Robotics and Autonomous Systems 122 (2019) 103295

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Abstraction based approach for segregation in heterogeneous robotic swarms☆

Edson B. Ferreira-Filho, Luciano C.A. Pimenta *

Graduate Program in Electrical Engineering - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil

ARTICLE INFO

Article history:
Received 21 December 2018
Received in revised form 23 August 2019
Accepted 23 September 2019
Available online 1 October 2019

Keywords:
Robot swarms
Heterogeneous swarms
Segregation behavior
Abstractions

ABSTRACT

The focus of this study is to design individual control laws that segregate multiple groups of mobile heterogeneous robots. Our approach is based on the use of abstractions to represent each group of robots and an artificial potential function to segregate the groups. Different from other works in the literature, we prove that with our controller the system will always converge to a state where robots of the same group will be together while separated from robots of different groups. We also propose a collision avoidance scheme which does not interfere in the segregation controller. Furthermore, our controller has a local property, meaning that the controller might not require global information of the whole swarm to converge to the segregated state. The approach is validated with simulations varying the number of robots and groups and experiments with real robots.

1. Introduction

Robotic swarms are systems formed of a large number of relatively simple robots that interact to each other to solve a task that is beyond each robot individual capabilities [1].

Robot swarms are commonly inspired by 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 restrictions of hardware.

Some researchers have been able to build real highly scalable swarms. In [2] a swarm of 100 real robots is used and in [3], for the first time, a swarm of 1000 real robots is presented. In [4] an operational system is designed for miniature robots with limited on-board resources, similar to the ones in [2] and [3].

An interesting application of a robot swarm is shown in [5], where a swarm formed of floating boats is connected to form a bridge. In [6] a swarm of boats is used to manipulate a floating object. In [7] a swarm autonomously construct a protective barrier. Other previously considered applications are perimeter surveillance [8], spill detection [9], image capturing for entertainment [10], target enclosure [11] and trapping [12], interaction with humans [13–15] and manipulation of objects [16]. 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 [17,18] and Intel's drone light show that are used for entertainment, as in the *2017 Superbowl halftime show* [19]. One can think of situations where it would be useful in both applications to have heterogeneous robots. Heterogeneous robot swarms are those formed of 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 [20].

☆ This work was in part supported by the project INCT (National Institute of Science and Technology) under the grant CNPq (Brazilian National Research Council) 465755/2014-3, FAPESP (São Paulo Research Foundation), Brazil 2014/50851-0. This work was also supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (Finance Code 88887.136349/2017-00), and CNPq, Brazil (grant number 311063/2017-9). Edson B. Ferreira Filho holds a scholarship from CAPES and Luciano C. A. Pimenta holds a scholarship from CNPq.

* Corresponding author.
E-mail addresses: edsonbffilho@ufmg.br (E.B. Ferreira-Filho), lucpim@cpdee.ufmg.br (L.C.A. Pimenta).

https://doi.org/10.1016/j.robot.2019.103295
0921-8890/

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, 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. To solve this problem one should formulate individual control laws to 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 paper 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 paper 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) [21] 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 paper focuses in the second layer responsible to solve the problem of spatially segregate 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 a formal proof that the system converges to segregation as desired. Our approach is based on the consideration of a double integrator robot model and the use of abstractions [22] and artificial potential functions [23]. Abstractions are virtual entities used to represent a group of robots. This paper extends the ideas proposed in our earlier work published in a conference [24], where we used abstractions and an artificial potential function to segregate groups of robots disregarding collisions among the robots.

This paper is organized as follows. The next section discusses some related work in the field and the contributions of this work. Section 3 gives a background and formally defines the problem while Section 4 presents our proposed solution. Simulations and experimental results are presented in Sections 5 and 6, respectively. Finally, in Section 7 we conclude and propose possible avenues for future work.

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 of the 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 [25]. 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 an offline designed control law 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 controller 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 [26,27] and [28]. Nonetheless, we believe the evolution based approaches might be used together with our method 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 controller 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 further important work in autonomous control.

2.1. Homogeneous robots and abstractions

Behavior-based models were the first used to control virtual swarms in the context of graphic computation [29]. These models define behaviors that are activated in preestablished conditions. In [30] behavior based models are used to control a group of robots, so that robots keep a desired formation. In these cases, the robots are considered to be identical agents, forming homogeneous groups.

Some of the algorithms used in homogeneous swarm coordination are based on artificial potential functions, where goals are attractive areas and obstacles are repulsive. In [31], potential functions are used to avoid collisions between robots while these robots move together. In [32] and [33] artificial forces are used to make robots spread along complex curves maintaining separation between robots. There are several other related approaches such as those based on the use of fluid dynamic equations to coordinate the group motion, [34–36] and more recently, in [37]. This work also uses potential functions and the details will be described in Section 3.3.

Some researchers have addressed the problem of controlling large robot groups by mapping the original problem into a problem in a lower dimensional space. In [22], virtual structures called abstractions are created to model a group of robots. When such structure is created, one can control a group of robots as if the whole structure were a single entity, which facilitates the problem of navigation of multiple robots. Other works, such as [38] and [39], have also used this notion of abstractions in the motion of multi-robot systems. This structure will be used here as well and it will be detailed in Section 3.1.

2.2. Heterogeneous robots

The study of heterogeneous systems has attracted the attention of different research groups more recently.

A nice example of such system is the project *Swarmanoid* [40], where aerial robots work together with ground robots to accomplish a given task.

In [41] and [42], heterogeneous robots are used in an area coverage task. The robots are different in the sense that the robots have different range in their sensing capabilities.

In [43], a communicating network with heterogeneous robots, aerial and ground robots, working as signal routers is presented. These robots move in the environment while maintaining the network connected.

Recently, in [44] a decentralized navigation method for heterogeneous swarms of robots with limited field of view is proposed. They propose a leader-following algorithm that allows the swarm to maintain connectivity during navigation where robots have different sensing ranges, fields of view, maximum velocities and accelerations.

A great advantage of heterogeneous robot swarms is the flexibility to divide the swarm into different groups so that each group of robots with certain common characteristics perform tasks that are better suited to its characteristics. In this context, heterogeneous swarms might be more likely to achieve better performance in comparison to homogeneous swarms. For robots with different characteristics to be used and perform their tasks, it may be necessary to physically separate the swarm into groups containing only robots of the same type, so that these robots can perform the task proposed for their group.

This problem of spatially separating robots of different types in a heterogeneous swarm is the focus of this work. This is the so-called swarm segregation problem. We show some works that aim to solve this problem next.

2.3. Segregation in robot swarms

In [45], a scheme to coordinate multiple robots is proposed where clusters of robots are formed hierarchically and those clusters are coordinated using the hierarchy to reach a desired formation. A fully actuated single integrator robot model is used and the control is done in a reactive centralized fashion. The scheme can be used to deal with the segregation in robot swarms although it is unclear if the scheme is scalable for swarms with lots of robots and groups.

An algorithm that deals with segregated navigation for single integrator robots in multiple groups is proposed in [46]. This algorithm is inspired by the concept of velocity obstacles, where robots can only choose velocities that will not result in collision and maintain segregation while navigating through the environment.

A theoretical foundation to make swarms of robots rendezvous at multiple locations is proposed in [47]. This foundation could be used to make groups of robots segregate although in this case robots aggregate based solely on their initial positions and a map with *attraction basins* is needed.

The most relevant works that deals directly with swarm segregation are [48–52] and [53].

In [48] a centralized algorithm capable of segregating robots is developed, based on the *Brazilian nut effect*. When a container with a big sphere and a lot of smaller spheres is shaken, the big sphere goes up even when it is denser than the other spheres. Similarly, a mixture of different size particles will segregate by the size of the particles when it is shaken. This is the effect called the *Brazilian nut effect* [54].

In the work of [48], despite robots having the same size, they simulate the behavior of different sized particles. In [50], this approach is implemented in *e-puck* robots successfully. A stability proof for the proposed controller is not shown.

In [52] a study is presented with robots that differ in their dynamics. The swarm is divided in two groups, one less maneuverable and other that can be accelerated faster than the first group. Segregative patterns emerge naturally with the proposed dynamics. This is the only work found in the literature in which the difference in the heterogeneous swarm is in the dynamics of the robots. Also, no convergence proof is shown.

In [53] a centralized algorithm is developed for robots modeled as single integrators. The algorithm is based in convex optimization, computing the convex hulls of each group and then segregating groups until there are no intersections among the convex hulls. A formal convergence proof to segregation is shown for single integrator robots disregarding collisions among robots in the proof even though a collision avoidance scheme is proposed.

Recently, in [55] a decentralized algorithm is proposed for single integrator robots. This approach is inspired on the Particle Swarm Optimization method (PSO) and its interesting in the sense of not requiring global information to converge although no convergence proof to segregation is shown. Our work is mostly related to the works that deal with the segregation problem and also use the double integrator robot model, these are [49] and [51].

In [49], an artificial potential function based on the biological differential adhesion cellular model is used. This was the first

work to reach segregation in a distributed way to appear in the literature. The convergence proof for segregation is shown for two groups of robots. In [51], a similar approach is shown, but for several groups of robots. Stability proof is shown, but convergence to segregation is not shown, which means that it may not always occur.

The segregation problem in swarms of robots is indeed a relevant problem in the field and it has been recently studied by different research groups. This work was motivated by the fact that none of the works found in the literature present a controller that can reach segregation in swarms of robots with double integrator agents, multiple groups of robots, integrated collision avoidance algorithm and formal proof of convergence.

2.4. Contributions

In this work a new segregation controller is proposed. In fact, this is an extended version of our conference paper [24]. In [24] the main contribution was a controller that achieves segregation for multiple groups of robots along with a convergence proof that guarantees that segregation will always occur. In [24] collisions between robots were totally disregarded.

In this paper we also propose a new approach to deal with inter-robot collisions while maintaining the convergence proof. We also provide more details regarding the proposed segregation controller. With the new collision avoidance scheme, a practical issue has arisen: in order to maintain the states of abstractions as desired, in some cases robots reach velocities much higher than the velocity of the mean of its abstraction, which is an undesired behavior. Thus, we also propose a scheme to treat this issue that mainly arises after collisions have been dealt with. This scheme could also be potentially useful in other scenarios that robots have undesirably acquired high velocities for some reason. In addition, experiments with real robots, with the new collision avoidance scheme, are shown for the first time. In this paper, three experiments are shown: one without the addition of intentional errors and two with the addition of intentional errors to show how the proposed controller is robust even when localization errors are expressive.

In comparison to other previously proposed distributed approaches such as [49] and [51], our controller may not need information from all the robots of the swarm. As it will be clear in the text, in most cases the robots need information from only a subgroup of the swarm.

3. Setup

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

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

Assume these N_j robots belong to a single group j out of M possible groups. The index j indicates to each group a robot belongs, $j = 1, 2, \ldots, M$, N_j indicates the number of robots in group j and $k = 1, 2, \ldots, N_j$ are the robots of group j.

Consider the double integrator model for each robot:

$$\dot{\boldsymbol{q}}_j^k = \boldsymbol{v}_j^k, \quad \dot{\boldsymbol{v}}_j^k = \boldsymbol{u}_j^k \quad k = 1, 2, \ldots, N_j, \tag{2}$$

in which $\boldsymbol{u}_j^k$ is the control input of robot k that belongs to group j.

Our goal is to devise control laws that allows this heterogeneous group of double integrator robots to segregate into the M homogeneous groups.

Remark 1. For the sake of simplicity in the presentation of the proposed methodology we consider mobile robots in a 2D plane. However, it is straight forward to apply our approach in 3D by considering an additional z component in the position vectors. In Section 5 we present simulations in 3D environments to show the scalability of the method.

Remark 2. In this paper we will design a controller for robots with the double integrator dynamics model (Eq. (2)) as in [49] and [51]. Nonetheless, the same controller can be used considering other models such as the differential drive model in a 2D plane. A feedback linearization 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 remark we omit robots and groups indexes. Consider the differential drive dynamics model

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \end{cases}, \tag{3}$$

where θ and w are the rotation angle and rotational velocity, respectively. We are assuming actuation in acceleration, therefore differentiating $\dot{x}$ and $\dot{y}$ in (3) we have that

$$\begin{cases} \ddot{x} = a\cos\theta - w\,v\sin\theta \\ \ddot{y} = a\sin\theta + w\,v\cos\theta \end{cases}, \tag{4}$$

where $a = \dot{v}$ 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

$$\begin{cases} \dot{x}_a = \dot{x} - d_a\,w\sin\theta \\ \dot{y}_a = \dot{y} + d_a\,w\cos\theta \end{cases} \tag{5}$$

and

$$\begin{cases} \ddot{x}_a = \ddot{x} - d_a\,\Upsilon\sin\theta - d_a\,w^2\cos\theta \\ \ddot{y}_a = \ddot{y} + d_a\,\Upsilon\cos\theta - d_a\,w^2\sin\theta \end{cases}, \tag{6}$$

where $\Upsilon = \dot{\omega}$ is the angular acceleration. Substituting (4) in (6) and considering its matrix form, we have

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -d_a\sin\theta \\ \sin\theta & d_a\cos\theta \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ \Upsilon \end{bmatrix}}_{B} + C \tag{7}$$

where

$$C = \begin{bmatrix} -w\,v\sin\theta - d_a\,w^2\cos\theta \\ w\,v\cos\theta - d_a\,w^2\sin\theta \end{bmatrix}. \tag{8}$$

Thus,

$$\begin{bmatrix} a \\ \Upsilon \end{bmatrix} = [A]^{-1}\{-C + u_v\} \tag{9}$$

where the control input u_v is the same that would be used to drive a double integrator robot:

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \end{bmatrix} = u_v. \tag{10}$$

Therefore, with this feedback linearization technique we can use the controller we will propose for double integrator dynamics in differential drive robots. Similar approaches can be devised for robots with different dynamics.

Before we formally state the segregation problem addressed in this work, we need first to describe our abstractions.

3.1. Abstractions

Some approaches commonly used to control robot swarms treat 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 class is used and the entity representing the group is called abstraction.

In [22] a large number of robots is controlled by means of an abstraction, mapping the configuration space into a space with lower dimension. In this work, we define an abstraction like the circular one presented in [22] 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

$$\boldsymbol{\mu_j} = \frac{1}{N_j}\sum_{k=1}^{N_j} \boldsymbol{q_j^k}. \tag{11}$$

The total number of robots in the system is then given by $\sum_{j=1}^{M} N_j$.

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).

In this paper, for the sake of simplicity, robots are always considered to be in a $2D$ plane. As it might be clear along the text, our results are easily extended to higher dimensions. Therefore $\boldsymbol{\mu_j}$ is given by:

$$\boldsymbol{\mu_j} = \begin{bmatrix} \mu_j^x \\ \mu_j^y \end{bmatrix}, \tag{12}$$

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. The third variable associated with each abstraction is given by

$$\sigma_j = \frac{1}{N_j}\sum_{k=1}^{N_j}[(x_j^k - \mu_j^x)^2 + (y_j^k - \mu_j^y)^2], \tag{13}$$

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 $Q \equiv \mathbb{R}^{2N_j}$ [56]. The variables of each abstraction define the map:

$$\phi_j = \begin{bmatrix} \mu_j^x \\ \mu_j^y \\ \sigma_j \end{bmatrix}. \tag{14}$$

Thus:

$$\phi_j : Q \to G \subset \mathbb{R}^3, \tag{15}$$

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 $\sqrt{N_j\sigma_j}$:

$$C_{\phi_j} = B(\boldsymbol{\mu_j}, \sqrt{N_j\sigma_j}), \tag{16}$$

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

Note that

$$\left\|\boldsymbol{q_j^k} - \boldsymbol{\mu_j}\right\|^2 \leq \sum_{k=1}^{N_j}\left\|\boldsymbol{q_j^k} - \boldsymbol{\mu_j}\right\|^2 = N_j\sigma_j, \tag{17}$$

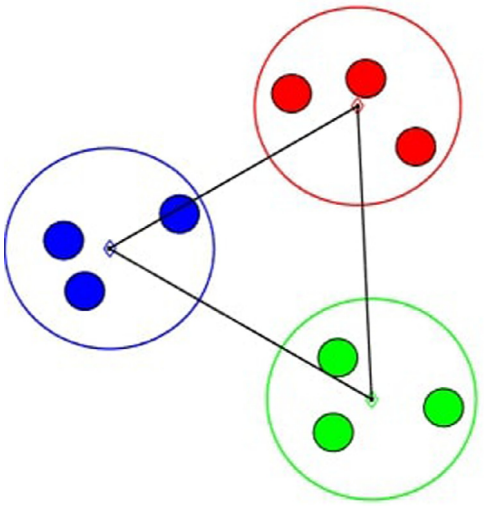

Fig. 1. Circular robots divided in 3 groups of 3 robots.

which implies

$$\left\|\boldsymbol{q_j^k} - \boldsymbol{\mu_j}\right\| \leq \sqrt{N_j\sigma_j}. \tag{18}$$

Eq. (18) 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 [57]. 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 4.3.

3.2. Problem definition

Now that abstractions and robot models are defined, we can formally present the segregation problem.

Problem definition: *Consider $\sum_{j=1}^{M} N_j$ robots of M types with dynamic model given by (2), devise individual control laws that enforce the robots to stay inside their abstractions and each abstraction associated with the robots converge to a state in which:*

$$\bigcap_{j=\{1,\ldots,M\}} C_{\phi_j} = \varnothing. \tag{19}$$

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

Fig. 1 shows a segregated system according to our definition, 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.

3.3. Artificial potential function

This section reviews a potential function proposed in [23]. The control law to be derived in this work is based on the artificial forces derived from this potential function. This function has an interesting property of having a *finite cut-off*, this means that there will not be any virtual force between agents that are very far from each other. This helps giving a local property to the segregation algorithm under some conditions. The potential function that was used in both [49] and [51] to achieve segregation does not have this property.

In the original context, the potential function proposed in [23] was applied directly in the individual robot controllers to achieve flocking behavior. In this 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:

$$\|z\|_\sigma = \frac{1}{\epsilon}[\sqrt{1+\epsilon\|z\|^2} - 1], \tag{20}$$

in which ϵ is a parameter that acts as a gain and is fixed throughout this work and z is the variable being mapped.

Now we can present the artificial potential function:

$$V(\boldsymbol{\mu}) = \frac{1}{2}\sum_j \sum_{i\neq j} \psi_\alpha(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma), \tag{21}$$

in which

$$\psi_\alpha(z) = \int_{d_\alpha}^{z} \gamma_\alpha(s)ds, \tag{22}$$

$$\gamma_\alpha(z) = \rho_h(z/r_\alpha)\frac{c(z-d_\alpha)}{\sqrt{1+(z-d_\alpha)^2}}, \tag{23}$$

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

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.

Function $\rho_h(z)$ is called a bump function and smoothly varies between 0 and 1:

$$\rho_h(z) = \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2}\left[1 + cos(\pi(\frac{z-h}{1-h}))\right] & z \in [h, 1] \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

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

$$\boldsymbol{F_i} = \underbrace{\sum_{j\in B_i} \gamma_\alpha(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma)\boldsymbol{n}_{ij}}_{-\nabla_{\mu_i} V(\boldsymbol{\mu})} + \underbrace{\sum_{j\in B_i} \rho_h(\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|_\sigma / r_\alpha)(\dot{\boldsymbol{\mu}}_\boldsymbol{j} - \dot{\boldsymbol{\mu}}_\boldsymbol{i})}_{\text{velocity consensus}} \tag{25}$$

in which,

$$\boldsymbol{n}_{ij} = (\boldsymbol{\mu_j} - \boldsymbol{\mu_i})/\sqrt{(1+\epsilon\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\|^2)}. \tag{26}$$

In (25), 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 neighbors of group i. Those neighbors are those groups in which the center of their abstractions are in a distance smaller than r from the center of abstraction i.

Fig. 2 shows an example of (22) and (23). Parameter c acts as a gain, while parameter h modifies the smoothness of the force. For any two agents, the force of repulsion/attraction will fade when they are exactly at the desired distance d and when the distance between them is greater than r.

An important Lemma regarding the function in (21) is given below:

Lemma 1. *If $d < r < \sqrt{3}d$, all local minimum of $V(\boldsymbol{\mu})$ implies in the formation of a α-lattice [23].*

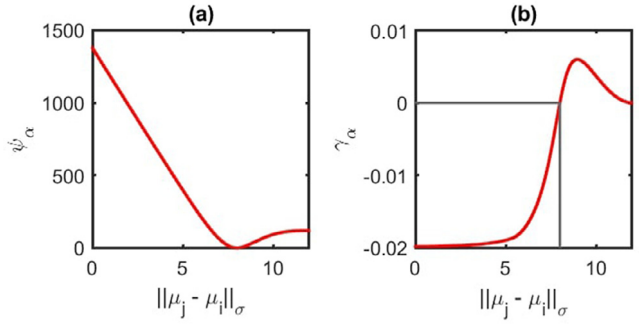

Fig. 2. Parameters: $c = 0.02$, $h = 0.4$, $d_\alpha = 8$, $r_\alpha = 1.5d_\alpha$. (a) Artificial Potential between two agents versus the distance between them. (b) Artificial force between two agents based on the gradient versus the distance between them.

These structures (α-lattices) are lattice shaped in which each vertex is at the same distance d of each other vertex belonging to its neighborhood. An example of an α-lattice can be seen in Fig. 1, in which the α-lattice is a triangle. Those formations must satisfy the set of algebraic restrictions given by

$$\|\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\| = d \quad \forall j \in B_i. \tag{27}$$

The formation of α-lattices, proven in [23] is crucial in the convergence proof to segregation shown in Section 4.3.

4. Robot controller

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 (14)

$$\dot{\phi}_j = D\phi_j \dot{\boldsymbol{q}}_\boldsymbol{j}, \tag{28}$$

where $\boldsymbol{q_j} = [\boldsymbol{q_j^1}^T, \ldots, \boldsymbol{q_j^{N_j}}^T]^T$. By using (11), (13) and (14) we can obtain $D\phi_j$:

$$D\phi_j = \frac{1}{N_j}\begin{bmatrix} 1 & 0 & 2(x^1 - \mu_j^x) \\ 0 & 1 & 2(y^1 - \mu_j^y) \\ \vdots & \vdots & \vdots \\ 1 & 0 & 2(x^{N_j} - \mu_j^x) \\ 0 & 1 & 2(y^{N_j} - \mu_j^y) \end{bmatrix}^T. \tag{29}$$

Since we have robots with double integrator dynamics, we need a relation between the abstraction motion and the robot acceleration. By differentiating (13) twice we have

$$\ddot{\sigma}_j = \frac{2}{N_j}\begin{bmatrix} x_j^1 - \mu_j^x \\ y_j^1 - \mu_j^y \\ x_j^2 - \mu_j^x \\ y_j^2 - \mu_j^y \\ . \\ . \\ . \\ x_j^{N_j} - \mu_j^x \\ y_j^{N_j} - \mu_j^y \end{bmatrix}^T \ddot{\boldsymbol{q}}_\boldsymbol{j} + 2\sigma_j', \tag{30}$$

where,

$$\sigma_j' = \frac{1}{N_j}\sum_{k=1}^{N_j}(\dot{x}_j^k - \dot{\mu}_j^x)^2 + (\dot{y}_j^k - \dot{\mu}_j^y)^2, \tag{31}$$

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

$$\ddot{\phi}_j = D\phi_j \ddot{\boldsymbol{q}}_{\boldsymbol{j}} + \begin{bmatrix} 0 \\ 0 \\ 2\sigma_j' \end{bmatrix}. \tag{32}$$

In order to cancel the dynamics that have emerged in (32) and consequently be able to directly control the states of the abstraction, we propose a controller for all the robots of group j:

$$\boldsymbol{u_j} = \underbrace{D\phi_j^T(D\phi_j D\phi_j^T)^{-1}\left\{-\begin{bmatrix} 0 \\ 0 \\ 2\sigma_j' \end{bmatrix} + \tilde{\boldsymbol{u}}_{\boldsymbol{j}}\right\}}_{\mathcal{S}_j} + \mathcal{N}(D\phi_j), \tag{33}$$

in which ϕ_j is the abstraction map with robot positions given by $\boldsymbol{q_j}$. Based on (2) we have $\ddot{\boldsymbol{q}}_{\boldsymbol{j}} = \boldsymbol{u_j}$, in which $\boldsymbol{u_j} = [(\boldsymbol{u_j^1})^T, (\boldsymbol{u_j^2})^T, \ldots, (\boldsymbol{u_j^{N_j}})^T]^T$. Note that each robot can compute their part of controller (33) without having to compute the controller 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 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 4.1. This term will be null when robots have no risk of colliding to each other.

We have that,

$$det(D\phi_j D\phi_j^T) = \frac{2\sigma_j}{(N_j)^3}, \tag{34}$$

so as long as $\sigma_j \neq 0$, this matrix inverse always exists. From (32), applying the individual control laws defined by the components of $\boldsymbol{u_j}$ in (33) in every robot, the abstraction state will evolve according to

$$\ddot{\phi}_j = \tilde{\boldsymbol{u}}_{\boldsymbol{j}}, \tag{35}$$

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

$$\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \begin{bmatrix} \boldsymbol{U_j^\mu} \\ U_j^\sigma \end{bmatrix}. \tag{36}$$

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 3.3, is given by

$$\boldsymbol{U_j^\mu} = \boldsymbol{F_j}, \tag{37}$$

where $\boldsymbol{F_j}$ is defined according to (25) 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

$$\sigma_j^{des} < \frac{d^2}{4N_j}. \tag{38}$$

We propose now the controller U_j^σ:

$$U_j^\sigma = \ddot{\sigma}_j^{des} + k_1(\dot{\sigma}_j^{des} - \dot{\sigma}_j) + k_2(\sigma_j^{des} - \sigma_j), \tag{39}$$

in which k_1 and k_2 are properly designed positive gains and

$$\dot{\sigma}_j = \frac{2}{N_j}\sum_{k=1}^{N_j}[(x_j^k - \mu_j^x)\dot{x}_j^k + (y_j^k - \mu_j^y)\dot{y}_j^k]. \tag{40}$$

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 (33), with $\mathcal{N}(D\phi_j) = 0$ and $\tilde{\boldsymbol{u}}_{\boldsymbol{j}}$ given by (36), (37), (38) and (39). After some simple manipulation, the elements of $\mathcal{S}_j$ related to robot k in (33) can be written as:

$$\mathcal{S}_j^k = \boldsymbol{U_j^\mu} + \frac{(\boldsymbol{q_j^k} - \boldsymbol{\mu_j})}{\sigma_j}[-2\sigma_j' - k_1\dot{\sigma}_j + k_2(\sigma_j^{des} - \sigma_j)]. \tag{41}$$

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

$$\boldsymbol{u_j^k} = \mathcal{S}_j^k, \tag{42}$$

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

$$\boldsymbol{u_j^k} = \mathcal{S}_j^k + \mathcal{N}(D\phi_j)^k. \tag{43}$$

The individual control laws (Eq. (42)), 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 3.3 thanks to the *finite cut-off* property of the artificial potential field.

If we consider the collision avoidance scheme (Eq. (43)), robots will also need information about all robots that belongs to the groups that are involved in the collision being treated as we will show in the next section.

4.1. Collision avoidance

The proposed control laws until now were designed to control point robots, that is, they do not consider robot sizes. To make this proposal more 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 matrix $\mathcal{N}(D\phi_j)$ pointed out in (33) and (43).

In [38] 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 by two.

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 robot in imminent collision. We use the projection:

$$\mathcal{N}(D\phi_j) = (I - D\phi_j^T(D\phi_j D\phi_j^T)^{-1}D\phi_j)\hat{\boldsymbol{u}}_{\boldsymbol{j}} = \mathcal{N}\hat{\boldsymbol{u}}_{\boldsymbol{j}}, \tag{44}$$

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.

$$\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\| < 2R_b + \delta. \tag{45}$$

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

Moreover, we check if robots are in trajectories that will result in collision, i.e.

$$(\dot{\boldsymbol{q}}_i^k - \dot{\boldsymbol{q}}_j^l)^T(\boldsymbol{q}_i^k - \boldsymbol{q}_j^l) < 0. \tag{46}$$

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 when there is a robot in imminent collision are determined by:

$$\boldsymbol{u}_j^k = \mathcal{S}_j^k + (I - D\phi_j^T(D\phi_j D\phi_j^T)^{-1}D\phi_j)\hat{\boldsymbol{u}}_j. \tag{47}$$

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 *Torricelli's* equation of motion

$$v_f^2 = v_0^2 + 2a\Delta s, \tag{48}$$

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 treated, minus the radius of the robots

$$\Delta s = \frac{1}{2}\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\| - R_b, \tag{49}$$

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:

$$a = (\boldsymbol{u}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|}, \tag{50}$$

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

$$v_0 = (\dot{\boldsymbol{q}}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|}. \tag{51}$$

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

$$a \leq \frac{0 - v_0^2}{2\Delta s}. \tag{52}$$

Fig. 3 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 (Eq. (49)), acceleration a (Eq. (50)) and velocity v_0 (Eq. (51)) related to robot k.

Replacing Δs, a and v_0 in (52), we have a constraint for robot k in relation to robot l:

$$(\boldsymbol{u}_i^k)^T \frac{(\boldsymbol{q}_i^k - \boldsymbol{q}_j^l)}{\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\|} \geq \frac{\left[(\dot{\boldsymbol{q}}_i^k)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|}\right]^2}{\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\| - 2R_b}, \tag{53}$$

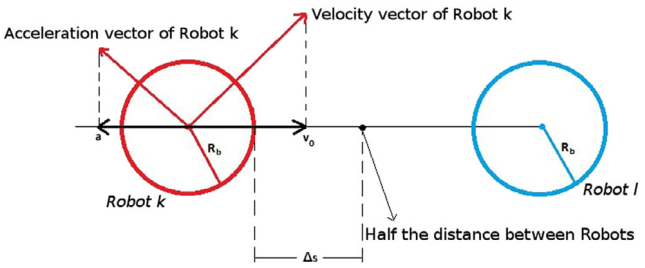

Fig. 3. Schematic illustrating the strategy to avoid collisions.

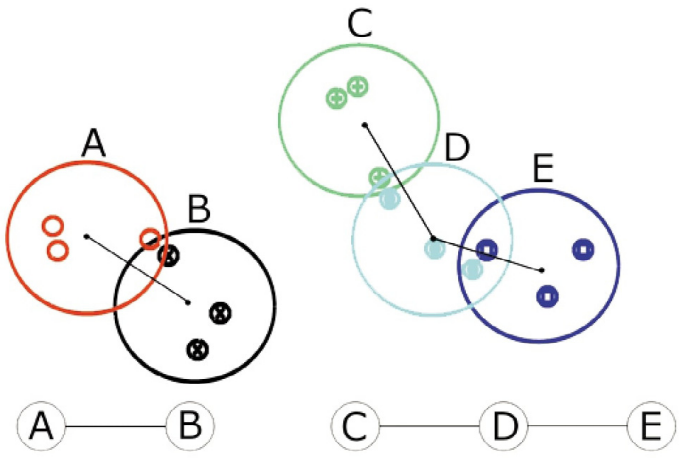

Fig. 4. Graph of Robots in imminent collision.

by analogy, we have for robot l in relation to robot k:

$$(\boldsymbol{u}_j^l)^T \frac{(\boldsymbol{q}_j^l - \boldsymbol{q}_i^k)}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\|} \geq \frac{\left[(\dot{\boldsymbol{q}}_j^l)^T \frac{(\boldsymbol{q}_i^k - \boldsymbol{q}_j^l)}{\left\|\boldsymbol{q}_i^k - \boldsymbol{q}_j^l\right\|}\right]^2}{\left\|\boldsymbol{q}_j^l - \boldsymbol{q}_i^k\right\| - 2R_b}. \tag{54}$$

Therefore we have two constraints that, if respected all the time, robots will never 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:

$$\begin{aligned} &\min \quad \sum_{s\in\Omega_p} \left\|\hat{\boldsymbol{u}}_s\right\|^2, \\ &\text{s. t.} \quad (53) \ \textit{and} \ (54), \end{aligned} \tag{55}$$

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 Fig. 4). In this graph, each abstraction is a node and there are edges between nodes that have imminent collisions between robots of their groups.

Fig. 4 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.

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 run a minimization problem internally.

Constraints of the minimization problem (55) are added for each pair of robots that comply with conditions of (45) and (46). If two robots of groups i and j are in imminent collision, for instance, the objective function of (55) becomes

$$\min \quad \left\|\hat{\boldsymbol{u}}_i\right\|^2 + \left\|\hat{\boldsymbol{u}}_j\right\|^2, \tag{56}$$

with constraints given by (53) and (54).

4.2. Implementation issues

After the minimization problem (55) 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 treated. This effect is perceived visually as robots "orbiting" the center of its own abstraction to maintain the abstraction state.

We now propose another controller with the objective of avoiding the robots to move with velocities much higher than the velocity of its abstraction mean, i.e. robots will tend to move cohesively.

This new controller is called *velocity dissipation*, velocity being the difference between the velocities of the abstraction mean and 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

$$^{f}\dot{\boldsymbol{q}}_j = [\,{}^{f}\dot{x}_j^1 \quad {}^{f}\dot{y}_j^1 \quad \cdots \quad {}^{f}\dot{x}_j^{N_j} \quad {}^{f}\dot{y}_j^{N_j}\,]^T. \tag{57}$$

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 $\hat{\boldsymbol{u}}_j$ as depicted in Algorithm 1.

Algorithm 1 Calculate future robot velocity

Input: Control vector $\hat{\boldsymbol{u}}_j$
Output: Future velocity vector $^{f}\dot{\boldsymbol{q}}_j$
1: Calculate $\mathcal{N}(D\phi_j)$ (Eq. (44))
2: Calculate $\boldsymbol{u}_j^k$ (Eq. (43))
3: Integrate one step (Using robot model, Eq. (2))
4: Obtain $\boldsymbol{v}_j^k$ one step in the future $= {}^{f}\dot{\boldsymbol{q}}_j$

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:

$$^{rel}\dot{\boldsymbol{q}}_j = {}^{f}\dot{\boldsymbol{q}}_j - \mathbb{1} \otimes \dot{\boldsymbol{\mu}}_j, \tag{58}$$

where $\mathbb{1}$ is a column vector given by: $\mathbb{1} = [1, 1, 1..., 1]$. Therefore, in $^{rel}\dot{\boldsymbol{q}}_j$ we have the velocities of the robots in relation to the center of its group's mean velocity.

We will 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

$$\min_{\hat{u}_j} \left\|^{rel}\dot{\boldsymbol{q}}_j\right\|^2. \tag{59}$$

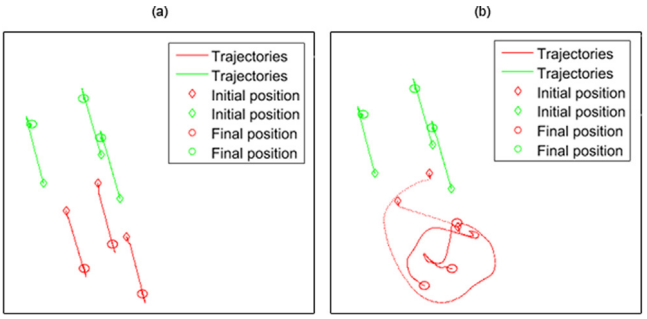

Fig. 5. Trajectories comparison with and without the velocity dissipation controller. (a) Velocity dissipation controller enabled. (b) Velocity dissipation controller disabled. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This minimization problem will generate a vector $\hat{\boldsymbol{u}}_j$ that will be added in the null space 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 of the *velocity dissipation* controller 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 very helpful in dissipating high velocities generated after collisions are avoided.

When conditions (45) and (46) are satisfied, the collision avoidance controller will be non-zero, thus turning off the *velocity dissipation* controller.

Fig. 5 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 Fig. 5(a) and (b) the same initial conditions were used and the simulation was stopped at the same time. In 5(a) the *velocity dissipation* controller is used and it makes the trajectories of red robots more cohesive than the trajectories in 5(b) where the *velocity dissipation* is not used. In 5(b), when a red robot have 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 Fig. 5(b) an undesirable "orbiting" behavior occurred, this can be better visualized in the video presented in https://youtu.be/UvjgwlV_PCE.

4.3. Controller 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 (43) for a group of $\sum_{j=1}^{M} N_j$ robots with dynamics given by (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 3.2 will be solved if (i) solving (55) 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 (39) are properly tuned.*

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

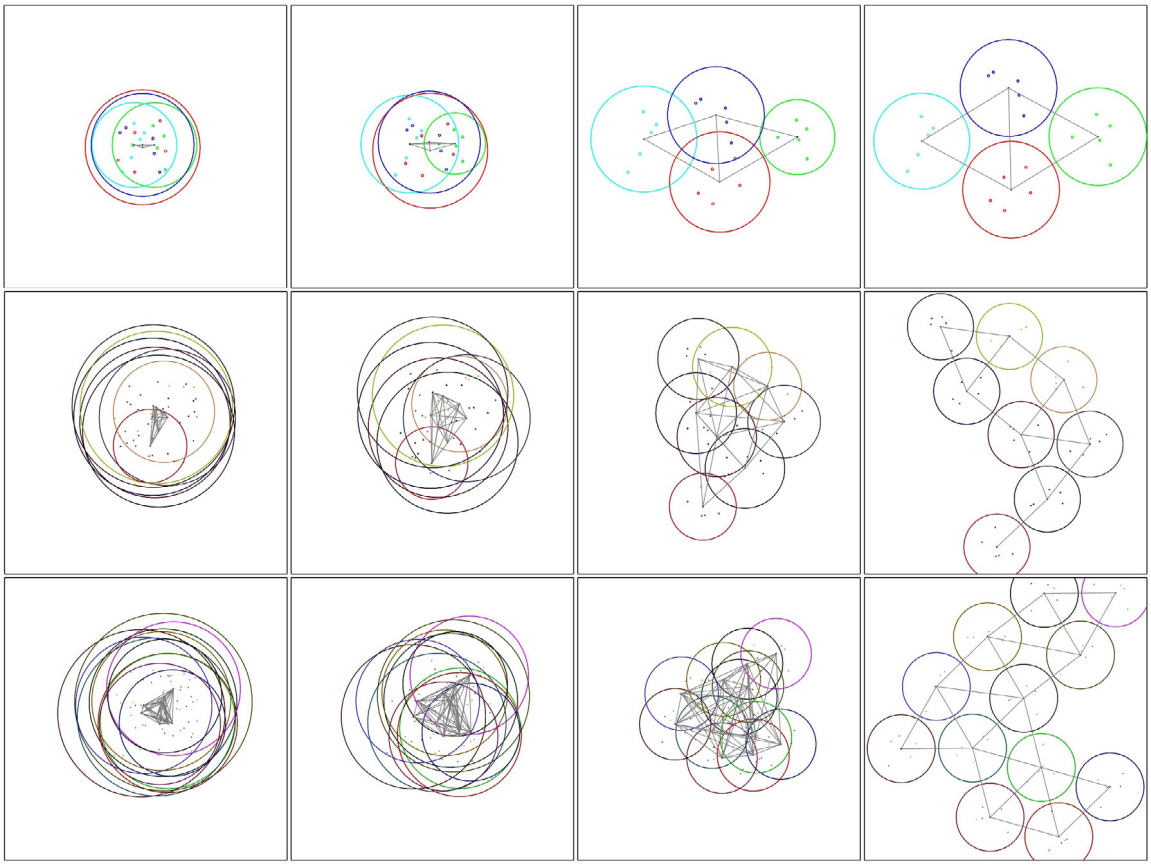

Fig. 6. Simulations in MATLAB, each group has $N_j = 5$ robots. From top to bottom: (a) M = 4 groups. (b) M = 8 groups. (c) M = 12 groups. From left to right, 4 snapshots of initial to final iterations. The snapshots also highlight the abstraction size and the formation of the α-lattices.

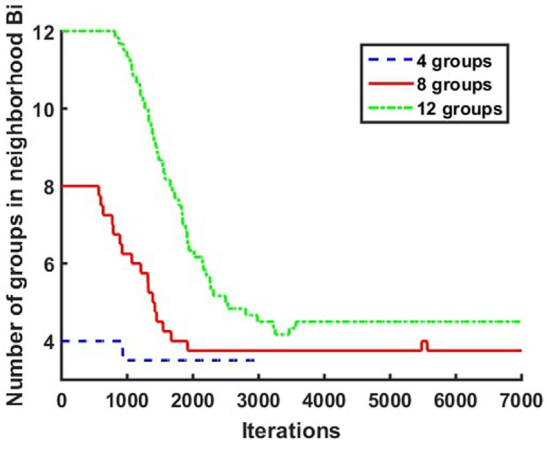

Fig. 7. Information of how many groups each robot needs (average) versus iterations. e.g. With 8 and 12 groups, after the iteration 6000, each robot needs information of up to 5 neighbor abstractions.

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 separated, without intersections.

Due to the assumptions that the robots do not start at the same position and the collision scheme in (55) 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 (33) always exists, then the motion of the abstraction will be given by (35). From (39) 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 [58]. Since the radius is defined according to $R_j = \sqrt{N_j \sigma_j}$, we know from Section 3.1 that the robots of 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 [23]. In this 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 (25) (see *Algorithm 1* in [23]) 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 3.3) 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 Eq. (38)) 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

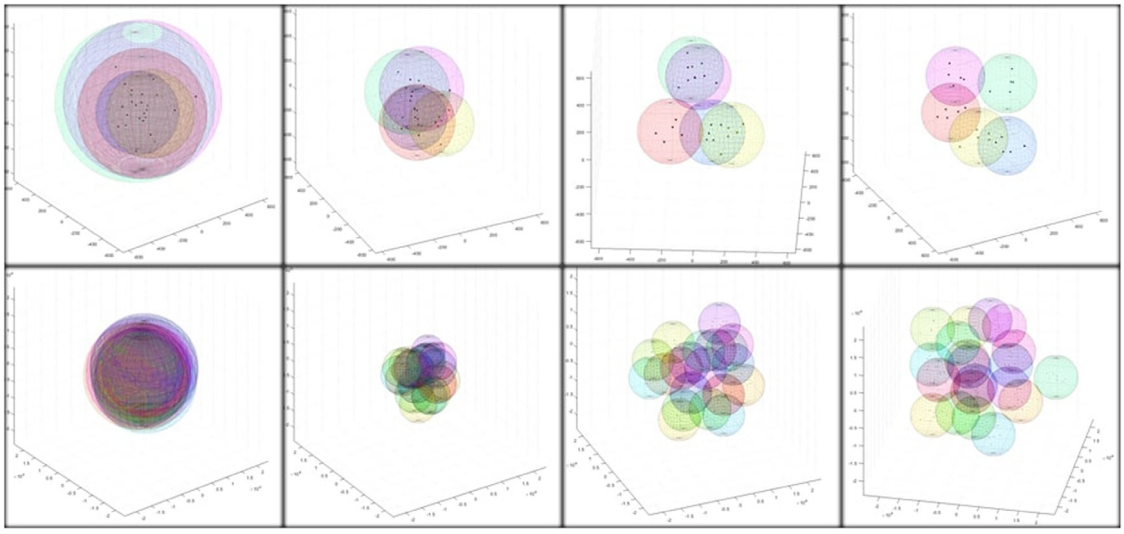

Fig. 8. 3*D* Simulations in MATLAB. Top: each group has $N_j = 5$ robots and the system has 5 groups. Bottom: each group has $N_j = 7$ and the system has 20 groups. From left to right, 4 snapshots of initial to final iterations. The snapshots also highlight the abstraction size (bigger spheres). Snapshots are rotated to help visualization.

Fig. 9. Sequence of snapshots of the unbalanced experiment with 10 real robots. Leftmost snapshot: t = 0, initial position. Middle left snapshot: t = 40 s, first segregation. Middle right snapshot: t = 73 s, second segregation. Rightmost snapshot: t = 98 s, third segregation. Snapshots also highlight the abstraction size and different markers for each type of robot.

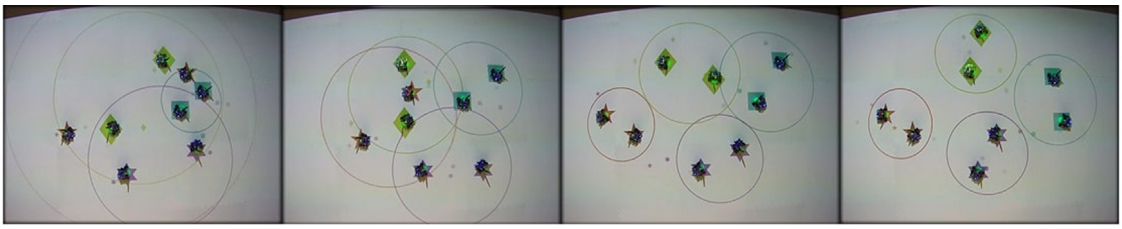

Fig. 10. Sequence of snapshots of the experiment with 8 real robots with added intentional errors. Leftmost snapshot: t = 0, initial position. Middle left snapshot: t = 4 s. Middle right snapshot: t = 24 s. Rightmost snapshot: t = 64 s, groups are segregated. Snapshots also highlight the abstraction size and different markers for each type of robot. Smaller markers show the position of the robots with added intentional errors.

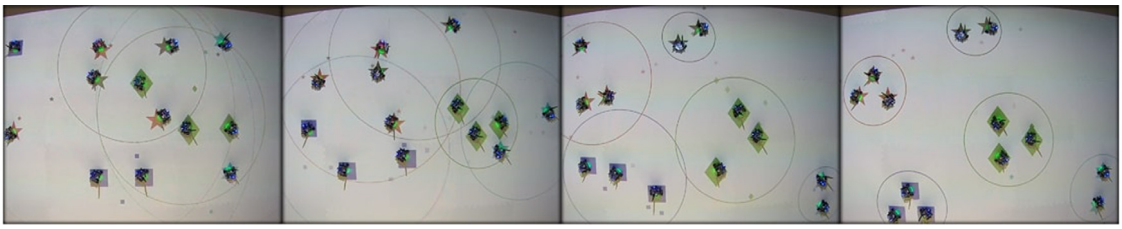

Fig. 11. Sequence of snapshots of the experiment with 13 real robots with added intentional errors. Leftmost snapshot: t = 0, initial position. Middle left snapshot: t = 9 s. Middle right snapshot: t = 34 s. Rightmost snapshot: t = 72 s, groups are segregated. Snapshots also highlight the abstraction size and different markers for each type of robot. Smaller markers show the position of the robots with added intentional errors.

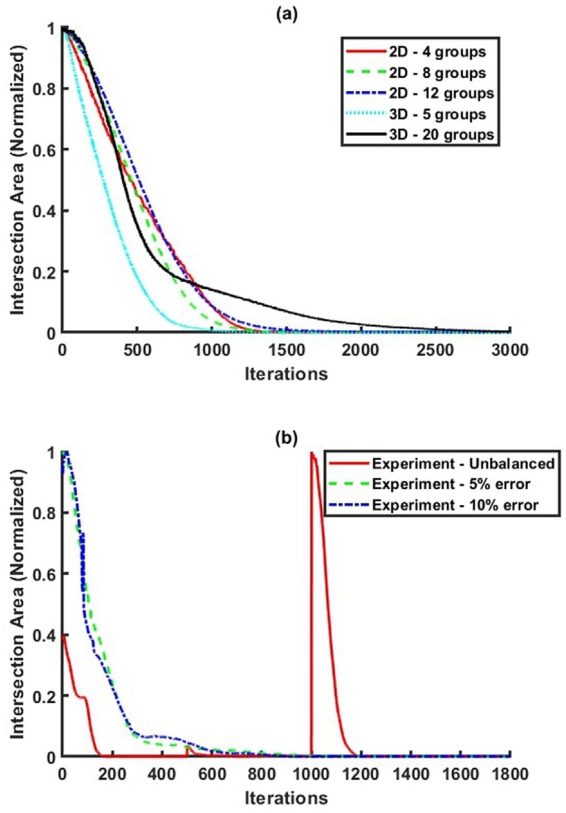

Fig. 12. Evolution of the intersection area related to the simulations of Figs. 6 and 8 and the experiments of Figs. 9–11. (a) Evolution of the intersection area of the 2*D* and 3*D* simulations. (b) Evolution of the intersection area of the three experiments with real robots.

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.

5. Simulations

The proposed controller was first tested using MATLAB [59] with our model of double integrator, holonomic robots as defined in Section 3. In this section we present three simulation runs in MATLAB in a two-dimensional space and two simulation runs in a three-dimensional space. Although our approach was presented considering a 2*D* setup, it is straight forward to derive the same equations in 3*D*, by adding the third coordinate z. We presented it in 2*D* only for the sake of simplicity in the derivation. 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 6 can be found in https://youtu.be/ox2LFwau1-A.

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 3.2, was visually reached and after the formation of the α-lattices was visually stable.

In the 2*D* 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 Fig. 6. We also show two 3*D* simulations in Fig. 8 to exemplify that the approach is scalable to higher dimensions.

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 [51] and [49], Fig. 7 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.

5.1. Discussions

We can see from the simulations that our approach is different from Kumar's and Santos's works and has valuable additions to the early conference work [24], such as the collision avoidance scheme. From Figs. 6 and 8 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 Fig. 12(a) it is possible to see that the intersection area between abstractions (19) in the 2*D* and in the 3*D* simulations converge 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.

From Fig. 7, 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 run the collision avoidance algorithm from robots that are outside the neighborhood B_i. In (55) it is necessary to have information from all the groups in the connected graph component Ω_p.

6. Experiments

The experiments with real robots were performed using the *Robotarium* testbed described in [60]. The testbed uses *GRITSBot X* robots, which is an improved version of the *GRITSBot* presented in [60]. These robots are differential drive robots, meaning that

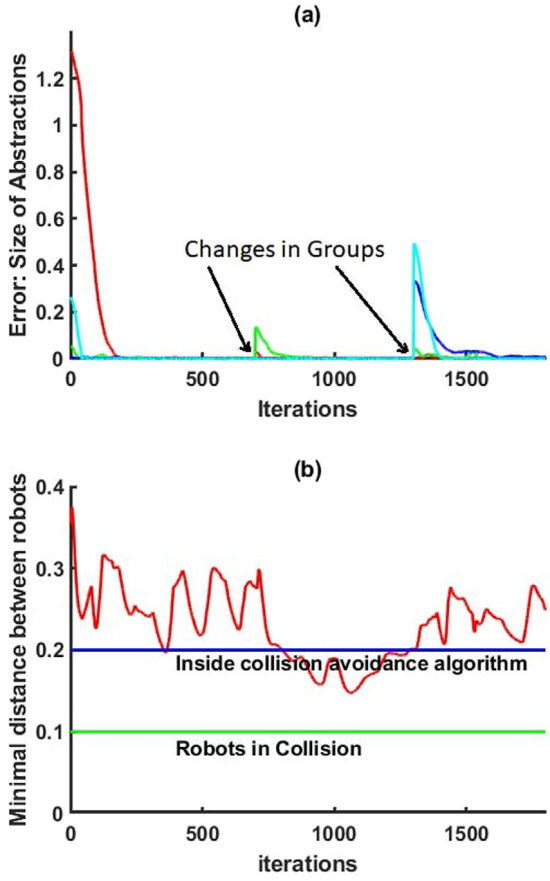

Fig. 13. Data plot related to the experiment of Fig. 9. (a) Evolution of abstractions size errors. (b) Evolution of the minimal distance between robots.

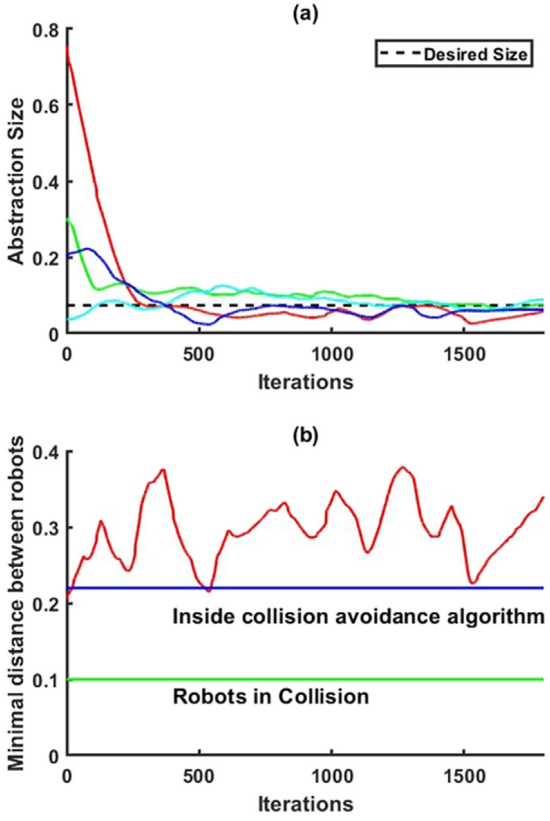

Fig. 14. Data plot related to the experiment of Fig. 10. (a) Evolution of abstractions sizes. (b) Evolution of the minimal distance between robots.

they only have two wheels (non-holonomic robots). We use a feedback linearization [61] technique so that we can use our controller, designed for holonomic robots, in differential drive as show in Remark 2.

We assumed that every robot have access to the information of its own position and 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 the robots of 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 3.3). As the used 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 the robustness of the proposed approach.

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 run for 1800 iterations which was equivalent to 98, 64 and 72 s 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 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 Eq. (55) are obtained with the help of the *fmincon* function of MATLAB.

In the first experiment, with unbalanced groups, we used 10 robots divided into 4 groups that change composition during the experiment. With $t = 0$ s, 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 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 controller (43) adding intentional errors drawn from a normal distribution to its own position and to the position of each other neighbor robot 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.

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 groups in an unbalanced manner, without changing groups compositions $\left(M = 5, N_j = 3 \quad \forall_j \in \{1, 2, 3\} \text{ and } N_j = 2 \quad \forall_j \in \{4, 5\}\right)$.

Frames of the first experiment are shown in Fig. 9, for the second experiment in Fig. 10 and for the third experiment in Fig. 11.

In Fig. 14(a) we show the evolution of the abstraction sizes (σ) for the second experiment. In Figs. 13(a) and 15(a) we show the evolution of the abstraction size errors for the first and third

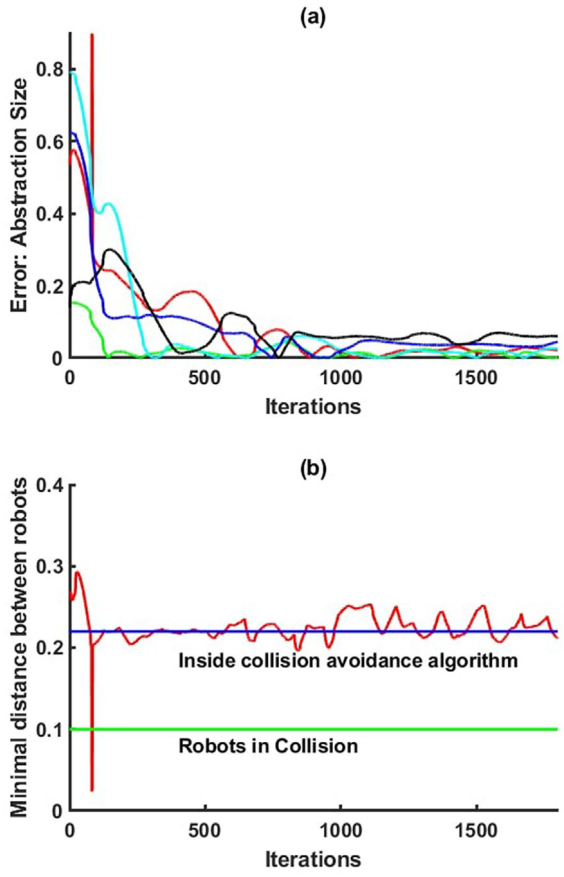

Fig. 15. Data plot related to the experiment of Fig. 11. (a) Evolution of abstractions size errors. (b) Evolution of the minimal distance between robots.

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 Figs. 13(a), 14(a) and 15(a) that the abstractions converge to the desired sizes.

In Figs. 13(b), 14(b) and 15(b) we show the evolution of the minimal distance between robots for the first, second and third experiments, respectively.

We can see in the second, third and fourth frames of Fig. 9 and in the last frame of Figs. 10 and 11 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 (Fig. 12(b)).

In Figs. 13(b), 14(b) and 15(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 Fig. 15. 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.

6.1. Discussions

With real robots, the implementation issues of Section 4.2 were not observed. That is because real robots have limited velocities. Thus, we chose not to use the *velocity dissipation* controller described in Section 4.2. Therefore, we use the controller of Eq. (43) 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 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 Figs. 10 and 11). 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.

7. Conclusions

In this paper we addressed the problem of segregation in robot swarms. We proposed a controller that makes the system reach a state where robots of the same group remain together while separated from robots of other groups. Our controller is based on the use of abstractions to represent each group and an artificial potential function to separate the groups. We have shown a proof of convergence which allows us to say that segregation will always be reached. We have provided a collision avoidance scheme that does not interfere with the segregation controller convergence proof. A *velocity dissipation* controller was also proposed to cope with some issues that have arisen with the collision avoidance scheme. We validated the effectiveness of the proposed controller by considering both simulations and experimental results.

Segregation can be seen as an interesting behavior that might be useful to exhibit in applications that employ heterogeneous swarms and for some reason have the need of bringing agents of the same type together. This might be the case when these agents need to exchange information and take collective decisions without the interference of agents of the other types.

Even with model uncertainties and external disturbances, the use of a feedback linearization approach validated experimentally the controller working with non-holonomic differential drive real robots, although it was originally proposed for perfect double integrator dynamics model.

We assume the robots are equipped with sensors and communication hardware to determine their own positions and velocities and the positions and velocities of other robots. In order to implement our controller in real applications it is necessary to rely on sensors, communication protocols, and probabilistic estimation approaches, such as EKF (Extended Kalman Filter) to provide robust robot state estimation. Addressing the issues related to robot localization and robot–robot communication is out of the scope of this work. Nevertheless, we have performed experiments adding intentional errors to show how our controller behave if we had more expressive errors associated with the localization of the robots. From those experiments, we can conclude that as long as the controller parameters are properly set, such as the desired group size and the desired distance among groups, the segregation controller can be used in real world scenarios.

Different from the previous works found in the literature that also consider the double integrator robot model, the proposed controller may not need global information all the time to reach segregation. In the proposed null-space scheme that deals with inter-robot collision avoidance, it might be necessary, although it is quite unlikely, to have information from all the other robots of the swarm in a given time step if one wishes to maintain the mathematical guarantees of convergence. Thus, in this case one could say the method had to rely on global information at that time step. Nonetheless, thanks to the finite cut-off of our potential function the part of the controller related to segregation convergence depends only on the information of the states of the neighbor abstractions. Moreover, the part of the controller related to collision avoidance depends only on the information from the robots of the groups that have robots in imminent collisions. In Fig. 4, for example, robots from groups A and B do not need information from robots from groups C, D, and E. In conclusion, although we cannot say our method is fully decentralized we can say that to the best of our knowledge our method is the only one available in the literature that provides mathematical convergence guarantees to multiple groups without requiring information from all the other robots of the swarm in every time step.

In future work we will investigate solutions to make the proposed controller fully local in the sense of requiring information from only a small subset of robots of the same group.

It is also important to mention that a current limitation of the proposed approach is the assumption of absence of environmental obstacles. One could think of a manner of modeling obstacles as virtual groups so that the actual groups were repelled from the obstacles. A manner of doing this without losing the mathematical guarantees is an open problem left for future investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2019.103295.

References

[1] M. Dorigo, E. Sahin, Guest editorial, Auton. Robots 17 (2) (2004) 111–113.
[2] J. Klingner, A. Kanakia, N. Farrow, D. Reishus, N. Correll, 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, 2014, pp. 846–851.
[3] M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm, Science 345 (6198) (2014) 795–799.
[4] S.M. Trenkwalder, Y.K. Lopes, A. Kolling, A.L. Christensen, R. Prodan, R. Groß, Openswarm: An event-driven embedded operating system for miniature robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017, pp. 4483–4490.
[5] I. O'Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, M. Yim, Self-assembly of a swarm of autonomous boats into floating structures, in: IEEE International Conference on Robotics and Automation, 2014, pp. 1234–1240.
[6] G. Sartoretti, S. Shaw, M.A. Hsieh, Distributed planar manipulation in fluidic environments, in: IEEE International Conference on Robotics and Automation, 2016, pp. 5322–5327.
[7] T. Soleymani, V. Trianni, M. Bonani, F. Mondada, M. Dorigo, Bio-inspired construction with mobile robots and compliant pockets, Robot. Auton. Syst. 74 (2015) 340–350.
[8] L. Pimenta, G. Pereira, M. Gonçalves, N. Michael, M. Turpin, V. Kumar, Decentralized controllers for perimeter surveillance with teams of aerial robots, Adv. Robot. 27 (9) (2013) 697–709.
[9] G. Zhang, G. Fricke, D. Garg, Spill detection and perimeter surveillance via distributed swarming agents, IEEE/ASME Trans. Mechatronics 18 (1) (2013) 121–129.
[10] B. Remes, D. Hensen, F. Van Tienen, C. De Wagter, E. Van der Horst, G.C. De Croon, Paparazzi: How to Make a Swarm of Parrot AR Drones Fly Autonomously Based on Gps, International Micro Air Vehicle Conference and Flight Competition, Toulouse, France, 2013.
[11] M. Kubo, H. Sato, T. Yoshimura, A. Yamaguchi, T. Tanaka, Multiple targets enclosure by robotic swarm, Robot. Auton. Syst. 62 (9) (2014) 1294–1304.
[12] S. Zhang, M. Liu, X. Lei, Y. Huang, F. Zhang, Multi-target trapping with swarm robots based on pattern formation, Robot. Auton. Syst. 106 (2018) 1–13.
[13] C.T. Recchiuto, A. Sgorbissa, R. Zaccaria, Visual feedback with multiple cameras in a uavs human–swarm interface, Robot. Auton. Syst. 80 (2016) 43–54.
[14] P. Walker, S. Amraii, N. Chakraborty, M. Lewis, K. Sycara, Human control of robot swarms with dynamic leaders, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1108–1113.
[15] J. Nagi, A. Giusti, L.M. Gambardella, Di Caro G, et al., Human-swarm interaction using spatial gestures, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 3834–3841.
[16] A. Marino, F. Pierri, A two stage approach for distributed cooperative manipulation of an unknown object without explicit communication and unknown number of robots, Robot. Auton. Syst. 103 (2018) 122–133.
[17] E. Guizzo, Three engineers, hundreds of robots, one warehouse, IEEE Spectrum 45 (7) (2008) 26–34.
[18] P.R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (1) (2008) 9.
[19] M. Burns, 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/, 2017.
[20] M. Knudson, K. Tumer, Coevolution of heterogeneous multi-robot teams, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, ACM, 2010, pp. 127–134.
[21] B.P. Gerkey, M.J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, Int. J. Robot. Res. 23 (9) (2004) 939–954.
[22] C. Belta, V. Kumar, Towards abstraction and control for large groups of robots, Control Probl. Robot. STAR 4 (2003) 169–182.
[23] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Trans. Automat. Control 51 (3) (2006) 401–420.
[24] E. Ferreira Filho, L. Pimenta, Segregating multiple groups of heterogeneous units in robot swarms using abstractions, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015, pp. 401–406, L..
[25] N. Bredeche, E. Haasdijk, A. Prieto, Embodied evolution in collective robotics: a review, Front. Robot. AI 5 (2018) 12.
[26] C.H. Yong, R. Miikkulainen, Coevolution of role-based cooperation in multiagent systems, IEEE Trans. Auton. Mental Dev. 1 (3) (2009) 170–186.
[27] G.S. Nitschke, M.C. Schut, A.E. Eiben, Evolving behavioral specialization in robot teams to solve a collective construction task, Swarm Evol. Comput. 2 (2012) 25–38.
[28] P. Trueba, A. Prieto, F. Bellas, P. Caamaño, R.J. Duro, Specialization analysis of embodied evolution for robotic collective tasks, Robot. Auton. Syst. 61 (7) (2013) 682–693.
[29] C.W. Reynolds, Flocks, herds and schools: A distributed behavioral model, ACM SIGGRAPH Comput. Graph. 21 (4) (1987) 25–34.
[30] T. Balch, R. Arkin, Behavior-based formation control for multirobot teams, IEEE Trans. Robot. Autom. 14 (6) (1998) 926–939.
[31] H. Tanner, A. Jadbabaie, G. Pappas, Flocking in teams of non-holonomic agents, in: Lecture Notes in Control and Information Sciences, Cooperative Control, Vol. 309, 2005, pp. 458–460.

[32] L. Chaimowicz, N. Michael, V. Kumar, Controlling swarms of robots using interpolated implicit functions, in: IEEE International Conference on Robotics and Automation, 2005, pp. 2487–2492.
[33] A. Hsieh, V. Kumar, L. Chaimowicz, Decentralized controllers for shape generation with robotic swarms, Robotica 26 (5) (2008) 691–701.
[34] J. Perkinson, B. Shafai, A decentralized control algorithm for scalable robotic swarms based on mesh-free particle hydrodynamics, in: IASTED International Conference on Robotics and Applications, 2005, pp. 1–6.
[35] L. Pimenta, N. Michael, R. Mesquita, G. Pereira, V. Kumar, Control of swarms based on hydrodynamic models, in: IEEE International Conference on Robotics and Automation, 2008, pp. 1948–1953.
[36] L. Pimenta, G. Pereira, N. Michael, R. Mesquita, M. Bosque, L. Chaimowicz, V. Kumar, Swarm coordination based on smoothed particle hydrodynamics technique, IEEE Trans. Robot. 29 (2) (2013) 383–399.
[37] A.A. Bandala, G.E. Faelden, J.M. Maningo, R.C.S. Nakano, R.R.P. Vicerra, E.P. Dadios, Implementation of varied particle container for smoothed particle Hydrodynamics-based aggregation for unmanned aerial vehicle quadrotor swarm, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, pp. 1715–1720.
[38] N. Michael, J. Fink, V. Kumar, Controlling a team of ground robots via an aerial robot, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 965–970.
[39] V. Santos, L. Chairmowicz, Hierarquical congestion control for robotic swarms, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4372–4377.
[40] M. Dorigo, D. Floreano, L.M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, D. Burnier, Swarmanoid: A novel concept for the study of heterogeneous robotic swarms, IEEE Robot. Autom. Mag. 20 (4) (2013) 60–71.
[41] L. Pimenta, V. Kumar, R. Mesquita, G. Pereira, Sensing and coverage for a network of heterogeneous robots, in: IEEE Conference on Decision and Control, 2008, pp. 3947–3952.
[42] Y. Kantaros, M. Thanou, A. Tzes, Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints, Automatica 53 (2015) 195–207.
[43] N. Bezzo, B. Griffin, P. Cruz, J. Donahue, R. Fierro, J. Wood, A cooperative heterogeneous mobile wireless mechatronic system, IEEE/ASME Trans. Mechatronics 19 (1) (2014) 20–31.
[44] R. Maeda, T. Endo, F. Matsuno, Decentralized navigation for heterogeneous swarm robots with limited field of view, IEEE Robot. Autom. Lett. 2 (2) (2017) 904–911.
[45] O. Arslan, D. Guralnik, D. Koditsheck, Coordinated robot navigation via hierarchical clustering, IEEE Trans. Robot. 32 (2) (2016) 352–371.
[46] F. Inácio, D. Macharet, L. Chaimowicz, United we move: Decentralized segregated robotic swarm navigation. Distributed Autonomous Robotic Systems, Proceedings in Advanced Robotics, 2018, pp. 313-326.
[47] K.N. Krishnanand, D. Ghose, Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations, Robot. Auton. Syst. 56 (7) (2008) 549–569.
[48] R. Groß, S. Magnenat, F. Mondada, Segregation in swarms of mobile robots based on the brazil nut effect, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4349–4356.
[49] M. Kumar, D. Garg, V. Kumar, Segregation of heterogeneous units in a swarm of robotic agents, IEEE Trans. Automat. Control 55 (3) (2010) 743–748.
[50] J. Chen, M. Gauci, M.J. Price, R. Groß, Segregation in swarms of e-puck robots based on the brazil nut effect, in: International Conference on Autonomous Agents and Multiagent Systems, 2012, pp. 163–170.
[51] V. Santos, L. Pimenta, L. Chaimowicz, Segregation of multiple heterogeneous units in a robotic swarm, in: IEEE International Conference on Robotics and Automation, 2014, pp. 1112–1117.
[52] K. Szwaykowska, L.M.Y.T. Romero, I.B. Schwartz, Collective motions of heterogeneous swarms, IEEE Trans. Autom. Sci. Eng. 12 (3) (2015) 810–818.
[53] V. Edwards, P. Rezeck, L. Chaimowicz, M.A. Hsieh, Segregation of heterogeneous robotics swarms via convex optimization. ASME Dynamic Systems and Control Conference, page V001T03A001, 2016.
[54] A. Rosato, K. Strandburg, F. Prinz, R. Swendsen, Why the brazil nuts are on top: Size segregation of particulate matter by shaking, Phys. Rev. Lett. APS 58 (10) (1987) 1038.
[55] F.R. Inácio, D.G. Macharet, L. Chaimowicz, Pso-based strategy for the segregation of heterogeneous robotic swarms, J. Comput. Sci. 31 (2019) 86–94.
[56] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot Motion-Theory: Algorithms, and Implementation, MIT Press, 2005.
[57] C. Belta, V. Kumar, Abstraction and control for groups of robots, IEEE Trans. Robot. 20 (5) (2004) 865–875.
[58] J.J.E. Slotine, W. Li, Applied Nonlinear Control, Prentice hall, Englewood Cliffs, NJ, 1991.
[59] Massachusetts. The MathWorks Inc., Natick. version 8.3.0.532 (r2014a), 2014.
[60] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, M. Egerstedt, The robotarium: A remotely accessible swarm robotics research testbed, in: IEEE International Conference on Robotics and Automation, 2017, pp. 1699–1706.
[61] J.P. Desai, J. Ostrowski, V. Kumar, Controlling formations of multiple mobile robots, in: IEEE International Conference on Robotics and Automation, 1998, pp. 2864–2869.

Edson B. Ferreira Filho received the B.S. degree in Control and Automation Engineering from the Universidade Federal de Ouro Preto (UFOP), Ouro Preto, Brazil, in 2013. He has received the M.Sc., degree in Electrical Engineering from the Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil, in 2015. Since 2016 he is a PhD candidate in electrical engineering from the Universidade Federal de Minas Gerais (UFMG). His research interests include robotics, multirobot systems and control theory.

Luciano C.A. Pimenta received the B.S., M.Sc., and Ph.D. degrees in Electrical Engineering from the Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil, in 2003, 2005, and 2009, respectively. From April 2007 to June 2008, he was a visiting Ph.D. student with the General Robotics, Automation, Sensing, and Perception Laboratory, University of Pennsylvania, Philadelphia. He is currently an Associate Professor with the Department of Electronic Engineering at UFMG. His research interests include robotics, multi-robot systems and control theory.