulsion function in Equation (2.3) causes the swarm to show an aggregation behavior.

3.2 Node Architecture

A variety of nodes run on a single robot to handle communication, tracking, sensor input, transformation of coordinate frames, navigation and message flow. The most important nodes running on each robot and for the whole swarm are shown with their respective connections in Figure 3.1. Note that this is not the exact ROS node graph but just an abstract figure that links all relevant parts of the software implementation together and shows their relations with each other condensed to the core. Therefore, communication concepts from ROS like topics and messages are omitted for the sake of clarity and intelligibility. Instead, specific nodes that serve a common purpose in our application are presented and explained in the following.

The *NN_Tracking* node references the localization framework described in Chapter 2.3.3 and provides the positions and headings of all tracked Spheros in the camera image. The *Map_Server* provides map data for the navigation stack. Both are the only nodes that run externally in our application and the data published by them is accessible for all robots. The other nodes run on each Sphero and can be categorized into three groups: hardware, navigation

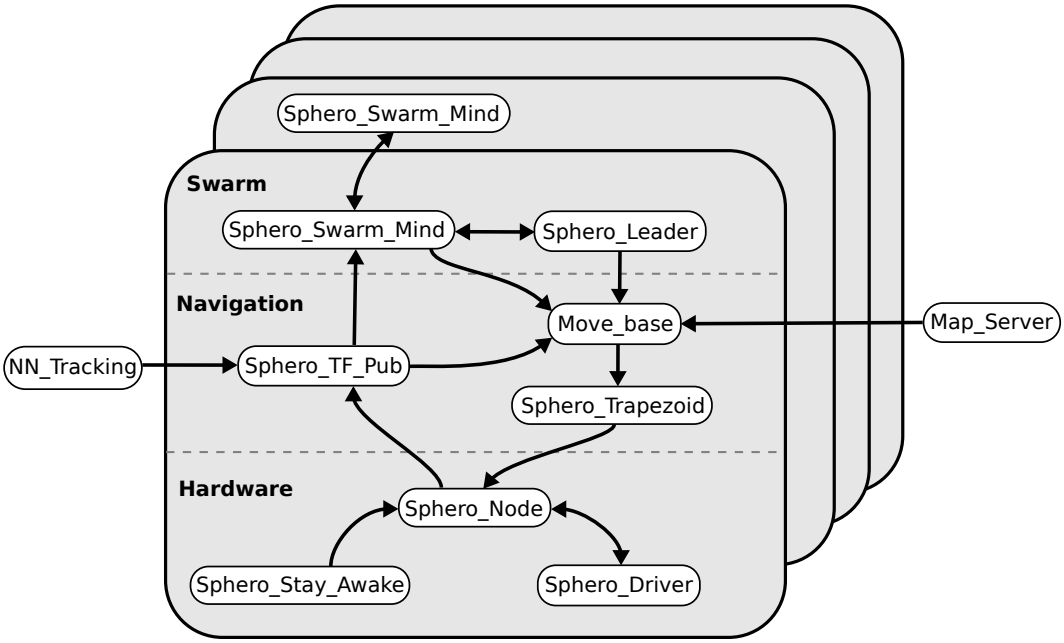


Figure 3.1: Graph of main ROS nodes

and swarm. The following subsections describe the grouped nodes per robot in detail.

3.2.1 Hardware

The hardware nodes mainly handle the communication with the Sphero robot. As the Sphero communicates via Bluetooth, the robot expects messages in a specific byte format. The byte format, commands and communication protocol are defined in the communication API [34]. Commands that are sent via ROS messages to control the robot have to be transformed accordingly. Additionally, the Sphero sends a variety of messages back to the device with which it is connected, e.g. a sensor stream or collision message. These messages need to be converted to ROS messages to publish them and make them available for other nodes to use.

Sphero_Driver provides a Python-based driver for interfacing with the Sphero via Bluetooth. The driver only needs the MAC address of a Sphero to connect and enable communication with it. All incoming messages from the connected device are handled and collected in a data list for further use. For

all defined control commands in [34], the driver builds and sends suiting messages in the specified byte format. All low-level functionalities will be accessed by the Sphero Node to interact with the Sphero.

Sphero_Node provides a straightforward ROS wrapper for the *Sphero_Driver*. The data stored in the data list by the driver is parsed into a ROS-compliant structure and published to be accessible for other nodes. Most of the data comes from the sensors and is used to check the robot's internal state. A starting routine is implemented that combines multiple commands for configuration, e.g. enabling data streaming and stabilization. By executing the starting routine the robot is connected and made ready for use.

Sphero_Stay_Awake sends a color command to the Sphero every 20 seconds when no other commands are sent, e.g. when each robot has reached comfortable distance. This prevents the Sphero from entering sleep mode which would shut down the robot and disconnect it from our application.

3.2.2 Navigation

As already mentioned in Chapter 2.2.4, in order to move the Sphero requires speed as a 1-dimensional linear velocity vector and heading as an angle in integer degree. The navigation stack outputs commands containing a 1-dimensional linear velocity and a 1-dimensional angular velocity. The nodes categorized as navigation handle the necessary message flow in ROS to convert the velocity commands from the navigation stack into roll commands for the Sphero. Additionally, the information provided by the localization framework and the sensor data are combined to calculate the position and the heading of the Sphero.

Sphero_TF_Pub calculates the necessary transformation of coordinate frames for the navigation package introduced in Chapter 2.3.2 and publishes the pose of the robot in the map. The transformation is an offset in terms of both translation and rotation between the map coordinate frame, called `map`, and the coordinate frame of our robot, called `base_link`. In general, the `base_link` provides an obvious point of reference and will be different for every hardware platform but for our small circular robot, we define it as the center of the Sphero. The translation offset is provided by the *NN_Tracking* that outputs the position of each Sphero in the environment. As the camera

image has a different coordinate system and origin than the map, we adjust the coordinate positions from the camera tracking by inverting the x-axis. For further information about the environment and coordinate systems see Section 4.3. The rotation offset is acquired by combining the provided heading by the *NN_Tracking* and the current heading provided by the IMU of the Sphero. Both offsets are combined to a single transformation from `map` to `base_link` and published to enable navigation for the Sphero. These offsets are furthermore used to publish the position and heading of the robot in the map for other nodes to access.

Move_base represents the ROS interface for interactions with the navigation stack. Path planning and navigation for the robot is accomplished by linking together a local and global planner [27]. The node takes in the published transformations from *Sphero_TF_Pub*, the published map from *Map_Server*, a target position from the swarm nodes and outputs velocity commands for the robot.

Sphero_Trapezoid combines the rotation offset from the *Sphero_TF_Pub* and the velocity commands from the *Move_base* to convert the angular velocity into a set angle for the Sphero by applying the trapezoidal rule. The trapezoidal rule is used to approximate the definite integral of the angular velocity over time. This integral represents the angle that the Sphero should be facing at a given time. The approximated angle in combination with the linear velocity from the original velocity command is send to the Sphero via *Sphero_Node*.

3.2.3 Swarm

The swarm nodes implement the swarm behavior, potential function and leader-follower principle and represent the core of our application. They access the current position of swarm members, calculate the position update accordingly and publish the desired position for an individual to the navigation.

Sphero_Swarm_Mind takes in all Sphero positions and headings provided by each *Sphero_Trapezoid*, based on this and with the use of the potential function from Equation (2.3), calculates the next position at which the robot should move. In the same way, the orientation for the robot to align with the swarm is calculated but with the alignment Equation (3.2). The new position and orientation are combined to a goal that is published as the current goal

to the *Move_ base*. A new goal is computed every time the position of any Sphero is published. As positions are published at very high frequency, the node would constantly publish new goals. This would only interfere with smooth navigation because for each goal the navigation would calculate a new path to reach a similar goal and discard the old path that the Sphero was already following. Therefore, only goals that are significantly distinct from the previous goal are set as the current goal for the robot. The node implements two different states which indicate whether swarm aggregation is active or not. In case an individual is selected as a leader, the node will deactivate aggregation and stop publishing new goals.

Sphero_ Leader randomly selects a member of the swarm, designates him as the leader and deactivates its swarm aggregation by changing the state of the *Sphero_ Swarm_ Mind*. The node loads the predetermined route for the leader as a list of points that define the route. Successively the next point to reach is published as a new goal once the leader has reached the previous goal. In case the selected leader robot fails, position updates will no longer be published by the *Sphero_ Trapezoid*. The node then directly selects a new leader for the swarm.

3.3 Integration

Due to the fact that we have no access to the microcontroller of the Sphero and, therefore, cannot run software on the robots itself, it is necessary to do the computation on remote machines for all individuals at the same time.

The localization framework and robot sensors serve as an information source to consistently estimate the state of a Sphero. By combining global external localization and robot sensors we get the position and orientation of each individual of the swarm at any time as they update frequently. This allows us to compute the next position, where each individual should move according to the attraction/repulsion function. With our navigation architecture, each robot will drive to its given next position, building the swarm formation over time. The interface between ROS and the Sphero allows precise control of each individual to assure the robots move in our real-world environment according to the planned path by the navigation.

Overall, each Sphero reacts on a position change of all swarm members, including itself, as a new goal is computed every time the localization framework

outputs the position of any robot. In case that a robot fails, it will not be tracked and, therefore, not considered for the swarm aggregation anymore by the swarm. All other individuals would act accordingly to adjust their positions and the swarm would continue to operate. Once a leader is selected, it moves on a given path and ignores all other robots in the swarm. While the other Spheros are influenced by the leader and try to aggregate around him. This leads to the effect that the whole swarm moves on the predetermined path because all individuals try to stay in comfortable distance to the leader on their path.

In general, the camera tracking and sensors provide necessary information for our swarm behavior but the localization approach in use is flexible for our application. It is possible to exchange the camera tracking for any other information source that outputs the current position of each Sphero. It is even possible to only use the local sensor information of the robot. However, the missing reference point for the dead reckoning approach and the unknown initial rotation in the environment pose problems that would need to be addressed accordingly. The same holds for the navigation nodes where any other navigation implementation or path planner could be used to navigate each robot to their goal.

The swarm would behave in the same way, even when certain groups of nodes were implemented differently but would produce the overall same abstract behavior relative to the other groups of nodes. Essentially, the swarm behavior nodes can be changed, because they are integrated in the hardware and navigation nodes, which can operate with any node that provides a goal for a robot. Another behavior could simply be implemented by replacing the *Sphero_Swarm_Mind* with a node that takes in the position of each member and outputs positions to reach in the environment. One example of a different behavior could be a foraging behavior for robots [4] where a swarm of robots has to collect resources that are scattered in the environment and bring them to the nest. Additionally, extensions to an already implemented behavior are possible as shown by our leader extension. With this extension, the static swarm aggregation behavior is changed to allow swarm movement without altering the base application.

4 Evaluation

This chapter discusses the applied metrics to evaluate swarm behavior and describes the experimental setup and experiment scenarios. A set of experiments is conducted to determine whether the swarm exhibits aggregation behavior. The effects of scaling up the number of individuals and the resulting impact on the overall performance of the swarm are examined. In addition, the leader-follower extension is tested and evaluated. The results of the experiments are then discussed in conclusion.

4.1 Metrics

The local interactions of the members of our swarm produce an observable behavior on a macro-level. The applied rules of attraction, repulsion and alignment result in the emergent behavior of aggregation where a circle is formed by the Spheros.

Emergent behaviors are non-obvious side effects that show up when different capabilities of individuals are combined. Emergence can be seen as self-organized order and evidently, the swarm can then be viewed as a self-organized system with the aim of creating order. We need a quantitative approach to characterize the degree of emergence to evaluate the level of order in the swarm. It has been shown that entropy can be used to evaluate the self-organizing properties of swarm intelligence algorithms [10]. Therefore, to characterize this degree of emergence we use entropy as a quantitative approach.

The concept of entropy in information theory was introduced by Shannon [44] in 1948. The entropy H of a discrete random variable X with n possible values is defined as

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{4.1}$$

where p_i is the probability of the value $x_i \in X$. The maximum value of entropy is $\log_2 n$ when all probabilities p_i are equal.

Entropy is a metric of order where generally lower entropy represents a higher level of order and vice versa. But the definition of order and disorder depends on the observation model. An observation model determines the attributes to be observed and their quantization. The observation model, together with the observed parameters, has a major influence on entropy since each model weighs the observed parameters differently.

To measure the aggregation behavior of the swarm, we have to measure whether all individuals are in comfortable distance towards each other. The individuals form a circle when they are in comfortable distance towards each other and, therefore, the circle formation of the swarm has to be measured.

To measure this geometric shape, a position observation model is used, where the positions of the individuals are the observed attribute. By definition, a circle is characterized as a shape consisting of equidistant points relative to the center of the circle. The center of the swarm is set as the center of the circle. The distance d_i from the position of each individual $x^i \in \mathbb{R}^2$ to the center of the swarm $\bar{x}$ is then measured as follows:

$$d_i = \|x^i - \bar{x}\| \tag{4.2}$$

where $\|\cdot\|$ represents the Euclidean norm.

The experimental determination of the entropy H relies on approximating the probabilities p_i. Relative frequencies are an approach to determine the probabilities from observed values [43]. These probabilities are normalized and measure the relative importance of a value compared to all other values. Hence, the probability p_i of an observed individual i is calculated as the relative distance of its own position to the center of the swarm:

$$p_i = \frac{d_i}{\sum_{j=1}^{M} d_j} \tag{4.3}$$

The computed entropy is based on the distribution of the relative distances of individuals towards each other and has its maximum value when all relative distances are equal which is the case when a circle is formed.

The previous observation only takes the positions of individuals into account, which is consistent for the rules of attraction and repulsion of the basic swarm model introduced in Chapter 2.1. But a different observation model is necessary to measure alignment. The observation model for alignment considers the orientation of the robots as the observed attribute. To evaluate whether the swarm aligns, only the orientation θ_i of each member of the swarm i needs to be considered.

The orientation is not measured as the absolute difference between the orientation of the individual and the average orientation of the swarm. In case two individuals have the same absolute difference but one has a positive difference and the other a negative one, they steer in different directions. But the absolute differences and also the probabilities are equally distributed in that case, which results in maximum entropy when the individuals are clearly not aligned. For this reason, only the individual's orientation is measured and the probability p_i of an observed individual i is calculated as the relative heading of this individual:

$$p_i = \frac{\theta_i}{\sum_{j=1}^{M} \theta_j} \tag{4.4}$$

Again the highest entropy is achieved when all probabilities are equal, which is the case when all robots in the swarm are aligned.

4.2 Scenario

Due to the limitation of simultaneous connections for a Bluetooth dongle, only five Spheros can connect at the same time. This significantly limits the possible number of active individuals in the swarm. Two or fewer robots are unreasonable because one robot does not represent a swarm and two individuals are always in the same distance from their center of the swarm. For this reason, we only consider scenarios with three, four and five Spheros.

Individuals are distributed randomly in the environment at the beginning of each experiment to ensure that the level of order is low. The respective swarm node, described in Chapter 3.2.3, is started for each robot as soon as all robots

are in position. Each experiment lasts 10 minutes, in which the entropy is recorded at a rate of 1 Hz.

The first series of experiments determine whether the swarm is able to exhibit aggregation behavior and which influence the size of the swarm has on this behavior. For this reason, three experiments are carried out, one for each possible swarm size, in which the robots are put into the arena, with the *Sphero_Swarm_Mind* running on all members of the swarm. In case that the individuals reach maximum entropy for a period of time, a random move command is sent to disperse them. The idea behind this is to test whether the aggregation behavior is robust and not prone to disruptions.

The second series of experiments evaluate the implemented leader-follower principle with four Spheros. Again, all robots start with the *Sphero_Swarm_Mind*. The *Sphero_Leader* node is then used to select a leader to drive a given predetermined route for the rest of the experiment. This is done for a rectangular and an ellipsoid route.

4.3 Environment

The experiments are conducted in an arena consisting of one continuous, 3 meters wide and 4 meters long closed off area as shown in Figure 4.1. This arena represents the known environment for the swarm. The camera, which is located in the ceiling above the arena, outputs a 1600×1200 pixel image of the arena and the space around it.

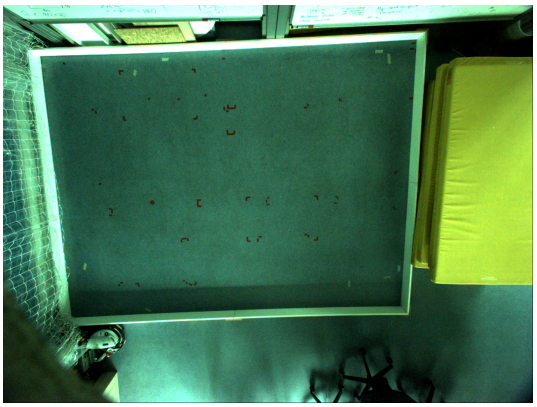


Figure 4.1: Camera view of the arena

As described in Chapter 2.3.3, the robot localization framework outputs the coordinates of each robot in the camera image. Note that the framework uses a heavily darkened image and not the one shown in Figure 4.1 to reduce the features of an image and increase the visibility of the robot's LEDs.

In ROS, the environment is represented as a map that is provided by the *Map_Server*. This map is a grayscale image of the same size as the camera image, in which black pixels are occupied, white pixels are free and pixels in between are unknown. We represent the space outside of the arena as occupied and the arena itself as free space. This results in the map shown in Figure 4.2.

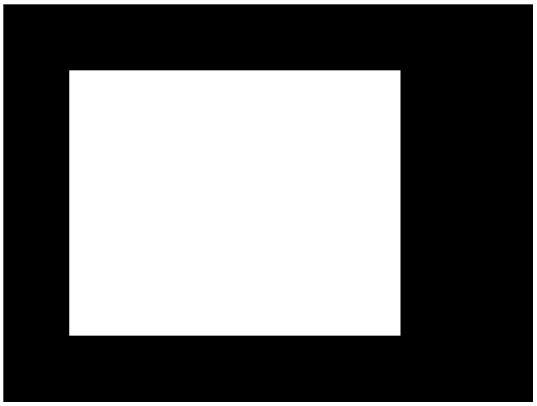


Figure 4.2: Resulting map that represents the arena

The camera image and the map do not have the same coordinate system. The camera has its origin in the top left and uses a left-handed coordinate system, while the map follows the ROS coordinate conventions and has a right-handed coordinate system. This results in a mirrored x-axis between the camera image and the map of the environment. Due to the slight oblique position of the camera, the map does not match the real arena perfectly, but this can be compensated for by applying a homography. Since this error is comparatively small, we use a small scaling instead, when transforming from the camera to ROS coordinates to compensate for this.

The robot must be navigated in this given environment, represented by the arena. We are using the navigation software introduced in Chapter 2.3.2 for this task. As with any navigation planner, however, the navigation stack requires suiting parameters to enable the precise movement of the robot. These parameters are always dependent on the respective environment, the consid-

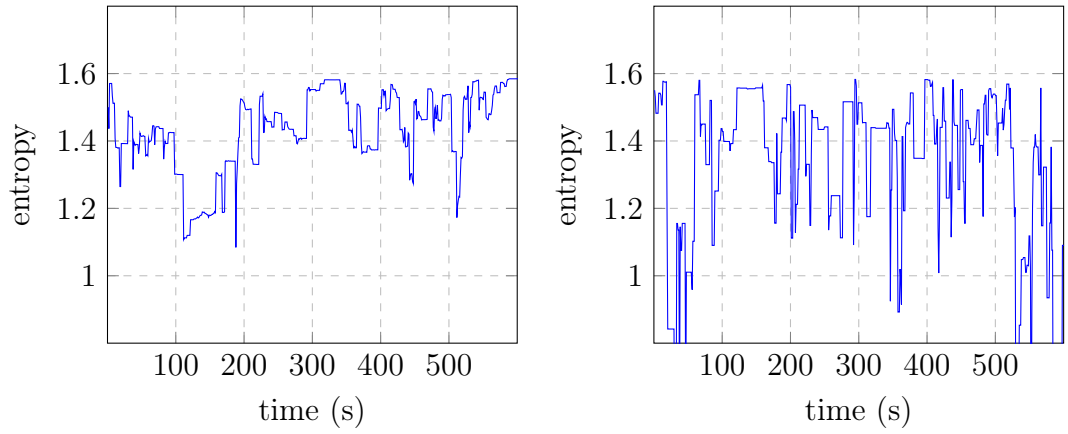
ered robots and the task. The configuration of the navigation stack for the experiments with the Spheros is given in the following parameter table:

parameter	value
base_global_planner	NavfnROS
base_local_planner	TrajectoryPlannerROS
controller_frequency	2.0
recovery_behavior_enabled	false
clearing_rotation_allowed	false
allow_unknown	false
default_tolerance	0.1
acc_lim_x	2.5
acc_lim_theta	3.2
max_vel_x	0.2
min_vel_x	0.1
max_vel_theta	0.7
min_vel_theta	-0.7
min_in_place_vel_theta	0.3
escape_vel	0.0
holonomic_robot	false
yaw_goal_tolerance	0.4
xy_goal_tolerance	0.2
latch_xy_goal_tolerance	true
sim_time	1.0
sim_granularity	0.025
angular_sim_granularity	0.025
vx_samples	3
vtheta_samples	20
meter_scoring	true
pdist_scale	0.6
gdist_scale	0.8
occdist_scale	0.01
heading_lookahead	0.325
heading_scoring	false
heading_scoring_timestep	0.8

Table 4.1: Parameters for the ROS navigation package

4.4 Results

In this section, the observations of the different scenarios are evaluated. Results are then normalized and put into perspective to make the effects of scaling up the number of individuals in the swarm more visible. In addition, the observations from the second series of experiments are compared to the previous results.



(a) Positions

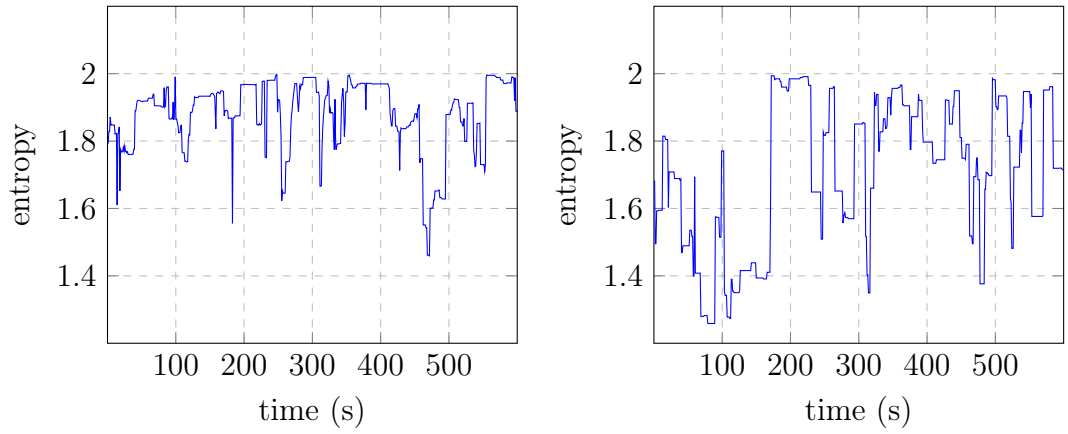
(b) Orientations

Figure 4.3: Observations on the aggregation behavior with three individuals.

The initial experiment of the first series with three Spheros already shows that the swarm is able to exhibit aggregation behavior. As shown in Figure 4.3 for the position observation model, the position entropy is not constantly increasing but the robots always try to get to a comfortable distance towards each other.

For this reason, entropy continues to increase even after breakdowns and peaks to the maximum value in certain periods of time. The acting forces of attraction and repulsion increase the entropy over time, while errors in the navigation and movement of the robot cause a spontaneous decrease in the entropy. Therefore, entropy fluctuates, which is a swarm that is constantly responding to malfunctions.

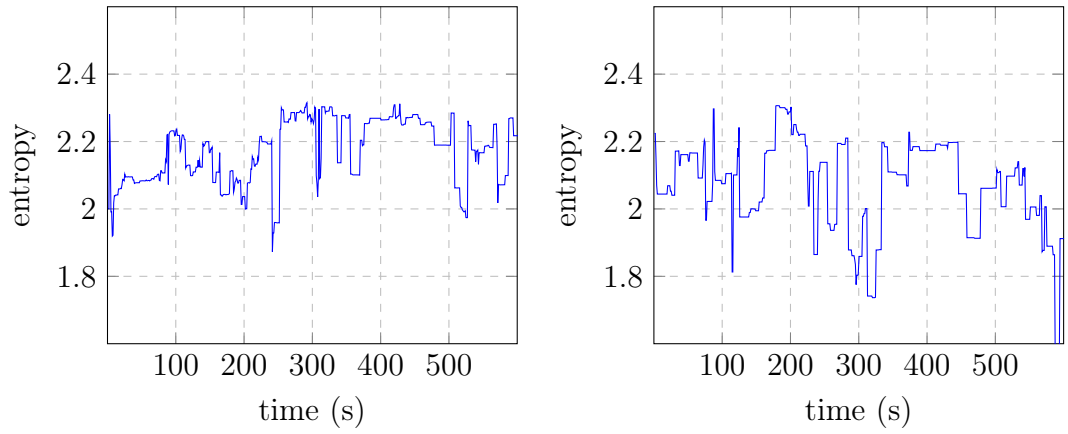
The orientation entropy fluctuates significantly more than the position entropy. Furthermore, the orientation entropy values show larger deviations in comparison. As the robots move around, they constantly change direction,



(a) Positions (b) Orientations

Figure 4.4: Observations on the aggregation behavior with four individuals.

which leads to extensive disturbances in the average orientation of the swarm. Alignment cannot compensate for this over long periods of time, because the system is dynamic and constantly moving.



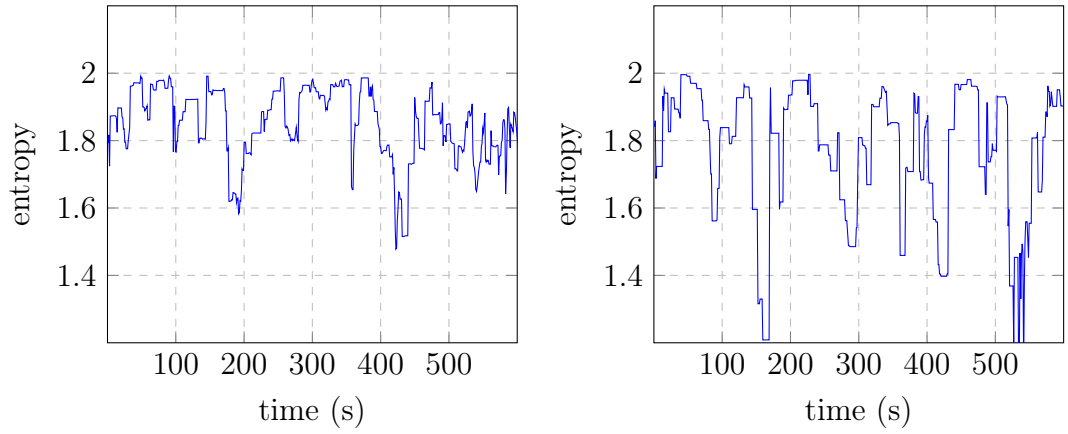
(a) Positions (b) Orientations

Figure 4.5: Observations on the aggregation behavior with five individuals.

Increasing the size of the swarm to four individuals causes the swarm to behave similarly to the previous experiment. As can be seen in Figure 4.4, the swarm reaches the maximum entropy more often in this scenario than in the previous

one. The position and orientation entropy are still fluctuating but no longer as frequent as with the previous swarm size. Even after a significant collapse, the swarm gets into formation, which is indicated by an increase in the position entropy over time. Compared to the previous experiment, the orientation entropy does not fluctuate as often but still shows an immense deviation.

The last experiment of the first series, which is carried out with five Spheros, shows the same scaling effect on the entropy of position and orientation as the previous observations. Compared to the other swarm sizes, the position entropy for five individuals more often reaches its maximum value and the entire entropy fluctuation subsides.

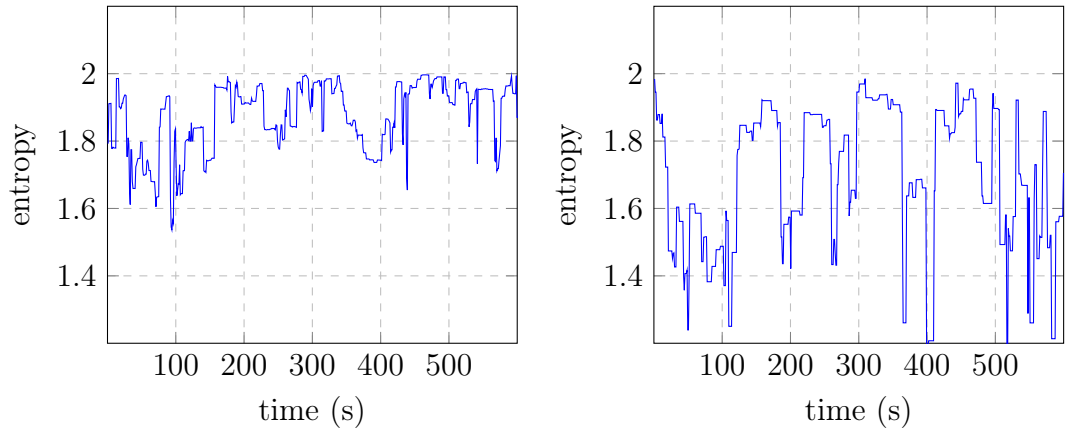


(a) Positions (b) Orientations

Figure 4.6: Observations on the leader-follower scenario with a rectangular route.

The results of the leader-follower scenario, in which one individual is constantly moving on a rectangular route, are shown in Figure 4.6. Accordingly, the observations for the ellipsoid route are shown in Figure 4.7. Despite the aggravation of a leader who ignores all other members of the swarm and is constantly moving on a predetermined route, individuals continue to aggregate.

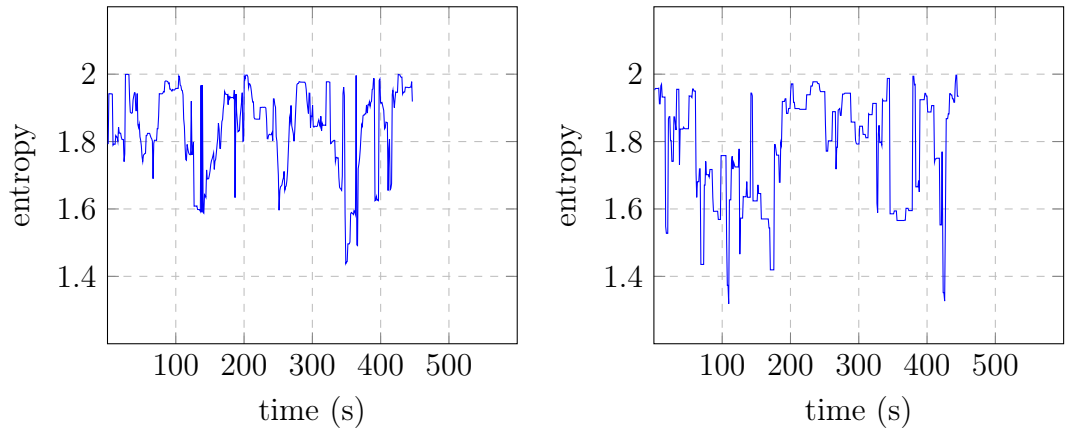
The aggregation behavior of the swarm is not stopped by the leader-follower extension. The visible impact on the swarm's performance is that individuals lag behind the leader trying to reach into comfortable distance towards him. For this reason, the swarm does not reach the maximum entropy as often as in the previous scenarios.



(a) Positions (b) Orientations

Figure 4.7: Observations on the leader-follower scenario with an ellipsoid route.

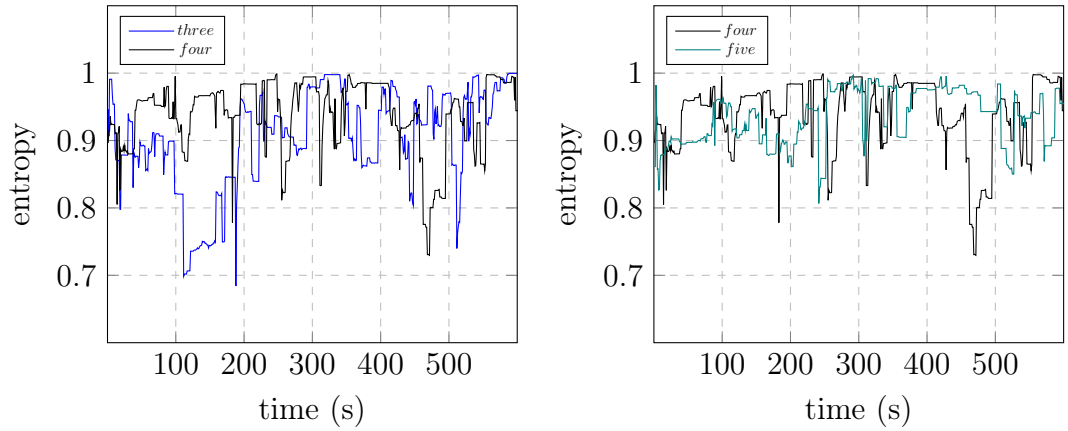
While robot swarms are generally robust and the failure of a single individual does not affect the overall performance of the swarm, observation models are not. During one run of the leader-follower experiment, a robot started rotating in place uncontrollably. The robot had to be reset to avoid damage and reduce the volume level, because the continuous rotation caused loud noise, so the experiment had to be stopped.



(a) Positions (b) Orientations

Figure 4.8: Observations on the leader-follower scenario with a robot failure.

Since the robot was still active and connected, the measured orientation of the robot would be indeterminable and would cause the rotation entropy to collapse. The swarm is unable to compensate for an individual that rotates in place uncontrollably. This misconduct occurred due to internal errors in the robot IMU. This problem was not previously addressed, because it is a specific internal error in the Sphero's own movement control that cannot be corrected, because there is no access to the microcontroller of the robot.



(a) Three and four individuals

(b) Four and five individuals

Figure 4.9: Comparison of the normalized position entropy for the aggregation behavior with different swarm sizes.

To put the results of the first experiment series into perspective, the entropies were normalized for different swarm sizes to make them more comparable. This is done by dividing each entropy by the respective maximum entropy. The resulting normalized entropies for the first series of experiments are shown in Figure 4.9. Two graphs were used for comparison for the sake of clarity and legibility. As already described above, with an increasing number of individuals in the swarm, the entropy fluctuation diminishes and the range of deviation decreases. Therefore, the overall performance of the swarm increases with the number of individuals in it.

4.5 Discussion

The task of getting three robots to form a circle and align is expected to be easier than the same task with four or five robots since fewer robots would have to be arranged. Therefore, the swarm of five people could be expected to perform worse on this task. However, the results of the experiments show that this does not apply to the aggregation behavior of the swarm shown in our experiments. As the number of robots increases, the swarm becomes less susceptible to small deviations and individual robots no longer have such a strong impact on the overall performance. Even with incomplete localization due to the inaccuracy of the robot sensor and the inaccuracy of the external localization, the Spheros continued to aggregate. In addition, disrupting the swarm with a selected leader following a given route and dragging the other individuals around does not prevent the swarm from displaying complex behavior that is only be implemented with simple rules. The disturbance that the leader can represent in terms of static aggregation only enables the swarm to dynamically aggregate on its path. Here, an individual invokes a new complex behavior in the swarm, without implicitly telling other individuals how to do so. The other members just continue aggregating based on the position and orientation update rule.

Despite all listed adversities, the swarm shows a stable aggregation behavior that requires the robot swarm to be robust, scalable and flexible. Therefore, it is possible to transfer the theoretical concepts of swarm aggregation to practice with a swarm of Sphero robots.

However, the results must be viewed critically and seen in terms of the overall setup of the application. The swarm behavior is stable in the sense that the swarm is able to converge to the maximum entropy value and form a circle. But with the physical limitations of the arena, the robots could not drift too far from the center of the swarm. The bounding box problem of the robot localization framework prevents the robots to get to close to each other because the robots are no longer tracked when they get to close to one another. The bounding box problem of the robot localization framework prevents the robots from getting too close to each other because the robots are no longer tracked in that case. The observation can then no longer correctly observe their positions. The unbound attraction and repulsion function is then physically bound by the environmental and localization constraints that prevent the robots from moving too close or too far apart.

The attraction and repulsion function requires an infinitely large environment, which is why robots get stuck at the boundary of the arena at times. This happens when the attraction and repulsion function calculates a point outside the arena as the next desired position. A robot with a goal outside of the arena continues to drive into the walls of the arena until it is attracted to the free arena space again.

The navigation is a critical part of the implementation because path planning tends to navigate robots in a way that reduces entropy during movement. Because Spheros are differential-driven robots, moving them to a destination means they have to turn, reducing the orientation entropy in the worst case. To increase the position entropy, the orientation entropy must, therefore, be reduced. Only when a stable position is reached, the robot can rotate in place to increase the entropy of orientation. For this reason, orientation entropy in the experiments collapses, because the robots are navigated towards a new goal that is not in their current direction. An alternative would be an easier navigation system since there are no obstacles in the arena and collisions are not a problem due to the robust casing of the Sphero. In addition, easier navigation would be more consistent with swarm intelligence theory, where particles only receive a direction to their destination. However, planners are often used in practice. By showing that complex navigation works, easier navigation always remains an option when the performance is affected too much.

Robot failures are currently a serious problem because they are unpredictable. In the event of uncontrolled in place rotation, the robot must be reset, although the swarm could continue to work. The noise level is not acceptable for a tech demo. The reason for this behavior is most likely the Sphero device's internal reference heading, which becomes inaccurate after a few minutes of driving. This means that the offset between the Sphero's reference heading and its calibrated reference heading will add up over time until the robot can no longer handle and eliminate it. This then requires recalibration of the reference heading which can be done by resetting the robot.

The size of the swarm is strictly limited by the robot localization framework. The color of the Sphero's LEDs serves as an identification marker but the framework is limited in the number of colors it can distinguish in an image. Overall, the camera tracking is very sensitive to noise in the camera image and ambient light. Bluetooth is another limiting factor for the swarm size, as

an additional Bluetooth dongle is required for every five additional Spheros. In addition, bandwidth is an aspect that must be taken into account when increasing the number of communicating robots. Because of these limitations, experiments could only evaluate scalability for small-sized swarms.

5 Conclusion

In this thesis, an application of swarm intelligence algorithms to a robot swarm was designed. This application enables the creation of a dynamic tech demo to demonstrate swarm robotics concepts. To implement the application, we first introduced basic concepts for swarm intelligence, including three rules for modeling a swarm and a system based on attraction and repulsion that leads individuals to aggregate. In addition, a concept was presented to extend the application by a leader-follower principle. Pivotal challenges and advantages of swarm robotics were discussed in connection with the change from theory to practice, from swarm intelligence to swarm robotics. We put the robots under consideration for this application into perspective with other swarm robotic platforms and gave an overview of current research in this area. Then, the concept of implementation was outlined and the respective implementation was presented.

Several experiments were carried out to determine whether the swarm shows the theoretical behavior in practice and to assess the possible scalability of the swarm and its influence on the performance. The results suggest that it is possible to apply the theoretical concepts of swarm aggregation to a swarm of Sphero robots, but with limitations. In addition, it was found that the swarm can be scaled up to the current limit of robots, which is possible in our setup. The overall performance of the swarm in terms of aggregation behavior increases with the number of individuals. Together with the proposed implementation, the robots represent the required tech demo that demonstrates swarm dynamics. The required increased movement is evoked in the swarm, which is indicated by the fluctuation in entropy. The individuals never aggregate statically but are always in motion.

The configuration and control of a single robot for the precise execution of a specific task are already very complex. Therefore, the effort to enable a swarm of robots to exhibit a certain behavior seems immense. Due to the fact that swarm control architectures are based on self-organization and local

interaction, the coordination mechanisms are scalable. This enables the design of behaviors for a swarm of robots without configuring and controlling every robot to behave in a specific way. This enables behaviors to be designed for a swarm of robots without having to configure and control each robot to execute a specific task. These advantages are shown in our application since several robots can be controlled simultaneously with the same swarm behavior implementation. The main effort then, however, is to overcome the challenges of swarm robotics to increase the size of the swarm, which means that more and more robots have to be located and controlled at the same time.

To conclude, while the performance of the swarm increases with the number of individuals, typical swarm robotic challenges aggravate with the swarm size. Therefore, the results show that theoretical concepts of swarm intelligence can be applied to robot swarm. However, the extent of this application is based on the respective handling of occurring swarm robotic challenges. Although our application solves the problem definition, there are still many possible improvements. We would like to suggest topics for future research and share some ideas for extending our application. The number of simultaneously connected Spheros can be increased up by using multiple Bluetooth dongles. This would require further research to determine the maximum usable bandwidth and the impact of interference on message throughput. The number of learned identification markers limits the localization framework in the number of robots it can differentiate. In addition, swarm behavior is negatively affected when robots are not recognized in certain situations. The task of object detection and identification for several small objects in one image remains an open topic in research. One approach that is more invariant to object size would be Faster R-CNN. This would require a comparison with the MobileNetV2 architecture currently in use. In future work, any misconduct of the Sphero could be remedied by sending a reset command to the robot. This would make manual intervention unnecessary and allow the swarm to continue its behavior.

Bibliography

[1] F. Alkhateeb, E. A. Maghayreh, and S. Aljawarneh. A multi agent-based system for securing university campus: Design and architecture. In *2010 International Conference on Intelligent Systems, Modelling and Simulation*, pages 75–79, Jan 2010.

[2] Bitcraze. Crazyflie 2.0.

[3] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

[4] Alexandre Campo and Marco Dorigo. Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life*, pages 696–705. Springer Berlin Heidelberg, 2007.

[5] Natalie Cheung. Technology behind the intel drone light shows. *Robotics Tomorrow*, 09 2017.

[6] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. Technical report, Department of ComputerEngineering, Middle East Technical University Ankara, 2005.

[7] Şahin E., Girgin S., Bayindir L., and Turgut A.E. Swarm robotics. In Blum C. and Merkle D., editors, *Swarm Intelligence. Natural Computing Series.*, pages 87–100. Springer, Berlin, Heidelberg, 2008.

[8] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2 edition, 2010.

[9] M. Faria, A. Costigliola, P. Alves-Oliveira, and A. Paiva. Follow me: Communicating intentions with a spherical robot. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 664–669, Aug 2016.

[10] Gianluigi Folino and Agostino Forestiero. Using entropy for evaluating swarm intelligence algorithms. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, volume 284, pages 331–343, Berlin, Heidelberg, 01 2010. Springer Berlin Heidelberg.

[11] Beni G. and Wang J. Swarm intelligence in cellular robotic systems. In Dario P. and Sandini G.and Aebischer P., editors, *Robots and Biological Systems: Towards a New Bionics?*, volume 102 of *NATO ASI Series (Series F: Computer and Systems Sciences)*. Springer, Berlin, Heidelberg, 1993.

[12] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31, 10 2007.

[13] V. Gazi and K. M. Passino. A class of attraction/repulsion functions for stable swarm aggregations. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 2842–2847 vol.3, Dec 2002.

[14] Veysel Gazi and Kevin Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48:692 – 697, 05 2003.

[15] Veysel Gazi and Kevin Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48:694, 2003.

[16] Veysel Gazi and Kevin Passino. Stability analysis of social foraging swarms. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34:539 – 557, 03 2004.

[17] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 1134–1140, 1999.

[18] Deborah M. Gordon, Richard E. Paul, and Karen Thorpe. What is the function of encounter patterns in ant colonies? *Animal Behaviour*, 45(6):1083 – 1100, 1993.

[19] Self-Organizing Systems Research Group. Kilobots.

[20] Lukas Hoyer, Christoph Steup, and Sanaz Mostaghim. A robot localization framework using cnns for object detection and pose estimation. *2018*

IEEE Symposium Series on Computational Intelligence (SSCI), pages 1388–1395, 2018.

[21] Intel Corporation. *Intel Shooting Star Drones Light Up the Sky*, 6 2019. Rev. 3.

[22] M. Ioannou and T. Bratitsis. Teaching the notion of speed in kindergarten using the sphero sprk robot. In *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, pages 311–312, July 2017.

[23] Belkacem Khaldi and Cherif Foudil. An overview of swarm robotics: Swarm intelligence applied to multi-robotics. *International Journal of Computer Applications*, 126:31–37, 09 2015.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[25] A. Loria, J. Dasdemir, and N. Alvarez Jarquin. Leader–follower formation and tracking control of mobile robots along straight paths. *IEEE Transactions on Control Systems Technology*, 24(2):727–732, March 2016.

[26] Vladimir Lumelsky and Alexander Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[27] Eitan Marder-Eppstein. move_base - ros wiki.

[28] Eitan Marder-Eppstein. navigation - ros wiki.

[29] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35), 2019.

[30] Francesco Mondada, Giovanni C. Pettinaro, Andre Guignard, Ivo W. Kwee, Dario Floreano, Jean-Louis Deneubourg, Stefano Nolfi, Luca Maria Gambardella, and Marco Dorigo. Swarm-bot: A new distributed robotic concept. *Autonomous Robots*, 17:193–221, 09 2004.

[31] Simon Andreas Engstrøm Nistad. Sphero nav - robotic navigation and control platform. Master's thesis, The Arctic University of Norway, 6 2014.

[32] Shervin Nouyan, Alexandre Campo, and Marco Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2, 03 2007.

[33] Hyondong Oh, Ataollah R. Shiraz, and Yaochu Jin. Morphogen diffusion algorithms for tracking and herding using a swarm of kilobots. *Soft Computing*, 22:1833 – 1844, 03 2018.

[34] Orbotix Inc. *Orbotix Communication API*, 8 2013. revision 1.50.

[35] Marian-Nicolae Pinzariu and Adrian Iftene. Sphero - multiplayer augmented game (smaug). In *RoCHI*, 2016.

[36] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[37] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery.

[38] Open Robotics. About ros.

[39] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. *Proceedings of 2012 IEEE International Conference on Robotics and Automation (IRCA 2012)*, pages 3293–3298, May 2012.

[40] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[41] D. Sakai, H. Fukushima, and F. Matsuno. Leader–follower navigation in obstacle environments while preserving connectivity without data transmission. *IEEE Transactions on Control Systems Technology*, 26(4):1233–1248, July 2018.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[43] Randall Schumacker and Sara Tomek. *Probability*, pages 11–41. Springer New York, New York, NY, 2013.

[44] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[45] Onur Soysal and Erol Şahin. Probabilistic aggregation strategies in swarm robotic systems. Technical report, Department of ComputerEngineering, Middle East Technical University Ankara, 2005.

[46] Sphero SPRK+. Sphero.

[47] Ying Tan and Zhong-yang Zheng. Research advance in swarm robotics. *Defence Technology*, 239, 03 2013.

[48] Eitan Marder-Eppstein Tully Foote and Wim Meeussen. tf - ros wiki.

[49] Trianni V. and Campo A. Fundamental collective behaviors in swarm robotics. In Kacprzyk J. and Pedrycz W., editors, *Springer Handbook of Computational Intelligence*, pages 1377–1394. Springer, Berlin, Heidelberg, 2015.

[50] Melonee Wise. Sphero ros - sphero_ros 0.1 documentation.

[51] Kaiyu Zheng. ROS navigation tuning guide. *CoRR*, abs/1706.09068, 2017.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Christian Wustrau Magdeburg,