mαcrϕ@ufmg

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Research group MACRO - Mechatronics, Control and Robotics

DECENTRALIZED SEGREGATION IN SWARMS OF ROBOTS

Edson Bernardes Ferreira Filho

Belo Horizonte, Brazil
2020

Edson Bernardes Ferreira Filho

DECENTRALIZED SEGREGATION IN SWARMS OF ROBOTS

Thesis submitted to the Graduate Program in Electrical Engineering of Escola de Engenharia at the Universidade Federal de Minas Gerais, in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

Advisor: Luciano Cunha de Araújo Pimenta

Belo Horizonte, Brazil
2020

Edson Bernardes Ferreira Filho

SEGREGAÇÃO DESCENTRALIZADA EM ENXAMES DE ROBÔS

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito para a obtenção do grau de Doutor em Engenharia Elétrica.

Orientador: Luciano Cunha de Araújo Pimenta

Belo Horizonte, Brasil
2020

To my mom, Ângela.
To my sisters, Nina and Aninha
To my girlfriend, Priscis.

Acknowledgements

This work would not be possible without the support of all my friends and family, who encouraged and motivated me during the whole time of its elaboration.

I would like to thank my mother, who always supported me and respected my decisions. Without her none of this would be possible.

To my father (in memoriam), which would be very proud to read this work.

To my sisters Nina and Aninha for trusting me and for being present in my life.

To my girlfriend Priscis, for her caring and love.

To my niece Marina and my goddaughter Manuela.

To Edu for his help and trust.

To all the other relatives, uncles, aunts, cousins, to whom I have always had an immeasurable friendship.

To all my friends, from Ouro Branco, Ouro Preto and Belo Horizonte for being part of my human formation and making my leisure days more joyful.

To my glorious fraternities, families from Ouro Preto, República Exílio and República Imprevisto.

To all colleagues from MACRO Lab, for the shared moments along the years.

To all professors at PPGEE with whom I had the pleasure of absorbing and sharing knowledge.

I would like to express my gratitude to my advisor, Prof. Luciano Pimenta for all the teachings.

I would also like to thank all the other teachers that I have had in my academic life for having contributed to my academic training.

Finally, I would like to acknowledge CNPq, FINEP and FAPEMIG for their financial support.

Resumo

O uso de enxames robóticos pode ser melhor explorado em um futuro próximo devido a suas vantagens, como a redundância por construção e a tendência de redução de custos na fabricação de robôs. Para esse tipo de sistema se tornar viável, precisamos de algoritmos de navegação eficientes para resolver muitos problemas que ainda estão abertos. Este trabalho aborda o problema de segregação em enxames de robôs. Esse problema consiste em segregar robôs heterogêneos, onde grupos menores de robôs homogêneos são formados a partir de um grupo heterogêneo maior e os grupos menores devem segregar. Para resolver o problema da segregação em um enxame robótico deve-se projetar leis de controle individual para fazer com que todos os robôs de um mesmo tipo se agrupem, mantendo distância de outros grupos.

Uma vantagem desejável em um enxame de robôs é o controle descentralizado, ou seja, cada robô faz uso apenas de informações locais, disponíveis em seus arredores. Todos os controladores propostos neste trabalho têm algum grau de descentralização, ou seja, os robôs não precisam de informações sobre todo o sistema o tempo todo. Propomos diferentes controladores para robôs modelados como duplo integradores (atuados em aceleração). Apresentamos controladores baseados em duas ideias principais diferentes.

A primeira ideia consiste em criar abstrações que representam cada grupo e depois separar as abstrações. Cada abstração é formada usando informações sobre a posição de todos os robôs de um grupo. Depois que as abstrações são criadas, o centro das abstrações é segregado através de forças artificiais provenientes de uma função potencial artificial. Com essa ideia, dois controladores são propostos. O primeiro controlador apresenta alguma descentralização em alguns momentos. Também é apresentado para este controlador um esquema para evitar colisões e uma prova formal de convergência para segregação. No segundo controlador, há mais interesse em usar informações locais. Assim, os estados das abstrações serão obtidos por estimadores descentralizados. As colisões entre robôs não são evitadas e não há prova de convergência, embora o algoritmo seja desenvolvido para facilitar a obtenção de uma possível prova de convergência.

A segunda ideia é o uso de pontos virtuais conectados aos robôs, juntamente com um algoritmo de consenso para orientar o movimento dos robôs. Além disso, propomos uma heurística para alterar os pontos virtuais de forma que cada grupo de robôs permaneça

coeso enquanto segregam dos robôs de outros grupos. Essa ideia também é usada para lidar com um problema de segregação ligeiramente diferente: segregação radial. Nesse problema, todos os robôs devem convergir para um estado onde robôs do mesmo tipo estão posicionados a uma mesma distância em relação a um determinado ponto, enquanto essas distâncias são diferentes para robôs de diferentes tipos.

Para todos os controladores, mostramos simulações e, para alguns, também mostramos resultados experimentais. Simulações e experimentos validam as estratégias que permitem que um enxame de vários robôs heterogêneos se segregue em grupos.

Abstract

The use of robotic swarms might be better exploited in a near future due to its advantages, such as the redundancy by construction and the tendency of cost reduction in robots fabrication. To this type of system become viable, we need efficient navigation algorithms to solve many issues that are still open. This work tackles the problem of segregation in robot swarms. This problem consists in segregating heterogeneous robots, where smaller groups of homogeneous robots are formed from a bigger heterogeneous group and the smaller groups should segregate. To solve the problem of segregation in a robotic swarm one should design individual control laws to make all robots of the same type form clusters while maintaining distance from other groups.

A desirable advantage in a robot swarm is the decentralized control, that is, each robot should make use of only local information, available from its surroundings. All the controllers proposed in this work have some degree of decentralization, that is, robots do not need information about all the elements in the system all the time. We propose different controllers for robots modeled as double integrators (actuated in acceleration). We present controllers based on two different main ideas.

The first idea consists in creating abstractions that represent each group and then separate the abstractions. Each abstraction is formed using information about the position of all robots in a group. After the abstractions are created, the center of the abstractions are segregated through artificial forces that come from an artificial potential function. With this idea two controllers are proposed. The first controller presents some decentralization in some moments. It is also presented for this controller is a scheme to avoid collisions and a formal proof of convergence to segregation. In the second controller, there is more interest in using local information. Thus, the states of the abstractions are obtained by decentralized estimators. Collisions between robots are not avoided and there is no proof of convergence, although the algorithm is developed in order to facilitate future proof of convergence.

The second idea is the use virtual points attached to robots together with a consensus algorithm to guide the movement of the robots. Furthermore, we propose a heuristic to change the virtual points in a way that each group of robots stays together while segregating from robots of other groups. This idea is also used to deal with a slightly

different segregation problem: radial segregation. In this problem, all the robots should converge to a state where robots of the same type are positioned at the same distance from a given point while these distances are different for robots of different types.

For all the controllers we show simulations, and for some, we also show experimental results. Simulations and experiments validate the strategies that allow a swarm of multiple heterogeneous robots to segregate into groups.

Contents

List of Figures

List of Tables

1 Introduction

Robotic swarms are systems formed by numerous relatively simple robots that interact with each other to solve a task that is beyond each robot individual capabilities Dorigo & Sahin (2004).

Robot swarms are commonly inspired in nature; in the way birds fly in formation or in the way fish schools aggregate to accomplish a common goal. In a robot swarm, usually each robot is a simpler one if compared with state-of-the-art robots. That is because applications with robot swarms are focused on what a swarm can do as a group rather than in what each robot can accomplish individually. Furthermore, one of the most important characteristics of a swarm of robots is its intrinsic tolerance to individual robot failures, since the global effect of a few damaged robots is usually attenuated by the large number of robots in the swarm. Those systems are controlled via local control laws and usually have limited communication and sensing capabilities due to hardware restrictions.

Another advantage of robot swarm applications is the possibility of adding and subtracting robots. As individual robot technology improves, the swarm can be increased without having to replace all robots in the swarm, thereby increasing its ability to perform tasks as a whole. It also means that, if needed, robots can be removed to reduce the swarm size.

Some researchers have been able to build real highly scalable swarms. In Klingner et al. (2014) a swarm of 100 real robots is used and in Rubenstein et al. (2014), for the first time, a swarm of 1000 real robots is presented. In Trenkwalder et al. (2017) an operational system is designed for miniature robots with limited on-board resources, similar to the

ones in Klingner et al. (2014) and Rubenstein et al. (2014).

An interesting application of a robot swarm is shown in O'Hara et al. (2014), where a swarm formed of floating boats is connected to form a bridge. In Sartoretti et al. (2016) a swarm of boats is used to manipulate a floating object. In Soleymani et al. (2015) a swarm autonomously construct a protective barrier. Other previously considered applications are perimeter surveillance (Pimenta et al., 2013a), spill detection (Zhang et al., 2013), image capturing for entertainment (Remes et al., 2013), target enclosure (Kubo et al., 2014) and trapping (Zhang et al., 2018), interaction with humans (Recchiuto et al., 2016), (Walker et al., 2014), (Nagi et al., 2014) and manipulation of objects (Marino & Pierri, 2018).

1.1 Motivation

There are many real world applications that can make use of robotic swarms, most of these applications are still in research phase. One can think of a robotic swarm locating victims of a natural disaster in unsafe scenarios, or, in the field of bio-medical engineering, several robots inside a patient checking the functioning of internal organs.

Although most swarm applications are still conceptual, it is possible to see some real world applications emerging nowadays. Two examples are the *Kiva* robots at *Amazon's* warehouse that are used to sort delivery packages (Guizzo, 2008), (Wurman et al., 2008) and Intel's drone light show that are used for entertainment, as in the *2017 Superbowl halftime show* (Burns, 2017). One can think of situations where it would be useful in both applications to have heterogeneous robots. Heterogeneous robot swarms are those formed by different types of robots, these differences can be in the available sensors and/or actuators, locomotion capabilities or even in the role to be played when performing a task. Generally, if the global task can be decomposed into smaller sub-tasks, it is beneficial for the system's performance to have teams of heterogeneous robots (Knudson & Tumer, 2010). In *Amazon's* warehouse robots could have different payloads and Intel's drones could have different light effects. We can imagine that in these scenarios one could wish to assign different tasks to different types of robots. Depending on the task, the robots of the same type might have to exchange information and take decisions together autonomously. Thus, in order to have real autonomous systems and to guarantee good communication performance among the agents it seems to be interesting to endow such heterogeneous swarms with the ability to autonomously segregate, i.e grouping robots of the same type while staying apart from robots of other types.

Swarm robotics is a recent field of study with several interesting problems to be addressed in order to allow its massive use in real world. One of these problems is this so-called swarm segregation. As described in the last paragraph, this problem appears whenever it is necessary to separate a swarm of heterogeneous robots into different groups composed of robots of the same type.

There are two different types of segregation: radial segregation and cluster segregation. To solve these problems one should formulate individual control laws to guide the movements of the robots. Regarding the radial segregation, the controller must guide all the robots to a configuration in which the robots of the same type are positioned at the same distance in relation to a global reference point and robots of different types are positioned at different distances in relation to that point. Furthermore, to solve the problem of cluster segregation, the controller must make all the robots of the same type group together while maintaining segregation from robots of different types.

As an example of a more practical application of heterogeneous swarms and segregation algorithm, one can think of the use of these systems in real world surveillance tasks. We can imagine a swarm composed of heterogeneous drones: some with good surveillance equipment (cameras, sensors, etc.), some with worse sensors but better battery autonomy, some with hardware suitable to communicate with the authorities at longer distances, etc. Imagine that two groups of perpetrators are identified by the robots and have to be monitored by these robots to help the authorities. Suppose the perpetrators are fleeing in different directions: one group fleeing on foot and the other group fleeing by car. Although in this work we do not address the issue of forming groups according to the tasks to be performed, it seems logical to form different groups of robots to follow the perpetrators and help the authorities. For our scenario, drones with better autonomy can follow the perpetrators fleeing by car and drones with better surveillance equipment can follow the perpetrators on foot. The drones with good communication hardware can divide themselves into two groups and each one of these groups can be incorporated in one of those groups previously defined to follow the targets to provide communication. After the assignment of the tasks to each group, it might be interesting to spatially segregate the groups so that the robots of the same group can interact with each other without the interference of nonmember agents in order to autonomously make important decisions related to the task. This could be done manually by the authorities, but if we imagine scenarios with dozens or more robots this would be a slow and tedious task, thus there is the need of an autonomous segregation algorithm such as the one proposed in this work.

Note in the last example that each group of drones is not necessarily formed by drones with the same hardware. In this work we consider robots to be of the same type if they were assigned to the same group regardless of its construction or role in the task. In this work we consider the segregation of robots to be an intermediary step in the execution of a task which is given to the swarm by a high level mission planner. This step might be needed whenever it is beneficial for the task to have a *meeting* of the robots of the same group to make collective decisions before actually executing the task.

We envision automated systems that use heterogeneous swarms of robots to perform multiple independent complex tasks. Those systems could be built by combining three hierarchical layers. In the first layer *Multi-Robot Task Allocation* problems (MRTA)

(Gerkey & Matarić, 2004) are solved to assign robots to groups; in the second layer the groups spatially segregate to have inner group interactions; and in the third layer each group accomplishes the task that better matches the capabilities of the group. This work focuses in the second layer responsible to solve the problem of spatially segregating robots.

There are some advantages in spatially segregating robots in groups before proceeding to do a task. A clear advantage is that if robots are spatially close it is easier for them to communicate with each other, reducing communication interference. Another advantage is that they will form smaller sub-swarms, thus reducing the complexity of treating inter-robot collisions.

Unlike other works in swarm segregation, this work proposes distributed techniques to segregate multiple groups, instead of only two groups, and presents formal proofs that the system converges to segregation as desired.

1.2 Objectives

The general objective of the work is to find ways to solve the problem of segregation in swarms of heterogeneous robots composed of several groups in which it is possible to make the proof of convergence for segregation of this system.

The specific objectives are:

1. Develop techniques to use less information from the point of view of each robot.
2. Couple controllers to avoid collisions that do not interfere with the developed techniques.
3. Test, via simulations, the feasibility of the proposed algorithms.
4. Conduct experiments with real robots using the developed algorithms.

1.3 Contributions

This doctoral thesis is the continuation of the master's thesis of the same author that have originated two articles:

1. Segregating Multiple Groups of Heterogeneous Units in Robot Swarms using Abstractions. Presented at IEEE/RSJ International Conference on Intelligent Robots and Systems 2015 (IROS), September 2015, Hamburg, Germany. (Ferreira Filho & Pimenta, 2015b).
2. Segregação de Enxames de Robôs Heterogêneos do Tipo Integrador Simples em Múltiplos Grupos usando Abstrações. Presented at the *Simpósio Brasileiro de*

Automação Inteligente 2015 (SBAI), October 2015, Natal, Rio Grande do Norte, Brasil. (Ferreira Filho & Pimenta, 2015a).

Both articles use an approach that is based on the use of abstractions (Belta & Kumar, 2003) and artificial potential functions (Olfati-Saber, 2006). Abstractions are virtual entities used to represent a group of robots. The articles differ mainly in the assumed robot model. Collisions were treated only in the second article, in which robots are modeled as first order integrators.

In this thesis we present further contributions in the approach using abstractions such as:

- Collision avoidance integrated in the segregation controller for the second integrator robot model;
- A controller to dissipate high velocities generated mainly by the new collision avoidance mechanism;
- A proof of convergence for the new controller in which groups segregate, avoid collisions and also avoid high velocities;
- New simulation and experiments including experiments with noisy measurements;
- The use of estimators to reduce the exchanged information each robot uses to achieve segregation.

Those new contributions (excluding the estimators) have generated a new journal article:

- Abstraction based approach for segregation in heterogeneous robotic swarms. Published at the Robotics and Autonomous Systems from *Elsevier* in December 2019. (Ferreira Filho & Pimenta, 2019a).

In this document, we dedicate a chapter to present the original controller for robots modeled as double integrators along with the new contributions (Chapter 3).

Furthermore, a new approach to segregate groups of robots have been proposed (Chapter 4). This approach uses virtual points attached to the robots and a consensus algorithm to segregate the groups based on a hierarchy between groups. This new approach have generated two new articles:

1. Decentralized Radial Segregation in Heterogeneous Swarms of Robots. Presented at the IEEE 58th Conference on Decision and Control (CDC), December 2019, Nice, France. (Ferreira Filho & Pimenta, 2019b).
2. Segregation of Heterogeneous Swarms of Robots in Curves. To be presented at the International Conference on Robotics and Automation (ICRA), June 2020, Paris, France. (Ferreira Filho & Pimenta, 2020).

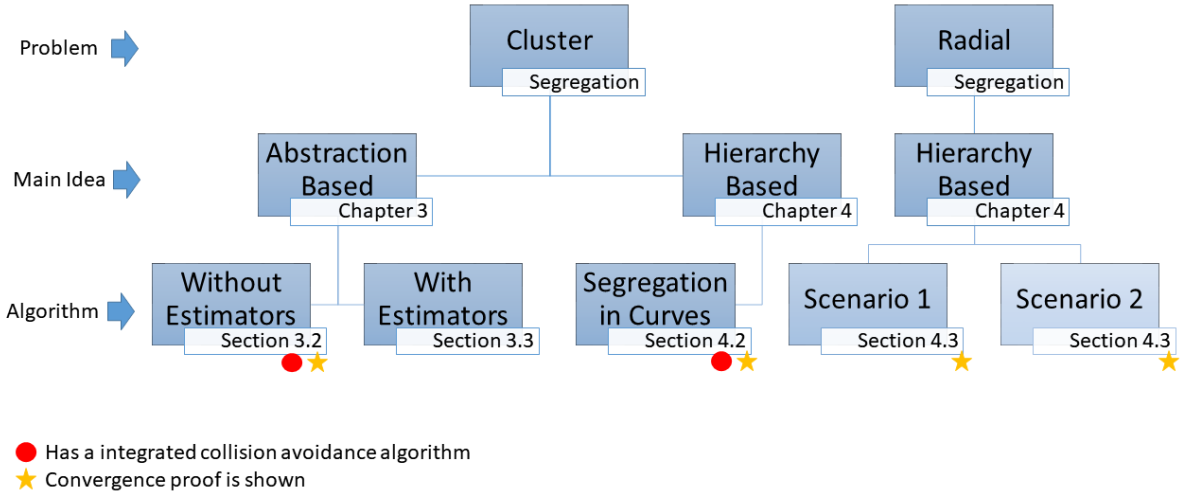


Figure 1.1: Fluxogram of the organization of the methodology.

Both articles are based on similar ideas, however, the first article focuses on the segregation of groups of robots radially and the second article focuses on the segregation of groups of robots in which the groups are constrained to pre-established curves.

1.4 Document Organization

This document is organized as follows. The next Chapter discusses some related work in the field, focusing on works that aim to solve the problem of segregating swarms of robots into multiple groups. In Chapters 3 and 4 the methodology is presented, in which we present in Chapter 3 the approach based on abstractions and in Chapter 4 the approach based on virtual points attached to the robots. Figure 1.1 shows a scheme detailing how the methodology is distributed in Chapters 3 and 4. In Chapter 5, simulations and experiments for both approaches are presented. In Chapter 6 we conclude this work and propose possible avenues for future work. Finally, the Appendix A gives some fundamental principles that were useful in both methodologies of Chapters 3 and 4.

2

Related Work

Given the recent technological advances that made it possible to construct robotic swarms and the vast number of possible applications, we can predict that swarms will play an important role in the society of the future. In this section we discuss some important related works previously developed in swarm robotics.

There are several ways to formulate control laws to coordinate a multi-robotic system to accomplish a task. For example, one could individually guide each robot to a desired position so that the desired coordination is achieved. This is unpractical or even impossible depending on the number and capabilities of robots and operators and the task to be accomplished. A better solution is to endow the robots with autonomous capabilities so that they autonomously make decisions and navigate themselves over the environment. The autonomous control of robots can be subdivided into: offline design of behaviors and embodied evolution (Bredeche et al., 2018). Offline design are those sets of control laws that are embedded into the robots before they are deployed and can be activated when a certain emergent behavior of the swarm is needed to accomplish a task. Embodied evolution deals with robots that can change their control laws *on-the-fly* according to the requirements of the task. In this work we propose offline designed control laws that can be embedded in robots and activated whenever it is required by a task.

The advantage in our design in comparison to a pure embodied evolution approach is that we can formally prove that with our controllers the proposed behavior will emerge, which means it will always happen, what is not usually the case in embodied robotics systems, as it is the case in (Yong & Miikkulainen, 2009), (Nitschke et al., 2012) and

(Trueba & Duro, 2013). Nonetheless, we believe the evolution based approaches might be used together with our methods in real applications in which our work can be used as an intermediary step between the high-level decision on the composition of the groups and the low-level execution of the tasks by each group.

Once the members of the groups are properly chosen it might be necessary that the robots of the groups get together before the execution of the task to communicate and take decisions regarding this execution. In this case, it makes sense to activate a segregative behavior so that the members of the groups can exchange information without the interference of non-members agents. Thus, our controllers can be used to guarantee that the groups will be segregated in the way required by the application before proceeding to actually do a given task.

Next, we review some important work in autonomous control, focusing on methods to navigate swarms of robots and providing some examples to illustrate possible applications with such swarms.

2.1 Swarms of Robots

Navigation algorithms can be divided from the point of view of the information needed for the controller: centralized control and decentralized control. In the centralized approach, there is a unit responsible for sending control actions to each robot obtained with the knowledge of global information. The great advantage of this type of system is that it is easier to solve navigation problems, because the controller unit has the information of all robots of the system. In decentralized control, each robot generates its control action from local information only. This can be a great advantage, as communication and sensing systems usually have limited range. There are also proposals that combine some aspects of each method.

Modeling the swarm navigation problem in individual navigation problems with centralized solution is usually not very efficient, because one should plan the movement of a very large system, which requires very elaborate algorithms with high computational cost. In addition, the size of the problem grows as robots are added. In Choset et al. (2005) and LaValle (2006) a review of these navigation algorithms is presented for a single robot, but they do not directly address robot swarm problems. There are some works that address robot swarm problems using different methods to formulate control laws that make robots navigate according to the established problem.

We further divide the navigation methods into two types: with homogeneous robots and with heterogeneous robots. Heterogeneous robot swarms are those where there are robots that are different, either in its construction or in the role each robot will play in pursuit of the swarm's goal. Swarms with homogeneous robots are those in which their robots have no distinction.

2.1.1 Swarms with Homogeneous Robots

Behavior-based models were the first types used in an attempt to control swarms virtually in the context of computer graphics (Reynolds, 1987). These models define behaviors that must be activated under predetermined conditions. In Reynolds (1987) the goal was to simulate animal movements in "flocks" for computer graphics purposes. A behavior-based model is used in which each virtual animal has three possible behaviors: avoiding collisions with surrounding animals, trying to stay close to the surrounding animals, and trying to achieve the same speed as the surrounding animals. Each of these behaviors is turned on or off as needed, giving rise to a global "flock" behavior. Balch & Arkin (1998) also uses a behavior-based model, but to control a group of robots, causing them to maintain desired formations, which are based on those used by the US Army platoons on the battlefield.

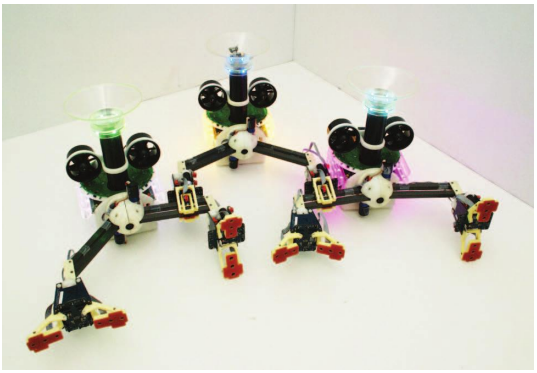
Alternatively, some works are based on artificial forces derived from potential functions. These methods that use potential functions usually make use of the gradient of these functions to guide the robot to the goal. Thus, the targets of each robot act as attractive forces and obstacles as repulsive forces, and the other robots in the swarm are modeled as obstacles from the point of view of the robot of interest. In (Tanner et al., 2005) potential function-based forces are used to avoid collisions between non-holonomic robots as they cluster together. In Chaimowicz et al. (2005) and Hsieh et al. (2008) such artificial forces are used to make robots spread out in complex curves (in two dimensions) while maintaining a separation between robots. Some of the approaches proposed in this work also makes use of artificial forces, and they will be further detailed in the section 3.1.3.

There are other approaches, such as using smoothed particle hydrodynamic properties to control robots in a distributed way, making them "flow" through the environment, as in Perkinson & Shafai (2005), Pimenta et al. (2008b) and Pimenta et al. (2013b). In addition, some properties of the robot swarm can be used to create a structure with smaller dimension than the dimension of the global swarm configuration space. In Belta & Kumar (2003) this strategy is also used, creating virtual structures called abstractions. This work also make use of such structures, that will be detailed in Section 3.1.1. By creating these structures, one can control several robots as a whole, which facilitates navigation problems for multiple robots, but in contrast you have less control over the behavior of each robot separately. In Santos & Chairmowicz (2011) and Santos & Chaimowicz (2011), abstractions of this type are used in the context of robot swarms and in Chaimowicz & Kumar (2004) unmanned aerial vehicles (UAVs) are used to gather the abstraction states and then send commands to swarms of robots as a whole for split and regroup tasks.

2.1.2 Swarms with Heterogeneous Robots

The use of heterogeneous robots has recently attracted the attention of researchers in robotics. There are some advantages to systems of this type, for example, one can design a

(a) Simple mobile robots.



(b) Robots with grippers.

(c) Aerial robots with cameras.

Figure 2.1: Project *Swarmanoid*.

system with some robots having only one type of actuator and sensor, while other robots have different actuators and sensors. Such a system is highly scalable because you expand the system by only adding the robots that are needed. It also has high fault tolerance due to the redundancy of robots of the same type. An example of such a system is the *Swarmanoid* (Dorigo et al., 2013) project. In this project there are three groups of robots. Some have cameras that act as system sensors (Figure 2.1c), some robots have grippers as actuators (Figure 2.1b) and there are also robots that have the function of moving the robots with grippers robots, (Figure 2.1a).

There are other works that address various problems in the area of heterogeneous robot swarms. In (Prorok et al., 2017), the authors propose a metric to evaluate the impact of the heterogeneity of robots on a swarm when performing a task where each robot can only perform part of it.

In Pimenta et al. (2008a) and Kantaros et al. (2015) control laws are proposed to address the problem of space coverage with heterogeneous robots. Robots are different in that their sensors have different detection capabilities. These works are interesting for some applications, for instance, using both air and ground robots to coverage tasks.

In the use of heterogeneous robots an important skill for the system that can be useful

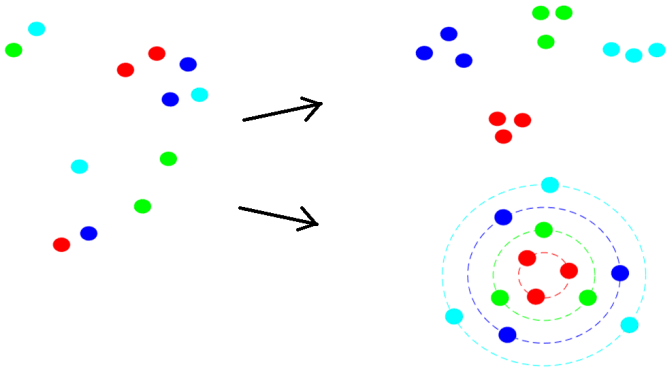


Figure 2.2: Segregation example. Left: initial configuration of robots. Top right: cluster segregation. Bottom right: radial segregation.

in many applications is the ability to autonomously segregate. This is the ability to form groups, each containing only robots of the same type. In order to provide this capability to the system, individual control laws must be designed that make robots of the same type form clusters, or, radial rings. Figure 2.2 shows an example of an initial configuration of robots (left) and two segregated systems: system segregated in clusters (top right) and system radially segregated (bottom right). Note that robots of the same type are represented by the same color.

In Kumar et al. (2010) and Santos et al. (2014) heterogeneous robots are also used and the problem of autonomously segregating swarms of robots in clusters is addressed, as will be better described in Section 2.2.

Application with swarms of robots can be more interesting when the robots do not need the knowledge of all system's information, i.e. global information. Next, we present some works in which the control is performed in a decentralized manner, that is, each robot individually does not have information about the whole system.

2.1.3 With Decentralized Control

In this section, we will cover some work in the robot swarm literature where the control is done in a decentralized manner. In this work we assume that decentralized control is the one in which robots individually do not have information about the whole system and there is no central unit controlling the whole system.

In Tanner et al. (2005) the problem of flocking is studied, where is desirable that all the robots achieve the same speed, this is done by exchanging information between robots. The problem of flocking is also studied in Li & Xi (2008) and Erfianto et al. (2016). In those works, the authors are interested in studying the flocking behaviour while maintaining robots close so that they are always able to communicate with each other, i.e. maintaining the connectivity. In Erfianto et al. (2016) flocking is performed, maintaining connectivity and also avoiding obstacles, as shown in the Figure 2.3. In this figure the group of robots

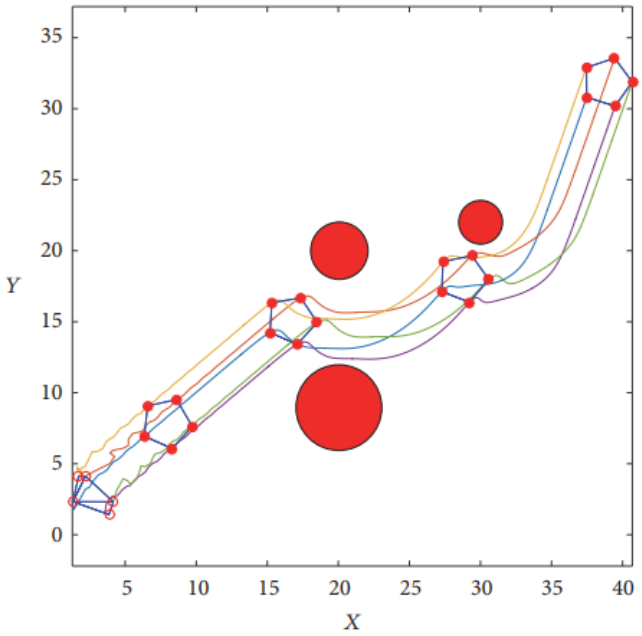


Figure 2.3: Flocking with obstacle avoidance (Erfianto et al., 2016).

reaches a flocking formation, maintaining connectivity while dodging various obstacles. Note that the figure shows which robots are connected to each other.

There are works that aim only to maintain or modify the connectivity graph between robots. In De Gennaro & Jadbabaie (2006) an algorithm that increases the connectivity of a robot swarm is proposed and Dimarogonas & Johansson (2008) proposes an algorithm to maintain connectivity when inputs are limited. Both algorithms are decentralized, each robot only have communication up to a fixed distance defined *a priori*. This simulates the operating radius of communication equipment, which is usually limited.

In Bezzo et al. (2014), communication networks with a heterogeneous robot system are presented, in which aerial and terrestrial robots are used as signal routers and these robots move in an environment while keeping the communication network connected. Recently, Maeda et al. (2017) have proposed a decentralized navigation method for a swarm of heterogeneous robots with limited vision capabilities. A leader-following algorithm has been proposed that allows the swarm to maintain its connectivity while navigating with robots differing in their sensor range, vision capabilities and maximum velocities and accelerations.

The works of Freeman et al. (2006), Niccolini et al. (2008), Antonelli et al. (2013) and Morbidi et al. (2011) use estimators simultaneously with robot control. Freeman et al. (2006) proposes a decentralized control strategy in which each robot estimates the variables representing the whole system using a consensus strategy and using the idea of an abstraction, as previously described, and controlling the abstraction rather than directly controlling the robots themselves. Much of the work that addresses control in a decentralized manner applies consensus algorithms. In the context of cooperative control,

consensus can be defined as an agreement between group members for a common goal. The variable defined as information state is used to model the collective view of the common goal and can be used to represent some idea of coordination variable, such as location of agent formation or position/time of *rendezvous* (Ordoñez et al., 2012).

In Niccolini et al. (2008) a very similar proposal is shown, but with the addition of a scheme to avoid collisions between robots and obstacles. The proposal in Morbidi et al. (2011) is similar to Freeman et al. (2006) and Niccolini et al. (2008) but instead of moving to a desired position robots should now distribute themselves in order to perform monitoring tasks. In Antonelli et al. (2013) the proposal is also similar, however, a state observer is used to estimate the states of the robot group and the algorithm control its centroid. In Antonelli et al. (2013), both time invariant topologies and time variants are considered, i.e. the graph of connectivity between robots may or may not vary over time.

The next section describes some works that deals with the segregation problem in robot swarms, both in a centralized manner and with some decentralization.

2.2 Segregation in Swarms of Robots

For robots with different characteristics to be used to perform their tasks, it may be necessary to separate the swarm into groups containing only robots of the same type, so that these robots can perform the task proposed for their group. One of the first works in this topic was developed by Groß et al. (2009).

Groß et al. (2009) has developed an algorithm that is able to segregate robots based on the *Brazilian nut effect*. When a container containing a large sphere and numerous small spheres is shaken, the larger sphere rises even when it is denser than the other spheres. Similarly, a mixture of different sized particles will segregate by size when agitated. This effect is called *Brazilian nut effect* Rosato et al. (1987). Figure 2.4 shows a sequence of frames illustrating this effect.

In the work of Groß et al. (2009), although robots have the same size, the behavior of different sized particles is simulated. In Chen et al. (2012) this approach is successfully implemented in *e-puck* robots. The proposal is based on three behaviors: random movements that simulate container shaking, attraction to the center of gravity and repulsion to other robots. Thus, each robot needs a global information, which is the center of gravity of the system. In the *e-pucks* implementation, to simulate this gravitational point of attraction, a light bulb is used as the radiation source. Figure 2.5 shows one frame of the experiments in Chen et al. (2012), where one can see the radiation source used. The need for this global information is a disadvantage, as in many applications it is impracticable for each robot to obtain this information. Also, there is no proof of the stability of the proposed controller, which motivates the development of controllers that converge to segregation regardless of the number of robots and groups.

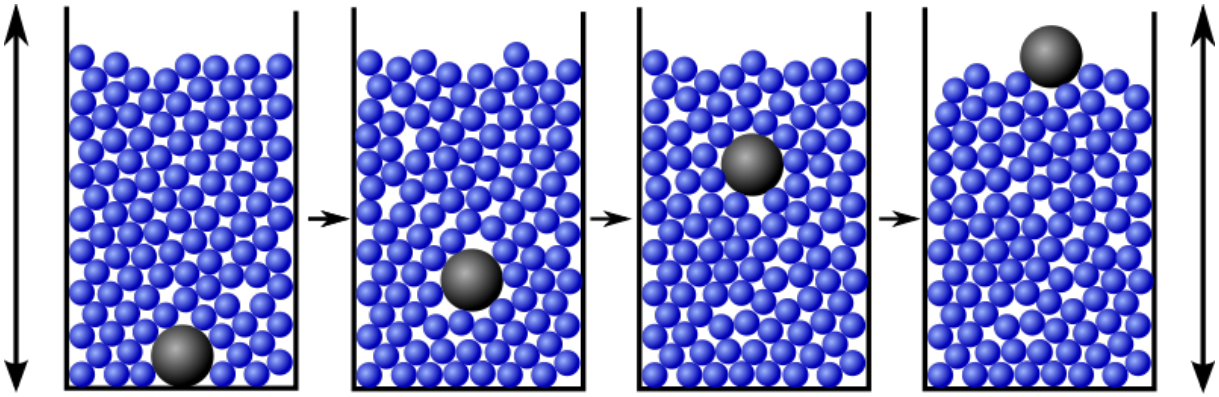


Figure 2.4: *Brazilian nut effect (Wikimedia, 2010)*

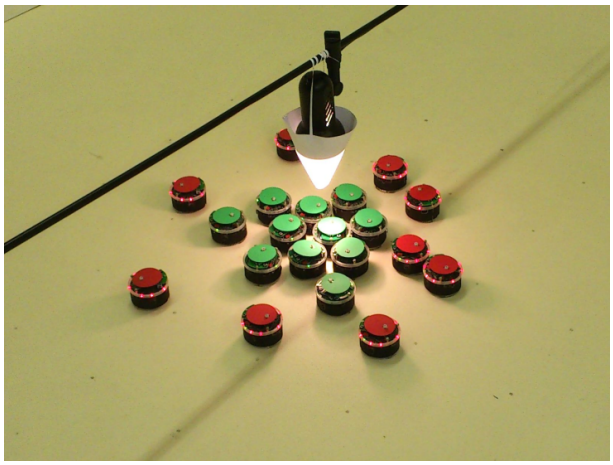


Figure 2.5: Radial segregation using the *Brazilian Nut Effect* (Chen et al., 2012).

We now review relevant works found in literature that intent to solve the problem of segregating swarms of robots. We divide the works into two categories: in Section 2.2.1 we review works that intent on segregating groups into clusters, that is, robots of the same group must be close while apart from robots of other groups and in Section 2.2.2 we review the works found in literature that aim to solve the problem of radial segregation, that is, robots of the same group must converge to the same distance in relation to a reference point while this distance must be different between different groups.

2.2.1 Segregation in Clusters

This section reviews the works that attempts to solve the problem of autonomous segregation of heterogeneous robot swarms in clusters. The most relevant studies in solving this problem are, Kumar et al. (2010), Santos et al. (2014), Edwards et al. (2016), Inácio et al. (2019) and Mitrano et al. (2019). In addition to works previously developed in the context of the master thesis of this author: Ferreira Filho & Pimenta (2015a), Ferreira Filho & Pimenta (2015b) and the algorithm that will be showed in Section 4.2.

In Edwards et al. (2016) an algorithm is developed for robots modeled as single

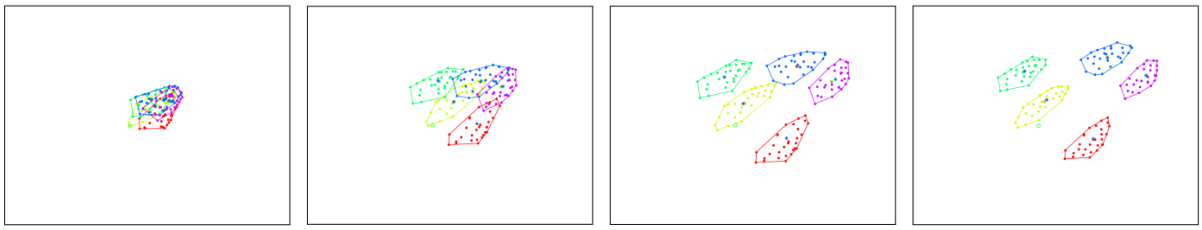


Figure 2.6: Convex Optimization Based Robot Segregation. From left to right: 5 groups of robots reaching segregation (Edwards et al., 2016).

integrators, assuming access to global information. The algorithm is based on convex optimization, calculating the convex hulls of each group of robots and then segregating the groups until there is no intersection between the convex hulls. Formal convergence proof is shown not considering collisions between robots although a collision avoidance scheme is shown. Figure 2.6 shows a simulation present in Edwards et al. (2016) in which it is possible to note the convex hulls of each group and also the segregated groups.

Recently, in Inácio et al. (2019), a decentralized algorithm is proposed considering also single integrator robots. This approach uses a combination of the Particle Swarm Optimization method (PSO) (Eberhart & Kennedy, 1995) with the ORCA (Van Den Berg & Manocha, 2011) algorithm and it is interesting in the sense of not requiring global information to reach segregation. According to Inácio et al. (2019), the convergence cannot be proven but if one consider infinite time, the algorithm is probabilistically complete.

Also recently, a reactive minimalistic approach was proposed (Mitrano et al., 2019). In this work, each robot is equipped only with a ternary sensor capable of detecting the presence of a single nearby robot, and, if that robot is present, whether it belongs to the same group as the sensing robot. Although in this work segregation may occur using minimalistic information, there is no mathematical guarantee of such fact.

This work is best related to other works that address the segregation problem in robot swarms assuming the double integrator robot model. Furthermore, we are interested in works that present a convergence proof to segregation, such as: Kumar et al. (2010) and Santos et al. (2014). Both are based on the gradient of potential functions.

In Kumar et al. (2010) an artificial potential function based on the differential adhesion model of biological cells is used. This was the first work to achieve distributed segregation to appear in the literature showing a convergence proof. Asymptotic convergence proof for segregation and stability analysis for robot swarms with only two groups are shown. The work only deals with two groups of robots, using potential function composed of three parts: potential between robots of one group, potential between robots of the other group, and potential between robots of different groups. Thus, there is a parameter that assumes different values for each part of this potential function, and for the potential between robots of different groups this parameter must be greater than for other cases, which may have equal parameters. In the work of Kumar et al. (2010) a simulation is presented with

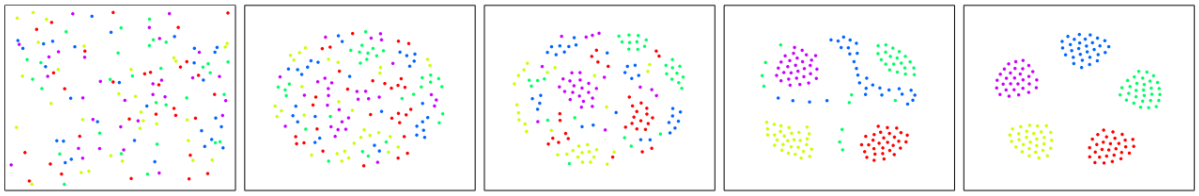


Figure 2.7: Robot segregation using potential functions. From left to right: 5 groups of robots reaching segregation (Santos et al., 2014).

two groups reaching segregation. In addition, data from just over 100 simulations that also achieved segregation according to the proposed definition are shown. This work does not consider collisions between robots.

The potential function used in Santos et al. (2014) is very similar to the potential function of Kumar et al. (2010) but has the addition of one term that helps in the segregative behavior for the system with more than two groups. In Santos et al. (2014), three simulations with 150 robots in total divided into groups of 5, 10 and 15 robots are shown. The mean and standard deviation of 100 other simulations, also with groups of 5, 10 and 15 robots are also shown. In all simulations the groups segregate successfully. The proof of system stability with the application of the proposed controller is shown, but no proof of convergence to segregation is presented, which means that there is a possibility that the system converges to a stable situation, in which the groups are not segregated. According to Santos et al. (2014) these situations can occur when groups are very unbalanced, i.e. there are some groups with few robots while there are other groups with many robots. Figure 2.7 shows snapshots of a simulation with 150 robots equally divided into 5 groups in which it is possible to see the segregation of the groups taking place. Note that in Edwards et al. (2016) (Figure 2.6) collisions are not being addressed and in Santos et al. (2014) (Figure 2.7) collisions are avoided.

In both Kumar et al. (2010) and Santos et al. (2014) all robots need information from all the other robots of the system at all times, this is due to the fact that in both potential functions the information of all robots is always used even when the robots are already segregated or too far apart, whether in the same group or not.

The table 2.1 shows a comparison between the algorithms for segregating robot swarms into clusters. Note that there is no algorithm with a formal proof of convergence for multiple groups of double integrator robots using only local information. This is one of the main contributions of our approach (presented in Ferreira Filho & Pimenta (2020) and in Section 4.2 of this document) in which robots must have the knowledge of a common reference point or there must exist an underlying connected topology.

Table 2.1: Comparison between algorithms for segregation in robot swarms in clusters.

Work	**Robot Dynamics**	**Groups**	**Approach**	**Convergence Proof**	**Information**	**Experiments**
Kumar et al. (2010)	Double Integrator	Two	Artificial Potential Forces	Yes [1]	Global	No
Santos et al. (2014)	Double Integrator	Multiple	Artificial Potential Forces	Stability	Global	No
Ferreira Filho & Pimenta (2015a)	Single Integrator	Multiple	Artificial Potential Forces and Abstractions	Yes	Mostly Global [2]	9 *e-pucks*
Ferreira Filho & Pimenta (2015b)	Double Integrator	Multiple	Artificial Potential Forces and Abstractions	Yes	Mostly Global [2]	13 *GRITS Bot X* [3]
Edwards et al. (2016)	Single Integrator	Multiple	Convex optimization	Yes [1]	Global	8 *e-pucks*
Inácio et al. (2019)	Single Integrator	Multiple	PSO and ORCA	Probabilistic	Local	No
Mitrano et al. (2019)	Differential Drive	Multiple	Reactive Minimalistic	No	Local	No
Ferreira Filho & Pimenta (2020)	Double Integrator	Multiple	Consensus Based	Yes	Mostly Local [4]	11 *GRITS Bot X*

[1] Without collision Avoidance
[2] In some situations no global information is required.
[3] In Section 5.1.1 of this document an experiment with this same approach and up to 13 GRITS Bot X robots is shown.
[4] A common reference point or an underlying connected topology is required.

2.2.2 Radial Segregation

A system is said to be radially segregated if all the robots of the same group are positioned at the same distance from a reference point while robots from different groups are positioned at different distances from it.

In Groß et al. (2009) and Chen et al. (2012) the main idea is to segregate robots in clusters inspired in the *Brazilian Nut Effect*, however, they also show experiments where radial segregation occurs, as it can be seen in Figure 2.5.

In Wilson et al. (2004), the intention is to segregate objects using robots and not to directly segregate robots, nonetheless, it has contributions providing performance metrics for radial structures. Three mechanisms for sorting objects radially using robots are presented briefly, in which two mechanisms are based on the fact that under centripetal forces particles segregate according to their size. In Groß et al. (2009) the same fact was used.

Szwaykowska et al. (2014) investigate a swarm of robots in which robots have different dynamics. The swarm is divided into two groups, one group having only robots that are poorly maneuverable and the other group having only robots that can be accelerated faster than the robots in the first group. It is observed that segregative patterns emerge naturally with the proposed dynamics. This is the only work dealing with segregation found in the literature in which swarm heterogeneity is due necessarily to the difference in robot dynamics. Other works assume a broader concept in which the heterogeneity is not explicitly defined. A simulation is shown in which radial segregation occurs, but no proof of convergence is presented. Segregation is not observed when all robots have the same dynamics.

In this work, we propose a controller for robots with double integrator dynamics to radially segregate heterogeneous swarms. The methodology for this approach is shown in Section 4.3 and the simulations and experiments in Section 5.2.2. The same approach has originated a paper: Ferreira Filho & Pimenta (2019b). Robots do not have the knowledge of the number of robots nor the number of groups in the system. Different from other works we present a method with a proof of convergence. Thus, we can say that by using our controller the system will always reach a segregated state. With our controller, robots do not need information from all the other robots of the system to achieve segregation although robots must have the knowledge of a common reference point or there must be an underlying fixed communication topology.

Table 2.2: Comparison between algorithms for radial segregation in robot swarms.

Work	Robot Dynamics	Groups	Approach	Convergence Proof	Information	Experiments
Groß et al. (2009)	Differential drive	Multiple	Based on the *Brazilian Nut Effect*	No	Mostly Local [5]	No
Chen et al. (2012)	Differential drive	Multiple	Based on the *Brazilian Nut Effect*	No	Mostly Local [5]	20 *e-pucks*
Szwaykowska et al. (2014)	Double Integ. with Delay	Two	Dynamics with delays	No	Global	No
Ferreira Filho & Pimenta (2019b)	Double Integrator	Multiple	Consensus Based	Convergence	Mostly Local [6]	15 GRITS Bot X

[5] A common reference point to all robots is required.
[6] A common reference point or a underlying connected topology is required.

3

Abstraction Based Segregation

In this chapter, two algorithms are presented. Both algorithms are based on the same main idea, to use abstractions to represent the groups and use a potential function to separate the groups represented by the abstractions. The fundamental background for both algorithms is presented in Section 3.1. Also in Section 3.1 we formally define the problem as it will be considered in this chapter.

The first approach is an algorithm in which collisions are avoided and the use of local information occur only in certain situations. A proof of convergence is shown and the algorithm will be described in Section 3.2. The simulations and experiments related to this algorithm will be described in Section 5.1.1.

In the second algorithm, we have more interest in the use of local information. Thus, the states of the abstractions will be obtained by decentralized estimators. Collisions between robots are not avoided and there is no proof of convergence, although the algorithm is being developed in order to facilitate that a possible proof of convergence is obtained. This algorithm is described in Section 3.3 and its simulations can be found in Section 5.1.2.

3.1 Background

Consider $\sum_{j=1}^{M} N_j$ holonomic mobile robots moving freely in a $2D$ plane with position of each robot given by the vector

$$\boldsymbol{q_j^k} = \begin{bmatrix} x_j^k \\ y_j^k \end{bmatrix}, \quad k = 1, 2, ..., N_j. \tag{3.1}$$

The index j indicates to which group a robot belongs, $j = 1, 2, ..., M$, N_j indicates the number of robots in group j, and $k = 1, 2, ..., N_j$ are the robots in group j.

Consider also robots modeled as double integrators:

$$\dot{\boldsymbol{q}}_{\boldsymbol{j}}^{\boldsymbol{k}} = \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{k}}, \quad \dot{\boldsymbol{v}}_{\boldsymbol{j}}^{\boldsymbol{k}} = \boldsymbol{u}_{\boldsymbol{j}}^{\boldsymbol{k}}, \quad k = 1, 2, ...N_j, \tag{3.2}$$

in which $\boldsymbol{u}_{\boldsymbol{j}}^{\boldsymbol{k}}$ is the control input of the k-th robot of group j. Throughout this paper, superscript indexes k and l are used as robot indexes and subscripts i and j are used as indexes of groups (abstractions). This distinction is important mainly when several groups are being addressed in the collision avoidance algorithm in Section 3.2.1.

Our goal is to devise control laws that allow this heterogeneous group of robots to segregate into the M homogeneous groups, as will be formally stated in Section 3.1.2.

Remark 1. *For the sake of simplicity in the presentation of the proposed methodology, in this chapter, we consider mobile robots in a $2D$ plane. However, it is straight forward to apply the approach in $3D$ by considering an additional z component in the position vectors, as long as the underlying robot dynamics are those of a point mass robot. In Section 5.1.1 we present simulations in $3D$ environments to show that the method can be extended to higher dimensions.*

Remark 2. *All the algorithms of this work are developed for robots modeled as double integrators. However, it is possible to apply the same algorithms to other robot models. In the Appendix A.1.2 we show how to apply the algorithms shown in this work to other robot models.*

3.1.1 Abstractions

Some approaches commonly used to control robot swarms consider the swarm, or part of it, as a single virtual entity. Usually in this type of approach it is easier to control the swarm, even though one might have less control over individual robots. In this work an approach of this type is used and the entity representing the group is called abstraction.

In Belta & Kumar (2003), numerous robots are controlled by means of an abstraction, mapping the configuration space into a space with lower dimension. We define, in this

section, a circular abstraction as proposed in Belta & Kumar (2003) to represent each group of robots.

Each abstraction has state variables associated with the mean of the positions of the robots and the covariance matrix of the positions of the robots belonging to the same group. The covariance matrix related variable quantifies the dispersion of a group.

The mean of the positions of each group is given by

$$\boldsymbol{\mu_j} = \frac{1}{N_j} \sum_{k=1}^{N_j} \boldsymbol{q_j^k}. \tag{3.3}$$

For robots in the $2D$ plane, $\boldsymbol{\mu_j}$ is given by:

$$\boldsymbol{\mu_j} = \begin{bmatrix} \mu_j^x \\ \mu_j^y \end{bmatrix}, \tag{3.4}$$

in which μ_j^x and μ_j^y are the components x and y of $\boldsymbol{\mu_j}$, respectively.

Each abstraction is made symmetric, defined by a circle. Another variable associated with each abstraction is given by

$$\sigma_j = \frac{1}{N_j} \sum_{k=1}^{N_j} [(x_j^k - \mu_j^x)^2 + (y_j^k - \mu_j^y)^2], \tag{3.5}$$

and reflects the dispersion of robots in relation to the mean of the group.

The configuration space of a system with N_j planar robots is given by $Q \equiv \mathbb{R}^{2N_j}$ Choset et al. (2005). Variables of each abstraction define the map:

$$\phi_j : Q \to G \subset \mathbb{R}^3, \qquad \phi_j = \begin{bmatrix} \mu_j^x \\ \mu_j^y \\ \sigma_j \end{bmatrix}, \tag{3.6}$$

in which the dimension of the manifold G is not dependent of the quantity of robots in the group.

The variables of the abstraction implicitly define a circle C_{ϕ_j} that contains all robots of the group. The center of this circle is the mean of the positions of the robots and the radius is given by $\sqrt{N_j \sigma_j}$:

$$C_{\phi_j} = B(\boldsymbol{\mu_j}, \sqrt{N_j \sigma_j}), \tag{3.7}$$

where $B(a, b)$ defines a ball centered in a with radius b.

Note that

$$\left\| \boldsymbol{q_j^k} - \boldsymbol{\mu_j} \right\|^2 \leq \sum_{k=1}^{N_j} \left\| \boldsymbol{q_j^k} - \boldsymbol{\mu_j} \right\|^2 = N_j \sigma_j, \tag{3.8}$$

which implies

$$\left\| \boldsymbol{q_j^k} - \boldsymbol{\mu_j} \right\| \leq \sqrt{N_j \sigma_j}. \tag{3.9}$$

Equation (3.9) means that, by construction, all the robots associated with ϕ_j always remain inside the abstraction C_{ϕ_j}.

Other shapes of abstractions are possible, such as ellipses and squares in the $2D$ plane, as presented in Belta & Kumar (2004). In this work, a simple circular abstraction is used because with this abstraction it is guaranteed that all robots will always stay inside it. This fact will help in the convergence proof in Section 3.2. There exists other geometrical forms that can have this property, we have chosen the circle because it can be defined with only three variables and the circle can be defined to comprise all the robots with tighter fits than other geometrical forms.

3.1.2 Problem Definition

Having defined the abstractions and robot models, we can formally present the problem of segregating robots into clusters.

Problem definition: *Consider $\sum_{j=1}^{M} N_j$ robots of M types with dynamic model given by (3.2), devise individual control laws that enforce the robots to move in a way so that each abstraction associated with the robots converge to a state in which:*

$$\bigcap_{j=\{1,\ldots,M\}} C_{\phi_j} = \varnothing. \tag{3.10}$$

When segregated, all robots of the same type will stay together while separated from robots of other types.

Figure 3.1 shows a segregated system according to the definition above, in this Figure robots of the same group have the same color and are inside the same abstraction. Lines connect the centers of the abstractions. Note that, by definition, robots of a group always remain inside the abstraction of this group.

3.1.3 Potential Function

This section reviews a potential function proposed by Olfati-Saber (2006). The control law to be derived in this work is based on the artificial forces derived from this potential function. The function has an interesting property of having a *finite cut-off*. It means that there will not be any virtual force between agents if they are very far from each other. This helps giving a local property to the segregation algorithm under some conditions, i.e. robots may not use global information all the time. The potential function that was used in both Kumar et al. (2010) and Santos et al. (2014) to achieve segregation does not have this property.

In the original context, the potential function proposed in Olfati-Saber (2006) was applied directly in the individual robot controllers to achieve a flocking behavior. In this

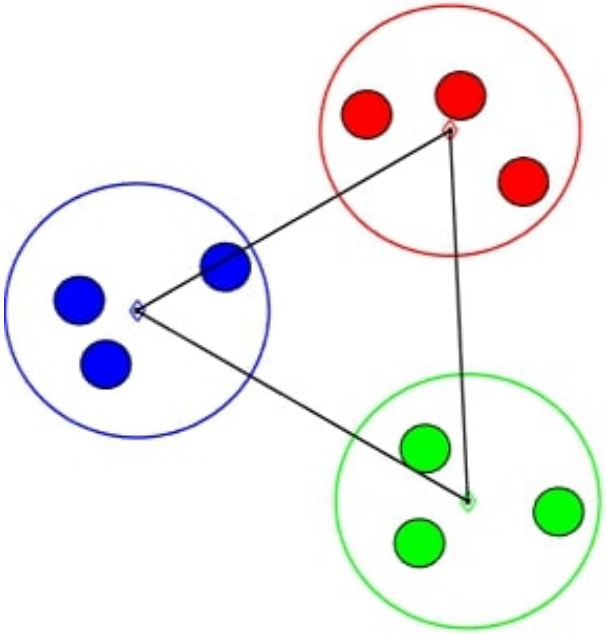


Figure 3.1: Circular robots divided in 3 groups of 3 robots.

work this function will be applied to control the center of each abstraction. Thus, we consider the multi-agent system in which the abstractions are the agents.

Before presenting the potential function, the definition of a non-negative mapping is required so that the potential function is differentiable everywhere. This mapping is called σ-norm (Olfati-Saber, 2006):

$$\|z\|_\sigma = \frac{1}{\epsilon}\left[\sqrt{1+\epsilon\|z\|^2}-1\right], \tag{3.11}$$

in which ϵ is a parameter that acts as a gain and is fixed throughout this work and z is the variable being mapped. Considering the vector of abstraction mean positions, μ, we can now present the artificial potential function:

$$V(\boldsymbol{\mu}) = \frac{1}{2}\sum_j \sum_{i\neq j} \psi_\alpha(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma), \tag{3.12}$$

in which

$$\psi_\alpha(r) = \int_{d_\alpha}^{r} \gamma_\alpha(s)ds, \qquad s, y \in \mathbb{R}; \tag{3.13}$$

and $\gamma_\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ is given by

$$\gamma_\alpha(s) = \rho_h(s/r_\alpha)\frac{c(s-d_\alpha)}{\sqrt{1+(s-d_\alpha)^2}}, \tag{3.14}$$

in which c is a positive constant, d_α is a positive constant defining the global minimum of ψ_α with $d_\alpha = \|d\|_\sigma$.

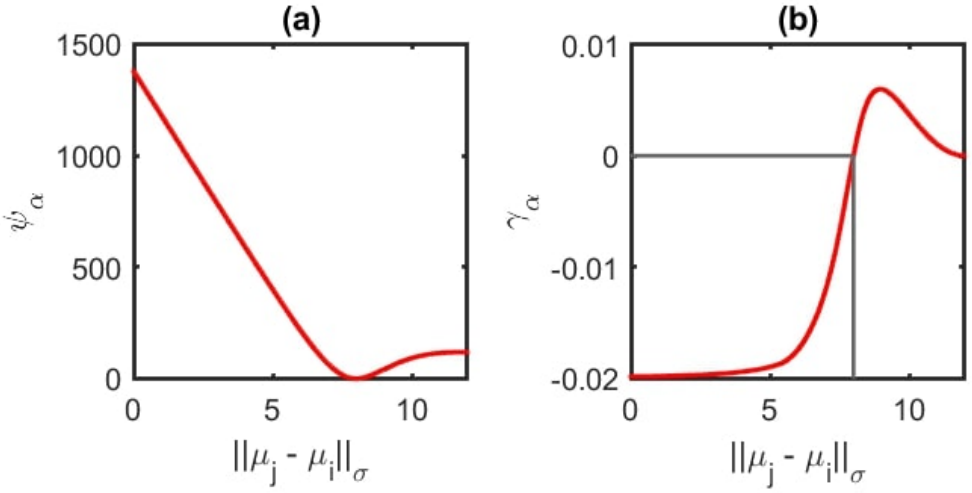


Figure 3.2: Parameters: $c = 0.02$, $h = 0.4$, $d_\alpha = 8$, $r_\alpha = 1.5d_\alpha$. (a) Artificial Potential between two agents versus the distance between them. (b) Artificial force between two agents based on the gradient versus the distance between them.

Function $\rho_h(z)$ is called a bump function and varies smoothly between 0 and 1 (Olfati-Saber, 2006):

$$\rho_h(s/r_\alpha) = \begin{cases} 1, & s/r_\alpha \in [0, h) \\ \frac{1}{2}\left[1 + \cos(\pi(\frac{(s/r_\alpha)-h}{1-h}))\right] & s/r_\alpha \in [h, 1] \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

where $h \in (0, 1)$.

The parameter r_α defines the *finite cut-off* $r_\alpha = \|r\|_\sigma$. It means that if two agents are in a distance greater than r_α from each other, there will not be any repulsive or attractive artificial force between them.

Using the artificial potential function, the following forces can be obtained:

$$\boldsymbol{F_i} = \underbrace{\sum_{j \in B_i} \gamma_\alpha(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma)\boldsymbol{n}_{ij}}_{-\nabla_{\mu_i} V(\boldsymbol{\mu})} + \underbrace{\sum_{j \in B_i} \rho_h(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma / r_\alpha)(\dot{\boldsymbol{\mu}}_j - \dot{\boldsymbol{\mu}}_i)}_{\text{velocity consensus}} \tag{3.16}$$

in which,

$$\boldsymbol{n}_{ij} = (\boldsymbol{\mu_j} - \boldsymbol{\mu_i})/\sqrt{(1 + \epsilon \|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|^2)}. \tag{3.17}$$

In (3.16), the term indicated as velocity consensus is used so that all agents have the same velocity when the artificial force is close to zero. This can be seen as a damping term. B_i is the set of neighboring abstractions of abstraction i. Those neighbors are those groups in which the center of their abstractions are in a distance smaller than r_α from the center of abstraction i.

Figure 3.2 shows an example of (3.13) and (3.14). Parameter c acts as a gain, while parameter h modifies the smoothness of the force. For any two agents, the force of

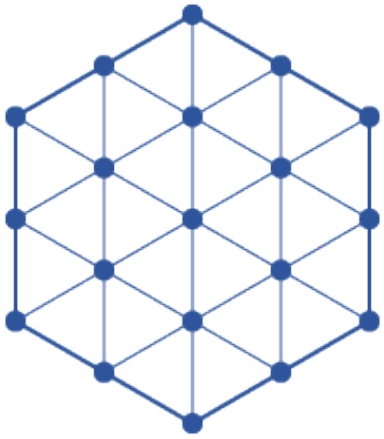


Figure 3.3: A α-lattice example.

repulsion/attraction will fade when they are exactly at the desired distance d_α and when the distance between them is greater than r_α.

An important Lemma regarding the function in (3.12) is given below:

Lemma 1. *If $d < r < \sqrt{3}d$, all local minimum of $V(\boldsymbol{\mu})$ implies in the formation of an α-lattice (Olfati-Saber, 2006).*

These structures (α-lattices) are lattice shaped in which each vertex is at the same distance d_α of each other vertex belonging to its neighborhood. An example of an α-lattice can be seen in Figure 3.1, in which the α-lattice is a triangle. Another example is show in Figure 3.3. Those formations must satisfy the set of algebraic restrictions given by

$$\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| = d, \quad \forall j \in B_i. \tag{3.18}$$

The formation of α-lattices, proven in Olfati-Saber (2006) is crucial in the convergence proof to segregation shown in Section 3.2.

3.2 Methodology - Segregation without Estimators

In this section, the algorithm for the problem of segregating groups of robots into clusters with convergence proof is shown. We also show a scheme to deal with collisions between robots without affecting segregation and another scheme to deal with implementation problems that arise after collisions maneuvers are executed.

The proposed control algorithm is based on the use of abstractions to represent each group of robots and an artificial potential function to command the motion of such abstractions.

In order to design our individual controllers it is important to relate the motion of the

abstraction with the motion of the robots. Thus, differentiating (3.6)

$$\dot{\phi}_j = D\phi_j \dot{\boldsymbol{q}}_{\boldsymbol{j}}, \tag{3.19}$$

in which $\dot{\boldsymbol{q}}_{\boldsymbol{j}} = [\dot{x}_j^1, \dot{y}_j^1, ..., \dot{x}_j^{N_j}, \dot{y}_j^{N_j}]^T$. By using (3.3), (3.5) and (3.6) we can obtain $D\phi_j$:

$$D\phi_j = \frac{1}{N_j} \begin{bmatrix} 1 & 0 & 2(x^1 - \mu_j^x) \\ 0 & 1 & 2(y^1 - \mu_j^y) \\ \vdots & \vdots & \vdots \\ 1 & 0 & 2(x^{N_j} - \mu_j^x) \\ 0 & 1 & 2(y^{N_j} - \mu_j^y) \end{bmatrix}^T. \tag{3.20}$$

Since we have robots with double integrator dynamics, we need a relation between the abstraction motion and the robot acceleration. By differentiating (3.5) twice we have

$$\ddot{\sigma}_j = \frac{2}{N_j} \begin{bmatrix} x_j^1 - \mu_j^x \\ y_j^1 - \mu_j^y \\ x_j^2 - \mu_j^x \\ y_j^2 - \mu_j^y \\ . \\ . \\ . \\ x_j^{N_j} - \mu_j^x \\ y_j^{N_j} - \mu_j^y \end{bmatrix}^T \ddot{\boldsymbol{q}}_{\boldsymbol{j}} + 2\sigma_j', \tag{3.21}$$

where,

$$\sigma_j' = \frac{1}{N_j} \sum_{k=1}^{N_j} (\dot{x}_j^k - \dot{\mu}_j^x)^2 + (\dot{y}_j^k - \dot{\mu}_j^y)^2, \tag{3.22}$$

in which $\ddot{\boldsymbol{q}}_{\boldsymbol{j}} = [\ddot{x}_j^1, \ddot{y}_j^1, ..., \ddot{x}_j^{N_j}, \ddot{y}_j^{N_j}]^T$. We can write:

$$\ddot{\phi}_j = D\phi_j \ddot{\boldsymbol{q}}_{\boldsymbol{j}} + \begin{bmatrix} 0 \\ 0 \\ 2\sigma_j' \end{bmatrix}. \tag{3.23}$$

In order to cancel the dynamics that have emerged in (3.23) and consequently be able to directly control the states of the abstraction, we propose a controller for all the robots of group j:

$$\boldsymbol{u_j} = \underbrace{D\phi_j^T(D\phi_j D\phi_j^T)^{-1}\left\{-\begin{bmatrix}0\\0\\2\sigma_j'\end{bmatrix}+\tilde{\boldsymbol{u}}_{\boldsymbol{j}}\right\}}_{s_j} + \mathcal{N}(D\phi_j), \tag{3.24}$$

in which ϕ_j is the abstraction map with robot positions given by $\boldsymbol{q_j}$. Based on (3.2) we have $\ddot{\boldsymbol{q}}_{\boldsymbol{j}} = \boldsymbol{u_j}$, in which $\boldsymbol{u_j} = [(\boldsymbol{u_j^1})^T, (\boldsymbol{u_j^2})^T, ..., (\boldsymbol{u_j^{N_j}})^T]^T$. Note that each robot can compute their part of the control law (3.24) without having to compute the control actions for all robots in its group. The virtual input $\tilde{\boldsymbol{u}}_{\boldsymbol{j}}$ must be designed to control the abstraction state. The term $\mathcal{N}(D\phi_j)$ is a vector defined in the null space of $D\phi_j$ and will be responsible for a collision avoidance scheme and will be defined later, in Section 3.2.1. This term will be null when robots have no risk of colliding to each other.

We have that,

$$\det(D\phi_j D\phi_j^T) = \frac{2\sigma_j}{(N_j)^3}, \tag{3.25}$$

and as long as $\sigma_j \neq 0$, this matrix inverse exists. By analysing the definition of σ_j, (equation 3.5) we can conclude that the case in which $\sigma_j = 0$ is only possible if all the robots of a group are located in the same position at the same time.

From (3.23), applying the individual control laws defined by the components of $\boldsymbol{u_j}$ in (3.24) in every robot, the abstraction state will evolve according to

$$\ddot{\phi}_j = \tilde{\boldsymbol{u}}_{\boldsymbol{j}}, \tag{3.26}$$

in which $\tilde{\boldsymbol{u}}_{\boldsymbol{j}}$ is a virtual input to abstraction j and is now defined by:

$$\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \begin{bmatrix}\boldsymbol{U_j^\mu}\\ U_j^\sigma\end{bmatrix}. \tag{3.27}$$

The term $\boldsymbol{U_j^\mu}$ is an artificial force that guides the mean of the positions of the group and U_j^σ determines the evolution of the size of the abstraction. The design of $\boldsymbol{U_j^\mu}$ and U_j^σ will determine if the approach will be successful. The artificial force to segregate the groups, as defined in Section 3.1.3, is given by

$$\boldsymbol{U_j^\mu} = \boldsymbol{F_j}, \tag{3.28}$$

where $\boldsymbol{F_j}$ is defined according to (3.16) so that the abstraction centers will form α-lattices.

To control the size of each abstraction, we propose a controller U_j^σ so that each abstraction converges to a desired size. The desired size σ_j^{des} is such that, when all abstractions reach this size and the α-lattices are formed, the system will be considered to be segregated according to our definition. We know that each abstraction radius is given by $R_j = \sqrt{N_j\sigma_j}$. Therefore, we should design σ_j^{des} so that the radius R_j of each abstraction

will be smaller than half of the distance d, where d is the size of the edges of the α-lattice, as follows

$$\sigma_j^{des} < \frac{d^2}{4N_j}. \tag{3.29}$$

We propose now the controller U_j^σ:

$$U_j^\sigma = \ddot{\sigma}_j^{des} + k_1(\dot{\sigma}_j^{des} - \dot{\sigma}_j) + k_2(\sigma_j^{des} - \sigma_j), \tag{3.30}$$

in which k_1 and k_2 are properly designed positive gains and

$$\dot{\sigma}_j = \frac{2}{N_j}\sum_{k=1}^{N_j}[(x_j^k - \mu_j^x)\dot{x}_j^k + (y_j^k - \mu_j^y)\dot{y}_j^k]. \tag{3.31}$$

We can make σ_j^{des} constant which implies that $\ddot{\sigma}_j^{des}$ and $\dot{\sigma}_j^{des}$ are equal to zero.

Finally, in the absence of imminent collisions, each robot will be guided by the individual control laws from (3.24), with $\mathcal{N}(D\phi_j) = 0$ and $\tilde{\boldsymbol{u}}_{\boldsymbol{j}}$ given by (3.27), (3.28), (3.29) and (3.30). After some simple manipulation, the elements of $\mathcal{S}_j$ related to robot k in (3.24) can be written as:

$$\mathcal{S}_j^k = \boldsymbol{U}_{\boldsymbol{j}}^{\boldsymbol{\mu}} + \frac{(\boldsymbol{q}_{\boldsymbol{j}}^{\boldsymbol{k}} - \boldsymbol{\mu}_{\boldsymbol{j}})}{\sigma_j}[-2\sigma_j' - k_1\dot{\sigma}_j + k_2(\sigma_j^{des} - \sigma_j)]. \tag{3.32}$$

The gains k_1 and k_2 will be fixed for all abstractions. Without avoiding collisions, we then have the individual control law:

$$\boldsymbol{u}_{\boldsymbol{j}}^{\boldsymbol{k}} = \mathcal{S}_j^k, \tag{3.33}$$

meaning this is the pure segregation controller for robot k, disregarding the collision avoidance scheme, which will be zero in situations where no imminent collisions are detected.

If we consider the collision avoidance scheme, the complete controller is defined as

$$\boldsymbol{u}_{\boldsymbol{j}}^{\boldsymbol{k}} = \mathcal{S}_j^k + \mathcal{N}_k(D\phi_j), \tag{3.34}$$

in which $\mathcal{N}_k(D\phi_j)$ is the k-th and (k+1)-th elements of the full vector $\mathcal{N}(D\phi_j)$, i.e. the elements related to robot k.

The individual control laws (equation (3.33)), when the collision avoidance scheme is not being used, depends on the state of the robot itself, on the state of the abstraction of the robot and on the states of neighboring abstractions. This control law does not depend on the states of robots and abstractions that are far from the robot being controlled, i.e. outside neighborhood B_i defined in Section 3.1.3 thanks to the *finite cut-off* property of the artificial potential field.

If we consider the collision avoidance scheme, (equation 3.34), robots will also need information about all robots that belongs to the groups that are involved in the collision being avoided as we will show in the next section.

3.2.1 Collision Avoidance Strategy

The proposed control laws until now were designed to control point robots, that is, they do not consider robot sizes. To make this proposal feasible to be applied in actual robots, the size of each robot must be considered and a collision avoidance scheme must be used.

In order to guarantee that the collision avoidance scheme will not interfere in the convergence to segregation, which is our primary goal, we will consider a controller in the null space of $D\phi_j$. This term is related with the vector $\mathcal{N}(D\phi_j)$ introduced in (3.24) and (3.34).

In Michael et al. (2007) a similar approach is used, also in the null space of $D\phi_j$, however using aggregations of three arbitrary robots to avoid collisions. When a robot is in imminent collision, two other robots are chosen to "correct" the state of the abstraction while the robot in imminent collision changes its route to avoid collision. Those grouping are based on the distances among the three robots taken two at a time. Note that in this case, to exist a null space, there must be at least three robots in the system.

In this work, we propose a different scheme, where those aggregations are not needed. We consider the null space of the whole group of the robots in imminent collision. We use the projection:

$$\mathcal{N}(D\phi_j) = (I - D\phi_j^T(D\phi_j D\phi_j^T)^{-1}D\phi_j)\hat{\boldsymbol{u}}_{\boldsymbol{j}} = \mathcal{N}\hat{\boldsymbol{u}}_{\boldsymbol{j}}, \tag{3.35}$$

where I is the identity matrix.

This new controller term is only activated if an imminent collision is detected. This term will be equal to zero if none of the robots are in imminent collision in the system. We define two conditions to be satisfied so that a robot is considered in imminent collision. First, we check if robots are close enough, i.e.

$$\|\boldsymbol{q}_{\boldsymbol{i}}^{\boldsymbol{k}} - \boldsymbol{q}_{\boldsymbol{j}}^{\boldsymbol{l}}\| < 2R_b + \delta. \tag{3.36}$$

In this work, we consider circular robots with radius R_b and a safety factor δ.

Moreover, we check if, with current velocities, robots are in trajectories that would likely result in collision, i.e.

$$(\dot{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \dot{\boldsymbol{q}}_{\boldsymbol{j}}^{\boldsymbol{l}})^T(\boldsymbol{q}_{\boldsymbol{i}}^{\boldsymbol{k}} - \boldsymbol{q}_{\boldsymbol{j}}^{\boldsymbol{l}}) < 0. \tag{3.37}$$

Indexes i and j indicate to which group those robots belong and indexes k and l indicate which robots are in imminent collision.

Therefore, the individual control laws are determined by:

$$\boldsymbol{u}_{\boldsymbol{j}}^{\boldsymbol{k}} = \mathcal{S}_j^k + \mathbf{1}_A(I - D\phi_j^T(D\phi_j D\phi_j^T)^{-1}D\phi_j)\hat{\boldsymbol{u}}_{\boldsymbol{j}}. \tag{3.38}$$

in which $\mathbf{1}_A$ is a indicator function that is equal to 1 when there is a robot in imminent collision, i.e. conditions 3.36 and 3.36 are satisfied. This indicator function is equal to 0

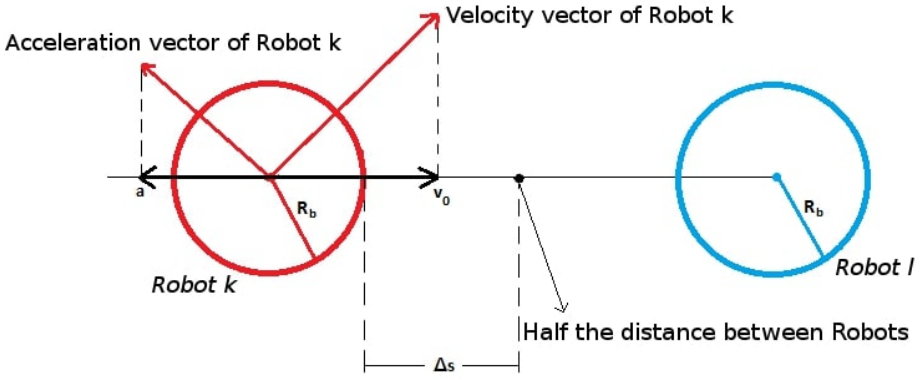


Figure 3.4: Schematic illustrating the strategy to avoid collisions.

when robots there are no robots in imminent collision.

Vector $\hat{\boldsymbol{u}}_j$ is chosen to guarantee absence of collisions as shown next.

As robots are actuated in acceleration, one should search for new accelerations that generate collision free trajectories. This is done based on the equations of motion for one-dimensional movements with constant acceleration (*Torricelli's* equation of motion)

$$v_f^2 = v_0^2 + 2a\Delta s, \tag{3.39}$$

where v_f is the final velocity of the robot in consideration, v_0 is the initial velocity, a is the acceleration and Δs is the difference in the displacement in a given time interval.

To obtain the acceleration a so that robots in imminent collision do not collide, we consider a conservative approach using $v_f = 0$. This means that, in the worst case, both robots should stop moving right before collision. We then define Δs as half the distance between two robots being considered, minus the radius of the robots

$$\Delta s = \frac{1}{2}\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\| - R_b, \tag{3.40}$$

assuming both robots with radius R_b. Considering the acceleration components a of robot k of group i in the direction of robot l of group j:

$$a = (\boldsymbol{u}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|}, \tag{3.41}$$

and considering the component of the velocity v_0 of robot i in the direction of robot j:

$$v_0 = (\dot{\boldsymbol{q}}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|}. \tag{3.42}$$

Replacing $v_f = 0$ in *Torricelli's* equation, we define the constraint

$$a \leq \frac{0 - v_0^2}{2\Delta s}. \tag{3.43}$$

Figure 3.4 shows an example with two robots in imminent collision (robots k and l) that may or may not be of the same group. In this Figure, we highlight Δs (equation (3.40)), the component of the acceleration a (equation (3.41)) and the component of the velocity v_0 (equation (3.42)) related to robot k.

Replacing Δs, a and v_0 in (3.43), we have a constraint for robot k in relation to robot l:

$$(\boldsymbol{u}_i^k)^T \frac{(\boldsymbol{q}_i^k - \boldsymbol{q}_j^l)}{\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\|} \geq \frac{\left[(\dot{\boldsymbol{q}}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\|}\right]^2}{\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\| - 2R_b}, \tag{3.44}$$

similarly, we have for robot l in relation to robot k:

$$(\boldsymbol{u}_j^l)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\|} \geq \frac{\left[(\dot{\boldsymbol{q}}_j^l)^T \frac{(\boldsymbol{q}_i^k - \boldsymbol{q}_j^l)}{\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\|}\right]^2}{\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\| - 2R_b}. \tag{3.45}$$

Therefore we have two constraints that, if respected all the time, robots will not collide. In spite of the use of indexes i and j to treat groups, if two robots of the same group are in imminent collision, we make $i = j$ and those same constraints will be used to treat collisions between these robots.

To make the strategy energetically efficient, $\hat{\boldsymbol{u}}_j$ is minimized for every group involved:

$$\begin{aligned} \min \quad & \sum_{s \in \Omega_p} \|\hat{\boldsymbol{u}}_s\|^2, \\ \text{s. t.} \quad & (3.44) \quad \text{and} \quad (3.45), \end{aligned} \tag{3.46}$$

where the set Ω_p is the set of groups in which there are robots in imminent collision, and index p indicates the corresponding connected component in the collision graph (see Figure 3.5). In this graph, each abstraction is a node and there are edges between nodes that have imminent collisions between robots of their groups. Furthermore, $\hat{\boldsymbol{u}}_s$ is the vector of accelerations chosen to avoid collisions for group s, and s is an element of the set Ω_p. Figure 3.5 shows an example of a situation where the collision graph has two connected components ($p \in \{1, 2\}$). In this scenario, there are two optimization problems to be solved independently, one with two groups involved and another one with three groups involved. Please refer to the Appendix A.1.1 to review the basic concepts in graph theory.

In each optimization problem, we consider in the objective function all groups in which there are robots involved in imminent collisions. With this information, the problem can be solved in a distributed manner as each robot can execute a minimization problem internally.

Constraints of the minimization problem (3.46) are added for each pair of robots that comply with conditions of (3.36) and (3.37). If two robots of groups i and j are in imminent

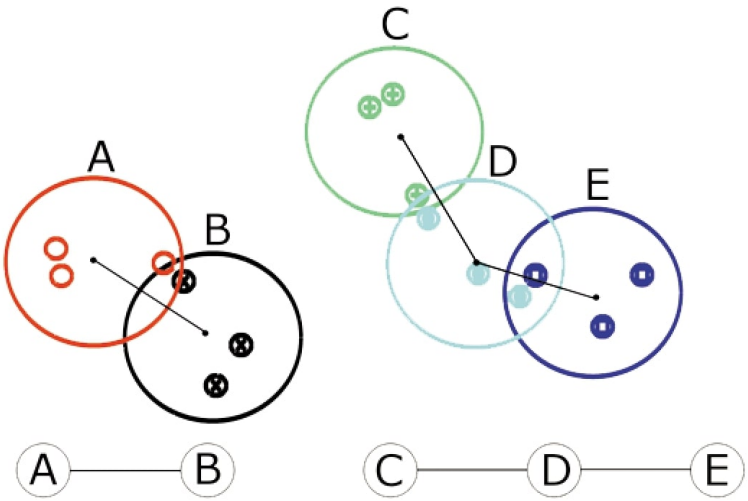


Figure 3.5: Graph of Robots in imminent collision.

collision, for instance, the objective function in (3.46) becomes

$$\min \quad \|\hat{\boldsymbol{u}}_i\|^2 + \|\hat{\boldsymbol{u}}_j\|^2, \tag{3.47}$$

with constraints given by (3.44) and (3.45).

3.2.2 Implementation Issues

After the minimization problem (3.46) is solved, we have a vector $\hat{\boldsymbol{u}}_j$ for each group. This vector changes trajectories of a robot in imminent collision and potentially of all the other robots of its group. We have seen in simulations that this can generate trajectories with undesired high speeds. This effect usually occurs only after an imminent collision is avoided. This effect is perceived visually as robots "orbiting" the center of its own abstraction to maintain the abstraction state.

In this work we propose another controller with the objective of avoiding the robots to move with velocities much higher than the velocity of its abstraction mean.

This new controller is called *velocity dissipation controller*, velocity being the difference between the velocities of the abstraction mean and the velocities of the robots that belong to this abstraction. This controller is associated with the collision avoidance controller and is non-zero only if the collision avoidance controller is zero. That is, if we have an imminent collision, only the collision avoidance algorithm is turned on and if we do not have imminent collisions, only the *velocity dissipation* controller is turned on. It should be clear that both controllers are defined in the null space of $D\phi_j$ so that our primary goal, segregation, is not impacted.

We define now a *future velocity* vector

$$^f\dot{\boldsymbol{q}}_j = [{}^f\dot{x}_j^1 \quad {}^f\dot{y}_j^1 \quad \cdots \quad {}^f\dot{x}_j^{N_j} \quad {}^f\dot{y}_j^{N_j}]^T. \tag{3.48}$$

This *future velocity* vector is obtained after integrating each robot trajectory for one integration step, i.e. the velocity that each robot will acquire after applying a given control vector $\hat{\boldsymbol{u}}_j$ as depicted in **Algorithm 3.1**.

Algorithm 3.1 Calculate future robot velocity

Require: Control vector $\hat{\boldsymbol{u}}_j$
Ensure: Future velocity vector ${}^f\dot{\boldsymbol{q}}_j$
1: Calculate $\mathcal{N}(D\phi_j)$ (Equation (3.35))
2: Calculate $\boldsymbol{u}_j^k$ (Equation (3.34))
3: Integrate one step (Using robot model, equation (3.2))
4: Obtain ${}^f\dot{\boldsymbol{q}}_j$, which is $\boldsymbol{v}_j^k$ one step in the future

With ${}^f\dot{\boldsymbol{q}}_j$ and mean velocities ($\dot{\boldsymbol{\mu}}_j = [\dot{\mu}_j^x \quad \dot{\mu}_j^y]^T$) we can now define a relative velocity vector for group j:

$$ {}^{rel}\dot{\boldsymbol{q}}_j = {}^f\dot{\boldsymbol{q}}_j - \mathbb{1} \otimes \dot{\boldsymbol{\mu}}_j, \tag{3.49}$$

where $\mathbb{1}$ is a column vector given by $\mathbb{1} = [1, 1, 1..., 1]^T$ and $\otimes$ is the Kronecker product operation. So that, in ${}^{rel}\dot{\boldsymbol{q}}_j$ we have the velocities of the robots in relation to the center of its group's mean velocity.

One must search for a vector $\hat{\boldsymbol{u}}_j^*$, that, in the next step, will generate velocities ${}^f\dot{\boldsymbol{q}}_j$ that minimizes ${}^{rel}\dot{\boldsymbol{q}}_j$.

Minimizing those relative velocities in the null space of $D\phi_j$ will make robots reduce those potential high velocities dissipating the velocity of the robots without affecting group formation. This unconstrained minimization problem is now defined

$$ \hat{\boldsymbol{u}}_j^* = \arg\min_{\hat{u}_j} \left\| {}^{rel}\dot{\boldsymbol{q}}_j \right\|^2. \tag{3.50}$$

This minimization problem will generate a vector $\hat{\boldsymbol{u}}_j^*$ that will be added to the control action of the original segregation controller whenever there is no imminent collisions. One different vector $\hat{\boldsymbol{u}}_j^*$ will be generated for each group of robots.

Note that in many cases, when robots are already moving cohesively, the minimization problem will generate a trivial all zero vector, because robot velocities and its abstraction mean velocity are the same. The *velocity dissipation* controller has shown to be helpful in dissipating high velocities generated after collisions are avoided. When conditions (3.36) and (3.37) are satisfied, the collision avoidance controller will be non-zero. Thus turning off the *velocity dissipation* controller.

Figure 3.6 shows a comparison between trajectories, so we can qualitatively analyze the *velocity dissipation* controller. In this Figure, we have six robots divided into two balanced groups (red and green groups). In both Figures 3.6(a) and 3.6(b) the same initial conditions were used and the simulation was stopped at the same time. In 3.6(a) the *velocity dissipation* controller is used and it makes the trajectories of red robots more cohesive than the trajectories in 3.6(b) where the *velocity dissipation* is not used. In 3.6(b),

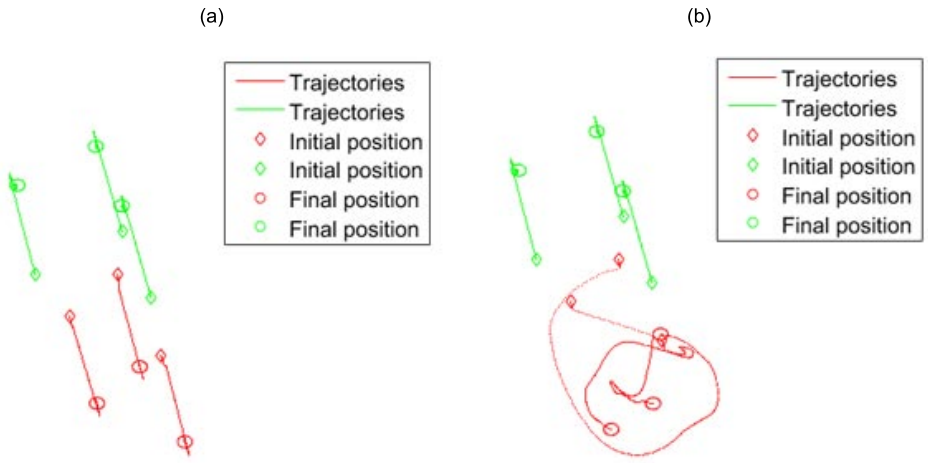


Figure 3.6: Trajectories comparison with and without the velocity dissipation controller. (a) Velocity dissipation controller enabled. (b) Velocity dissipation controller disabled.

when a red robot has to avoid a collision, it accelerates to a high velocity and starts to "orbit" around the abstraction center together with all the other robots of its own group to maintain the group state. In both cases segregation was successfully achieved without collisions but in the case of Figure 3.6(b) an undesirable "orbiting" behavior occurred, this can be better visualized in the video presented in https://youtu.be/UvjgwlV_PCE.

3.2.3 Control Analysis

In this section, we formally analyze the proposed controller to show its capability in solving the problem of segregation in robot swarms.

Theorem 1. *The application of individual control laws given by (3.34) for a group of* $\sum_{j=1}^{M} N_j$ *robots with dynamics given by (3.2) divided in* M *groups will enforce the convergence of the multi-robot system to a state where all robots of a same group are grouped together while segregated from robots of other groups, i.e, the problem defined in Section 3.1.2 will be solved if (i) solving (3.46) to avoid collisions is always feasible; (ii) the system does not start in a local maximum or saddle point of* $V(\boldsymbol{\mu})$*; (iii) the robots start at different positions in the environment; and (iv)* k_1 *and* k_2 *in (3.30) are such that the related characteristic equation has roots with negative real parts.*

Proof. The approach was elaborated to guarantee the solution of the problem; this means that the proof is straight forward. The analysis is conducted in two parts.

First, we have to prove that all robots in an abstraction will stay inside it and the abstraction state will converge to the desired size. The second part is to show that the abstractions will end up separated, without intersections.

Due to the assumptions that the robots do not start at the same position and the collision scheme in (3.46) is always feasible we can assure that the robots will never be at

the same position at the same time. Thus, the determinant of $D\phi_j D\phi_j^T$ is different from zero and the inverse in (3.24) always exists, then the motion of the abstraction will be given by (3.26). From (3.30) it should be clear that if k_1, k_2 are properly designed the dynamics given by $\ddot{\sigma}_j = U_j^{\sigma}$ will be such that σ_j converges to σ_j^{des} exponentially (Slotine & Li, 1991). Since the radius is defined according to $R_j = \sqrt{N_j \sigma_j}$, we know from Section 3.1.1 that the robots in ϕ_j will remain inside the abstraction during all the time.

For the second part of the proof, we consider the proof of *Theorem 1* in Olfati-Saber (2006). In that theorem, *LaSalle's invariance principle* is used to show that a set of agents with double integrator dynamics subject to the artificial potential force in (3.16) (see *Algorithm 1* in Olfati-Saber (2006)) asymptotically converges to a configuration which is an equilibrium of function V. Since we assume that the system does not start at a local maximum or at a saddle point of V, and these are unstable equilibria we can guarantee that the system asymptotically converges to a local minimum of V. By using *Lemma 1* (see Section 3.1.3) we can conclude that the system asymptotically converges to an $\alpha-$lattice formation.

As the abstractions converge to the desired size, with all the robots of the abstraction inside, together with the fact that the other parameters (see equation (3.29)) were specified to guarantee absence of intersections among abstractions when they converge to the $\alpha-$lattice, the problem of segregation as defined in the *Problem Definition* will be solved as $t \to \infty$. □

In this theorem we made some assumptions on the initial conditions. We consider those assumptions to be plausible in real scenarios. In real situations, the system hardly begins and stays at a critical point of $V(\boldsymbol{\mu})$ as these are unstable equilibrium points, except for the minima which are the solution of the problem. An example of a critical point would be more than one abstraction center starting at the same point, it is reasonable to assume that this will hardly happen in practice and even if it does, this is an unstable condition. Most approaches based on artificial potential fields described in the literature rely on the same assumption. Furthermore, a scheme to avoid collisions between robots was developed, formulated through a minimization problem. If this problem always has viable solutions, robots will never collide. However, it is known that there are situations in which this problem has no viable solution. One can think of an example in which a robot from one group is surrounded by several robots from another group. In this case, the surrounded robot would not be able to escape from collisions without "disturbing" the abstraction state of the robots that surround it. Nevertheless, it is reasonable to assume that this type of situation is rare in practice.

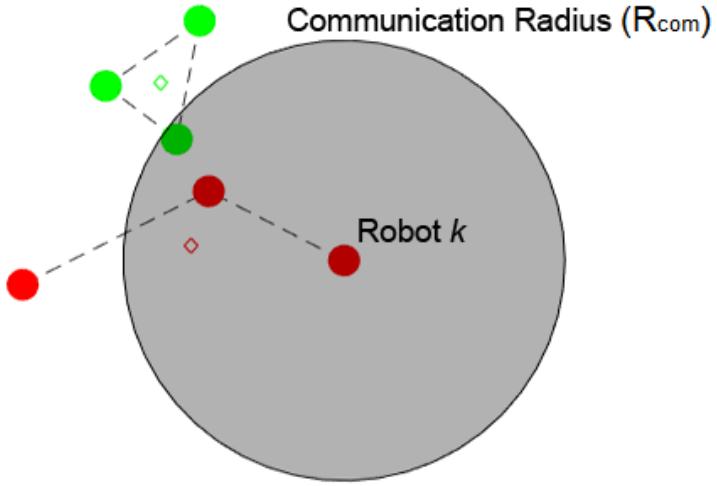


Figure 3.7: Example of a robot's communication radius (R_{com}).

3.3 Methodology - Segregation with Estimators

Figure 3.7 shows 2 groups of 3 robots in each, highlighting the communication radius (R_{com}) of a robot (robot k). Note that robot k has communication with only one robot in its group and one robot in a neighboring group. This Figure also shows the communication graphs of both the red robot group and the green robot group in dotted lines. Both graphs are connected and the green robot group graph is also complete, meaning that all robots of the green group can exchange information with each other. Also shown in the Figure are the average of the positions of the robots for both groups, marked as diamonds.

In this section we propose a new controller where collisions between robots will not be considered, thus, in order to be able to use it in practice a local controller to avoid collisions must be implemented. The collision avoidance controller that does not change the state of abstractions proposed in Section 3.2.1 cannot be used directly, as it uses information from all robots in the groups involved in collisions and, in this controller, the focus is to control robots with information provided only by the corresponding neighborhood. An adaptation of the proposed controller to avoid collisions in Section 3.2.1 to work locally is a suggestion of future work.

The controller in this section deals with the problem defined in Section 3.1.2, but no proof of convergence or formal stability of the proposed controller will be presented, just some considerations in the direction of such proof. The formal proof is also a suggestion of future work.

The controller presented in this section is built upon the controller presented in Section 3.2. More specifically, in this section the same controller of the equation (3.32) is used and

reproduced here:

$$\boldsymbol{u}_j^{\boldsymbol{k}} = {}^{\mu}\boldsymbol{U}_j + \frac{(\boldsymbol{q}_{\boldsymbol{k}} - \boldsymbol{\mu}_j)}{\sigma_j}[-2\sigma'_j - k_1\dot{\sigma}_j + k_2(\sigma_j^{des} - \sigma_j)].$$

However, all states of the abstractions, its derivatives and σ' will be estimated. Each robot now estimates abstraction states independently. It cannot be said that robots in the same group estimate exactly the same value for a state. Thus, we use superscript k index to represent the estimate of the robot k. For the k-th robot of the j-th group we have the individual control given by

$$\boldsymbol{u}_j^{\boldsymbol{k}} = {}^{\hat{\mu}}\boldsymbol{U}_j^{\boldsymbol{k}} + \frac{(\boldsymbol{q}_{\boldsymbol{k}} - \hat{\boldsymbol{\mu}}_j^{\boldsymbol{k}})}{\hat{\sigma}_j^k}[-2\hat{\sigma'}_j^k - k_1\dot{\hat{\sigma}}_j^k + k_2(\sigma_j^{des} - \hat{\sigma}_j^k)], \tag{3.51}$$

in which $\hat{\boldsymbol{\mu}}_j^{\boldsymbol{k}}$ is the estimate of the average of the positions of all robots of group j, from the k-th robot standpoint. Similarly, one has that $\hat{\sigma}_j^k$, $\dot{\hat{\sigma}}_j^k$ and $\hat{\sigma'}_j$ are estimated values of σ_j, $\dot{\sigma}_j$ and σ'_j, respectively, computed by robot k.

Note that in (3.32) all robots in the same group consider exactly the same value as the states of abstraction ($\boldsymbol{\mu}_j$, σ, $\dot{\sigma}_j$ and σ'_j) and in (3.51) each robot individually calculates each of these states. This is due to the fact that, previously, variables required for the calculation of individual control were calculated centrally and supplied to the robots, and now each robot makes this calculation in a decentralized manner.

Thus, the term ${}^{\mu}\boldsymbol{U}_j$ is changed to ${}^{\hat{\mu}}\boldsymbol{U}_j^{\boldsymbol{k}}$ in the equation (3.51). This means that the potential forces for separation of group averages are now calculated from the estimated values of the average of groups positions and velocities from the point of view of each robot:

$${}^{\hat{\mu}}\boldsymbol{U}_j^{\boldsymbol{k}} = \sum_{i\in B_j} \gamma_\alpha(\|\hat{\boldsymbol{\mu}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}}\|_\sigma)\hat{\boldsymbol{n}}_{ji}^k + \sum_{i\in B_j} \rho_h(\|\hat{\boldsymbol{\mu}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}}\|_\sigma / r_\alpha)(\dot{\hat{\boldsymbol{\mu}}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \dot{\hat{\boldsymbol{\mu}}}_{\boldsymbol{j}}^{\boldsymbol{k}}), \tag{3.52}$$

in which

$$\hat{\boldsymbol{n}}_{ji}^k = (\hat{\boldsymbol{\mu}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}})/\sqrt{(1 + \|\hat{\boldsymbol{\mu}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}}\|^2)}. \tag{3.53}$$

This controller that guides the movement of the centers of abstractions, ${}^{\hat{\mu}}\boldsymbol{U}_j^{\boldsymbol{k}}$, depends on the positions and velocities of the centers of all neighboring abstractions. This information is not available from the point of view of a robot. Thus, robots obtain the information of these states of other groups from estimates of the nearest robots belonging to these other groups. That is, robots from other groups transmit the states of their abstractions to nearby robots.

No connectivity to other groups is assumed. If all robots in one group do not "see" any robots in another group, this suggests that these groups are already segregated. If a robot "sees" a robot of another group, this robot will move in such a way as to "repel" this other group. Thereby, the other robots in its group will also move in order to maintain the abstraction. This may cause another robot to "see" other groups, and also "repel" these

other groups.

3.3.1 Consensus as Estimator

In the area of swarm robotics with decentralized control, consensus strategies are widely used for swarm navigation, whether in flocking, rendezvous or other tasks. In the controller that will be proposed in this section, a consensus is not used to move robots directly [1].

In this section, consensus protocols are used for robots to exchange information with their neighbors so that certain variables are estimated, those variables are the ones that each robot would not have access on its own. This information exchange is done until all robots reach a certain common value for the estimated variable. A consensus protocol is a process used for all robots to agree on a common value without a centralized agent coordinating this information exchange.

Thus, using only local information, robots will have access to an estimate of these variables and may use these estimates to define a decentralized controller to move each robot to solve the segregation problem in robot swarms as defined in Section 3.1.2.

Note that in this document, when we use estimators, we are dealing with consensus protocols used as estimators. We do not use Kalman filters or other types of classic estimators.

In this section, only consensus protocols with no time delays and continuous time will be considered, and the related graphs are always undirected. In (Zhu et al., 2017) a complete review of the latest advances in consensus theory can be found. Variables used in consensus protocols will be indicated by bars over the variables, e.g. $\bar{x}$. All the variables that have a bar are only used to exchange information and are not states of the robots.

Consider the dynamics

$$\dot{\bar{x}}_k(t) = \bar{u}_k(t) \quad k = 1, 2, ..., \bar{N}, \tag{3.54}$$

in which $k = 1, 2...\bar{N}$ is the number of agents involved in the consensus protocol and $\bar{x}_k, \bar{u}_k \in \mathbb{R}$.

Considering the dynamics (3.54) we define the following linear consensus protocol:

$$\bar{u}_k = \sum_{l \epsilon \bar{N}} \bar{a}_{kl}(\bar{z}_l - \bar{z}_k), \tag{3.55}$$

in which $\bar{a}_{kl}$ are the elements of the Adjacency Matrix defined in Section A.1.1, specifying the neighborhood of agent k, and $\bar{z}_l$ and $\bar{z}_k$ are scalar decision variables for robots l and k, respectively. Those variables will be different depending on the estimative that is being computed, as will be clear along the text.

[1] Except in part for the potential function of the Section 3.1.3 where there is a consensus term used to match the velocity of the groups.

The protocol (3.54) is called the *average consensus*, because this protocol always implies in the convergence to the average of the initial conditions of the agents (3.54) if the robot connection graph is an undirected connected graph (Olfati-Saber & Murray, 2003).

Note that this protocol is a first order one while the dynamics of the robots are modeled with a second order model. This is possible because the consensus will be used as an estimator and not to move the robots.

Convergence time to average is directly related to the algebraic connectivity, and also depends on initial conditions $\bar{x}(0)$. Recent advances in consensus theory have made possible to ensure convergence to the mean in finite-time (Wang & Xiao, 2010), and in fixed-time (Parsegov et al., 2013).

Finite time convergence ensures that there is a constant T where convergence to the mean will be achieved for all agents at a time less than T (Cao et al., 2013). The transient period in the finite time consensus depends on the initial conditions. If the consensus is reached in finite time and the transient period does not depend on the system initial conditions, then this protocol is called the fixed time consensus (Zhu et al., 2017).

In this section, a fixed time consensus will be used as shown in Zhang & Jia (2015) and reproduced below:

$$\bar{u}_k = \bar{\alpha} \sum_{l \epsilon \bar{N}} \bar{a}_{lk} (\bar{z}_l - \bar{z}_k)^{\bar{m}/\bar{r}} + \bar{\beta} \sum_{l \epsilon \bar{N}} \bar{a}_{lk} (\bar{z}_l - \bar{z}_k)^{\bar{p}/\bar{q}}, \tag{3.56}$$

in which $\bar{\alpha} > 0$, $\bar{\beta} > 0$ and $\bar{m}$, $\bar{r}$, $\bar{p}$, $\bar{q}$ are odd positive integers such that $\bar{m} > \bar{r}$ and $\bar{p} < \bar{q}$. With this protocol there is a time T_{max} which can be calculated and does not depend on the initial system conditions, for all agents to agree on the mean value of the initial conditions, i.e. the consensus of the mean is always obtained[2].

Considering that in this work the communication graphs between robots are such that the elements of the adjacency matrix are always 1 if there is a connection and 0 if there is not, we can simplify the calculation of T_{max} found in Zhang & Jia (2015) to

$$T_{max} = \frac{\pi q (N_j^{(q-p)/2q})}{2\Big(\lambda_2 (q-p) \sqrt{(\bar{\alpha}\bar{\beta})}\Big)}, \tag{3.57}$$

in which N_j is the number of robots in group j and $\bar{q}$, $\bar{p}$, $\bar{m}$, $\bar{n}$, $\bar{\alpha}$ e $\bar{\beta}$ are protocol parameters to be regulated *a priori*.

To estimate the states of the abstraction in a decentralized way, an estimator is proposed using a consensus protocol for each state to be estimated. In this work, the consensus protocol is always used with all robots in the same group. Thereby, we have $\bar{N} = N_j$ for the group j, i.e. the number of robots used in the protocol will always match the number of robots in the group being considered.

[2]The proof can be found at (Zhang & Jia, 2015).

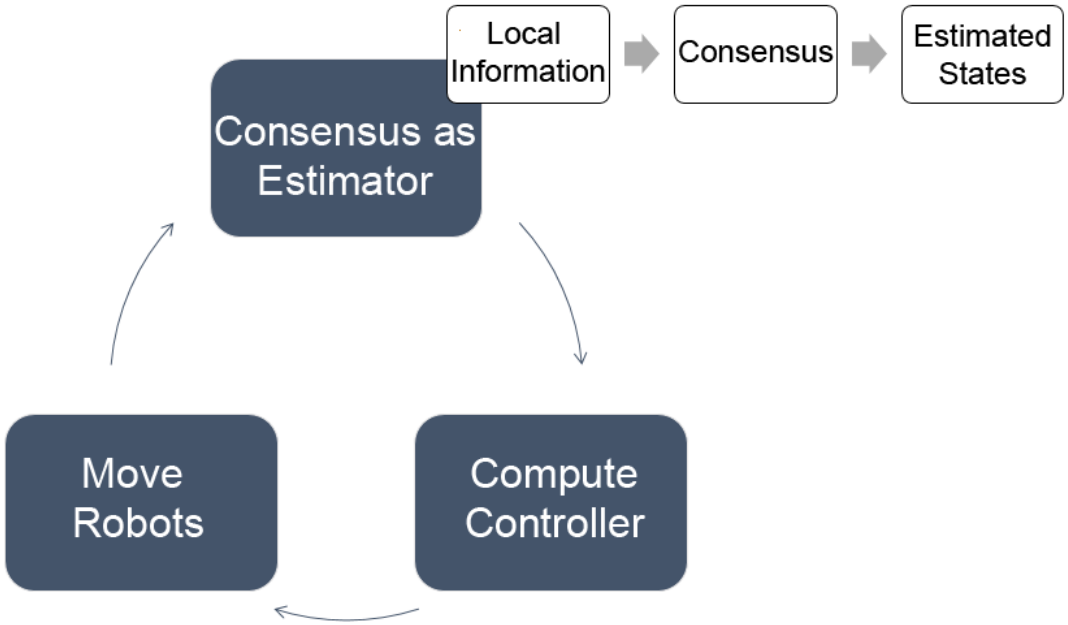


Figure 3.8: Flowchart on how to move robots using estimated states to compute the controllers.

From a robot's point of view, first, the estimated states of its abstraction are obtained using only local information, using consensus protocols. Then, the individual control action is calculated using the estimated states (equation (3.51)). Finally, the robots move with the computed control action. The Figure 3.8 exemplifies this whole process.

The fixed time consensus protocol will be used as shown in the equation (3.56). This protocol will be computed internally for each robot for a fixed time.

Every time a robot calculates its individual control action (equation (3.51)) it must have previously executed a consensus for each of the states required in this control law, i.e. the consensus is executed on an internal loop and it is only virtual while segregation control is executed in an external loop.

Therefore, for the individual control to be fully defined, at each moment one must estimate, from the point of view of a robot of the j-th group, seven variables:

1. $\hat{\mu}_j^x$ - Component x of the average of the positions of the robots of group j;
2. $\hat{\mu}_j^y$ - Component y of the average of the positions of the robots of group j;
3. $\dot{\hat{\mu}}_j^x$ - Component x of the average of the velocities of the robots of group j;
4. $\dot{\hat{\mu}}_j^y$ - Component y of the average of the velocities of the robots of group j;
5. $\hat{\sigma}_j$ - Third variable of the abstraction, associated with the size of the abstraction;
6. $\dot{\hat{\sigma}}_j$ - Variable required for the computation of the individual control (given by equation (3.31));

7. $\hat{\sigma}'_j$ - Variable required for the computation of the individual control (given by equation (3.22)).

This means that for each calculation of the individual control law (3.51) one has to execute seven consensus protocols internally to estimate these values. In addition, robots must receive from the closest robots from neighboring groups the states of these groups.

Note that the communication graph of robots may be time varying, but the graph must remain connected at all times so that it is guaranteed that the consensus converges to the average (Olfati-Saber & Murray, 2004). Also note that the communication graph is not related to the collision graph used in Section 3.2.2.

In this section, it was assumed that all groups have connected communication graphs, this means that each robot has communication with at least one other robot of its group. The communication graph being connected is equivalent to saying that the Laplacian matrix $\mathcal{L}$ always has a positive second smallest eigenvalue (see Section A.1.1). This assumption is strong in the sense that there is nothing guaranteeing that robots will not lose this connection. An interesting addition to this work may be to try to ensure that graphs always remain connected. Strategies for maintaining the connectivity of a group of robots exists, such as the works of Dimarogonas & Johansson (2008) and Yang et al. (2010), but are beyond the scope of this work.

To estimate the averages of positions and velocities of group j ($\hat{\mu}_j^x$, $\hat{\mu}_j^y$, $\hat{\dot{\mu}}_j^x$ and $\hat{\dot{\mu}}_j^y$), the fixed time consensus protocol (equation (3.56)) is executed on each robot individually for the time determined by T_{max} (equation (3.57)). Knowing that fixed-time consensus estimator gains are regulated *a priori*, this consensus protocol will always converge to averages (μ_j^x, μ_j^y, $\dot{\mu}_j^x$ and $\dot{\mu}_j^y$) (Zhang & Jia, 2015). This is done considering the variables $\bar{z}_l$ and $\bar{z}_k$ of the equation (3.56) as, respectively:

- x_j^l and x_j^k to estimate μ_j^x,
- y_j^l and y_j^k to estimate μ_j^y,
- $\dot{x}_j^l$ and $\dot{x}_j^k$ to estimate $\dot{\mu}_j^x$,
- $\dot{y}_j^l$ and $\dot{y}_j^k$ to estimate $\dot{\mu}_j^y$,

in which x e y are the components of the position of the robots in the plane, and $\dot{x}$ and $\dot{y}$ are the components of their velocities in the plane (equation (3.2)). Index k indicates which robot is executing the consensus protocol, and the l index indicates its neighbors (see equation (3.56)).

To calculate the maximum time T_{max} for the protocol to converge to the average of the initial values, information on the number of robots of group N_j, the algebraic connectivity λ_2 and consensus protocol gains (see equation (3.57)) are required. Consensus protocol gains can be fixed a priori and can be the same for all robots. Other information (N_j and

λ_2) are not available from a robot's point of view and may vary over time, given that the robot communication graph may change as robots move and may change when robots are added or removed from the group.

It is known that an upper bound of the second smallest eigenvalue of the Laplacian matrix related to a connectivity graph (upper bound of λ_2) is given by the number of nodes in this graph, which in this case is the number of robots in the group itself (N_j) (Gross & Yellen, 2003). Thus, it is assumed that the robots know an upper bound for N_j which is also an upper bound of λ_2. This means that the robots must know or estimate how many robots their own group can have and from this data the robot calculates the maximum time for the consensus protocol to converge to the mean of the initial values. This is a conservative approach, which makes the time for the consensus protocol to execute from the initial instant up to T_{max} to be longer. Note that one must execute a consensus protocol for each of the estimated variables.

It is also necessary to compute $\hat{\sigma}_j$, $\dot{\hat{\sigma}}_j$ and $\hat{\sigma'}_j$ in order that the individual controller of equation (3.51) to be completely defined. These values appear as a sum divided by the number of robots, so these values can also be viewed as a "mean". This can be observed by analyzing their definitions in the equations (3.5), (3.31) and (3.22). Thus, the computation of $\hat{\sigma}_j$, $\dot{\hat{\sigma}}_j$ and $\hat{\sigma'}_j$ can be done using auxiliary variables ($\bar{z}_\sigma$, $\bar{z}_{\dot{\sigma}}$ and $\bar{z}_{\sigma'}$ respectively), defining them:

$$\bar{z}_\sigma^k = (q_j^k - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}})^T (q_j^k - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}}) \tag{3.58}$$

$$\bar{z}_{\dot{\sigma}}^k = 2(q_j^k - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}^{\boldsymbol{k}})^T \dot{q}_j^k \tag{3.59}$$

$$\bar{z}_{\sigma'}^k = (\dot{q}_j^k - \dot{\hat{\boldsymbol{\mu}}}_{\boldsymbol{j}}^{\boldsymbol{k}})^T (\dot{q}_j^k - \dot{\hat{\boldsymbol{\mu}}}_{\boldsymbol{j}}^{\boldsymbol{k}}) \tag{3.60}$$

These auxiliary variables are initialized on each robot and each one executes the fixed time consensus protocol (equation (3.56)) long enough (T_{max}) so that these auxiliary variables converge to the average. These averages of the auxiliary variables are the values of σ_j, $\dot{\sigma}_j$ and σ'_j themselves.

Note that robots never stop moving to wait for the fixed-time consensus protocol to converge, robots continue to move and use the latest results, in which the protocol converged to the average, to calculate its controller and move. This works as a zero-order hold (ZOH), that is, each time one have more recent estimates, these are used for the computation of the individual control action in equation (3.51). Figure 3.9 shows this process and the information needed for each robot to fully define its individual control action (equation (3.51)).

Note that position and velocity averages must be calculated before $\hat{\sigma}_j$, $\dot{\hat{\sigma}}$ and $\hat{\sigma'}_j$. The average can be calculated in parallel and after that, also in parallel $\hat{\sigma}_j$, $\dot{\hat{\sigma}}_j$ and $\hat{\sigma'}_j$. Thus, the total time to calculate all estimated states is $2T_{max}$. One can first execute the fixed time consensus in parallel to get $\hat{\mu}_j^x$ (Consensus 1), $\hat{\mu}_j^y$ (Consensus 2), $\dot{\hat{\mu}}_j^x$ (Consensus 3) and $\dot{\hat{\mu}}_j^y$ (Consensus 4), and with these values execute again in parallel to get $\hat{\sigma}_j$ (Consensus

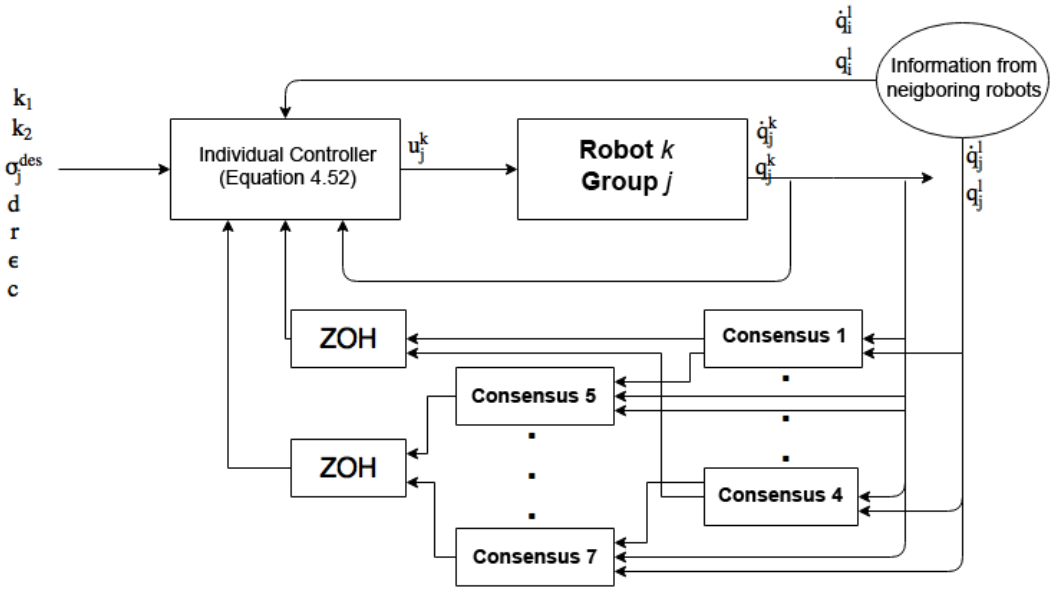


Figure 3.9: Individual control calculation flowchart for robot k of group j.

5), $\dot{\hat{\sigma}}_j$ (Consensus 6) and $\hat{\sigma'}_j$ (Consensus 7). This whole process is shown in Figure 3.9, in which the necessary information for each calculation is highlighted.

3.3.2 Considerations

In this section considerations are made about the decentralized controller for segregation of robot swarms considering some unrealistic assumptions, in order to visualize a future proof of convergence. The complete proof with more realistic assumptions is a suggestion of future work.

Consider $\sum_{j=1}^{M} N_j$ robots with dynamics given by the equation (3.2) and with a limited communication radius, given by R_{com}, divided into M groups. The following assumptions were made:

1. There are no collisions between robots and therefore no situations where all robots in the same group are at one point at the same time
2. The communication graphs of the groups are always connected;
3. The system does not start at a saddle point or global maximum of $V(\mu)$;
4. Consensus estimators are executed instantaneously ($T_{max} = 0$);
5. Perfect and immediate communication between robots;

Thus, it is desired that, by applying the equation controller (3.51), the system converge to a state in which all robots in the same group are grouped while separated from robots in the other groups.

These considerations are based on the proof of Section 3.2, but with some clearly unrealistic assumptions, such as the fact that one can execute consensus estimators in null time (assumption 4). Assumptions 1 and 2 are strong, but not unrealistic. It is possible to design swarms of robots having these characteristics. Assumption 5 is not very strong in the sense that we can consider that the robots have fast and reliable communication, therefore disregarding lost packages and delays.

Assumption 3 and the part of assumption 1 that states that robots of the same group are not in the same point at the same time are the same assumptions as those in Section 3.2 and are related to the conditions for the potential function to converge to situations where there are no groups at a distance smaller than d_α when $t \to \infty$.

Considering the assumption 4 and knowing that the fixed time consensus always converges to the average, one has that all the states of abstractions necessary for the calculation of individual control (equation (3.51)) need to be computed instantly and accurately. Thus, one has that all estimated variables will be exactly the same as the corresponding variables for all robots: $\hat{\mu}_j^x = \mu_j^x$, $\hat{\mu}_j^y = \mu_j^y$, $\dot{\hat{\mu}}_j^x = \dot{\mu}_j^x$, $\dot{\hat{\mu}}_j^y = \dot{\hat{\mu}}_j^y$, $\hat{\sigma}_j = \sigma_j$, $\dot{\hat{\sigma}} = \dot{\sigma}$ e $\hat{\sigma'}_j = \sigma'_j$.

Note that robots get state information from other groups from the nearest robots belonging to these other groups to calculate the control action ${}^{\hat{\mu}}\boldsymbol{U}_j^k$ to separate groups. However, each robot can communicate with robots from different groups, so even with "ideal" estimators each robot gets a ${}^{\hat{\mu}}\boldsymbol{U}_j^k$ computation individually.

The controller that dictates the evolution of the size of abstractions (${}^{\sigma}U_j^k$) will be exactly the same as the controller with the same purpose in Section 3.2, only the control action that dictates the separation of groups (${}^{\hat{\mu}}\boldsymbol{U}_j^k$) will be different. This difference is because each robot can have communication with different robots from other groups. It is not assumed that robots in one group maintain communication with robots in other groups. If a robot has no communication with any robot from another group, then the part of its control action that dictates the separation of the groups (${}^{\hat{\mu}}\boldsymbol{U}_j^k$) will be equal to zero. The potential function acts by applying a virtual force in the center of the group of robot k to every robot from another group within its communication radius, as exemplified in the Figure 3.11. This makes this robot and consequently, its group, move in order to "repel" neighboring groups.

To guarantee that the k-th robot of group i has information about at least one robot of the j-th group, when the j-th group is in the neighborhood of i, we consider that:

$$R_{com}^2 \geq (R_i + r)^2 + R_j^2 \tag{3.61}$$

$$R_{com} \geq \sqrt{(R_i + r)^2 + R_j^2}. \tag{3.62}$$

As it can be seen in Figure 3.10, with this restriction, in the worst case, when a robot is in the furthest point from the other abstraction, it can still "see" the mean of the other

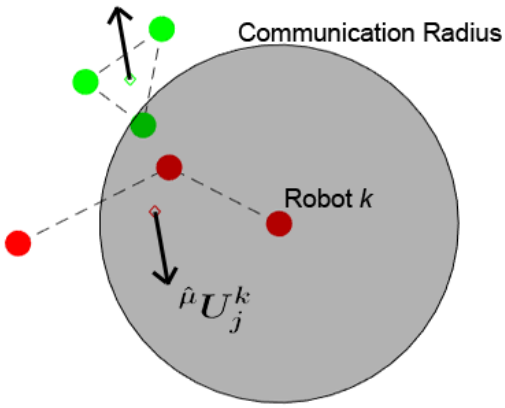


Figure 3.10: Potential force acting at the center of the groups.

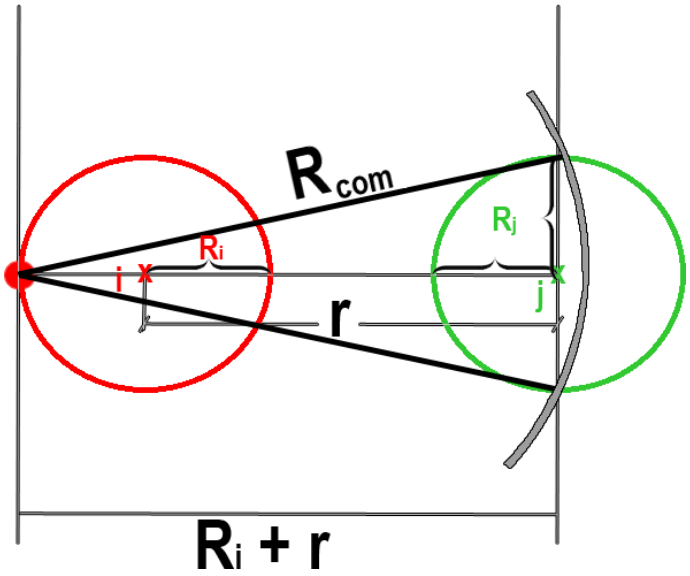


Figure 3.11: Relation between the communication radius and the parameters of the abstractions and potential function: (R_{com}, r, R_i, R_j).

abstraction, which means that it can "see" at least a robot from the other abstraction.

Thus, as the abstractions will reach the desired size (σ^{des}) and this size is such that there are no intersections between the abstractions (see Section 3.2) and the robots will always "repel" neighboring abstractions that are invading its abstraction, the problem of segregating swarms of robots using local information as defined in Section 3.1.2 is solved, as exemplified by the simulations in Section 5.1.2.

Note that this controller has some disadvantages compared to the controller of Section 3.2. First, collisions are not handled and a local controller would be needed to avoid collisions so that this controller could be implemented in real robots. In addition, this new controller has a clearly unrealistic assumption (assumption 4), because it is not possible to make calculations that demand T_{max} seconds instantly. In practice, after this calculation is performed, the robot has already moved so it would be necessary to recalculate. This

consideration is less unrealistic if each robot has great internal processing capabilities. Finally, there is no proof of convergence for this decentralized controller.

As future work, we intend to model the errors of the estimates as uncertainties so that we can consider the real effects of T_{max}. With that, we want to develop a convergence proof that is robust to these effects.

This controller also has a great advantage over the controller of the Section 3.2 and some controllers for segregation found in the literature (see Section 2.2): each robot do not need global information, and yet, the system can apparently reach segregation.

4

Hierarchy Based Segregation

In this chapter a new idea is introduced in order to obtain controllers to reach segregation in swarms of robots. The idea consists in using a consensus based algorithm to position robots and a heuristic in which robots exchange information in order to decide where they should be positioned according to which group they belong. This idea is different from the one presented in Chapter 3, however, the intention is to solve similar problems.

The same idea is used to propose controllers for two problems: to segregate groups of robots into clusters (similar to the problem defined in Chapter 3) and to segregate groups of robots radially. Next, in Section 4.1, we lay a background that will be common for both problems. In Sections 4.2 and 4.3 we present the methodology for each problem.

In the methodology of Section 4.2, only one scenario is considered, which requires robots to communicate by means of an underlying fixed topology and have knowledge of a previously defined parameterized curve. Also, collisions amongst robots are considered. In the methodology of Section 4.3, we consider two different scenarios, in which the first scenario has the same assumptions of the scenario considered in Section 4.2, except that in that section the knowledge of a curve is not required. The second scenario does not require an underlying communication topology but does require that all the robots have the knowledge of the same fixed point in the environment. In this section, collisions amongst robots are disregarded.

4.1 Background

Consider N holonomic robots moving in a two-dimensional Euclidean obstacle-free environment.

The dynamics of each robot is given by the double integrator, as in Chapter 3.

$$\dot{\boldsymbol{q}}_{\boldsymbol{i}} = \boldsymbol{v}_{\boldsymbol{i}}, \quad \dot{\boldsymbol{v}}_{\boldsymbol{i}} = \boldsymbol{u}_{\boldsymbol{i}}, \quad i = 1, 2, ..., N; \tag{4.1}$$

in which $\boldsymbol{q_i} = [x_i; y_i]^T$, $\boldsymbol{v_i} = [\dot{x}_i; \dot{y}_i]^T$ and $\boldsymbol{u_i} = [u_{xi}; u_{yi}]^T$ are the position, velocity and control input vectors for the 2D case, respectively.

Each robot is assigned to a group N_k, $k \in \mathcal{M} = \{1; 2; ..., M\}$ and M is the number of groups. Therefore, the system is composed of N robots divided into the groups $N_1, N_2, ..., N_M$. Robots of the same group are considered to be robots of the same type.

Note that there are some differences in the initial definitions in relation to the definitions in Chapter 3, namely, in this chapter we do not use indexes to assign robots to groups as the controller in this chapter will be defined for all the robots of the system regardless of the robot's group.

4.1.1 Required Information

Consider that each robot has a communication radius R_{com} that is the same for all the robots in the system. An example of this communication radius is showed in Figure 4.1, in which the bigger blue circle shows the radius R_{com} for the blue robot. It can be seen that the blue robot has communication with two other robots (one red and one green) and does not have communication with the green robot that is outside the radius R_{com}.

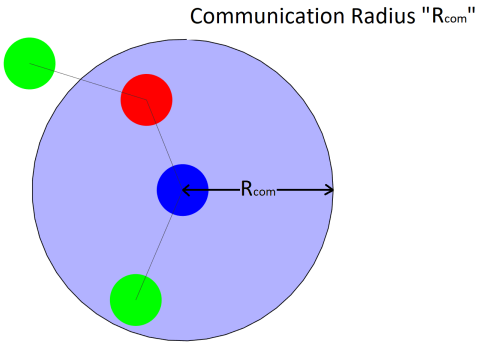


Figure 4.1: Communication radius example.

Consider the group of all robots in the system: $\mathcal{R}$. Also consider the previously defined groups of robots of the same type N_k. We now define an ordered set of groups: $\mathcal{G} = \{N_1, N_2, ..., N_M\}$. We assume that this set of groups is a totally ordered set with a pre-defined binary relation ($<$). Consider the mapping that associates each robot to its

corresponding group: $h : \mathcal{R} \rightarrow \mathcal{G}$. As $\mathcal{G}$ is a totally ordered set with a pre-defined binary relation, we can define a hierarchy such that:

$$h(R_{N1}) < h(R_{N2}) < ... < h(R_{NM}), \tag{4.2}$$

where R_{Nk} is any arbitrary robot of group N_k. We also define that $h(N_k)$ returns the corresponding value to the robots of group k. Given the order in (4.2) when a robot from group 1 has communication with a robot from group 2, the robot from group 1 knows that the robot from group 2 is from a group with higher order in the hierarchy ($h(N_2) > h(N_1)$) and exchange information based on this hierarchy, when they are close enough. **We assume that robots do not have the information of how many groups there are in the system or how these robots are distributed in groups. Moreover, although we do not assume the robots know the whole set order, we do assume that they are able to compute the result of a comparison with robots of other groups according to the binary relation ($<$).** Thus, when robot i (R_i) meets robot j (R_j) they are able to access the result of the comparison $h(R_i) < h(R_j)$. This ability to compare will be useful when defining a heuristic to dynamically allocate different desired positions to different groups.

In the next two sections we present two methodologies: in Section 4.2 we deal with the problem of segregating groups of robots into clusters and in Section 4.3 we deal with the problem of segregating groups of robots radially.

4.2 Methodology - Segregation in Curves

All the development in this section can be directly extended for robots in a three dimensional environment. Note that this is not the case for the development in Section 4.3.

The dynamics of each robot is given by (4.1). In which, for the 3D case, the position, velocity and control input vectors are, respectively: $\boldsymbol{q_i} = [x_i; y_i; z_i]^T$, $\boldsymbol{v_i} = [\dot{x}_i; \dot{y}_i; \dot{z}_i]^T$ and $\boldsymbol{u_i} = [u_{xi}; u_{yi}; u_{zi}]^T$.

4.2.1 Problem definition

Consider an unlimited open curve without self $\boldsymbol{s}(d_i) : \mathbb{R} \rightarrow \mathbb{R}^D$ We assume that each robot has the knowledge of the parametric equations of this open curve with the parameter given by the length from a point considered to be the origin, i.e. they can retrieve the coordinates in the curve when given the length from its origin.

Our goal is to investigate the problem of segregation in swarms of heterogeneous robots. All the robots should converge to a state where robots of the same type are close while they are separated from robots of different types, i.e., robots must form clusters with other robots of the same type.

In this section we present a new approach to segregate swarms of robots into clusters given the knowledge of a curve. The approach is based on the use of a formation control scheme where the desired formation pattern changes according to the robot groups in a way that each group travel different distances from the origin of the curve $\boldsymbol{s}(d_i)$.

4.2.2 Specific Required Information

In this section, we consider that robots can exchange information in two manners: (i) through an underlying fixed communication topology; (ii) when they are close enough. The underlying topology does not depend on the position of the robots, and we assume that the topology is fixed and connected all times. A communication graph is induced using both (i) and (ii). This graph is built considering the robots as nodes and defining edges between two robots if they are connected via the underlying fixed communication topology or if they are in the communication range of one another. Please refer to the Appendix A.1.1 to review the basic concepts in graph theory.

Moreover, we assume that robots can compute the comparison as stated in Section 4.1.1.

Finally, all robots must have the knowledge of the parametric equations of the same open curve.

4.2.3 Formation Control

Our method builds upon a formation control algorithm with collision avoidance adapted from the work of Mondal & Jamshidi (2017), in which the author proposes a trajectory tracking consensus algorithm with collision avoidance and connectivity assurance. In our work, as we want to segregate groups of robots, connectivity assurance is disregarded. Furthermore, as we are interested in reducing the quantity of information each robot requires, we use an absolute velocity damping term instead of the relative velocity damping term and the formation velocity used in Mondal & Jamshidi (2017).

Consider that the i-th robot has a desired position vector associated with it. Each desired position is given by the terms of a parametric equation of a given open curve, $\boldsymbol{s}(d_i)$, plus a component $\boldsymbol{w_i}$. The parameter d_i can be seen as the distance that the robot has to travel along the curve from its origin. Component $\boldsymbol{w_i}$ is a random vector to be added to a point on the curve so that robots converge to a region near that point. Those parameters will be better explained in Section 4.2.4. In this section, as in Mondal & Jamshidi (2017), we use the position error as the consensus variable,

$$\boldsymbol{e_i} = \boldsymbol{q_i} - \boldsymbol{s}(d_i) - \boldsymbol{w_i}, \tag{4.3}$$

in which $\boldsymbol{s}(d_i) + \boldsymbol{w_i}$ is the desired position for the i-th robot.

Consider the following formation maintenance control law with relative position error measurements and an absolute velocity damping term

$$\boldsymbol{u}_i^{for} = \underbrace{-\sum_{j=1}^{N} a_{ij}(\boldsymbol{e_i} - \boldsymbol{e_j})}_{\text{Formation Control}} - \underbrace{\gamma \dot{\boldsymbol{q}}_{\boldsymbol{i}}}_{\text{Velocity Damping}}, \tag{4.4}$$

in which $a_{ij} = a_{ji}$ is given by the elements of an adjacency matrix from an arbitrary connected communication topology and $\gamma > 0$ is a fixed gain. The gain γ is associated in the performance of the consensus. It can be regulated to improve the speed in which the consensus is achieved. Any positive value ensure the consensus convergence.

4.2.4 Distance travelled

Each robot must compute its own parameter d_i to be able to compute (4.3) and then (4.4). The parameter d_i, also called the distance travelled, will be the composition of two terms:

$$d_i = r_i b_i. \tag{4.5}$$

The term r_i can be seen as robot's i estimated position of its group in the group hierarchy, and will be better explained in Section 4.2.4.1. The term b_i will be responsible to keep increasing the distance among groups while they are not segregated, and will be better explained in Section 4.2.4.2.

4.2.4.1 Estimated position on the group hierarchy

To assign the estimated position on the group hierarchy r_i to each robot we propose a heuristic that dynamically changes robot's r_i when a robot is able to exchange information with other robots that are within the communication radius R_{com}.

In Algorithm 4.1 we show the local control algorithm for robot i in which it is possible to see the heuristics to change its estimated position in the group hierarchy.

Each robot can perceive other robots within its communication radius and broadcast its own h_i, r_i and $\boldsymbol{w_i}$ (line 3). Furthermore, robots broadcast its own $\boldsymbol{e_i}$ and robots that are interconnected in the underlying topology can receive it (line 5). The robots also receive the broadcasted h_j, r_j, $\boldsymbol{q_j}$ and $\boldsymbol{w_j}$ from all the other robots within its communication radius (line 8).

In Algorithm 4.1, lines 9-13, when a robot i meets a robot j that belongs to a group that is lower in the hierarchy than the group of robot i ($h_i > h_j$), with an estimated position on the group hierarchy that is greater or equal to r_i, robot i change its estimated position on the group hierarchy with an increment of 1 with respect to r_j. This means that robot i, of the group higher in the hierarchy will move away from the beginning of

Algorithm 4.1 Control Algorithm for robot i.

1: Initialize $r_i = 0$, $\boldsymbol{w_i} = [0,0]^T$ or $[0,0,0]^T$, $W_i = 0$;
2: **while** Active **do**
3: Broadcast h_i, r_i, $\boldsymbol{w_i}$, $\boldsymbol{e_i}$;
4: **for all** $\boldsymbol{q_j}$ such that $a_{ij} = 1$ **do**
5: Receive $\boldsymbol{e_j}$;
6: **end for**
7: **for all** $\boldsymbol{q_j}$ such that $\|\boldsymbol{q_j} - \boldsymbol{q_i}\| < c$ **do**
8: Receive h_j, r_j, $\boldsymbol{q_j}$, $\boldsymbol{w_j}$
9: **if** $h_i > h_j$ **then** ▷ *Robot i belongs to higher hierarchy group in comparison to robot j.*
10: **if** $r_j >= r_i$ **then**
11: $r_i \leftarrow r_j + 1$
12: **end if**
13: **end if**
14: **if** $h_i \neq h_j$ **then** ▷ *Robots i and j are from different groups.*
15: Increment b_i in (4.6) ▷ *At the discrete times in which Robot i is within range of any other robot of a different group*
16: **end if**
17: **if** $h_i = h_j$ **then** ▷ *Robots i and j are from the same group.*
18: **if** $r_j > r_i$ **then**
19: $r_i \leftarrow r_j$
20: **end if**
21: **if** $\boldsymbol{w_j}$ is such that $\|\boldsymbol{w_j} - \boldsymbol{w_i}\| < c_{out}$ **then**
22: $W_i \leftarrow \text{sat}(W_i + \delta_W, \frac{c}{2\sqrt{2}})$
23: $\boldsymbol{w_i} \leftarrow \text{rand}(-W_i, W_i)$
24: **end if**
25: **end if**
26: **end for**
27: Keep running consensus (4.7)
28: Compute (4.5) with r_i and b_i
29: Move according to control law (4.11)

the curve, thus segregating itself from the robot j, that belongs to the group lower in the hierarchy.

In lines 14-16, when a robot i meets a robot j from a different group, robot i increases its segregation distance, as it will be clear in (4.6).

In lines 17-20, when a robot i meets a robot j from the same group, robot i receives the value of the estimated position on the group hierarchy of robot j and if this value is greater than the one robot i already has then sets its position to the same value received from robot j. This means that robot j had met another robot from another group that is lower in the hierarchy and is now broadcasting this information to robot i. Furthermore, in lines 21-24, if robots of the same group have desired positions that would activate the collision avoidance controller when they converge to this position, robot i chooses a new random position from an increased region centered at the point given by $s(d_i)$. Moreover, W_i is

the size of the region for the i-th robot, δ_W is a small fixed parameter that dictates how much the region grows each time and rand$(-W_i, W_i)$ is a function that returns values for the components of a vector from a uniform distribution in the interval $(-W_i, W_i)$. Column vector w_i is $2D$ or $3D$ depending on the case considered and W_i is a scalar. The square region is saturated by the function $sat(W_i + \delta_W, c/2\sqrt{2})$, thus, has maximum side size of $c/\sqrt{2}$. The saturation occurs to guarantee that robots of a group will remain in a region near a point in the curve related to that group, so that robots from other groups are guaranteed to have encounters at this region. The reason we consider this region will be better explained in Section 4.2.7.1.

Note that, in Algorithm 4.1, robots communicate in two different ways: in lines 4-6, robots are communicating only through the underlying fixed communication topology ($a_{ij} = 1$) and in lines 7-24 robots are communicating only when they are close enough ($\|\boldsymbol{q_j} - \boldsymbol{q_i}\| < c$).

If one robot has encountered at least one robot from every other group, the estimated position on the group hierarchy of robot i (r_i) will converge to its true position on the hierarchy. Our scheme was designed to make sure that robots have meetings with at least one robot of each of the other groups, or get this information from other groups, therefore, we have that the estimated position on the group hierarchy of robot i will always converge to the true position on the hierarchy. This fact will help in the convergence proof in section 4.2.7.

Figure 4.2 shows an example with 3 robots and 3 groups segregating using Algorithm 4.1 and a horizontal line. In this figure, all robots initialize its estimated position in the group hierarchy believing that its group is the first one in the hierarchy. Their current estimation are represented by the boxes. As robots have encounters, they update their estimated position in the group hierarchy and move accordingly. With enough encounters, robots segregate. As there are only one robot per group, lines 17-24 of algorithm 4.1 are never executed.

4.2.4.2 Segregation distance

Since we have interest in developing algorithms in which robots do not need to exchange information with all the other robots of the system, we will propose a decentralized scheme to compute a segregation distance (b_i) between groups so that groups keep segregating while there are robots from different groups "seeing each other", within range R_{com}. This is also interesting in the sense that one could change the system size (number of robots and groups) and this parameter would adjust so that this new system would also converge to a segregated state.

This segregation distance is initialized with zero for all robots and will increase when two robots, i and j, from different groups are within range from each other by performing

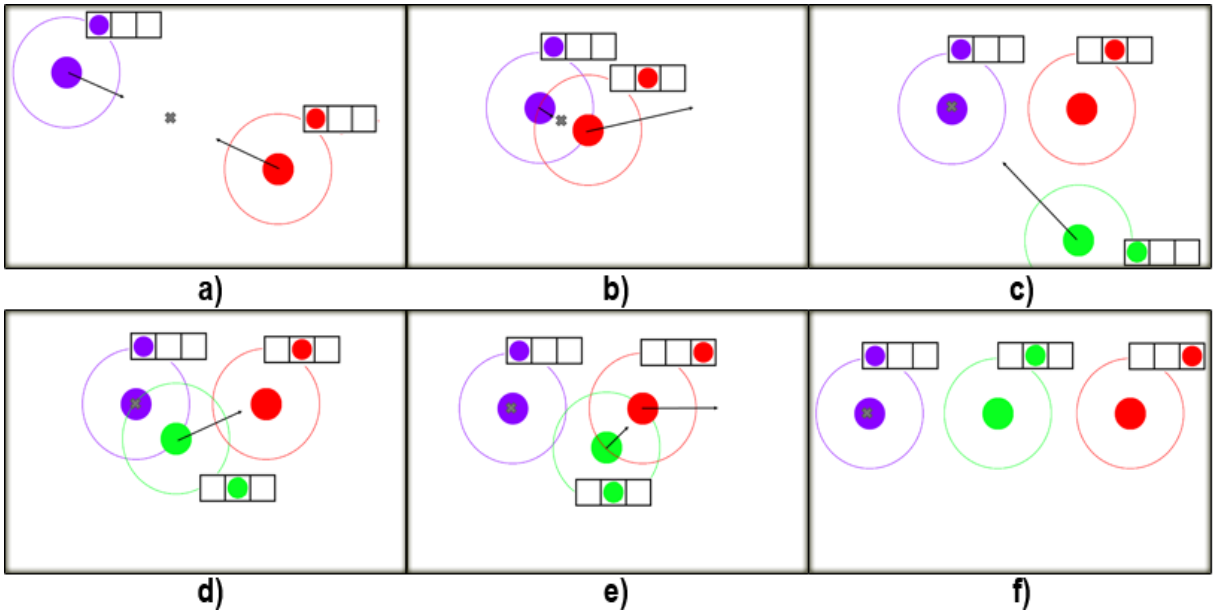


Figure 4.2: Example of the heuristics with 3 robots. Group order in the hierarchy in ascending order is: purple group, green group and red group. Smaller circles represent the robots and bigger circles represent the communication radius of the robots. Boxes indicate the estimated position on the group hierarchy of each robot. The curve used in a horizontal line. (a) Two robots of two different groups initiating consensus. (b) Robots are communicating for the first time, they compare their hierarchy. The red robot discover that it is higher in the hierarchy than the purple robot. (c) A third robot arrives, the green one. As the green robot has never meet another robot, it proceeds to the origin of the curve. (d) In the origin, the green robot and the purple robot have an encounter and the green robot update its estimation and goes to the next position in the curve. (e) The green robot have an encounter with the red robot. The red robot update its estimation and move to the next position in the curve. (f) Groups are segregated.

the following computation:

$$b_i \leftarrow b_i + \delta_b, \quad \text{and} \quad b_j \leftarrow b_j + \delta_b, \tag{4.6}$$

in which δ_b is a fixed small parameter.

After robots have increased b_i, if necessary, we have proposed an information consensus scheme to make all robots agree on the same segregation distance. Each robot will run a simple first order consensus algorithm:

$$b_i(N_s + 1) = b_i(N_s) - \sum_{j=1}^{N} a_{ij}(b_i(N_s) - b_j(N_s)), \tag{4.7}$$

in which $a_{ij} = 1$ if robots i and j are connected in the underlying communication topology and $a_{ij} = 0$, otherwise and N_s is the number of samples in Algorithm 4.1.

This consensus should always be running, and at discrete time events, b_i changes discretely according to (4.6) if there are robots of other groups within

the range R_{com}.

This exchange of information will guarantee that all the robots in the system will eventually agree with the same value for b_i and consequently, d_i, which will be important for the proof of convergence in section 4.2.6.

After updating its estimated position on the group hierarchy (r_i) and the segregation distance (b_i), robots can compute the distance travelled in the curve by means of (4.5), and consequently, robots can compute the consensus error (4.3) and then move accordingly. This can be seen in Algorithm 4.1, lines 27-29.

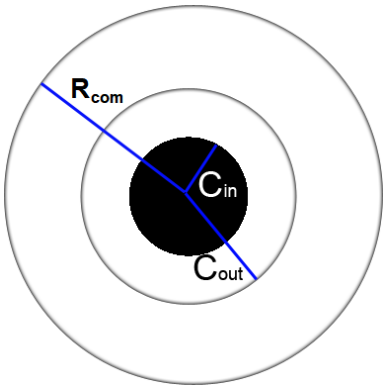


Figure 4.3: Scheme showing the distance in which the collision avoidance is activated (c_{out}), the considered robot size for the collision avoidance (c_{in}) and the communication range of a robot (R_{com}).

4.2.5 Collision Avoidance

Also consider a collision avoidance controller based on an artificial potential function, integrated with our controller, as in Mondal & Jamshidi (2017). We consider two circular regions around each robot, with radii c_{in} and c_{out} as can be seen in Figure 4.3. The collision avoidance region is bounded by those two circles. The collision avoidance term is active when a pair of robots i and j are inside the collision avoidance region, i.e., when two robots collision avoidance region intersect. This controller is zero when $\|\boldsymbol{q}_i - \boldsymbol{q}_j\| \geq c_{out}$. The potential function with a finite cutoff at c_{out} is given by Mondal & Jamshidi (2017):

$$\psi_{col}(x) = \begin{cases} \int_{c_{out}}^{x} \phi_{col}(s)ds, & \text{for } x \in [c_{in}, c_{out}) \\ 0, & \text{otherwise,} \end{cases} \tag{4.8}$$

in which $\phi_{col}(s)$ is such that $\psi_{col}(x)$ is strictly decreasing and has maximum value at c_{in}:

$$\phi_{col}(s) = -\frac{\|\boldsymbol{q}_{ij}\|}{(\|\boldsymbol{q}_{ij}\| - c_{in})^2 + \frac{1}{Q_{col}}}. \tag{4.9}$$

The collision avoidance for robot i is defined, as in Mondal & Jamshidi (2017):

$$\boldsymbol{u_i^{col}} = -\sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi^{col}(\|\boldsymbol{q_{ij}}\|). \tag{4.10}$$

in which $\mathcal{N}_i$ is the set of all the robots within a distance smaller than c_{out} from robot i, $\|\boldsymbol{q}_{ij}\|$ is the distance between robots i and j and $Q_{col} > 0$ in 4.9 is a fixed parameter that will be better explained in section (4.2.7.2).

4.2.6 The Complete Control Law

Consider:

1. The formation control to make all robots converge to a region near a curve (section 4.2.3);
2. The heuristics to decide where in the curve robots should converge depending on its group (section 4.2.4);
3. The collision avoidance controller (section 4.2.5).

By combining those controllers, we can now define the complete control that will guide the movement of each robot. First using the heuristics of section (4.2.4) robots can compute (4.5). Then, using the definition (4.3), robots can completely define (4.4). Finally, composing (4.4) and (4.10) we can move the robots modeled by the dynamics of (4.1). Thus, each robot will be guided by the control law

$$\boldsymbol{u_i} = \underbrace{-\sum_{j=1}^{N} a_{ij}(\boldsymbol{e_i} - \boldsymbol{e_j}) - \gamma\dot{\boldsymbol{q}}_{\boldsymbol{i}}}_{\boldsymbol{u_i^{for}}} \underbrace{- \sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi^{col}(\|\boldsymbol{q_{ij}}\|)}_{\boldsymbol{u_i^{col}}}. \tag{4.11}$$

in which a_{ij} are the elements of the adjacency matrix of the fixed underlying topology. Also, $\boldsymbol{u_i^{for}}$ is the formation control controller given by (4.3) and $\boldsymbol{u_i^{col}}$ is the collision avoidance controller given by (4.10).

4.2.7 Controlled System Analysis

In order to show how our approach will lead groups of robots to a segregated state, first we show how the controller (4.11) will lead each robot to its desired position without collisions. After that we show how our heuristics to compute the desired position for each robot will lead groups to a segregated state.

4.2.7.1 Ensuring formation control and collision avoidance

Our controller (4.4) is a simplified version of the controller proposed by Mondal & Jamshidi (2017). In comparison to the controller used in Mondal & Jamshidi (2017), in our controller we disregard the connectivity maintenance term, we also disregard the formation velocity consensus term, and we set the desired formation velocity to zero.

Furthermore, in our approach, the desired formation will change at discrete times. Robots from the same group will acquire a new desired position whenever their desired position will mean that the collision avoidance term will be active, when the desired position is reached. As the robots draw new desired position vectors randomly from an increasing area centered at a point in the curve, eventually all the robots will acquire points that will mean that they have reached a valid formation. We are assuming uniform distribution in our sampling, and thus if the relation between R_{com} and c_{out} is adequate, arguments similar to the ones to show probabilistic completeness of sampling based planners can be used to show that the probability of sampling a valid configuration tends to 1 as the number of samples goes to infinity. A valid formation is the one in which all robots are at a distance greater than c_{out}, meaning that the collision avoidance term is not active. Also, as robots only draw new positions when they are close and the area that they draw from increases with small increments, and saturated at $c/2\sqrt{2}$, robots of the same group will remain close. Note that the valid formation is equivalent to the formations considered in Mondal & Jamshidi (2017), disregarding the connectivity maintenance part.

In Figure 4.4 we show an example of a curve, a region in which robots draw desired positions, and two robots (one blue and one red). The desired point in the curve ($s(d_i)$) is depicted by a black dot. Considering that the region is saturated at $c/2\sqrt{2}$, this is the case in which the robots are farther apart, inside the region. As one can see in the Figure 4.4, in this case, the worst one, robots are still in the communication range of each other. This is the reason for the use of the saturation function (line 22, Algorithm 4.1).

The scheme to make a robot acquire a new position constrained in a square region is designed so that when other robots are in the same region, there will be a encounter. This fact will ensure that the robots of each group converges to the region they should converge, as it will be show in 4.2.7.3.

In order to show that robots will converge to the desired positions, without the occurrence of collisions between robots we rely on the same reasoning of Mondal & Jamshidi (2017) but using the following positive semi-definite function instead of the one used in Mondal & Jamshidi (2017):

$$V_E = \frac{1}{2}\sum_{i=1}^{N}\left[\sum_{j\in\mathcal{N}_i}\psi_{col}(\|\boldsymbol{q}_{ij}\|) + \boldsymbol{v}_i^T\boldsymbol{v}_i\right] + \frac{1}{2}\boldsymbol{e}^T(L\otimes I_D)\boldsymbol{e}. \tag{4.12}$$

in which L is the Laplacian matrix related to the fixed underlying topology of the system,

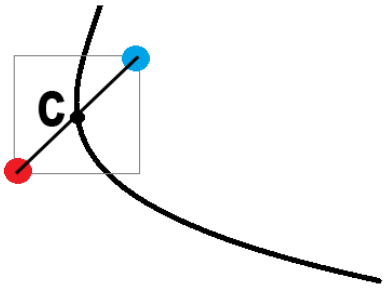


Figure 4.4: Example of a curve, two robots and the region that robots draw desired positions from.

$I_D \in \mathbb{R}^{D\times D}$ is the identity matrix, D is the dimension of the system (2 or 3), $\otimes$ denotes the Kronecker product and $\boldsymbol{e}$ is the stacked vector of the errors $\boldsymbol{e} = [\boldsymbol{e}_1^T, \boldsymbol{e}_2^T, ..., \boldsymbol{e}_N^T]$. Also consider the time derivative of (4.12):

$$\dot{V}_E = \sum_{i=1}^{N} \left[\dot{\boldsymbol{q}}_i{}^T \sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi_{col}(\|\boldsymbol{q}_{ij}\|) + \boldsymbol{v}_i^T \dot{\boldsymbol{v}}_i \right] + \boldsymbol{e}^T(L\otimes I_D)\dot{\boldsymbol{e}}. \tag{4.13}$$

Following a similar development as in Mondal & Jamshidi (2017), and using (4.1), we have that

$$\dot{V}_E = \sum_{i=1}^{N} \left[\boldsymbol{v}_i{}^T \sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi_{col}(\|\boldsymbol{q}_{ij}\|) + \boldsymbol{v}_i^T \boldsymbol{u}_i \right] + \boldsymbol{e}^T(L\otimes I_D)\boldsymbol{v}. \tag{4.14}$$

Substituting from (4.11)

$$\begin{aligned}\dot{V}_E = \sum_{i=1}^{N} \Bigg[& \boldsymbol{v}_i^T \sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi_{col}(\|\boldsymbol{q}_{ij}\|) \\ & + \boldsymbol{v}_i^T \left\{ - \sum_{j\in\mathcal{N}_i} \nabla_{q_i}\psi^{col}(\|\boldsymbol{q}_{ij}\|) - \sum_{j=1}^{N} a_{ij}(\boldsymbol{e_i} - \boldsymbol{e_j}) - \gamma \boldsymbol{v_i} \right\} \Bigg] \\ & + \boldsymbol{e}^T(L\otimes I_D)\boldsymbol{v}. \end{aligned} \tag{4.15}$$

Rearranging

$$\dot{V}_E = -\sum_{i=1}^{N} \boldsymbol{v}_i^T \sum_{j=1}^{N} a_{ij}(\boldsymbol{e_i} - \boldsymbol{e_j}) - \gamma \sum_{i=1}^{N} \boldsymbol{v}_i^T \boldsymbol{v}_i + \boldsymbol{e}^T(L\otimes I_D)\boldsymbol{v}. \tag{4.16}$$

Using the property of Laplacians

$$\sum_{i=1}^{N} \boldsymbol{v}_i^T \sum_{j=1}^{N} a_{ij}(\boldsymbol{e_i} - \boldsymbol{e_j}) = \boldsymbol{v}^T(L\otimes I_D)\boldsymbol{e} \tag{4.17}$$

and the fact that $\boldsymbol{e}^T(L \otimes I_D)\boldsymbol{v}$ can be writen as $\boldsymbol{v}^T(L \otimes I_D)\boldsymbol{e}$ we have that

$$\dot{V}_E = -\gamma \sum_{i=1}^{N} \boldsymbol{v}_i^T \boldsymbol{v}_i \tag{4.18}$$

$$\dot{V}_E \leq 0. \tag{4.19}$$

$\dot{V}_E$ is negative semidefinite, thus, V_E is a nonincreasing function and will never become unbounded.

Also, following the same reasoning as Mondal & Jamshidi (2017), applying *Barbalat's* lemma, one can show that $\dot{V}_E = -\gamma \sum_{i=1}^{N} \boldsymbol{v}_i^T \boldsymbol{v}_i \to 0$ as $t \to \infty$. This implies that $\boldsymbol{v}_i \to 0$ as $t \to \infty$ for all i and $\dot{\boldsymbol{v}}_i = \boldsymbol{u}_i \to 0$, which is possible only when all the individual control parts become zero. Using (4.11), $\boldsymbol{e}_i \to \boldsymbol{e}_j$ and $\boldsymbol{v}_i \to 0$ as $t \to \infty$ $\forall j \in \mathcal{N}_i$. Thus, the formation position error consensus is achieved and the velocity of all robots goes to zero. This means that robots will converge to a region near the point $s(d_i)$. This region is constructed in a way that robots are close to the point $s(d_i)$ and from other robots from the same group but are not close enough to activate the collision avoidance term $\boldsymbol{u}_i^{col}$.

4.2.7.2 Design of collision avoidance potential function

To ensure that there will be no collisions between robots, We can compute Q_{col} according to Mondal & Jamshidi (2017). Assume

$$\bar{\phi}_{col}(\|\boldsymbol{q}_{ij}\|) = \frac{\|\boldsymbol{q}_{ij}\|}{(\|\boldsymbol{q}_{ij}\| - c_{in})^2} \tag{4.20}$$

and

$$\bar{\psi}_{col}(\|\boldsymbol{q}_{ij}\|) > \psi_{col}(\|\boldsymbol{q}_{ij}\|). \tag{4.21}$$

Also assume

$$\bar{V}_E = \frac{1}{2} \sum_{i=1}^{N} \left[\sum_{j \in \mathcal{N}_i} \bar{\psi}_{col}(\|\boldsymbol{q}_{ij}\|) + \boldsymbol{v}_i^T \boldsymbol{v}_i \right] + \frac{1}{2} \boldsymbol{e}^T (L \otimes I_D)\boldsymbol{e}, \tag{4.22}$$

Clearly $\bar{V}_E(t) \geq V_E(t)$, $V_E(t) \leq V_E(0)$, thus we can choose

$$E_0 = \bar{V}_E(0). \tag{4.23}$$

The collision avoidance potential function ψ_{col} is designed so that

$$V_E(t) \leq V_E(0) \leq E_0 < \psi_{col}^{max}(\|\boldsymbol{q}_{ij}\|) \tag{4.24}$$

$\forall i, j \in \mathcal{N}_i$. The potential will give its maximum value when $\|\boldsymbol{q}_{ij}\| = c_{in}$, hence

$$\psi_{col}^{max}(\|\boldsymbol{q}_{ij}\|) = Q_{col} * c_{in} * (c_{out} - c_{in}) \tag{4.25}$$

$\forall i, j \in \mathcal{N}_i$.

$$Q_{col} * c_{in} * (c_{out} - c_{in}) > E_0 \geq V_E(0) \tag{4.26}$$

which implies

$$Q_{col} > \frac{E_0}{c_{in}(c_{out} - c_{in})}. \tag{4.27}$$

For a given desired position and initial position of the robots and c_{in}, c_{out}, value of Q_{col} can be calculated using (4.27). This designed value ensures collision avoidance. For V_E to become unbounded, ψ_{col} has to become unbounded for at least one robot. The collision avoidance tends to become unbounded when

$$\|\boldsymbol{q}_{ij}\| \to c_{in}. \tag{4.28}$$

But we have shown in (4.18) that V_E is always bounded, so ψ_{col} never becomes unbounded, for any robot i. Thus for any pair of robots i and j,

$$c_{in} < \|\boldsymbol{q}_{ij}\| \tag{4.29}$$

which ensures no collision between robots.

Finally, as now we know that the robots will converge to their desired positions, with zero velocity and without collisions, we need to show that the proposed heuristics is able to find such desired positions to ensure segregation between groups.

4.2.7.3 Ensuring segregation between groups

Now, we show how our heuristics to dynamically assign a desired position on the curve (d_i) to each robot will make the system always reach a segregated state.

Theorem 2. *Assume the following hypotheses:*

(i) Individual robots are governed by the dynamics in (4.1) with communication radius R_{com};

(ii) There is a connected underlying communication topology and a global knowledge of an unlimited curve;

(iii) Groups and a binary relation between groups are defined in such a way that a strictly totally ordered set of groups is induced;

(iv) Each robot i is able to compute if the order of its group is greater, equal, or less than the order of the group of any other robot j according to the predefined binary relation when the information about the group of robot j is made available;

(v) The relation between R_{com} and c_{out} is such that all the robots in each group can fit in a square region of side given by $c/\sqrt{2}$ in a way that collision avoidance is not activated.

Then, by applying the Algorithm 4.1 in the control of each individual robot, the probability of the multi-robot system converge to a segregated state as defined in Section 4.2.1 tends to 1 as $t \to \infty$.

Proof. From facts (iii) and (iv) we can assume that Algorithm 4.1 can run as all the comparisons can be properly computed. From facts (ii), (v) and the analysis in section 4.2.7.1 we know that robots will converge to a region near the point in the curve $\boldsymbol{s}$ at a distance d_i from the origin of the curve with zero velocity, and the probability of convergence to a valid configuration goes to 1 as N_s (number of samples in Algorithm 4.1) goes to infinity, assuming the proper relation between R_{com} and c_{out}. Therefore, in order to show segregation we need to show that the distance d_i of each group is such that robots of the same group remain close while apart from robots of different groups.

We will show that robot's estimated position on the group hierarchy (r_i) will converge to the real position of the group in the hierarchy and we will show that the segregation distance (b_i) will keep increasing in a way that group segregation is achieved and will be the same for all robots in the system.

We will first show that the first group's estimated position on the group hierarchy will converge to $r_i = 0$. Then, employing induction, we follow to show that the other groups estimation will increase according to its group order in the group hierarchy.

According to Algorithm 4.1, all the robots start with $r_i = 0$ and the only possible changes in the parameter r implies that $r_i = \lambda$, where $\lambda \in \mathbb{N}$. The changes can only occur when robots meet within a radius R_{com}, which is the same for every robot. Moreover, the parameter r never decreases, it might only increase in case robot i receives the information about the existence of another robot j of a different group so that $h_i > h_j$ and $r_j \geq r_i$ or another robot j of the same group so that $r_j \geq r_i$. As the set of groups is a strictly totally ordered set, and the changes are given by $r_i = r_j + 1$ for $h_i > h_j$ or $r_i = r_j$ for $h_i = h_j$ it is guaranteed that the parameter r of the robots of the group which is the least element of the set, i.e. $h_1 < h_j \; \forall j$, never changes, i.e., $r_1 = 0$. This implies in the convergence of the first group's estimated position on the group hierarchy to its real position on the group hierarchy, consequently, it implies in the convergence of the first group to the beginning of the curve $\boldsymbol{s}$, i.e $d_1 = 0$.

Now consider the hypothesis: all the robots of groups $1, 2, ..., k$, where $h(N_1) < h(N_2) < ... < h(N_k)$, have converged to the corresponding $r_{N1} = 0, r_{N2} = 1, r_{N3} = 2, ..., r_{Nk} = k - 1$. According to Algorithm 4.1 and the strict total order it is impossible to have a change in r_{Nk+1} of a robot of group N_{k+1} when meeting robots of groups $N_{k+2}, N_{k+3}, ..., N_M$ as $h(N_{k+1}) < h(N_{k+2}) < h(N_{k+3}) < ... < h(N_M)$. From this and the initial conditions, $r_i = 0 \; \forall i$,

we can conclude that r_{Nk+1} of group N_{k+1} must converge to $r_{Nk+1} = \lambda$ where $\lambda \in \{0, 1, 2, ..., k\}$.

From (4.7) and fact (ii), we can conclude that there is a finite time t and a small value ϵ such that the variables of the system reach a value with a distance smaller than ϵ from the mean of initial values, from the instant in which b_i stopped increasing, $\forall i$. This means that, eventually, $\lim_{t\to\infty} d_i = b\lambda$, $\lambda \in \{0, 1, 2, ..., k\}$ in which b acts only as a gain and is the average value of b_i, for all robots in the system. Therefore, robots will only move to regions near fixed points in the curve. Given this fact and the hypothesis of convergence of groups $N_1, ..., N_k$ we can guarantee that a robot from group N_{k+1} will always receive information about the existence of other robots in the same curve and the corresponding value $h(N_l)$ for comparison when $r_i = \lambda$ with $\lambda \in \{0, 1, 2, ..., k\}$. From this we can conclude that it is impossible for a robot of group N_{k+1} to converge to a region near the point on the curve equivalent to $r_i = \lambda$ with $\lambda \in \{0, 1, 2, ..., k-1\}$. According to Algorithm 4.1 and the convergence of the consensus protocol (4.7), being aware of robots already in their correct region near the position on the curve implies in the increment of the radius of the robots of group N_{k+1}. Therefore, we can conclude that the only possible region near the position on the curve for convergence is the one where $r_i = k$, i.e. $d_i \approx bk$.

By induction, we can conclude that each robot i of group N_l in the group hierarchy will converge to $r_i = (l-1)$, $\forall i, \forall l$. Also, given the fact that b_i tends to be the same in the limit for all the robots and the fact that b_i will only increase when robots from one group are seeing robots from other groups (equation (4.6)), we have that b_i will keep increasing while robots from different groups are near. Therefore, considering that the desired position of each robot in the curve (d_i) will always be such that no robot from different groups are within range from each other, $\boldsymbol{u_i^{for}}$ will guide robots of the same group to be close to the same point in the curve. Nevertheless, as robots do not collide, they will form clusters near the desired position on the curve for its group. If in the formed cluster a robot is within range from another robot from other group, b_i will increase and make sure the distance from different groups increase, thus, segregation will always be achieved. □

4.3 Methodology - Radial Segregation

In chapter 3 and in the last section we have considered the so called cluster segregation. Now we consider a different problem: radial segregation.

One possible application of the radial segregation are scenarios where the task of multiple targets enclosure (Kubo et al., 2014) is considered, in which different targets have to be enclosed by different groups of robots. Furthermore, the formation of annular structures, and of center-periphery patterns in particular, might be useful in a range of applications. Examples include re-configurable nested membrane structures in biomedical applications and dynamically constructed defense structures in military applications (Chen et al., 2012).

4.3.1 Problem definition

We aim to investigate the problem of radially segregating autonomous swarms of robots considering only the use of local information. All the robots should converge to a state where robots of the same type are positioned at the same distance with respect to a given point while these distances are different for robots of different types.

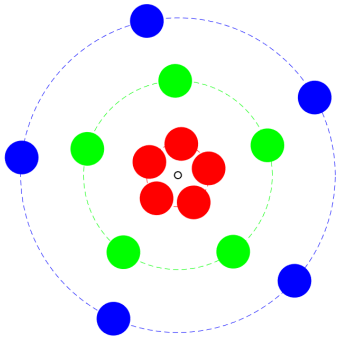


Figure 4.5: System of 15 robots divided in 3 groups radially segregated.

Figure 4.5 shows a system with 15 radially segregated robots divided in 3 groups in which robots of the same group have the same color. In Figure 4.5, dashed circles indicate that robots have indeed the same distance with respect to a given point that is represented by the small black circle. It is preferable to have robots of the same group equally distributed in the corresponding circle, as in figure 4.5, even though this is not a requirement.

In this section we present a new approach to radially segregate swarms of robots. The approach is based on the control of virtual points associated with the robots. The approach is similar to the approach presented in Section 4.2, in the sense that both use consensus algorithms to guide the movement of the robots and a heuristic to decide where groups should converge to. To facilitate the comprehension of each approach, in this section the consensus algorithm is formulated in the context of a *rendezvous* problem and in Section 4.2 it is formulated as a formation control problem although both algorithms have some similarities as it will become clear next. Note that in the approach of this section collisions between robots are disregarded.

4.3.2 Specific Required Information

In this section, we will consider two scenarios. *Scenario 1* has similar considerations to those of Section 4.2, in which robots can exchange information in two manners: (i) through an underlying fixed communication topology; (ii) when they are close enough. In *Scenario 2* we consider that robots only exchange information when they are close, however, we also consider that all robots have the knowledge of the same point in the environment.

A different communication graph is built depending on the considered *Scenario*, as it will be clear in Section 4.3.4.

Furthermore, the same assumptions regarding the knowledge of a hierarchy induced by an order in the set of groups with the definition of a binary relation ($<$) are considered, as explained in Section 4.1.1.

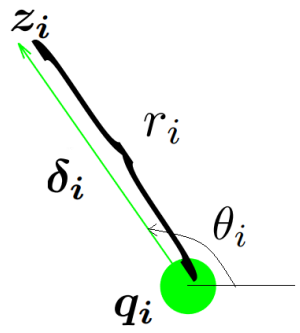


Figure 4.6: Example of virtual point for a robot.

4.3.3 Virtual points

Consider that the i-th robot has a virtual point z_i associated with it. Figure 4.6 shows an arrow pointing to the virtual point of a robot. Each virtual point has an angle θ_i and a radius r_i associated with it. Those variables are the polar coordinates of the point as seen from a frame attached to the robot. We assume that each robot's virtual point is initialized with a random angle θ_i and the same distance $r_i = d$, in which d is a parameter dependent of R_{com} and R_{com} is the communication radius of the robots. Therefore, for the i-th robot of the system we have that $\boldsymbol{z_i} = \boldsymbol{q_i} + \boldsymbol{\delta_i}$, in which $\boldsymbol{\delta_i} = [r_i \cos\theta_i \quad r_i \sin\theta_i]^T$. The robot dynamics is given by the double integrator (4.1), thus, to relate the motion of the robot with the motion of the virtual point we have that

$$\ddot{\boldsymbol{q}}_{\boldsymbol{i}} = \ddot{\boldsymbol{z}}_{\boldsymbol{i}} - \ddot{\boldsymbol{\delta}}_{\boldsymbol{i}}, \tag{4.30}$$

and differentiating $\boldsymbol{\delta_i}$ twice, with constant r_i, we have that

$$\ddot{\boldsymbol{\delta}}_{\boldsymbol{i}} = \begin{bmatrix} -r_i(\ddot{\theta}_i \sin\theta_i + \dot{\theta}_i^{\,2} \cos\theta_i) \\ r_i(\ddot{\theta}_i \cos\theta_i - \dot{\theta}_i^{\,2} \sin\theta_i) \end{bmatrix}. \tag{4.31}$$

In this differentiation we considered r_i as a constant. In our approach r_i will be allowed to change only at some specific discrete events instantaneously as it will be clear later when we present an algorithm to execute this change.

To achieve radial segregation in the system, we design a controller for the virtual points considering:

$$\ddot{\boldsymbol{z}}_{\boldsymbol{i}} = \hat{\boldsymbol{u}}_{\boldsymbol{i}}, \tag{4.32}$$

by enforcing $\boldsymbol{u_i} = \hat{\boldsymbol{u}}_i - \ddot{\boldsymbol{\delta}}_i$, from (4.1) and (4.30). This controller will be such that all virtual points converge to the same point eventually.

In this section, the main idea consists in using a consensus based algorithm to drive virtual points attached to the robots and a heuristic to define the radius, r_i, of the virtual point. Concomitantly, robots spread themselves along the virtual circles where virtual points *rendezvous*.

We consider two different scenarios in which we use the same main idea for radial segregation. The scenarios differ in the information which is available for the robots and the communication topology. Note that, in the controller shown in Section 4.2, only the *Scenario 1* is used.

4.3.4 Consensus Algorithm

In both scenarios the radius heuristics (Section 4.3.5) and the angle controller (Section 4.3.6) are used in the same way. The only difference is in the consensus algorithm. In both scenarios we use a consensus algorithm as it is usually done in the context of the multi-robot *rendezvous* (Jadbabaie & Morse, 2003). Next, we detail how the consensus controllers will differ from each other.

4.3.4.1 Scenario 1 - Underlying fixed communication topology

In this scenario, we consider that robots can communicate using a fixed underlying topology. Moreover, we assume that robots have no prior knowledge of a common reference point to guide the consensus algorithm. The communication topology must be connected and it is not dependent on the distance between pairs of robots. To control the virtual points we use the following consensus algorithm:

$$\hat{\boldsymbol{u}}_i = -\sum_{j=1}^{N} a_{ij}[(\boldsymbol{z_i} - \boldsymbol{z_j}) + \gamma(\dot{\boldsymbol{z}}_i - \dot{\boldsymbol{z}}_j)], \tag{4.33}$$

in which γ is a positive gain, $\dot{\boldsymbol{z}}_i = \dot{\boldsymbol{q}}_i + \dot{\boldsymbol{\delta}}_i$ and $a_{ij} = a_{ji}$ is given by the elements of an adjacency matrix associated with the connected communication topology.

4.3.4.2 Scenario 2 - Robots know a reference point

In this scenario we consider that all the robots know *a priori* exactly the location of a reference point. However, in this scenario robots cannot communicate outside a given range. We consider that the reference point is $\boldsymbol{o} = [0, 0]^T$, for the sake of simplicity.

We use the reference point in the consensus algorithm as if it were a fixed leader robot positioned at the reference point. Therefore, the global system configuration is given by: $\breve{\boldsymbol{q}} = [\boldsymbol{q_1}^T, \boldsymbol{q_2}^T, ..., \boldsymbol{q_N}^T, \boldsymbol{o}]^T$. Similarly, we consider also the extended $\breve{\boldsymbol{z}}$ and $\breve{\dot{\boldsymbol{z}}}$ (for $\boldsymbol{z}$ and $\dot{\boldsymbol{z}}$,

respectively) in which we consider $\breve{\boldsymbol{z}}_{N+1} = [0,0]^T$ and $\breve{\boldsymbol{z}}_{N+1} = [0,0]^T$. Thus, the consensus algorithm used for *Scenario 2* is:

$$\hat{\boldsymbol{u}}_{\boldsymbol{i}} = -\boldsymbol{z}_{\boldsymbol{i}} - \gamma\dot{\boldsymbol{z}}_{\boldsymbol{i}} - \sum_{j=1}^{N} a_{ij}[(\boldsymbol{z}_{\boldsymbol{i}} - \boldsymbol{z}_{\boldsymbol{j}}) + \gamma(\dot{\boldsymbol{z}}_{\boldsymbol{i}} - \dot{\boldsymbol{z}}_{\boldsymbol{j}})], \tag{4.34}$$

in which a_{ij} is given by:

$$a_{ij} = a_{ji} = \begin{cases} 1, & \text{if} \quad ||\boldsymbol{q}_{\boldsymbol{i}} - \boldsymbol{q}_{\boldsymbol{j}}|| \leq R_{com} \\ 0, & \text{otherwise.} \end{cases} \tag{4.35}$$

In (4.35), R_{com} is the communication radius of the robots. Note that when the virtual points converge to a *rendezvous* state, the distance from a robot to its virtual point will be the same as the distance from a robot to the reference point, as the rendezvous point is exactly the reference point.

Note also that (4.33) and (4.34) are direct applications of common consensus algorithms Ren & Atkins (2007), Ren (2007). In (4.33) and (4.34) we control virtual point positions to guide the virtual points (z) associated with all the robots to the same position.

Until now, nothing has been stated in regard to how the distances between the virtual points and the robots should be defined. In the next section we propose a heuristic to choose those distances for each robot dynamically to lead the system to radial segregation.

4.3.5 Radius Heuristics

To assign the virtual point radius r_i to each robot we propose a heuristic that dynamically changes robot's r_i when a robot is able to exchange information with other robots that are within the communication radius R_{com}. In *Scenario 1* robots exchange information through the underlying fixed communication topology (to be able to compute (4.33)) and also exchange information with other robots within the communication radius R_{com} (to be able to process Algorithm 4.2). In *Scenario 2* robots only exchange information with other robots within the communication radius R_{com}. The drawback for *Scenario 2* is that all the robots must know a common reference point. This is not a strong limitation since we can always think of a two-stage solution in which in the first stage a preliminary consensus protocol might be processed while the robots stay still in their initial positions to define the reference point as long as robots start in a connected topology. The second stage is then exactly the proposed approach for *Scenario 2*. In Algorithm 4.2 we show the local control algorithm for robot i in which it is possible to see the heuristics to change the radius.

Each robot can perceive other robots within its communication radius and broadcast its own h_i, r_i and h_i^d (line 3). In Algorithm 4.2, h_i^d is used to store information about robots from other groups and then broadcast this information to other robots. The robots also

Algorithm 4.2 Control Algorithm for robot i.

1: Initialize $h_i^d = 0$, $r_i = d$;
2: **while** Active **do**
3: Broadcast h_i, r_i, h_i^d
4: **for all** $\mathbf{q_j}$ such that $\|\mathbf{q_j} - \mathbf{q_i}\| < c$ **do**
5: Receive h_j, r_j, h_j^d
6: **if** $h_i > h_j$ **then** ▷ *Robot i belongs to a hierarchy higher group in comparison to robot j.*
7: **if** $r_j >= r_i$ **then**
8: $h_i^d \leftarrow 0$
9: $r_i \leftarrow r_j + d$
10: **end if**
11: **end if**
12: **if** $h_i = h_j$ **then** ▷ *Robots i and j are from the same group.*
13: **if** $r_j > r_i$ **then**
14: $h_i^d \leftarrow 0$
15: $r_i \leftarrow r_j$
16: **end if**
17: **end if**
18: **if** $h_j > h_i$ **then** ▷ *Robot i saving information.*
19: **if** $r_j > r_i$ **then**
20: $h_i^d \leftarrow h_i^d \cup \{(h_j, r_j)\}$
21: **end if**
22: **end if**
23: **if** $\exists (h_k, r_k) \in h_j^d$ such that $h_i > h_k$ and $r_i \leq r_k$ **then** ▷ *Robot i analyzing received information*
24: $h_i^d \leftarrow 0$
25: $r_i \leftarrow r_k + d$
26: **end if**
27: **end for**
28: Move according to control law (4.40);
29: **end while**

receive the broadcasted h_j, r_j and h_j^d from all the other robots within its communication radius (line 5).

In Algorithm 4.2, lines 6-11, when a robot i meets a robot j of a group that is lower in the hierarchy than the group of robot i ($h_i > h_j$), with a radius that is greater or equal to r_i, robot i change its radius to r_j plus a fixed parameter d. This means that robot i, of the group higher in the hierarchy will move away from the "rendezvous point" thus segregating from the robot j, of the group lower in the hierarchy.

In Algorithm 4.2, lines 12-17, when a robot i meets a robot j from the same group, robot i receives the value of the radius of robot j if this value is greater than the one robot i already has. This means that robot j had met another robot from another group that is lower in the hierarchy and is now broadcasting this information to robot i.

Due to the local nature of the approach, there can be situations where robots are "stuck" in the same radius with robots of groups with lower order. These situations occur

when robots converged to the "wrong" circle but do not have communication with other robots in this circle. To solve these situations, whenever a robot i communicates with a robot j of a group higher in the hierarchy that is more external in relation to the reference point, robot i stores robot's j hierarchy (h_j) and radius (r_j) in a list (lines 18-22).

When a robot i receives the broadcasted list from robot j, robot i analyzes the list to see if robot j has information of a robot k that is from a group lower in the hierarchy than robot's i group ($h_i > h_k$). If yes and the radius of robot k is greater or equal than the radius of robot i ($r_k \geq r_i$) then robot's i radius should increase (lines 23-27). Informally, it works like one robot told the other: I have seen a robot with a group hierarchy smaller than yours with a radius greater or equal to the radius that you have, then you should increase your radius. This exchange of information can be better visualized in the video presented in https://youtu.be/fohz_5DRmbI.

To guarantee that the robots that are "stuck" have meetings with other robots, we make all robots rotate around the reference point with a polar angular velocity that depends on its radius r_i. Therefore, as robots rotate with different angular velocities, they will eventually meet other robots.

When robots are rotating, the robots from an external radius will eventually exchange information with robots of immediate internal radius as long as some conditions are met, as shown next. The constant d regulates the distances between groups and must be such that:

$$d < 0.5R_{com}. \tag{4.36}$$

Equation (4.36) guarantees that sometimes robots have connections with at least one robot of an internal group (if an internal group exists). Given the initial condition $r_i = d$, if an internal group does not exist, equation (4.36) guarantees that robots are able to communicate with at least one other robot of the system when reaching the circle of radius d as exemplified in Figure 4.7.

Figure 4.7 shows a red robot and a blue robot. This figure captures the moment when two robots (a blue and a red) had converged to the first circle ($r_i = d\ \forall i$) and are meeting for the first time. After the blue robot communicates with the red, the blue robot would increase its radius (line 9) assuming the position of the group of the blue robot in the hierarchy is greater than the position of the group of the red robot. After updating its radius the robot moves according to the control law (4.40).

Now, to control the rotation of the robots around the reference point and to control the distribution of robots within the "desired radius" of its own group we define an angle controller as shown next.

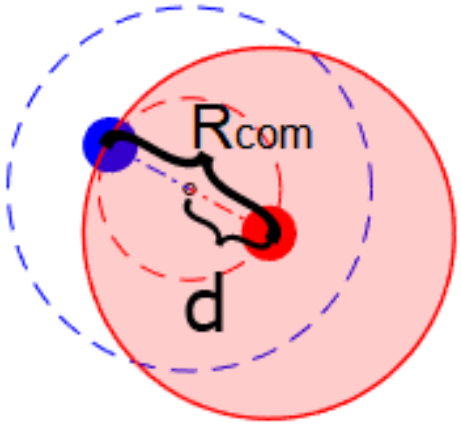


Figure 4.7: Example of constant d that satisfies (4.36).

4.3.6 Angle Control

It is preferable that robots of the same group distribute themselves uniformly along the virtual circle which is centered where the virtual points *rendezvous*, although it is not a requirement in the definition of the problem. Thus, we propose a controller for the dynamics of the angle θ_i, as follows

$$\ddot{\theta}_i = \tilde{u}_i = k_p\bar{\theta}_i + k_d\dot{\bar{\theta}}_i + k_\beta(\omega_i^d - \dot{\theta}_i), \tag{4.37}$$

in which k_p, k_d and k_β are positive gains. We also have that

$$\omega_i^d = k_\omega / r_i, \tag{4.38}$$

is the desired angular speed, in which k_ω is a gain that regulates the fixed tangential speed and is the same for all robots of the system. Each robot will move locally to the mean angle in relation to its left and its right neighbors, as in Gonçalves et al. (2011):

$$\bar{\theta}_i = \frac{^{Left}\theta_i + ^{Right}\theta_i - 2\pi}{2} \tag{4.39}$$

in which $^{Left}\theta_i = \text{argmin}_{\tilde{\theta}_j \in \Omega'_j}\{\tilde{\theta}_j\}$ and $^{Right}\theta_i = \text{argmax}_{\tilde{\theta}_j \in \Omega'_j}\{\tilde{\theta}_j\}$. The set Ω is: $\Omega = \{\bigcup_{i=1}^{N}\theta_i\}$ and $\Omega'_i = \Omega \setminus \theta_i$ is the set containing the angular positions, of all robots of the same group of robot i, except the angle of robot i. We also have that $\tilde{\theta}_j$ is the measure of θ_j taken with respect to θ_i, i.e. $\tilde{\theta}_i = 0$.

Figure 4.8 shows an example of robots of the same group and the left and right neighbors of a robot.

The fixed angular speed (ω_i^d) is always dependent on the robot radius r_i. The angular velocity is responsible for making robots eventually meet other robots if (4.36) is respected.

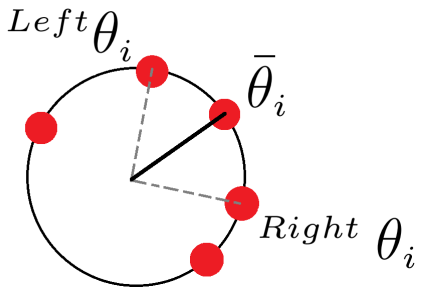


Figure 4.8: Example of right and left neighbors of a robot.

4.3.7 The Complete Control Law

We have proposed controllers for three different dynamics:

1. Consensus algorithm to control virtual point positions and velocities (Section 4.3.4);
2. Radius heuristics to dynamically set different radius for different groups (Section 4.3.5);
3. Angle controller to distribute robots of the same group (Section 4.3.6).

By combining those controllers, we can now define the complete control that will guide the movement of each robot. First we use the definition (4.37) and the heuristics described in Algorithm 4.2 to completely define (4.31). Then we use a consensus protocol ((4.33) or (4.34)) to control the dynamics (4.32). Finally, composing (4.32) and (4.31) we define (4.30) and we can move the robots given by the dynamics of (4.1). Thus, each robot will be guided by the control law

$$\boldsymbol{u_i} = \hat{\boldsymbol{u}}_{\boldsymbol{i}} - \begin{bmatrix} -r_i(\tilde{u}_i \sin\theta_i + \dot{\theta}_i^{\,2} \cos\theta_i) \\ r_i(\tilde{u}_i \cos\theta_i - \dot{\theta}_i^{\,2} \sin\theta_i) \end{bmatrix}, \tag{4.40}$$

in which $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ will be either given by (4.33) or (4.34) depending on the considered scenario.

4.3.8 Controlled System Analysis

Theorem 3. *Given the following assumptions:*

(i) Individual robots are governed by the dynamics in (4.1) with communication radius R_{com} and constant parameter d such that $d < 0.5c$;

(ii) Groups and a binary relation between groups are defined in such a way that a strictly totally ordered set of groups is induced;

(iii) Each robot i is able to compute if the order of its group is greater, equal, or less than the order of the group of any other robot j according to the predefined binary relation when the information about the group to which robot j belongs is made available.

Then, by applying the Algorithm 4.2 in the control of each individual robot, it is guaranteed that the multi-robot system will converge to a radial segregation state as defined in Section 4.3.1.

Proof. From facts (ii) and (iii) we can assume that Algorithm 4.2 can execute as all the comparisons can be properly computed.

In Algorithm 4.2 the motion of the robots is governed by (4.40). From (4.1) and (4.30) it is clear that the virtual points are driven by (4.32). In (4.32), $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ is determined according to the well-known consensus protocols in (4.33) or (4.34) which leads to the *rendezvous* of the virtual points. Given this fact, in order to show radial segregation we need to show that the radius r_i of each robot i converges to a given value which is the same value of the radius for robots of the same group and is a different value when compared to the radius of robots of other groups. We follow to show this with a similar procedure as in the proof of the controller of Section 4.2.

The rest of the proof employs induction, we first show that the first group will converge to the circle centered at the *rendezvous* point with radius given by d. Then, we proceed to show that the other groups will converge to circles also centered at the *rendezvous* point but with radius that increases accordingly to its group order in the group hierarchy.

According to Algorithm 4.2, all the robots start with $r_i = d$ and the only possible changes in the radius implies that $r_i = \lambda d$, where $\lambda \in \mathbb{N}^*$. The changes can only occur when robots meet within a radius R_{com}, which is the same for every robot. Moreover, a radius never decreases, it might only increase in case robot i receives the information about the existence of another robot j of a different group so that $h_i > h_j$ and $r_i \leq r_j$ or another robot j of the same group so that $r_j > r_i$. As the set of groups is a strictly totally ordered set, and the changes are given by $r_i = r_j + d$ for $h_i > h_j$ or $r_i = r_j$ for $h_i = h_j$ it is guaranteed that the radius of the robots of the group which is the least element of the set, i.e. $h_1 < h_j \ \forall j$, never changes, i.e., $r_i = d$ which implies in the convergence to the circle centered at the *rendezvous* point with radius given by d.

Now consider the hypothesis: all the robots of group $1, 2, ..., k$, where $h(N_1) < h(N_2) < ... < h(N_k)$, have converged to the corresponding radius $r_{N1} = d, r_{N2} = 2d, ..., r_{Nk} = kd$. According to Algorithm 4.2 and the strict total order it is impossible to have a change in the radius of a robot in group N_{k+1} when meeting robots in groups $N_{k+2}, N_{k+3}, ..., N_M$ as $h(N_{k+1}) < h(N_{k+2}) < h(N_{k+3}) < ... < h(N_M)$. From this and the initial conditions, $r_i = d \ \forall i$, we can conclude that the radius of group N_{k+1} must converge to $r_{k+1} = \lambda d$ where $\lambda \in \{1, 2, 3, ..., k+1\}$.

We have that $d < 0.5c$ and the desired polar angular velocity ω_i^d given by (4.38) is so that robots moving at circles with different radius will move with different angular velocities. Thus, it is guaranteed that robots at the circle with $r_i = \lambda d$ receive information from the other robots at consecutive circles, i.e. $r_j = (\lambda + 1)d$ as they meet and share

information periodically in finite time while they move in these circles. Moreover, in the first circle the robots have also access to the information from robots at the same circle (fact (i)). Thus, given the scheme of storing and broadcasting information of robots at consecutive circles in Algorithm 4.2 and given also the hypothesis of convergence of groups $N_1, ..., N_k$ we can guarantee that a robot from group N_{k+1} will always receive information about the existence of other robots at the same circle and the corresponding value $h(N_l)$ for comparison when $r_i = \lambda d$ with $\lambda \in \{1, 2, ..., k+1\}$. From this we can conclude that it is impossible for a robot of group N_{k+1} to converge to any circle of radius $r_i = \lambda d$ with $\lambda \in 1, ..., k$. According to Algorithm 4.2, being aware of the other robots already in their correct circle implies in the increment of the radius of the robots of group N_{k+1}. Therefore, the only possible circle for convergence is the one with $r_i = (k+1)d$.

By induction, we can conclude that each robot i of group N_l will converge to $r_i = ld$, $\forall i, \forall l$. Thus, segregation will always be achieved. □

We also have proposed an angle controller to distribute robots of the same group uniformly, although it is desirable to have this feature, in this section it is not a requirement of the segregation problem. As the angle controller only changes robots angles based on the proximity to other robots and virtual points remain unaltered, it does not interfere with the proof of *Theorem* 3.

5

Simulations and Experiments

In this chapter we show all the simulations and experiments related to the controllers presented in Chapters 3 and 4. This chapter is divided into two sections. In Section 5.1, simulations and experiments for the controllers based on abstractions (Chapter 3) are shown and discussed. In Section 5.2, simulations and experiments for the controllers based on consensus (Chapter 4) are shown and discussed.

5.1 Abstraction Based Segregation

In this section we present simulations and experiments for two different controllers that use similar approaches, based on the use of abstractions. In Section 5.1.1 we present simulation and experiments for segregating groups of robots (related to the methodology presented in Section 3.2). In Section 5.1.2 we present simulations for segregating groups of robots in a decentralized manner (related to the methodology presented in Section 3.3).

5.1.1 Segregation

5.1.1.1 Simulations

The proposed controller was first tested using MATLAB (The MathWorks Inc. (2014)) with our model of double integrator, holonomic robots as defined in Section 3.1. In this section we present three simulations in MATLAB in a two-dimensional space and two simulations in a three-dimensional space. Although our approach was presented considering a $2D$

setup, it is straight forward to derive the same equations in $3D$, by adding the third coordinate z. Furthermore, we also discuss in this section some advantages and limitations of our approach. A video of these simulations together with the experiments of section 5.1.1.2 can be found in https://youtu.be/ox2LFwau1-A.

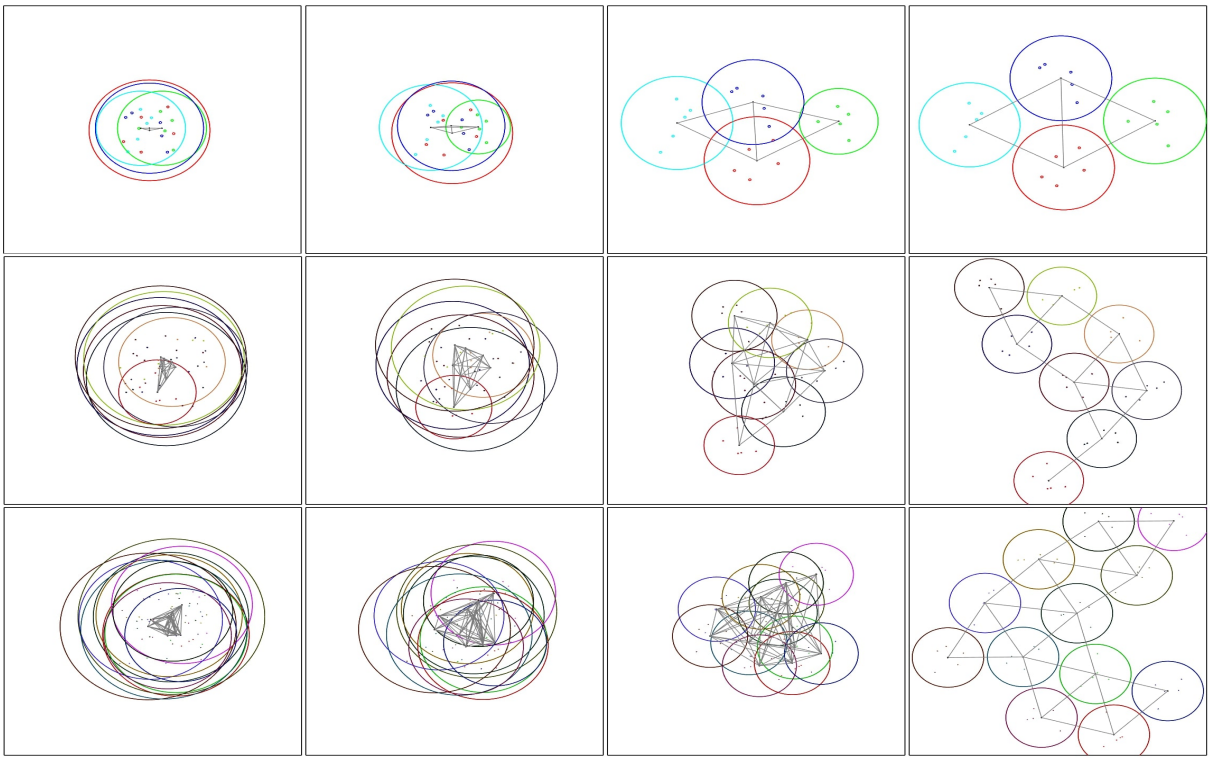


Figure 5.1: Simulations in MATLAB, each group has $N_j = 5$ robots. From top to bottom: (a) M = 4 groups. (b) M = 8 groups. (c) M = 12 groups. From left to right, 4 snapshots from initial to final iterations. The snapshots also highlight the abstraction size and the formation of the α-lattices.

In all simulations we assume that all robots start with zero velocity and the robots were positioned according to a random uniform distribution.

We used the following parameters to define the potential function: $r = 1.4d$, $h = 0.3$, $c = 10$. Gains k_1 and k_2 were set to 25 and 0.05, respectively. The desired abstraction size is used as 90% of the desired distance between groups, $\sigma_j^{des} = 0.9d^2/(4N_j)$. Other parameters, such as the desired distance between groups (d) and the normal distribution of robots were set in a way that we can better visually evaluate our approach and are dependent on the number of groups and robots.

The simulations were stopped after segregation, as defined in section 3.1.2, was visually reached and after the formation of the α-lattices was visually stable.

In the $2D$ simulations, we always used 5 robots per group, although this is not a requirement of our approach. We simulate 4, 8 and 12 groups of robots as shown in Figure 5.1. We also show two $3D$ simulations in Figure 5.2 to exemplify that the approach is scalable to higher dimensions.

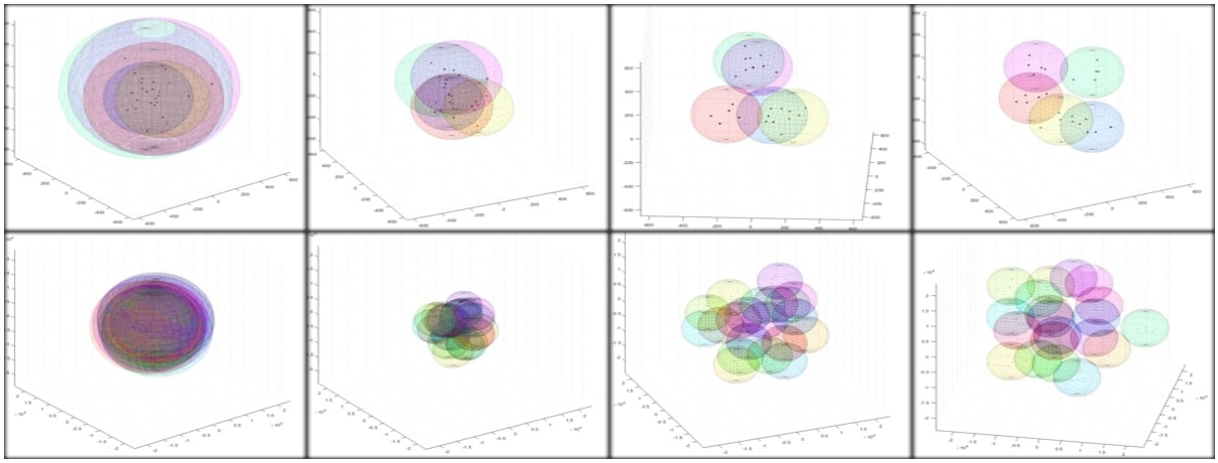


Figure 5.2: $3D$ Simulations in MATLAB. Top: each group has $N_j = 5$ robots and the system has 5 groups. Bottom: each group has $N_j = 7$ and the system has 20 groups. From left to right, 4 snapshots of initial to final iterations. The snapshots also highlight the abstraction size (bigger spheres). Snapshots are rotated to help visualization.

Each robot needs information about the position and velocity of all robots of its own group and of all the robots of neighboring abstractions.

In order to better depict the local property of our controller in comparison to the works in Santos et al. (2014) and Kumar et al. (2010), Figure 5.3 shows the average number of groups in neighborhood B_i versus the iterations, that is, the average amount of information needed for robots from initial time to the time segregation was reached.

No collision was verified in all the simulations, as desired.

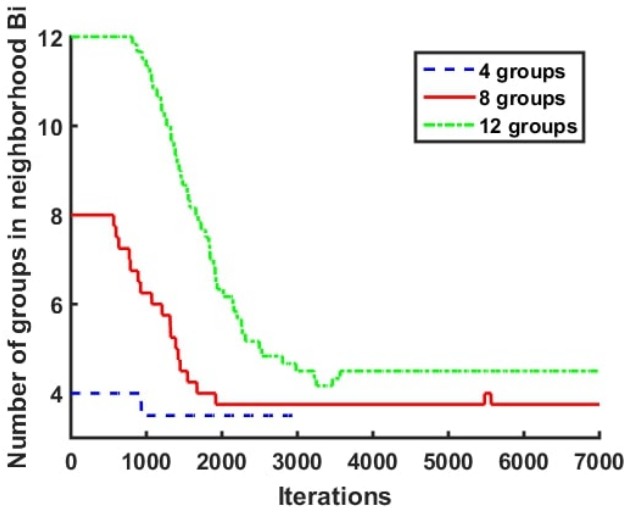


Figure 5.3: Information of how many groups each robot needs (average) versus iterations. e.g. With 8 and 12 groups, after the iteration 6000, each robot needs information of up to 5 neighbor abstractions.

5.1.1.1.1 Discussions

We can see from the simulations that our approach is different from Kumar's and Santos's works. From Figures 5.1 and 5.2 we see that groups have achieved segregation in various scenarios ranging from 20 to 140 robots. Note that our initial conditions make the segregation problem the hardest possible, because we purposely set robots to be very mixed. If we already have some kind of segregation, the problem becomes much easier to be solved.

In Figure 5.7 (a) it is possible to see that the intersection area between abstractions in the $2D$ and in the $3D$ simulations converges to zero as desired.

We assume that the initial distribution of robots are such that they are not in a local minimum. If we imagine a scenario of 6 robots divided in 2 groups of 3 robots each, arranged alternately at the corners of a regular hexagon, the system would not converge to segregation because this is a local minimum. This hexagon scenario is an unstable critical point and even the smallest disturbance in any position of any robot would make the system reach segregation as desired.

Another drawback of our approach is that, in order to maintain the state of the abstractions, we rely on a nonlinear programming solver to solve the collision avoidance minimization problem. It might happen that sometimes it does not find viable solutions. This would result in collisions between robots.

From Figure 5.3, we can see that as the abstractions begin to separate, the quantity of information needed for each robot decreases. This can be better seen with 8 and 12 groups, because with the chosen parameter and 4 groups, the groups remain mostly "connected".

It is important to note that the collision avoidance algorithm in practice does not increase the number of information each robot needs, because usually when robots are in imminent collision they belong to neighboring groups. In theory, the number of information needed could increase in scenarios with multiple groups involved in imminent collisions, which would make robots need information to compute the collision avoidance algorithm from robots that are outside the neighborhood B_i. In (3.46) it is necessary to have information from all the groups in the connected graph component Ω_p.

5.1.1.2 Experiments

The experiments with real robots were performed using the *Robotarium* testbed described in Pickem et al. (2017). The testbed uses *GRITSBot X* robots, which is an improved version of the *GRITSBot* presented in Pickem et al. (2017). These robots are differential drive robots, meaning that they only have two wheels (non-holonomic robots). We use a feedback linearization Desai et al. (1998) technique so that we can use our controller, designed for holonomic robots, in differential drive as show in Section A.1.2.

We assumed that every robot has access to the information of its own position and

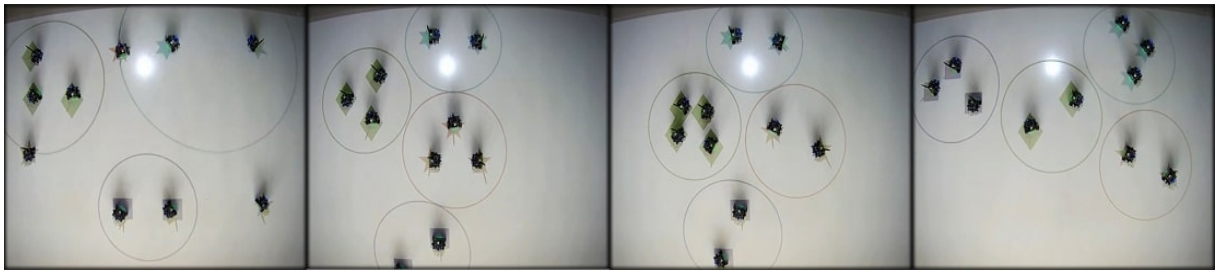


Figure 5.4: Sequence of snapshots of the unbalanced experiment with 10 real robots. In this experiment we purposefully change groups compositions two times after segregation is achieved. Leftmost snapshot: t=0, initial position. Middle left snapshot: t=40s, first segregation. Middle right snapshot: t=73s, second segregation. Rightmost snapshot: t=98s, third segregation. Snapshots also highlight the abstraction size and different markers for each type of robot.

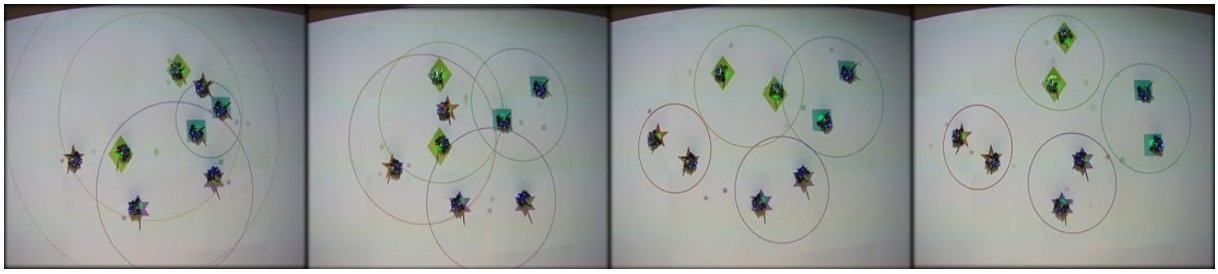


Figure 5.5: Sequence of snapshots of the experiment with 8 real robots with added intentional errors. Leftmost snapshot: t=0, initial position. Middle left snapshot: t=4s. Middle right snapshot: t=24s. Rightmost snapshot: t=64s, groups are segregated. Snapshots also highlight the abstraction size and different markers for each type of robot. Smaller markers show the position of the robots with added intentional errors.

velocity and the positions and velocities of every robot of its own group and every robot from neighboring groups.

In the testbed used to perform the experiments, robots do not have sensors. There is an overhead camera which is used to estimate the poses of all robots in the system. Therefore, we "emulate" a decentralized controller: each robot receives information (position, velocity, and group) of every other robot in the system in a centralized way but discards the information of robots that belong to non-neighbor groups according to the definition of neighborhood given by the parameter r_α (see Section 3.1.3). Since the testbed has a very reliable pose estimation system, we have also performed experiments in which we have added intentional errors in the measurement of the position of the robots to verify if the proposed approach is able to achieve segregation when the localization is not exact.

We performed three experiments: one without the addition of any intentional errors and two with the addition of intentional errors in the measurements of the position of the robots. All experiments were processed for 1800 iterations which was equivalent to 98, 64 and 72 seconds for the first, second and third experiments, respectively.

Robots are modeled as a square inscribed in a circle of radius equal to half of the

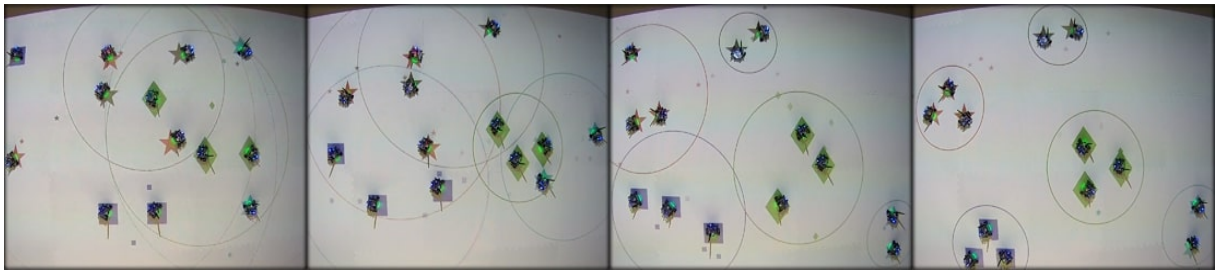


Figure 5.6: Sequence of snapshots of the experiment with 13 real robots with added intentional errors. Leftmost snapshot: t=0, initial position. Middle left snapshot: t=9s. Middle right snapshot: t=34s. Rightmost snapshot: t=72s, groups are segregated. Snapshots also highlight the abstraction size and different markers for each type of robot. Smaller markers show the position of the robots with added intentional errors.

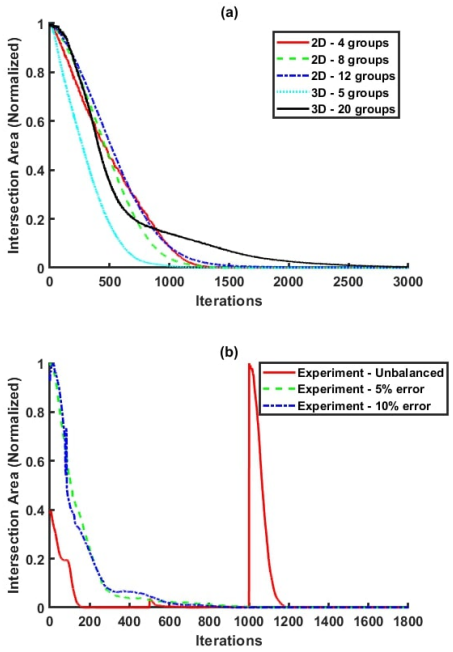


Figure 5.7: Evolution of the intersection area related to the simulations of Figures 5.1 and 5.2 and the experiments of Figures 5.4, 5.5 and 5.6. (a) Evolution of the intersection area of the $2D$ and $3D$ simulations. (b) Evolution of the intersection area of the three experiments with real robots.

diagonal of the square. For the purpose of the proposed methodology the size of the robots is considered to be the size of the circles.

The solution of the problems of minimization of equation (3.46) are obtained with the help of the *fmincon* function of MATLAB, with the *interior-point* optimization algorithm.

In the first experiment, with unbalanced groups, we used 10 robots divided into 4 groups that change composition during the experiment. With $t = 0s$, groups N_1, N_2, N_3 and N_4 have $3, 3, 2$ and 2 robots, respectively. After 700 iterations, groups compositions are changed to $N_1 = 2$, $N_2 = 4$, $N_3 = 2$ and $N_4 = 2$. Finally, after 1300 iterations, groups

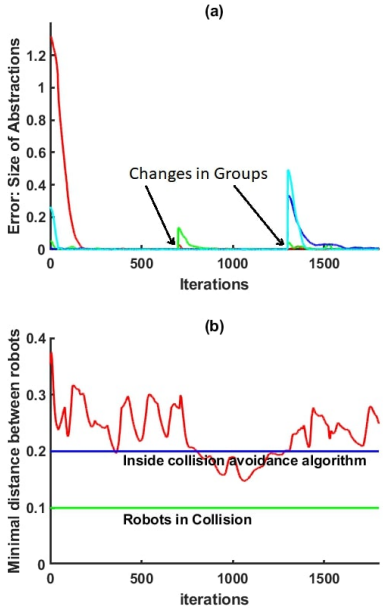


Figure 5.8: Data plot related to the experiment of Figure 5.4. (a) Evolution of abstractions size errors. (b) Evolution of the minimal distance between robots.

compositions are changed to $N_1 = 2$, $N_2 = 2$, $N_3 = 3$ and $N_4 = 3$.

In the second and third experiments we added intentional errors in the measurements acquired from the overhead camera from the *Robotarium* platform. Each robot computes its individual control action (3.34) adding intentional errors drawn from a normal distribution to its own position and to the position of each other neighbor robots at each time step. The errors have zero mean and standard deviation of 5% and 10% of the area of the environment for the second and third experiment, respectively. Note that neighbor robots of robot i are those in the neighborhood B_i, as explained in section 5.1.1.1.

In the second experiment, we used 8 robots divided equally into 4 groups, without changing groups compositions ($M = 4$, $N_j = 2, \forall j$).

In the third experiment, we used 13 robots divided into 5 unbalanced groups, without changing groups compositions ($M = 5$, $N_j = 3 \quad \forall_{j\in\{1,2,3\}}$ and $N_j = 2 \quad \forall_{j\in\{4,5\}}$) .

Frames of the first experiment are shown in Figure 5.4, for the second experiment in Figure 5.5 and for the third experiment in Figure 5.6.

In Figure 5.9 (a) we show the evolution of the abstraction sizes (σ) for the second experiment. In Figures 5.8 (a) and 5.10 (a) we show the evolution of the abstraction size errors for the first and third experiments, respectively, as the desired abstraction sizes (σ^{des}) changes over time or are different depending on the number of robots in the group. It is possible to see in Figures 5.8 (a), 5.9 (a) and 5.10 (a) that the abstractions converge to the desired sizes.

In Figures 5.8 (b), 5.9 (b) and 5.10 (b) we show the evolution of the minimal distance between robots for the first, second and third experiments, respectively.

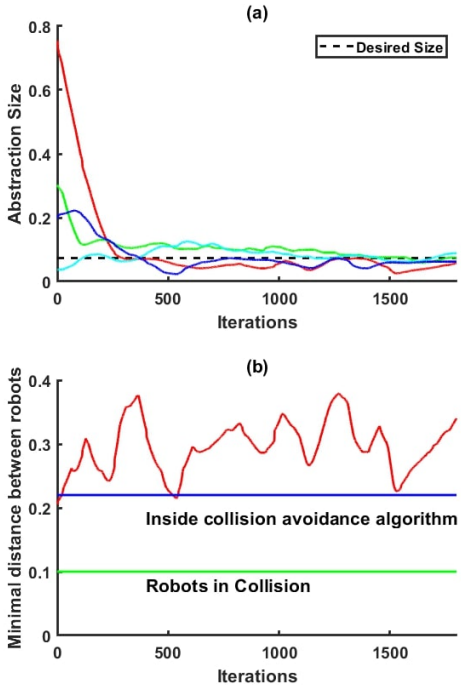


Figure 5.9: Data plot related to the experiment of Figure 5.5. (a) Evolution of abstractions sizes. (b) Evolution of the minimal distance between robots.

We can see in the second, third and fourth frames of Figure 5.4 and in the last frame of Figures 5.5 and 5.6 that segregation has occurred. We can also see that the system has reached segregation according to our definition analyzing the evolution of the intersection area between abstractions (Figure 5.7 (b)).

In Figures 5.8 (b), 5.9 (b) and 5.10 (b) it is possible to see that robots do not collide and when the collision avoidance algorithm was used, except for some brief moment in Figure 5.10. In this brief moment, there are some points that indicate erroneously the presence of collisions between robots. We have ignored those outliers that were in fact caused by a brief error in the localization of a robot. This brief error in the data can be confirmed watching the video of this experiment, where no real collisions have occurred (video can be found in `https://youtu.be/ox2LFwau1-A`). This example confirms the robustness of the method against usual errors in real scenarios.

5.1.1.2.1 Discussions

With real robots, the implementation issues of section 3.2.2 were not observed. As real robots have limited velocities, they do not reach velocities much higher than the mean velocity of its group. Thus, we chose not to use the *velocity dissipation* controller described in section 3.2.2. Therefore, we use the controller of equation (3.34) where the collision avoidance scheme is only turned on in imminent collision situations.

The experiments were important to validate our controller in real scenarios, where

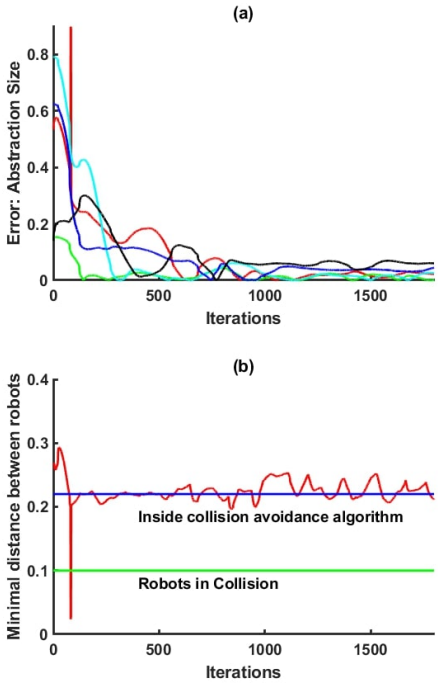


Figure 5.10: Data plot related to the experiment of Figure 5.6. (a) Evolution of abstractions size errors. (b) Evolution of the minimal distance between robots.

errors due to imperfections are commonly observed. Imperfections such as packet loss and time delays are usually observed in real world networks. Information from real sensors, such as the overhead camera, used for localization are subject to noise. These issues have direct impact in the information actually used by the robots to compute their control inputs. Furthermore, we can also have imperfections from the unmodeled dynamics of the differential drive robots and the discretization of the implementation, given that our controller was originally proposed for double integrator continuous dynamics. Actuator errors and input saturation may also be a source of degradation in the real system performance. Finally, differences in the robots such as small differences in motor power, in the battery charges, in the construction of the robots, etc. may introduce unexpected behavior. All these imperfections are usually disregarded in computer simulations which justifies the verification of the method in real robots. In the real experiments performed in this work we could indeed verify some robustness of the proposed method to errors caused by real world issues.

In the *Robotarium* testbed, although errors due to imperfections exist, we can usually assume that the overhead camera based localization system is quite precise. Therefore, to show that our controller can be robust even in the presence of more severe errors we have also performed experiments adding intentional errors to the position of the robots after receiving those information from the testbed (experiments shown in Figures 5.5 and 5.6). Those experiments can provide indication that the segregation controller will perform well

even in other real world environments where errors and imperfections are usually more expressive.

It is also important to mention that the experiment with unbalanced groups could attest that the method is robust to the dynamic addition and withdraw of robots from groups as long as $\sigma_j \neq 0$. In practice, this means that groups should have at least two robots.

5.1.2 Decentralized Segregation

5.1.2.1 Simulations

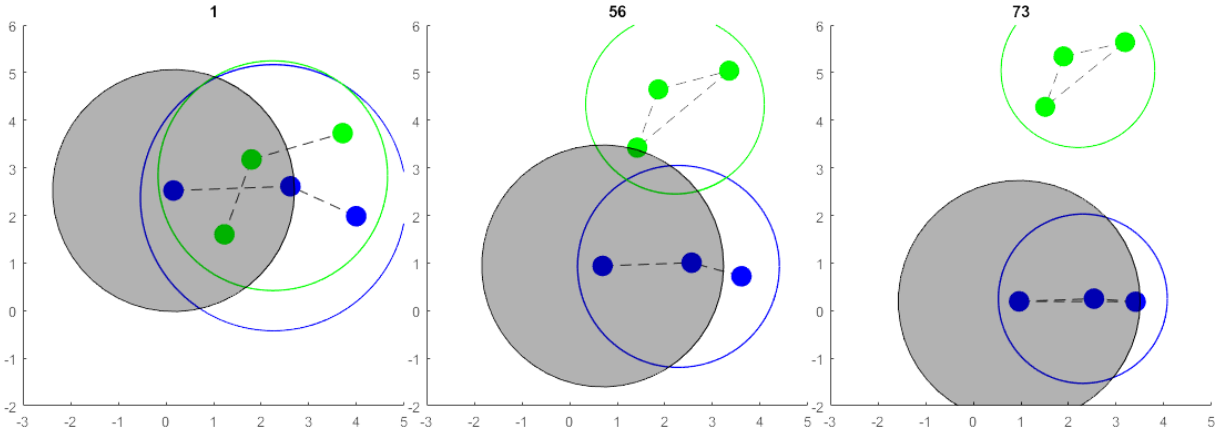


Figure 5.11: Sequence of frames from a simulation with 6 robots divided into two groups. Leftmost frame: Starting positions. Rightmost frame: Groups are segregated.

In all simulations in this section, the parameters of the potential function were defined: $r = 1.4d$, $h = 0.3$ and $c = 10$. The gains of the individual controller were defined: $k_1 = 3$ and $k_2 = 5$. The integration step used is 0.01 and the numerical integration is solved with the function *ode45*, which uses the explicit *Runge-Kutta* formula of orders 4 and 5. The size of abstractions in this section is always defined respecting equation (3.29) and the desired distance between abstraction centers (parameter d) is such that

$$d = 0.8r, \tag{5.1}$$

in which r is the radius of communication between robots and also the parameter that dictates the *finite cut-off* of the potential function. In all simulations, in this section, collisions were not considered.

Figure 5.11 shows a simulation with 6 robots divided equally into 2 groups in order to qualitatively visualize the proposal. This figure shows the communication radius of a robot (blue robot) and also the graph of communication between robots in the same group (dotted lines). In this simulation, the radius of communication between robots is

$r = 2.5$m, the radius of robots is $R_b = 0.1$m and the desired distance between the centers of abstractions is $d = 2$m.

Other simulations were made in order to analyze the controller with more groups and robots, Figures 5.12 and 5.13 are related to these simulations. In Figure 5.12, 3 simulations are shown, with 50, 100 and 150 robots. In all of these simulations, the stopping criteria was the absence of intersections between abstractions, as defined in Section 3.1.2. The radii of the robots is $R_b = 0.5$m and the radius of communication between robots is given by $r = 35$m, $r = 70$m and $r = 105$m for simulations with 50, 100 and 150, respectively. Note that we consider the environment to be unlimited.

In the simulations of this section, it was considered that the consensus protocols would be processed in zero time, as assumed in Section 3.3. Thus, robots estimate the values needed to define their individual controller, calculate its control action and move accordingly. In these simulations, the zero-order sampler was as defined in the section 3.3.1 was not used. The use of the zero-order sampler in simulations and with real robots and the consequences of this use are a suggestion for future work.

Figure 5.13 shows the amount of information required on average for robots from the initial instant until the instant that segregation was obtained. This figure is related to the 3 experiments in figure 5.12. The reference used in this figure was obtained through the use of the controller in Section 3.2 with the same initial conditions as the controller in this section.

All the simulations in this section can be found in a video at https://youtu.be/Nz4qJlKAgMA. In this video, some other information from each simulation is also shown, such as the evolution of the average error of the estimators used (see Section 3.3) and the evolution of the error of the size of the abstractions. The additional information is shown only in the video and not in the text because it is understood that they are more relevant accompanied by the movements of the robots and abstractions.

5.1.2.2 Discussions

It can be seen from the simulations found in Figures 5.11 and 5.12 that segregation was achieved in systems ranging from 6 to 150 robots. From Figure 5.13 and the data about the required information seen in the video (https://youtu.be/Nz4qJlKAgMA) it is possible to see that the number of information needed for each robot is very low. This number reduces on average throughout the entire simulation in relation to the controller described in the Section 3.2: 29.32% for the simulation with 50 robots, 34.58% for the simulation with 100 robots and 33.95% for the simulation with 150 robots.

The use of local information is the greatest advantage of this decentralized controller in relation to all other segregation controllers found in the literature and the controller of in Section 3.2. Assuming that robots of the same group have connected communication graphs, the only information required for each robot outside its communication range is a

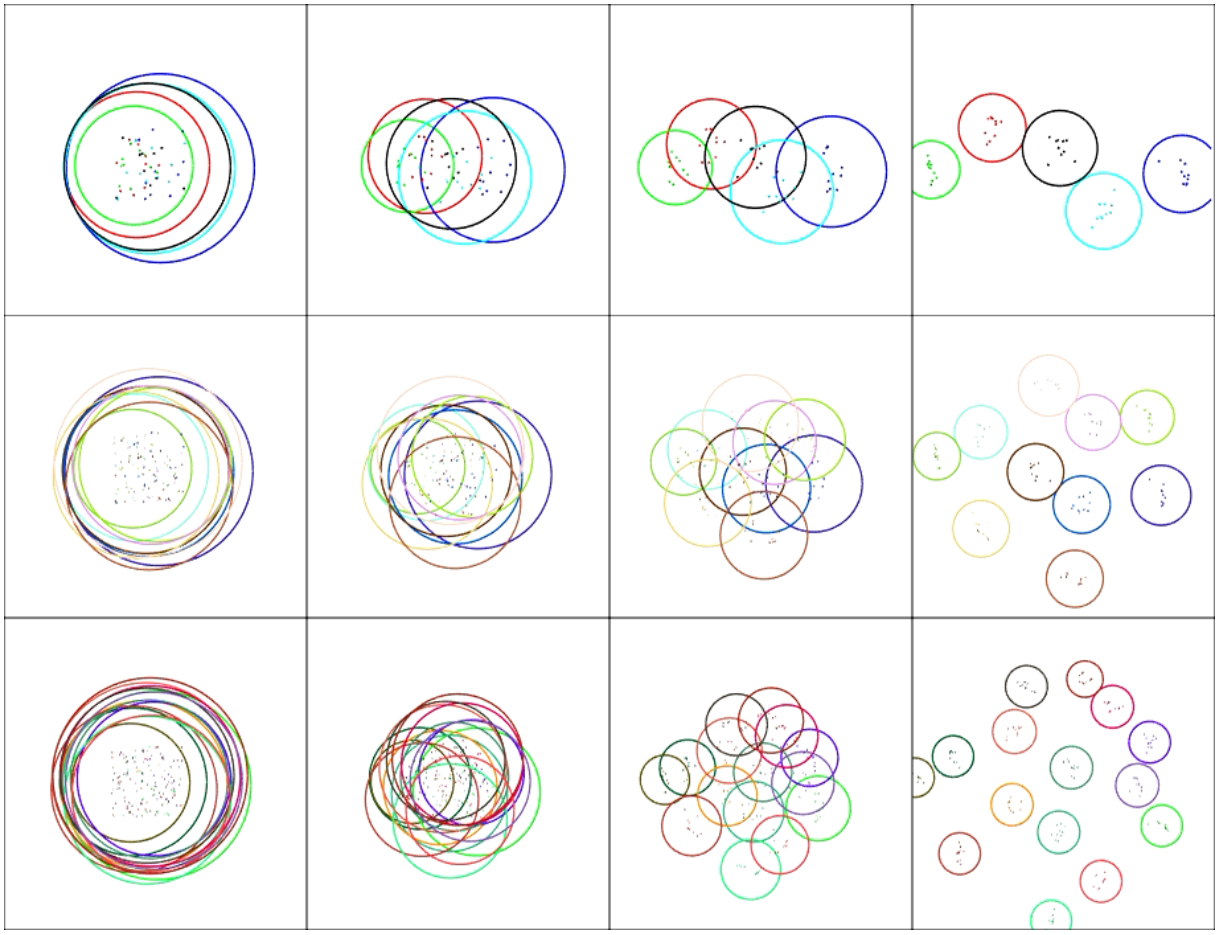


Figure 5.12: MATLAB simulations, each robot group has $N_j = 10$ robots. From top to bottom: (a) M = 5 groups. (b) M = 10 groups. (c) M = 15 groups. From left to right, 4 frames from the initial iteration to the final iteration.

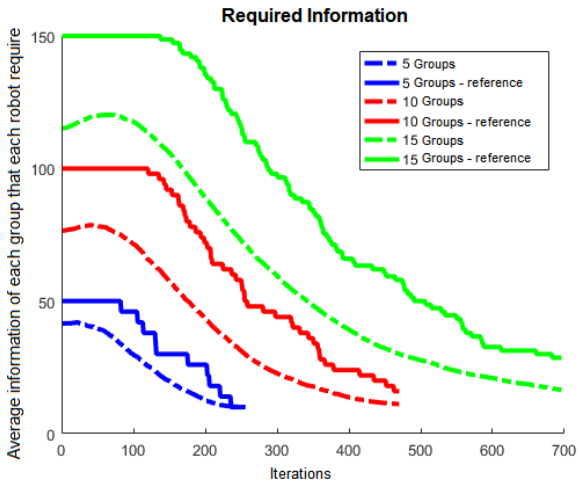


Figure 5.13: Average information of each group that each robot require per iteration.

limit on the number of robots in its group. In the simulations of this work, we assume that all robots can estimate this limit. This could be done using another distributed estimator but it remains as a suggestion for future work. As in the simulations we are assuming that

the consensus is processed in zero time, the accuracy of this estimate would not make a difference in the simulations because the robots wait for the consensus to be processed and their controller will be fully defined and then move.

In this work it was assumed that the communication graph of each group can vary, but it must always remain connected. This is a strong assumption, if the graph is disconnected at any time, the entire proposal is invalid. In the simulations, the parameters were regulated in a way that it was unlikely that the communication graphs would disconnect, for example, choosing the desired size of the group σ_{des} to be small in relation to the initial distribution of the robots. This caused the robots to move in the direction in which the size of their initial abstractions is reduced, thus maintaining the connectivity of the graph.

Note in Figure 5.13 that robots do not need information from the entire system anytime, unlike the reference (controller in Section 3.2), where in the initial moments all robots need information from all other robots in the system.

The greatest disadvantage of this controller is that it does not have a convergence proof. As a result, there may be situations in which segregation is not achieved or the system is unstable. In all tests there were no situations of this type, as long as group topology remained connected. In all situations where the system did not converge to segregation, with disconnected groups, it was enough to increase the radius of communication of robots for the system to converge to segregation with the same initial conditions.

Note that each robot calculates the artificial forces individually, this means that sometimes some robots are attracted or repelled by different groups and do not move according to the average of their group as in the controller in Section 3.2. Consequently, robots move with less cohesion, i.e. robots sometimes "orbit" the centers of their abstractions, but always maintaining their states.

5.2 Consensus Based Segregation

In this section we present simulations and experiments for controllers that use similar consensus based approaches to deal with different problems. In Section 5.2.1 we present simulation and experiments for segregating robots in curves (related to the methodology presented in Section 4.2). In Section 5.2.2 we present simulations and experiments for segregating robots radially (related to the methodology presented in Section 4.3).

5.2.1 Segregation in Curves

In this section we present our simulations and experiments with real robots. A video containing all the simulations and experiments shown in this paper, and some more examples, can be found in `https://youtu.be/JuUn4DIa0-w`. The segregation error is defined as in Santos et al. (2014). First, we compute the convex hull of each group of

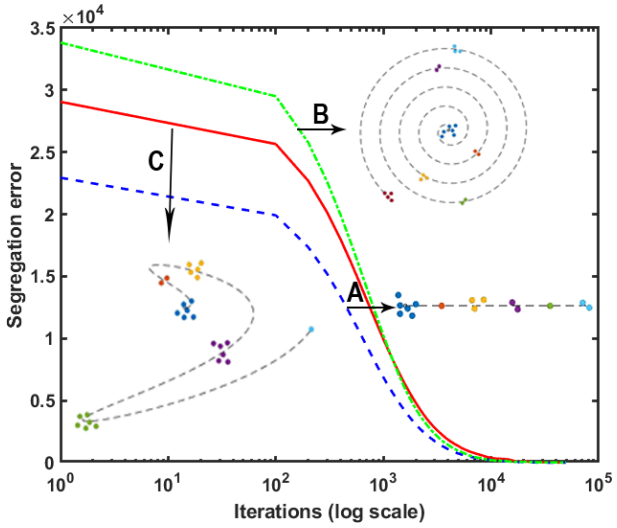


Figure 5.14: Mean segregation error for 150 simulations x Iterations (Log scale).

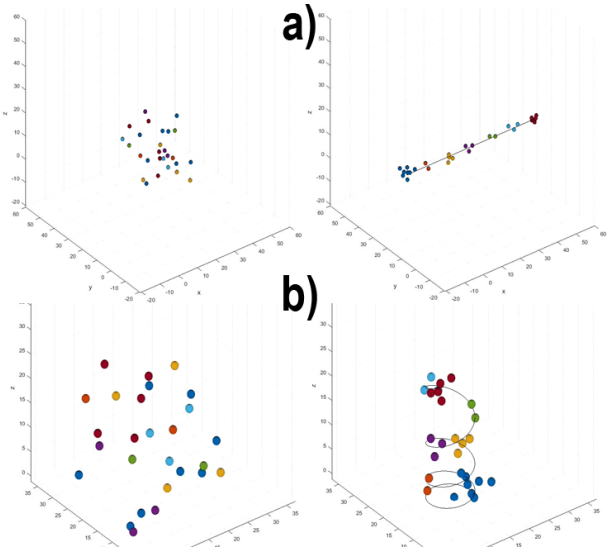


Figure 5.15: Initial and final snapshot of two $3D$ simulations with robots divided unevenly into 7 groups. a) Segregation in a line. b) Segregation in a helicoid.

robots using the position of all the robots of each group. The segregation error is then defined computing the intersection area or volume of all the convex hulls.

5.2.1.1 Simulations

We have executed an extensive series of $2D$ simulations with three different curves to analyze our approach quantitatively. We have also performed two $3D$ simulations to analyze our approach qualitatively.

In Figure 5.14 we show the simulation results. We have performed 50 simulations for each curve with a varying number of robots and groups. We randomly picked the number of robots for each group from the set $[1, 2, ..., 10]$ and we also picked the number of groups in the system from the same set. We used the curves: A: $s_i = [d_i \quad 0]^T$, B:

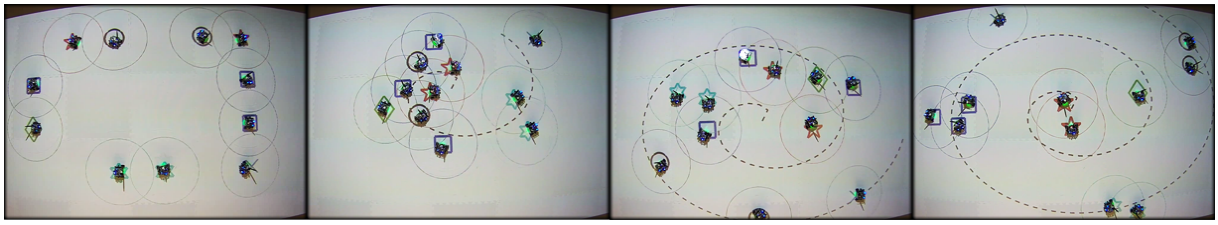


Figure 5.16: Experiment with 11 robots divided in 6 groups, $N_k = \{2, 1, 3, 2, 2, 1\}, k \in \{1, ..., 6\}$. Circles are the communication radius c.

$s_i = [\sqrt{2d_i}\ \cos(\sqrt{(2d_i)})\quad \sqrt{2d_i}\ \sin(\sqrt{(2d_i)})]^T$ and C: $s_i = [5d_i\ \cos(2d_i)\quad 5d_i\ \sin(d_i)]^T$. In Figure 5.14 we also show an example of the final step of a simulation run in which the curves A, B and C can be seen. We also use those examples as the legend for Figure 5.14.

In Figure 5.15 we show two examples of $3D$ simulations. We used the curves $s_i = [d_i \quad 0 \quad 0]^T$ in 5.15(a) and $s_i = [5\cos(\sqrt{2d_i}) \quad 5\sin(\sqrt{2d_i}) \quad 0.1d_i]^T$ in 5.15(b).

5.2.1.2 Experiments

In Figure 5.16 we show an experiment with *11 GRITSBot X* (Pickem et al., 2017) using the *Robotarium* testbed (Pickem et al., 2017). The experiment was conducted for 78s with parameters: $\delta_b = 0.0003$, $\delta_W = 0.001$, $\gamma = 3$, $c = 0.30$m, $c_{out} = 0.15$m, $c_{in} = 0.11$m and robots radii $R_b = 0.055\text{m} = 0.5c_{in}$. The curve used is the spiral $\boldsymbol{s} = [1.4\ \sqrt{d_i}\ \cos(\sqrt{140d_i}) \qquad \sqrt{d_i}\ \sin(\sqrt{140d_i})]^T$.

5.2.1.3 Discussions

From Figures 5.16, 5.14 and 5.15 it can be seen that segregation has occurred in several scenarios, in $2D$, $3D$ and with real robots. In all simulations and experiments there were no collisions between robots and a fixed connected underlying topology was used.

Our approach is an improvement over the work of Kumar et al. (2010) and Santos et al. (2014) in the sense that a convergence proof for multiple groups is provided and segregation for any number of robots and groups is guaranteed. Also, in our approach robots do not use information of all other robots in the system all the time and do not have problems with unbalanced groups as long as the considerations about the relation between c and c_{out} is adequate, as explained in Section 4.2.7.3.

Note that our approach is robust to adding and subtracting robots to the system, as the segregation distance b_i will potentially increase to guarantee that groups are segregated.

5.2.2 Radial Segregation

We have tested the proposed controller with extensive simulations using the double integrator dynamics (4.1) and experiments with real robots using the *Robotarium* platform (Pickem et al., 2017). We have tested the controller for both *Scenarios 1* and *2*. In this

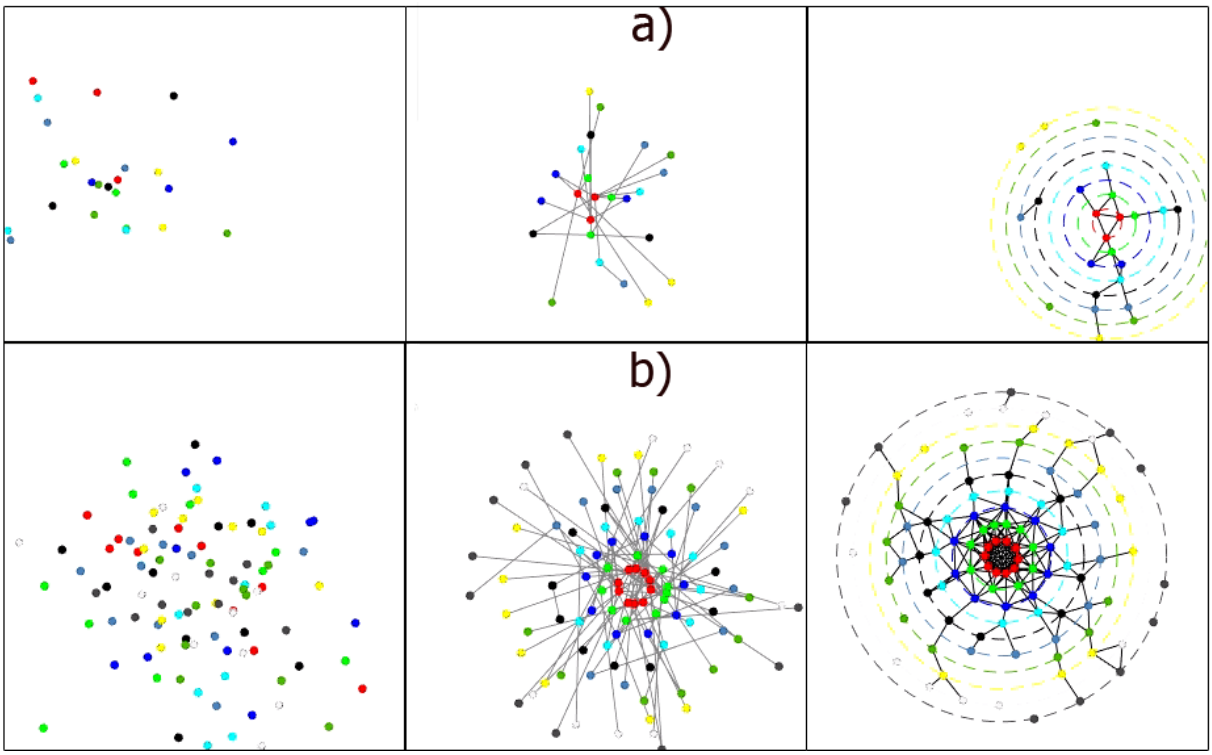


Figure 5.17: Simulations in MATLAB. From top to bottom: (a) N = 24 robots divided equally in M = 8 groups. (b) N = 100 robots divided equally in M = 10 groups. From left to right, 3 snapshots of initial to final iterations. In the middle snapshot of each simulation we highlight the fixed underlying topology. In the last snapshot of each simulation we highlight in dashed lines the circles of the groups after segregation is reached, and we also "connect" with black lines every robot that are within the communication radius in that instant.

section we present two simulation for *Scenario 1* to analyze the proposal qualitatively. For *Scenario 2* we present one of our trials of an experiment with real robots, and we also present data of 90 simulations to evaluate the proposal quantitatively. In addition, we briefly discuss the results.

A video with the simulations and the experiment can be found at: https://youtu.be/yLZyN9MpC18. In this video we show the evolution of the mean of information needed for each robot and the evolution of the segregation error (according to Groß et al. (2009)) and the uniformity error (according to Wilson et al. (2004)) related to the simulations of Figure 5.17. In the video we also show the experiment with real robots along with the evolution of the segregation error of the experiment.

The segregation error is defined as in Groß et al. (2009): the segregation error for the entire swarm is obtained by summing up the segregation errors for all possible pairs of robots, scaled from 0 (all robots in segregated order) to 1 (zero robots in the segregated order).

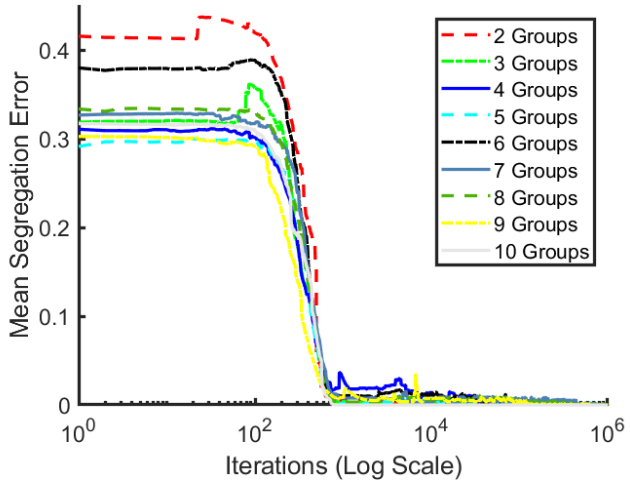


Figure 5.18: Mean segregation error of 90 simulations with a varying number of robots and groups for *Scenario 2*.

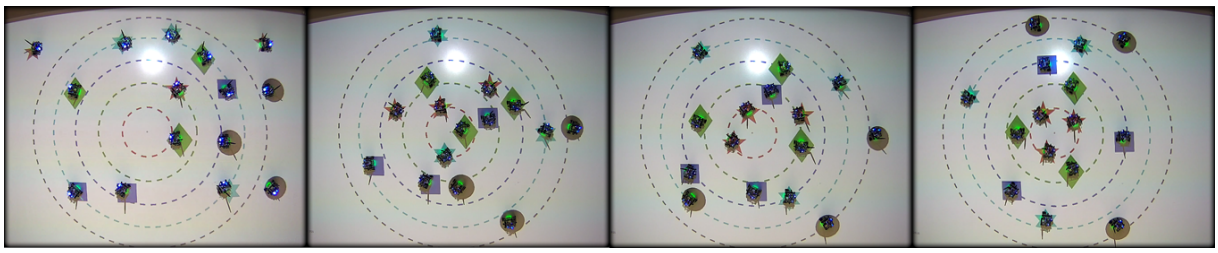


Figure 5.19: Experiment in *Robotarium* (Pickem et al., 2017). Robots of the same group have the same color and form markers. Dashed circles represent the virtual circles for each group. Leftmost snapshot: initial arbitrary positioned robots. Rightmost snapshot: robots are radially segregated.

5.2.2.1 Simulations

In all simulations we assume that all robots start with zero velocity and the robots were positioned according to a normal distribution.

In both *Scenarios* the parameters were: $c = 5\text{m}$, $d = 0.5c - \epsilon$, $\epsilon = 0.1$, $\gamma = 5$, $k_p = 0.01$, $k_d = 0.01$, $k_\omega = 0.1$, $k_\beta = 10$ in which ϵ is set to guarantee (4.36). Each simulation was performed for 50000 iterations with integration step equal to 0.02.

In Figure 5.17 we show two simulations for *Scenario 1* that can be visually analyzed, one simulation with 24 robots and the other one with 100 robots. The underlying topology was arbitrarily set and is fixed and connected in each simulation.

In *Scenario 2* we performed 90 simulations varying the number of robots and groups ranging from two robots to 100 robots divided in two to 10 groups. Starting with two groups and one robot per group up to 10 robots and then increasing the number of groups: $3, 4, ..., 10$ groups, therefore, the last simulation has 10 groups and 10 robots per group. In Figure 5.18 each line shows the mean segregation error for 10 simulations, from one to 10 robots per group and the number of groups is depicted in the legend.

5.2.2.2 Experiment

In Figure 5.19 we show an experiment with *15 GRITSBot X* (Pickem et al., 2017) divided equally into 5 groups. The experiment was conducted for *236 seconds* and segregation state was first reached around *100 seconds* after the start. Parameters of the experiment: $c = 0.48$, $d = 0.5c - \epsilon$, $\epsilon = 0.05$, $\gamma = 8$, $k_p = 0.001$, $k_d = 0.005$, $k_\omega = 0.03$, $k_\beta = 0.1$. A local collision avoidance algorithm already implemented in the *Robotarium* platform (Pickem et al., 2017) was used, the algorithm is based on barriers certificates (Ames et al., 2014). Robots have diameter of $R_d = 0.11$m and the collision avoidance algorithm was used whenever two robots were less than 0.33m from each other.

5.2.2.3 Discussions

After analyzing Figures 5.17, 5.18 and 5.19 we can see that in all cases robots have reached radial segregation. It can also be seen in Figure 5.17 that robots do not need information from all robots of the system to reach segregation. In average robots needed information of 10.43% and 24.20% of robots in the system, for the simulation in Figures 5.17 (a) and 5.17 (b), respectively.

Note in Figure 5.17 that robots can sometimes be unevenly distributed on the virtual circle of its group due to the local nature of the approach. Depending on the system's initial conditions some robots may never meet robots from the same group and consequently never reach perfect distribution. Nonetheless, they always reach the desired radius of their group. This would still be a radially segregated system according to our definition of the problem. As the size of groups increase they are more likely to reach uniform distribution.

6

Conclusions

In this work we have proposed several controllers for segregating groups of robots. All the controllers were developed for robots modeled as double integrators and in all controllers robots do not need information about all the system all the time. This fact alone shows that all the controllers presented in this document are improvements over the segregation controllers found in literature.

We have presented controllers to deal with the problem of radially segregating groups of robots and to deal with the problem of segregating groups of robots into clusters. All in all, five different controllers to segregate groups of robots are shown. Table 6.1 presents a summary of those controllers.

In Section 3.2 it was presented a controller that uses abstractions to represent each group of robots and an artificial potential function was used to separate groups. This controller requires that robots have the knowledge of the states of the centers of abstractions of neighboring groups (given by the parameter r). Also, robots need the knowledge of the states of its own abstraction, one way to acquire this is to have the knowledge of the states of all other robots on its own group. Moreover, an integrated collision avoidance algorithm was proposed, which allows robots to segregate maintaining the abstractions formation. This approach is an improvement over the works of Kumar et al. (2010) and Santos et al. (2014) because it does not require information about all the system all the time. The approach is also an improvement over the works of Edwards et al. (2016) and Inácio et al. (2019) because we present a convergence proof with an integrated collision avoidance while Edwards et al. (2016) present a convergence proof for robots modeled as

single integrators without a collision avoidance and Inácio et al. (2019) does not present a convergence proof.

Table 6.1: Summary of the controllers shown in this document.

In this document	Published	Main Ideas	Information Required	Collision Avoidance	Scenario	Dimension	Proof
Section 4.2	RAS 2019 (Ferreira Filho & Pimenta, 2019a)	Abstraction Based; Without Estimators; Potential Function;	The center of all the abstractions inside r; All robots of its own group;	Integrated	-	2D and 3D	Convergence
Section 4.2	-	Abstraction Based; With Estimators; Potential Function;	Nearby Robots; All robots of its own group;	No	-	2D and 3D	No
Section 5.2	ICRA 2020 (Ferreira Filho & Pimenta, 2020)	Consensus Based; Segregation in Curves; Formation Control; Heuristics;	The same open curve; Robots inside c; Fixed connected Topology	Integrated	-	2D and 3D	Convergence
Section 5.3	CDC 2019 (Ferreira Filho & Pimenta, 2019b)	Consensus Based; Radial Segregation; Virtual Points; Rendezvous; Heuristics	Robots inside c; Fixed connected Topology	No	1	2D only	Convergence
Section 5.3	CDC 2019 (Ferreira Filho & Pimenta, 2019b)	Consensus Based; Radial Segregation; Virtual Points; Rendezvous; Heuristics	Robots inside c; A fixed reference point	No	2	2D only	Convergence

In Section 3.3 another controller based on the use of abstractions and potential artificial functions was proposed. With this controller, robots use estimators to acquire the states of its own abstraction. Furthermore, each robot relies on local information to attract or repel robots from other abstractions, when they are close enough. This approach has improvements over the approach of Section 3.2 in the sense that robots require less information to reach a segregated state, although this was the only approach of this work that does not have a convergence proof and may not reach segregation in real scenarios where robots have low computation power and communications have delays.

Section 4.2 presents a controller to segregate groups of robots in which the main idea is the use of a consensus algorithm to guide the movements of the robots. In this controller, all the robots must have the knowledge of the same open curve. We consider that robots can communicate through a fixed underlying topology and also when they are within a certain distance. We propose a heuristic to compute the distance between the groups, i.e. the distance from the beginning of the curve, which makes groups segregate. The approach is local in the sense that each robot only needs communication with part of the system to converge to a segregated state. This approach has improvements over the works on segregation found in literature, as it combines a convergence proof with a reduction on the amount of required information required per robot.

Section 4.3 presents two controllers: one in which robots require a fixed underlying communication topology (*Scenario 1*) and another one in which robots only require the knowledge of the same fixed point (*Scenario 2*). This approach deals with a slightly different segregation problem: the radial segregation. This approach, considering any of the scenarios, is an improvement over all the works on radial segregation found in literature in the sense that none of them present a convergence proof to segregation.

For all the controllers presented in this document simulations and experiments validated

the results, except the controller of section 3.3, in which only simulations were shown.

6.1 Future Work

In this section, proposals for continuity of work for the controllers shown in this document will be suggested.

6.1.1 Abstraction Based Segregation

In Chapter 3, two controllers were shown to solve the problem of segregation in swarms of robots. The first controller (Section 3.2) might have some decentralization features at times, but this can not be guaranteed in every scenario. The second controller (Section 3.3) is really decentralized, i.e. robots do not use global information. The main proposal for continuing the work Chapter 3 is to obtain a formal proof of convergence for the decentralized controller, since without a formal proof there is no guarantee that the system will converge to segregation as desired.

As estimators have been added, it is intended to use the *Lyapunov* stability theory to extend the convergence proof of the first controller (see Section 3.2.3). It is intended to have the formation of different α-*lattices*, in which each node can be in a limited region instead of just one point. It is also intended to modify the desired size of the abstractions (σ_{des}), so that it is possible to define this size in a limited region, instead of being a fixed size. In this way it is intended to study techniques that show the convergence of the system for segregation even with these different formations. It is intended to use the parameter d, that separates the center of groups, to regulate the size of these regions and thus try to prove the convergence for segregation of the second controller.

Another idea that might be explored is the use of the discrete time framework with our consensus estimators in order to try to mathematically prove the convergence of our controller to segregation.

We also want to design a new controller to avoid collisions, that uses only local information, based on the controller shown in Section 3.2.1. The initial idea is to create a null space only with the neighboring robots and that these neighbors maintain the state of the abstraction, while the collision is avoided by a robot.

It is also intended to explore another idea in the sense that the states of abstractions do not necessarily need to be maintained while avoiding a collision. They can be changed, as long as it is in the "right" direction. That is, it is always desired that the abstractions keep approaching the desired size (σ_{des}) and moving away from the centers of other groups. This would mean that if a robot changes its trajectory to avoid a collision, and that this trajectory makes the abstraction closer to its "desired state", then the other robots in this group would not need to move to maintain the state of the abstraction.

It is also a proposal of future work, to improve the local properties of this controller. The controller shown in Section 3.3 requires the robots to have some information *a priori*: the number of robots in its group and the adjacency matrix of this group (which robot communicates with which other). We want to further reduce what information robots need to achieve segregation. One possible idea to be followed is to add new estimators, so that robots can acquire these information individually.

6.1.1.1 Controller Robustness

Another proposal for future work consists of giving robustness to the decentralized controller. In this controller, the states of abstractions are estimated in a decentralized manner, and these estimated values may contain uncertainties. Uncertainties can be due to practical aspects such as failures/delays in communication between robots or even theoretical aspects such as the delay in the convergence of the estimate when using the consensus protocol.

A possible idea is to model all uncertainties so that it is possible to obtain a model of robots of the type

$$\ddot{q} = u + \Delta u, \tag{6.1}$$

in which Δu is a sum of all these uncertainties.

Thus, we want to investigate robust control techniques and propose a u controller that is robust to these uncertainties without increasing the amount of information needed by the robots and that still takes the system to a so-called segregated state.

6.1.2 Consensus Based Segregation

In Chapter 4 we have presented two novel decentralized approaches to segregate swarms of heterogeneous robots. We have shown proof of convergence for both approaches, however, we only considered collision avoidance in the first approach (Section 4.2). Furthermore, only the first approach is scalable to a higher dimension. The radial segregation controller may not be directly used in a $3D$ Euclidean space because the virtual points are modeled according to polar coordinates. To extend to $3D$, one should formulate virtual points with spherical coordinates. Formulating as spherical coordinates, a problem arises: robots rotating in a $3D$ sphere would not have so many meetings as they have in $2D$. Therefore, one must formulate a controller in which robots of a group rotate in a way that they can eventually "see" robots in the immediately more intern sphere. As future work, we intend to extend the controllers for radial segregation in order to have an integrated collision avoidance and the extension to the $3D$ case.

Another idea to future work is to use abstractions similar to the ones proposed in Chapter 3 to represent the groups and then segregate the abstractions radially.

Moreover, we intend to explore strategies to segregate groups of robots in which even less information is used. In the curve segregation controller, one might explore the idea of

acquiring the knowledge of a curve in a decentralized way. In this case, robots would have to reach a consensus in the parameters of the curve they should converge to.

Other ideas that might be interesting exploring is to use auction or gossip algorithms so that robots are able to decide the group hierarchy in a decentralized manner. In this case one would eliminate the requirement of the existence of a global group hierarchy.

Finally, in the controller to segregate robots in curves, we have a constraint regarding the number of robots per group. Robots of a group must fit in a region so that we can guarantee that the communication between robots will be such that eventually all robots will have enough meetings to gather the information about where is the true position of its group in the hierarchy. This is a restrictive constraint that may limit how many robots each group can have. As future work, we intend to explore new ways to prove the convergence of the controller in which this constraint is not required.

Bibliography

Ames, A. D., Grizzle, J. W., & Tabuada, P. (2014). Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control* (pp. 6271–6278).: IEEE.

Antonelli, G., Arrichiello, F., Caccavale, F., & Marino, A. (2013). A decentralized controller-observer scheme for multi-agent weighted centroid tracking. *IEEE Transactions on Automatic Control*, 58(5), 1310–1316.

Balch, T. & Arkin, R. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6), 926–939.

Belta, C. & Kumar, V. (2003). Towards abstraction and control for large groups of robots. *Control Problems in Robotics*, STAR 4, 169–182.

Belta, C. & Kumar, V. (2004). Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5), 865–875.

Bezzo, N., Griffin, B., Cruz, P., Donahue, J., Fierro, R., & Wood, J. (2014). A cooperative heterogeneous mobile wireless mechatronic system. *IEEE/ASME Transactions on Mechatronics*, 19(1), 20–31.

Bollobás, B. (1998). Modern graph theory. *Springer Science & Business Media, vol. 184*.

Bredeche, N., Haasdijk, E., & Prieto, A. (2018). Embodied evolution in collective robotics: a review. *Frontiers in Robotics and AI*, 5, 12.

Burns, M. (2017). Intel powered the drones during super bowl halftime show [online]. https://techcrunch.com/2017/02/05/intel-powered-the-drones-during-lady-gagas-super-bowl-halftime-show/.

Cao, Y., Yu, W., Ren, W., & Chen, G. (2013). An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1), 427–438.

Chaimowicz, L. & Kumar, V. (2004). Aerial shepherds: Coordination among uavs and swarms of robots. *Distributed Autonomous Robotic Systems 6, pp. 242-252.*

Chaimowicz, L., Michael, N., & Kumar, V. (2005). Controlling swarms of robots using interpolated implicit functions. In *IEEE International Conference on Robotics and Automation* (pp. 2487–2492).

Chen, J., Gauci, M., Price, M. J., & Groß, R. (2012). Segregation in swarms of e-puck robots based on the brazil nut effect. In *International Conference on Autonomous Agents and Multiagent Systems* (pp. 163–170).

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., & Thrun, S. (2005). *Principles of Robot Motion-Theory: Algorithms, and Implementation.* MIT Press.

De Gennaro, M. C. & Jadbabaie, A. (2006). Decentralized control of connectivity for multi-agent systems. In *Decision and Control, 2006 45th IEEE Conference on* (pp. 3628–3633).: IEEE.

Desai, J. P., Ostrowski, J., & Kumar, V. (1998). Controlling formations of multiple mobile robots. In *IEEE International Conference on Robotics and Automation* (pp. 2864–2869).

Dimarogonas, D. V. & Johansson, K. H. (2008). Decentralized connectivity maintenance in mobile networks with bounded inputs. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 1507–1512).: IEEE.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., & Burnier, D. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4), 60–71.

Dorigo, M. & Sahin, E. (2004). Guest editorial. *Autonomous Robots*, 17(2), 111–113.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. *In MHS'95. Proceedings of the Sixth Int. Symposium on Micro Machine and Human Science.*, (pp. 39–43).

Edwards, V., Rezeck, P., Chaimowicz, L., & Hsieh, M. A. (2016). Segregation of heterogeneous robotics swarms via convex optimization. In *ASME 2016 Dynamic Systems and Control Conference* (pp. V001T03A001–V001T03A001).: American Society of Mechanical Engineers.

Erfianto, B., Bambang, R. T., Hindersah, H., & Muchtadi-Alamsyah, I. (2016). Design of connectivity preserving flocking using control lyapunov function. *Journal of Robotics*, 2016, 2.

Ferreira Filho, E. B. & Pimenta, L. (2015a). Segregação de enxames de robôs heterogêneos do tipo integrador simples em múltiplos grupos usando abstrações. *XII Simpósio Brasileiro de Automação Inteligente (SBAI).*

Ferreira Filho, E. B. & Pimenta, L. (2015b). Segregating multiple groups of heterogeneous units in robot swarms using abstractions. *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, (pp. 401–406).

Ferreira Filho, E. B. & Pimenta, L. (2019a). Abstraction based approach for segregation in heterogeneous robotic swarms. *Robotics and Autonomous Systems*, 122.

Ferreira Filho, E. B. & Pimenta, L. (2019b). Decentralized radial segregation in heterogeneous swarms of robots. *Decision and Control, 2019 58th IEEE Conference on*, (pp. 5723–5728).

Ferreira Filho, E. B. & Pimenta, L. (2020). Segregation of heterogeneous swarms of robots in curves. *IEEE International Conference on Robotics and Automation.*

Freeman, R. A., Yang, P., & Lynch, K. M. (2006). Distributed estimation and control of swarm formation statistics. In *American Control Conference, 2006* (pp. 7–pp).: IEEE.

Gerkey, B. P. & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9), 939–954.

Gonçalves, M., Pimenta, L., & Pereira, G. (2011). Coverage of curves in 3d with swarms of nonholonomic aerial robots. *18th IFAC World Congress*, (pp. 10367–10372).

Gross, J. L. & Yellen, J. (2003). *Handbook of graph theory.* CRC press.

Groß, R., Magnenat, S., & Mondada, F. (2009). Segregation in swarms of mobile robots based on the brazil nut effect. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4349–4356).

Guizzo, E. (2008). Three engineers, hundreds of robots, one warehouse. *IEEE spectrum*, 45(7), 26–34.

Hsieh, A., Kumar, V., & Chaimowicz, L. (2008). Decentralized controllers for shape generation with robotic swarms. *Robotica*, 26(5), 691–701.

Inácio, F. R., Macharet, D. G., & Chaimowicz, L. (2019). Pso-based strategy for the segregation of heterogeneous robotic swarms. *Journal of Computational Science*, 31, 86–94.

Jadbabaie, A., L. J. & Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules.s. *IEEE Trans. on automatic control*, 48(6), 988–1001.

Kantaros, Y., Thanou, M., & Tzes, A. (2015). Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints. *Automatica*, 53, 195–207.

Khalil, H. K. (2014). *Nonlinear control.* Pearson Higher Ed.

Klingner, J., Kanakia, A., Farrow, N., Reishus, D., & Correll, N. (2014). A stick-slip omnidirectional powertrain for low-cost swarm robotics: Mechanism for calibration and control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 846–851).

Knudson, M. & Tumer, K. (2010). Coevolution of heterogeneous multi-robot teams. In *In Proceedings of the 12th annual conference on Genetic and evolutionary computation* (pp. 127–134).: ACM.

Kubo, M., Sato, H., Yoshimura, T., Yamaguchi, A., & Tanaka, T. (2014). Multiple targets enclosure by robotic swarm. *Robotics and Autonomous Systems*, 62(9), 1294–1304.

Kumar, M., Garg, D., & Kumar, V. (2010). Segregation of heterogeneous units in a swarm of robotic agents. *IEEE Transactions on Automatic Control*, 55(3), 743–748.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Li, X. & Xi, Y. (2008). Flocking of multi-agent dynamic systems with guaranteed group connectivity. *Journal of Systems Science and Complexity*, 21(3), 337–346.

Maeda, R., Endo, T., & Matsuno, F. (2017). Decentralized navigation for heterogeneous swarm robots with limited field of view. *IEEE Robotics and Automation Letters*, 2(2), 904–911.

Marino, A. & Pierri, F. (2018). A two stage approach for distributed cooperative manipulation of an unknown object without explicit communication and unknown number of robots. *Robotics and Autonomous Systems*, 103, 122–133.

Michael, N., Fink, J., & Kumar, V. (2007). Controlling a team of ground robots via an aerial robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 965–970).

Mitrano, P., Burklund, J., Giancola, M., & Pinciroli, C. (2019). A minimalistic approach to segregation in robot swarms. *arXiv preprint arXiv:1901.10423*.

Mondal, A., B. C. B. L. & Jamshidi, M. (2017). Trajectory tracking by multiple agents in formation with collision avoidance and connectivity assurance. *IEEE Systems Journal*, 12(3), 2449–2460.

Morbidi, F., Freeman, R. A., & Lynch, K. M. (2011). Estimation and control of uav swarms for distributed monitoring tasks. In *American Control Conference (ACC), 2011* (pp. 1069–1075).: IEEE.

Nagi, J., Giusti, A., Gambardella, L. M., & Di Caro G., e. a. (2014). Human-swarm interaction using spatial gestures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3834–3841).

Niccolini, M., Pollini, L., & Innocenti, M. (2008). Decentralized control of swarms with collision avoidance implications. *IFAC Proceedings Volumes*, 41(2), 16002–16007.

Nitschke, G. S., Schut, M. C., & Eiben, A. E. (2012). Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm and Evolutionary Computation*, 2, 25–38.

O'Hara, I., Paulos, J., Davey, J., Eckenstein, N., Doshi, N., Tosun, T., Greco, J., Seo, J., Turpin, M., Kumar, V., & Yim, M. (2014). Self-assembly of a swarm of autonomous boats into floating structures. In *IEEE International Conference on Robotics and Automation* (pp. 1234–1240).

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3), 401–420.

Olfati-Saber, R. & Murray, R. M. (2003). Consensus protocols for networks of dynamic agents. *Proceedings of the 2003 American Control Conference*.

Olfati-Saber, R. & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9), 1520–1533.

Ordoñez, B., Moreno, U. F., Cerqueira, J., & Almeida, L. (2012). Generation of trajectories using predictive control for tracking consensus with sensing. *Procedia Computer Science*, 10, 1094–1099.

Parsegov, S., Polyakov, A., & Shcherbakov, P. (2013). Fixed-time consensus algorithm for multi-agent systems with integrator dynamics. *IFAC Proceedings Volumes*, 46(27), 110–115.

Perkinson, J. & Shafai, B. (2005). A decentralized control algorithm for scalable robotic swarms based on mesh-free particle hydrodynamics. In *IASTED International Conference on Robotics and Applications* (pp. 1–6).

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., & Egerstedt, M. (2017). The robotarium: A remotely accessible swarm robotics research testbed. In *IEEE International Conference on Robotics and Automation* (pp. 1699–1706).

Pimenta, L., Kumar, V., Mesquita, R., & Pereira, G. (2008a). Sensing and coverage for a network of heterogeneous robots. In *IEEE Conference on Decision and Control* (pp. 3947–3952).

Pimenta, L., Michael, N., Mesquita, R., Pereira, G., & Kumar, V. (2008b). Control of swarms based on hydrodynamic models. In *IEEE International Conference on Robotics and Automation* (pp. 1948–1953).

Pimenta, L., Pereira, G., Gonçalves, M., Michael, N., Turpin, M., & Kumar, V. (2013a). Decentralized controllers for perimeter surveillance with teams of aerial robots. *Advanced Robotics*, 27(9), 697–709.

Pimenta, L., Pereira, G., Michael, N., Mesquita, R., Bosque, M., Chaimowicz, L., & Kumar, V. (2013b). Swarm coordination based on smoothed particle hydrodynamics technique. *IEEE Transactions on Robotics*, 29(2), 383–399.

Prorok, A., Hsieh, M. A., & Kumar, V. (2017). The impact of diversity on optimal control policies for heterogeneous robot swarms. *IEEE Transactions on Robotics*, 33(2), 346–358.

Recchiuto, C. T., Sgorbissa, A., & Zaccaria, R. (2016). Visual feedback with multiple cameras in a uavs human–swarm interface. *Robotics and Autonomous Systems*, 80, 43–54.

Remes, B., Hensen, D., Van Tienen, F., De Wagter, C., Van der Horst, E., & De Croon, G. C. (2013). *Paparazzi: how to make a swarm of parrot AR drones fly autonomously based on gps.* Toulouse, France: International Micro Air Vehicle Conference and Flight Competition.

Ren, W. (2007). Second-order consensus algorithm with extensions to switching topologies and reference models. *In American Control Conf.*, (pp. 1431–1436).

Ren, W. & Atkins, E. (2007). Distributed multi vehicle coordinated control via local information exchange. *Int. Journal of Robust and Nonlinear Control: IFAC - Affiliated Journal*, 17(10-11), 1002–1033.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4), 25–34.

Rosato, A., Strandburg, K., Prinz, F., & Swendsen, R. (1987). Why the brazil nuts are on top: Size segregation of particulate matter by shaking. *Physical Review Letters, APS*, 58(10), 1038.

Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198), 795–799.

Santos, V. & Chaimowicz, L. (2011). Uso de hierarquias no controle de enxames robóticos. *Anais do X Simp. Brasileiro de Automação Inteligente, pp. 557-562.*

Santos, V. & Chairmowicz, L. (2011). Hierarquical congestion control for robotic swarms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4372–4377).

Santos, V., Pimenta, L., & Chaimowicz, L. (2014). Segregation of multiple heterogeneous units in a robotic swarm. In *IEEE International Conference on Robotics and Automation* (pp. 1112–1117).

Sartoretti, G., Shaw, S., & Hsieh, M. A. (2016). Distributed planar manipulation in fluidic environments. In *IEEE International Conference on Robotics and Automation* (pp. 5322–5327).

Slotine, J. J. E. & Li, W. (1991). *Applied nonlinear control.* Englewood Cliffs, NJ: Prentice hall.

Soleymani, T., Trianni, V., Bonani, M., Mondada, F., & Dorigo, M. (2015). Bio-inspired construction with mobile robots and compliant pockets. *Robotics and Autonomous Systems*, 74, 340–350.

Szwaykowska, K., Romero, L. M.-y.-T., & Schwartz, I. B. (2014). Collective motions of heterogeneous swarms. *arXiv preprint arXiv:1409.1042.*

Tanner, H., Jadbabaie, A., & Pappas, G. (2005). Flocking in teams of non-holonomic agents. *Lecture Notes in Control and Information Sciences, Cooperative Control*, 309, 458–460.

The MathWorks Inc., Natick, M. (2014). version 8.3.0.532 (r2014a).

Trenkwalder, S. M., Lopes, Y. K., Kolling, A., Christensen, A. L., Prodan, R., & Groß, R. (2017). Openswarm: An event-driven embedded operating system for miniature robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4483–4490).

Trueba, P., P. A. B. F. C. n. P. & Duro, R. J. (2013). Specialization analysis of embodied evolution for robotic collective tasks. *Robotics and Autonomous Systems*, 61(7), 682–693.

Van Den Berg, J., G. S. J. L. M. & Manocha, D. (2011). Reciprocal n-body collision avoidance. *Robotics research.*, (pp. 3–19).

Walker, P., Amraii, S., Chakraborty, N., Lewis, M., & Sycara, K. (2014). Human control of robot swarms with dynamic leaders. In *IEEE/RSJ International Conference on Intelligent Robots and Systems,* (pp. 1108–1113).

Wang, L. & Xiao, F. (2010). Finite-time consensus problems for networks of dynamic agents. *IEEE Transactions on Automatic Control*, 55(4), 950–955.

Wikimedia (2010). Brazilian nut effect.

Wilson, M., Melhuish, C., Sendova-Franks, A. B., & Scholes, S. (2004). Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies. *Autonomous Robots*, 17(2-3), 115–136.

Wurman, P. R., D'Andrea, R., & Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1), 9.

Yang, P., Freeman, R. A., Gordon, G. J., Lynch, K. M., Srinivasa, S. S., & Sukthankar, R. (2010). Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2), 390–396.

Yong, C. H. & Miikkulainen, R. (2009). Coevolution of role-based cooperation in multiagent systems. *IEEE Transactions on Autonomous Mental Development*, 1(3), 170–186.

Zhang, B. & Jia, Y. (2015). Fixed-time consensus protocols for multi-agent systems with linear and nonlinear state measurements. *Nonlinear Dynamics*, 82(4), 1683–1690.

Zhang, G., Fricke, G., & Garg, D. (2013). Spill detection and perimeter surveillance via distributed swarming agents. *IEEE/ASME Transactions on Mechatronics*, 18(1), 121–129.

Zhang, S., Liu, M., Lei, X., Huang, Y., & Zhang, F. (2018). Multi-target trapping with swarm robots based on pattern formation. *Robotics and Autonomous Systems*, 106, 1–13.

Zhu, B., Xie, L., Han, D., Meng, X., & Teo, R. (2017). A survey on recent progress in control of swarm systems. *Science China Information Sciences*, 60(7), 070201.

A

Appendix

A.1 Fundamental Principles

In this Appendix, we review some fundamental principles that were useful in the methodologies shown in Chapters 3 and 4.

A.1.1 Graph Theory

In this work we used graph theory concepts, in the context of avoiding collisions between robots and in the context of the use of consensus protocols. The following are the basic concepts in graph theory.

A graph is defined by:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \tag{A.1}$$

with $\mathcal{V}$ being the vertex or nodes set and $\mathcal{E}$ being the edge set, which connect the vertices.

Another graph:

$$\mathcal{G}' = (\mathcal{V}', \mathcal{E}') \tag{A.2}$$

is a subgraph of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subset \mathcal{V}$ and $\mathcal{E}' \subset \mathcal{E}$ (Bollobás, 1998).

A path is a sequence of vertices where there are edges between each of the consecutive vertices in the sequence. A graph or subgraph is connected if any two of its vertices can be connected by a path. If there are two vertices that cannot be connected then the graph or subgraph is disconnected. The connected components of a graph are the maximal

connected subgraphs of this graph, i.e. are subgraphs that are not contained in other connected subgraphs. Figure A.1 shows an example of a connected graph and an example of a disconnected graph.

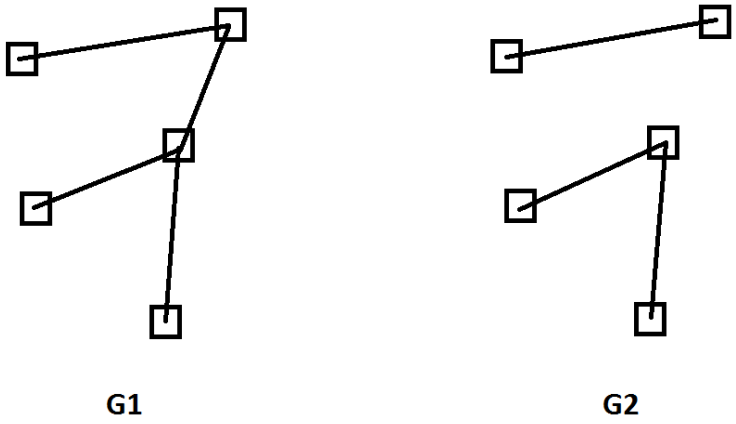


Figure A.1: Example of connected and disconnected graph. Two graphs: G1 and G2. Graph G1 is connected and graph G2 is disconnected.

From a graph one can construct an adjacency matrix $\mathcal{A}$. The adjacency matrix $\mathcal{A} = [\bar{a}_{kl}]$, associated with a graph $\mathcal{G}$ is a square matrix of size $N \times N$, where N is the number of graph nodes, and k and l range from 1 to N. The adjacency matrix elements $(\bar{a}_{kl})$ are weights that indicate whether there is a link between agents k and l.

In this work we will consider only undirected graphs, those where the adjacency matrix $\mathcal{A}$ is symmetric. In this work we will also always consider graphs with weight $\bar{a}_{kl} = 1$ if there is a connection between agents k and l and weight $\bar{a}_{kl} = 0$ if there is no such connection.

The Laplacian matrix of a graph can be defined as

$$\mathcal{L} = \mathcal{D} - \mathcal{A}, \tag{A.3}$$

in which $\mathcal{D}$ is a diagonal matrix which contains information about the number of edges attached to each vertex, for graph $\mathcal{G}$. The diagonal elements of $\mathcal{D}$ are given by

$$d_{kk} = \sum_{l=1}^{N} \bar{a}_{kl}. \tag{A.4}$$

If $\mathcal{G}$ is connected, the second smallest eigenvalue of $\mathcal{L}$ is always real and positive ($\lambda_2 > 0$) (Olfati-Saber & Murray, 2004). This second smallest eigenvalue is the so-called algebraic connectivity of the graph $\mathcal{G}$.

Note that in this work graph theory is used at two different moments: in the context of collisions between robots and in the context of communication between robots.

In Section 3.2 a collision graph is defined, which is used in avoiding collisions between

robots and does not depend on the communication radius between robots, which is not used in Section 3.2. In the Section 3.3 and in Chapter 4 other controllers are defined, in those controllers the collision graph is not used as collisions are avoided in another manner, but a robot communication graph is defined, which defines the neighborhood of each robot.

A.1.2 Other Robot Models

In this document, all the controllers are designed for robots with the double integrator dynamics model, as in Kumar et al. (2010) and Santos et al. (2014):

$$\dot{\boldsymbol{q}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = \boldsymbol{u}, \tag{A.5}$$

in which $\boldsymbol{q}$, $\boldsymbol{v}$ and $\boldsymbol{u}$, are the position, velocity and control input vectors, respectively. Nonetheless, the same controllers can be used considering other models such as the differential drive model in a $2D$ plane. A feedback linearization (Khalil, 2014) technique can be used in order to apply the controller designed for double integrator dynamics model in robots with the differential drive dynamics model. For the sake of simplicity, in this section we omit robot and group indexes. Consider the differential drive dynamics model

$$\begin{cases} \dot{x} = v \cos\theta, \\ \dot{y} = v \sin\theta, \\ \dot{\theta} = w, \end{cases} \tag{A.6}$$

in which θ and w are the rotation angle and rotational velocity, respectively. We are assuming actuation in rotational and translation acceleration, therefore differentiating $\dot{x}$ and $\dot{y}$ in (A.6) we have that

$$\begin{cases} \ddot{x} = a \cos\theta - w\, v \sin\theta \\ \ddot{y} = a \sin\theta + w\, v \cos\theta \end{cases}, \tag{A.7}$$

in which $a = \dot{v}$ is the linear acceleration of the robot. Choosing a point $(x_a = d_a cos\theta + x, y_a = d_a sin\theta + y)$ at a distance d_a of the center of the robot we have

$$\begin{cases} \dot{x}_a = \dot{x} - d_a\, w \sin\theta \\ \dot{y}_a = \dot{y} + d_a\, w \cos\theta \end{cases} \tag{A.8}$$

and

$$\begin{cases} \ddot{x}_a = \ddot{x} - d_a\, \Upsilon \sin\theta - d_a\, w^2 \cos\theta \\ \ddot{y}_a = \ddot{y} + d_a\, \Upsilon \cos\theta - d_a\, w^2 \sin\theta \end{cases}, \tag{A.9}$$

where $\Upsilon = \dot{\omega}$ is the angular acceleration.

Figure A.2 shows a schematic illustrating the chosen point (x_a, y_a), as well as the robot's

linear velocity (v), the rotation angle (θ) and the rotation speed (ω).

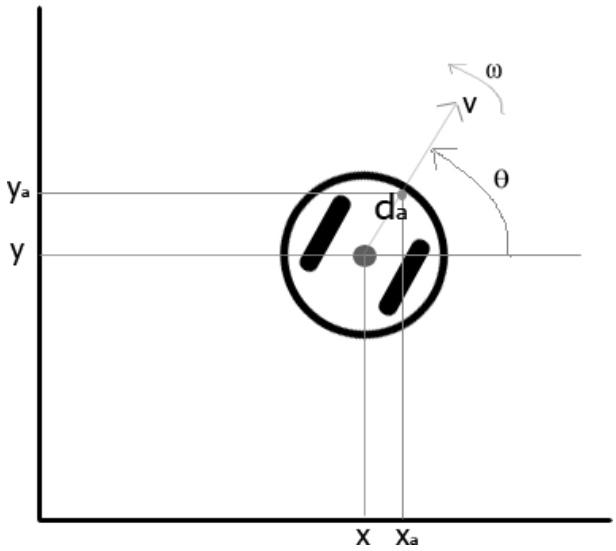


Figure A.2: Scheme illustrating the control of a robot with differential drive dynamics.

Substituting (A.7) in (A.9) and considering its matrix form, we have

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \end{bmatrix} = \underbrace{\begin{bmatrix} cos\,\theta & -d_a\ \sin\,\theta \\ sin\,\theta & d_a\ \cos\,\theta \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ \Upsilon \end{bmatrix}}_{B} + C \tag{A.10}$$

where

$$C = \begin{bmatrix} -w\,v\,sin\,\theta - d_a\,w^2\ \cos\,\theta \\ w\,v\,cos\,\theta - d_a\,w^2\ \sin\,\theta \end{bmatrix}. \tag{A.11}$$

Thus,

$$\begin{bmatrix} a \\ \Upsilon \end{bmatrix} = \Big[A\Big]^{-1} \Big\{-C + u_v\Big\} \tag{A.12}$$

in which the control input u_v is the same that would be used to drive a punctual robot positioned at the point d_a with the double integrator dynamics model:

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \end{bmatrix} = u_v. \tag{A.13}$$

Therefore, with this feedback linearization technique we can use the controllers proposed in this thesis in differential drive robots. Similar approaches can be devised for robots with different dynamics.