United We Move: Decentralized Segregated Robotic Swarm Navigation

Fabrício R. Inácio, Douglas G. Macharet, Luiz Chaimowicz

Abstract A robotic swarm is a particular type of multi-robot system that employs a large number of simple agents in order to cooperatively perform different types of tasks. In this context, a topic that has received much attention in recent years is the concept of segregation. This concept is important, for example, in tasks that require maintaining robots with similar features or objectives arranged in cohesive groups, while robots with different characteristics remain separated on their own groups. In this paper we propose a decentralized methodology to navigate heterogeneous groups of robots whilst maintaining segregation among different groups. Our approach consists of extending the ORCA algorithm with a modified version of the classical flocking behaviors to keep robots segregated. A series of simulations and real experiments show that the groups were able to navigate in a cohesive fashion in all evaluated scenarios. Furthermore, the methodology allowed for a faster convergence of the group to the goal when compared to state-of-the-art algorithms.

Keywords: Swarm robotics, segregative navigation, flocking, ORCA.

1 Introduction

The use of multi-robot systems in different contexts can bring several advantages over single-robot systems. In this sense, we consider the specific scenario of robotic swarms, which are systems composed of a large number of agents seeking to cooperatively accomplish a particular task. Inspired by colonies of social insects that cooperate with each other to carry out tasks of common interest, robotic swarms emerged as an alternative to solve complex problems.

The authors are with the Computer Vision and Robotics Laboratory (VeRLab), Department of Computer Science, Universidade Federal de Minas Gerais, Brazil. e-mails:{fabricio.rod, doug, chaimo}@dcc.ufmg.br. This work was developed with the support of CNPq, CAPES, FAPEMIG.

In general, swarms usually consist of simple agents with little processing power and able to perform a limited set of actions. Another common feature in swarms is the limitation of perception and communication between agents. Normally, each agent is capable of sensing a small portion of the environment and communicating locally, i.e., the communication takes place only between agents who are close to each other.

Similarly to different swarms found in nature (e.g. some insects colonies, bird flocks or fish schools), an important aspect in robotic swarms is the absence of a central entity responsible for coordinating the entire swarm. Hence, each individual behavior contributes to emergent group behaviors that are able to guide the swarm to its objectives [9].

One of these group behaviors is segregation, a natural phenomenon that is commonly used as a sorting mechanism by several biological systems and can be useful in many different tasks and scenarios. This behavior can be important for maintaining robots with similar features or objectives arranged in cohesive groups, while the groups with different characteristics remain separated [11].

In this paper we present a methodology capable of navigating heterogeneous groups of robots whilst maintaining segregation among distinct groups. Our approach consists of extending the Optimal Reciprocal Collision Avoidance (ORCA) algorithm [14] with a modified version of the classical flocking behaviors [8]. The set of possible velocities is informed with the flocking demands, which enables different robot groups to avoid collisions and navigate in a cohesive manner. This methodology improves our previous approach, which was based on velocity obstacles [10], presenting a much better performance. Moreover, differently from other segregation works found in the literature [7, 11], the proposed technique is fully decentralized and assumes local perception only.

2 Related Work

2.1 Navigation and collision avoidance

Finding feasible paths for mobile agents that are either length or time optimized is fundamental in any task that requires the navigation from an initial to a goal position in the environment. However, in general, path planners consider single-agent systems, a static and known environment with a given map, and may not be able to adapt to dynamic issues that may arise during the execution of the path.

An approach that has received much attention in recent years is called Velocity Obstacles (VO). Proposed in [4], this technique uses the agents' velocities and obstacles' positions to calculate the VO-set of all velocities that, if applied to the agents, will result in a collision and then must be considered obstacles. Next, linear and angular speeds that do not belong to the VO-set must be chosen, which ensures a safe navigation to the agents. Several extensions over the VO algorithm have been

proposed in the literature [6, 12, 13, 15], which aim to improve its performance and increase the possibilities for use.

The concept of using the agents velocity space in order to choose acceptable velocities, i.e., the ones that avoid collisions, was further extended resulting in the ORCA algorithm [14]. This technique computes for each other robot a half-plane of allowed velocities. The robot must select its optimal velocity from the intersection of all permitted half-planes, which can be done efficiently by solving a low-dimensional linear system. Moreover, differently from the VO, the ORCA algorithm guarantee local collision-free navigation [14].

2.2 Group segregation in robotic swarms

A characteristic found in social insect communities is the existence of specific roles that segregate the individuals into distinct groups (e.g. bees and ants). Each group of individuals has a distinct set of activities and responsibilities that promote the perpetuation of the colony.

In robotic swarms, as well, it might also be interesting to keep different groups of agents segregated, allowing for example to assign specific tasks according to certain characteristics of each group.

One of the first works to deal with this problem was [7]. It presented an algorithm capable of segregating distinct types of agents by applying different potential functions on the robot according to the type of the agents which are in its neighborhood. The technique produced satisfactory results, however, it can only be applied in scenarios where there are only two different types of agents. In applications that use a very large number of robots, there may be the need to segregate the swarm in more than two groups.

The aforementioned technique was later extended in [11] in order to deal with more than two types of robots. In order to calculate the potential to be used, each agent evaluates whether the neighbors are of their same type or of a different type. Thus, it is possible to obtain the artificial force which must be applied to each agent in order to make the robots with the same type come closer, while robots of different types remain spaced apart. The main restrictions of this technique are the need for global sensing and a balanced number of agents in all groups. More recently, an approach based on abstractions and using an artificial potential function to segregate the groups was proposed [3]. Differently from other works on swarm segregation, it is mathematically guaranteed that the system will always converge to a state where multiple dissimilar groups are segregated. A different segregation algorithm, based on the *Brazil Nut Effect*, is discussed in [5]. A distributed controller considers robots as having distinct virtual sizes, and local interactions make "larger" robots move outwards, segregating the robots in annular structures.

A segregative navigation strategy is proposed in [10], combining the concepts of flocking [8] and the use of abstractions to represent the groups. Built upon the classical VO algorithm [4], it introduces a novel concept called Virtual Group Velocity

Obstacles (VGVO). In this algorithm, a robot i senses the relative position and velocity of every robot j within its neighborhood n_i and builds shapes containing other teams of robots, with the exception of its own. These shapes can be considered as the smallest enclosing disc, the convex hull, or the more general class of α-shapes [2]. In the workspace of robot i, these shapes are considered as virtual obstacles moving at the average velocity of their respective underlying robots. Thus, robot i can build a virtual velocity obstacle specifying all velocities that will lead to a collision with these shapes, assuming that they maintain their current average velocities. However, the cost of computing the virtual shape for each group can be prohibitive, slowing down the navigation.

To cope with this problem, in this paper we propose a decentralized methodology in order to navigate heterogeneous groups of robots whilst maintaining segregation among different groups. We propose a modification over the classical flocking model established by Reynolds [8] and combine it with the ORCA algorithm [14] in order to avoid collisions and keep the group cohesion.

3 Methodology

As mentioned, our methodology is based on a combination of flocking behaviors with the ORCA algorithm. Using local sensors, each robot is able to sense its neighborhood and compute the velocity that will guide it to the goal, while keeping it close to its group. This velocity is used as the preferred velocity by the ORCA algorithm, which is responsible for navigating the robot while avoiding other agents. As in the original algorithms, we assume that an agent has access to the position and velocity of all other agents present in its neighborhood, or can infer these values based on its observations. They can also detect if its neighbors belong to its group and infer their state. Finally, we consider that the robots start in a segregated state and know the direction of a specific goal in the environment. The next sections detail our methodology.

3.1 Robot and Group Modeling

We consider a scenario in which a swarm $\mathscr{R} = \{\mathscr{R}_1, ..., \mathscr{R}_\eta\}$ of η robots must navigate in a static 2D environment. Each robot i is represented by its pose $q_i = \langle x_i, y_i, \theta_i \rangle$, with kinematic model given by $\dot{q}_i = u_i$.

Robots are holonomic and can move in any direction given by the velocity vector u_i. Thus, θ_i is the direction of the movement of robot i in a global frame.

The entire swarm is formed by distinct types (groups) of robots, which we represent by the partition $\Gamma = \{\Gamma_1, ..., \Gamma_m\}$, where each Γ_k contains all agents of type k. We assume that $\forall j,k : j \neq k \rightarrow \Gamma_j \cap \Gamma_k = \emptyset$, i.e., each robot is uniquely assigned to a single type.

3.2 Flocking

In order to maintain a cohesive behavior during navigation, we use a slightly modified version of the classical flocking behaviors presented in [8]. These behaviors are obtained by applying three simple rules, which are described next.

Initially, we define a robot's neighborhood based on its sensing range. During navigation, robot i maintains a neighborhood $\mathscr{N}_i$ around its current position. A neighborhood consists of a circular region of radius λ around the current position of a robot. Within the neighborhood, we may have robots from the same group and robots from different groups. Therefore, we define $\mathscr{N}_i^+ = \{\mathscr{R}_j : \|p_j - p_i\| \leq \lambda \wedge \mathscr{R}_j, \mathscr{R}_i \in \Gamma_k\}$ as the set of all robots of the same group currently located in the neighborhood. Similarly, we define $\mathscr{N}_i^- = \{\mathscr{R}_j : \|p_j - p_i\| \leq \lambda \wedge (\mathscr{R}_j \in \Gamma_u \wedge \mathscr{R}_i \in \Gamma_k, u \neq k)\}$ the set of all robots of other groups in its neighborhood.

The first flocking rule is related to cohesion. This rule is used to keep the agents close to each other. Therefore, agent i must compute the midpoint of the positions of all agents in $\mathscr{N}_i^+$. Next, a velocity vector that moves the agent to the computed point is calculated. Formally:

$$v_{\text{cohesion}} = \left(\frac{1}{|\mathscr{N}_i^+|} \sum_{j \in \mathscr{N}_i^+} p_j \right) - p_i. \tag{1}$$

The second rule was originally proposed to keep a safe distance among agents, with the objective of avoiding collisions between them. However, as will be further detailed, we use the ORCA algorithm in order to ensure obstacle avoidance during navigation. Therefore, we use the separation rule in order to keep distinct groups of agents apart from each other. The agent should move to a position that respects a minimum distance from agents of other groups in its neighborhood, given by:

$$v_{\text{separation}} = \sum_{j \in \mathscr{N}_i^-} (p_i - p_j). \tag{2}$$

We emphasize that the safety distance among robots of the same group are guaranteed by the ORCA algorithm. Therefore, the proposed algorithm does not use the separation rule for agents that belong to the same type.

Finally, the third rule evaluates the alignment of agents of the same group. In order to respect this rule, each agent calculates the average of the spatial orientation of all agents in its group inside its neighborhood. After that, the agent is guided by the calculated average alignment, i.e.,

$$v_{\text{alignment}} = \frac{1}{|\mathscr{N}_i^+|} \sum_{j \in \mathscr{N}_i^+} \theta_j. \tag{3}$$

The final resulting vector v_{flock} that will be used in the control phase is given by:

$$v_{\text{flock}} = k_c \cdot v_{\text{cohesion}} + k_s \cdot v_{\text{separation}} + k_a \cdot v_{\text{alignment}}, \tag{4}$$

which is composed by the weighted sum of the three vectors. Constants k_c, k_s, and k_a are determined empirically, and normally k_c receives a larger value.

3.3 Setting Robot Velocities

The robot velocities can be adjusted according to the different situations an agent may face during its navigation. For example, it can be surrounded by agents from its own group only, or it may be in a position where there are no other agents between itself and the goal or may be close to agents who belong to another group. Thus the control input is composed by a combination of different velocities:

$$u = \alpha.v_{goal} + \beta.v_{flock} + \gamma.v_{aux}. \tag{5}$$

v_{goal} is an attractive velocity that drives the agent towards its goal, v_{flock} is a flocking velocity set according to Equation 4 and v_{aux} is an auxiliary velocity. Constants α, β and γ, as well as the auxiliary velocity, are set according to the scenario faced by the robot. This is controlled by a finite state machine depicted in Figure 1 and explained next.

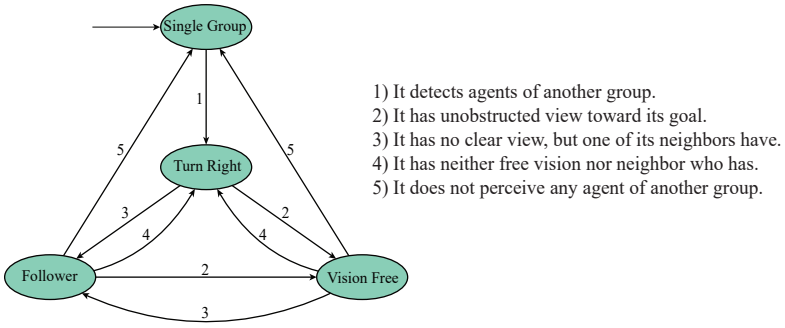


Fig. 1: Finite state machine describing the possible situations faced by each robot.

Single Group: This first state is enabled whenever a robot does not perceive any agent of a different group within its neighborhood. In this case, its action is to move straight to the goal while also keeping cohesion with its group. For this behavior, the constants are set in such way that $\alpha \gg \beta$ and $\gamma = 0$.

Vision Free: The state *Vision Free* is enabled if the agent perceives the presence of one or more robots of different groups, but still has a free line of sight to its goal. More specifically, in a sector defined from the current position of the agent towards its goal and within his field of view there is no agent of a different group. In this case, the agent still tends to move to the goal, but also gives more importance to the flocking factor. Thus, constants are set so that $\alpha > \beta$ and $\gamma = 0$.

Follower: If the robot does not have a clear view towards the goal but detects a neighbor of its own group that is in one of the previous states, it enters in a follower mode, following the agent that has a clear path to the goal, but also keeping the flocking behavior. In this case, the v_{aux} component is set to move the robot towards this neighbor, with $\alpha = 0$ and $\beta < \gamma$.

Turn right: Finally, if the agent does not fit into any of the situations described above, it probably has a congested situation in front of it and should move to avoid this. So, a "traffic rule" is imposed that makes the robot move perpendicularly to its goal direction, *i.e.*, v_{aux} is set in a direction that is perpendicular to v_{goal}. With this behavior, the agents try to get around this congested area rather than trying to cross it. Constants are set to balance this velocity and the flocking component: $\alpha = 0$ and $\beta \simeq \gamma$.

3.4 ORCA

The ORCA algorithm [14] is a velocity-based navigation strategy based on the concept of velocity obstacles [4].

Consider two robots $\mathscr{R}_A$ and $\mathscr{R}_B$ with radii r_A and r_B, positions p_A and p_B and velocities v_A and v_B, respectively. Robot $\mathscr{R}_A$ tries to reach an assigned goal point g_A by selecting a *preferred velocity* v_A^{pref}. The objective is to choose an optimal v_A^*, which lies as close as possible to v_A^{pref}, such that collisions among the robots are avoided for at least a time horizon τ.

The velocity obstacle $VO_{A|B}^{\tau}$ for $\mathscr{R}_A$ induced by $\mathscr{R}_B$ in the local time interval $[0,\tau]$ is the set of velocities of $\mathscr{R}_A$ relative to $\mathscr{R}_B$ that will cause a collision between $\mathscr{R}_A$ and $\mathscr{R}_B$ at some moment before time τ has elapsed. It is assumed that both robots maintain a constant trajectory within that time interval. Formally:

$$VO_{A|B}^{\tau} = \{v | \exists t \in [0,\tau] :: t(v - v_B) \in D(p_B - p_A, r_A + r_B)\} \tag{6}$$

where $D(p_B - p_A, r_A + r_B)$ denote an open disc of radius $(r_A + r_B)$ centered at position $(p_B - p_A)$. If $\mathscr{R}_A$ and $\mathscr{R}_B$ each choose a velocity outside $VO_{A|B}^{\tau}$ and $VO_{B|A}^{\tau}$, respectively, then they will be collision-free for at least the period of time τ.

The half-plane of velocities $ORCA_{A|B}^{\tau}$ can be constructed geometrically as follows. Let us assume that $\mathscr{R}_A$ and $\mathscr{R}_B$ adopt velocities v_A and v_B, respectively, and that these velocities causes $\mathscr{R}_A$ and $\mathscr{R}_B$ to be on collision course, i.e. $v_A - v_B \in VO_{A|B}^{\tau}$. Let $\mathbf{w}$ be the vector from $v_A - v_B$ to the closest point on the boundary of the velocity obstacle:

$$\mathbf{w} = (\underset{v \in \partial VO_{A|B}^{\tau}}{\arg\min} ||v - (v_A - v_B)||) - (v_A - v_B). \tag{7}$$

Then, $\mathbf{w}$ is the smallest change required to the relative velocity of $\mathscr{R}_A$ and $\mathscr{R}_B$ to avert collision within τ time. To "share the responsibility" of avoiding collisions

among the robots, $\mathscr{R}_A$ adapts its velocity by (at least) $\frac{1}{2}w$ and assumes that B takes care of the other half. Hence, the set $ORCA^{\tau}_{A|B}$ of permitted velocities for $\mathscr{R}_A$ is the half-plane pointing in the direction of $\mathbf{n}$ starting at the point $v_A + \frac{1}{2}w$, where $\mathbf{n}$ is the outward normal of $VO^{\tau}_{A|B}$ at $v_A - v_B + w$ [14]:

$$ORCA^{\tau}_{A|B} = \{v|(v-(v_A+\frac{1}{2}\mathbf{w})).\mathbf{n} \geq 0\}. \quad (8)$$

In our methodology, the finite state machine described in the previous section is used to calculate the preferred velocity. This velocity is passed along to the classic ORCA algorithm that sets the new velocity to be applied to the agent. In other words, the velocity computed by equation 5 is fed to ORCA as the *preferred velocity*, which is a value used as a reference to determine the actual velocity that will be assigned to the robot during navigation. Consequently, the value belonging to that half-plane that most closely matches the preferred velocity is calculated using linear programming and passed as the new velocity that the robot should take.

4 Experiments

To evaluate the proposed methodology with regards to its performance and feasibility, we executed a series of simulations and compared it with the classical versions of the ORCA and VGVO algorithms. We also performed proof-of-concept experiments with a group of e-puck robots to show its applicability with real robots.

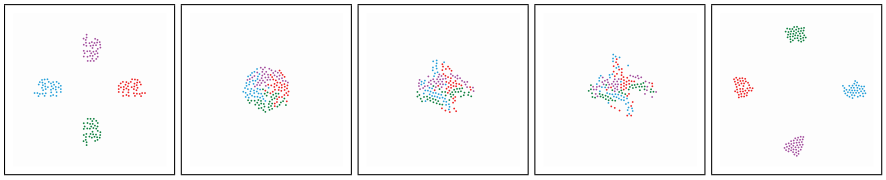
The parameters used in these experiments are shown in Table 1. Our main objective when selecting the values was to keep the robots segregated, even if this requires spending more time for the groups to achieve their goals.

Agent Status	k_c	k_s	k_a	α	β	γ
Single Group	10	0	10	10	1	0
Vision Free	15	5	5	3	1	0
Follower	15	10	0	0	20	50
Turn Right	5	2.5	1	0	20	30

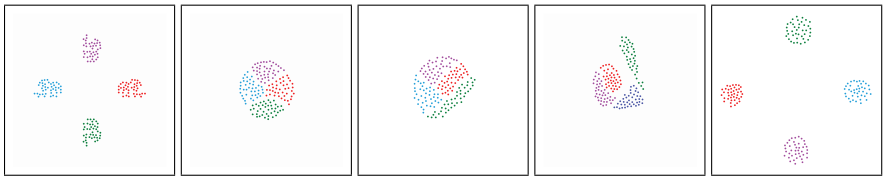
Table 1: Parameters used in the experiments.

In Figure 2, we show snapshots of the execution of the three algorithms. In this first example, we have 4 distinct groups, represented by different colors, each one formed by 40 agents. Each group must navigate to its opposite direction, changing sides with another group. As can be seen in Figure 2a, as expected, ORCA does not keep robots segregated when groups meet in the center of the scenario. Figures 2b and 2c show, respectively, the results obtained by the VGVO algorithm and the proposed methodology for the same scenario. As shown, with both algorithms, robots

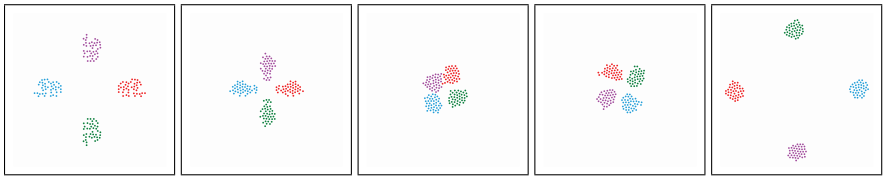
belonging to the same group remain united during the entire navigation, while seeking to move away from other groups.



(a) Execution of the ORCA algorithm.



(b) Execution of the VGVO algorithm.



(c) Execution of the proposed algorithm.

Fig. 2: Execution of the evaluated algorithms in a scenario composed of 160 agents evenly distributed into 4 groups.

To better evaluate the segregation, we use a metric proposed in [7], which consists of calculating the average distances between agents of the same group and agents from different groups. According to this metric, two different groups of agents, e.g. A and B, are said to be segregated if the average distance between the agents of the alike types (type A or type B) is less than the average distance between the agents of the unlike types (i.e., between the agents of type A and type B). More formally, we should have:

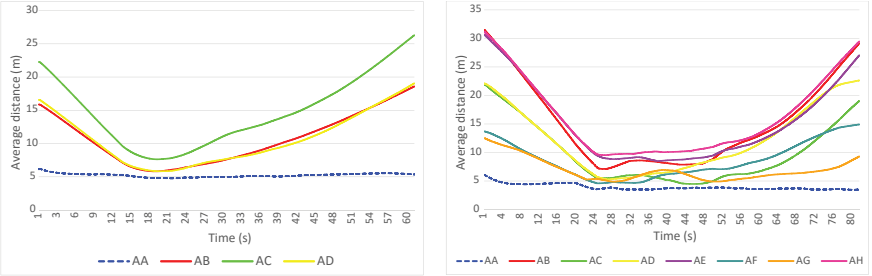
$$d_{AA} < d_{AB} \quad \text{and} \quad d_{BB} < d_{AB}, \tag{9}$$

where d_{XY} is the average distance between the agents of types X and Y. Formally:

$$d_{XY} = \frac{1}{|\Gamma_X|} \sum_{i \in \Gamma_X} \left(\frac{1}{|\Gamma_Y|} \sum_{j \in \Gamma_Y} (p_i - p_j) \right). \tag{10}$$

The segregative behavior of the methodology may be confirmed by using the aforementioned metric pairwise. As can be seen in Figure 3, the average distance

between agents in the group A are smaller than the average distance between agents from group A relative to agents of all other groups, i.e., $d_{XX} < d_{XY} \; \forall \, \Gamma_X, \Gamma_Y \in \mathscr{R}$.



(a) 4 distinct groups. (b) 8 distinct groups.

Fig. 3: Average distance among agents during the execution of the proposed algorithm. Two scenarios were considered: (a) 4 distinct groups and (b) 8 distinct groups.

In the following experiments, we varied the number of robots and groups. Initially, the algorithm was evaluated considering a fixed number of groups and an increasing number of agents in each group. Next, we have fixed the number of agents and varied the quantity of distinct groups in the environment. Finally, we have evaluated the methodology in a scenario where each group has a different number of agents.

In the first experiment we investigated the performance of the approach relative to the variation in the number of agents on each group. The simulations were performed with 4 groups consisting of 10, 20, 30 and 40 agents. We emphasize that on these first set of simulations all the groups have the same number of agents.

The methodology was evaluated considering the average time the groups take to converge to their goals and the results were compared to the ones obtained by the ORCA and the VGVO algorithms. We recall that ORCA does not sort (segregate) agents into distinct groups, i.e., each agent is considered as a single entity and no group information is used during its execution. The main objective of this evaluation is to verify if the restriction to keep the groups segregated would affect the navigation time and if our approach is able to increase the performance of the VGVO algorithm.

Since the proposed technique (as well as the VGVO algorithm) may produce different results during executions (due to a non-deterministic factor on the velocity selection), we performed 100 simulations for each experiment and the average time was used for comparison. Figure 4 presents the results for each technique for this scenario.

As might be expected, the time required for all the agents to reach their goals increases with the number of agents. Also, we can observe that the proposed algorithm has a much better performance than ORCA and VGVO when the number of

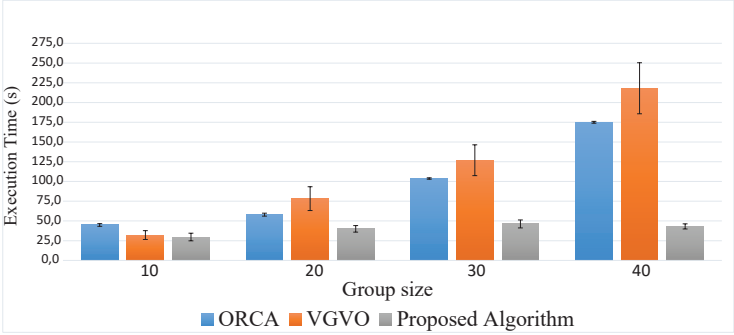


Fig. 4: Average execution time of the algorithms varying group size.

robots increase. This may be explained by the fact that in ORCA, robots have to deal with a very congested situation when all groups meet, which makes it difficult for the algorithm to find feasible velocities. Regarding VGVO, as explained in [10], the algorithm prioritizes slower speeds in order to maintain stable relative distances and velocities among robots. We can also note that, due to its characteristics, VGVO has a large variation in its running time within the 100 runs, while in ORCA and our algorithm the standard deviation is relatively small.

Next, the second experiment aims to evaluate our approach in scenarios with a fixed number of agents that were distributed in a different number of groups in each set of runs. The simulations consist of scenarios composed by a total of 160 agents evenly distributed into 2, 4, 6, and 8 groups. As shown in Figure 5, the increase in the number of groups causes the time required for the agents to reach their targets to also increase. However, we observe that the proposed approach improves considerably the average navigation time. Another point to note is the difference between the times obtained using VGVO algorithm and the proposed algorithm. The VGVO algorithm makes the groups to be positioned uniformly in the conflict area, causing the agents to take a long time to find a new path that would allow the group to contour the obstacles that hinder its movement. On the other hand, our approach allows the agents to start diversion maneuvers when a new group is detected, causing a reduction on the time needed to solve the problem.

Finally, the last experiment was performed in order to investigate the methodology behavior in scenarios with groups of different sizes. One of the experiments evaluated a scenario with 220 agents divided into 8 groups with sizes ranging from 10 to 40 agents. Figure 6 shows snapshots of the navigation over time.

It is possible to observe that the proposed algorithm has the same segregating behavior when used in environments having different group sizes. We call attention to the trajectory performed by the pink group during this simulation. As shown in Figures 6(c), (d) and (e), despite the definition of an explicit rule that requires the agents to move to the right when another group is blocking its way, the pink group managed to find a better path than the path resulting from applying the rule. This

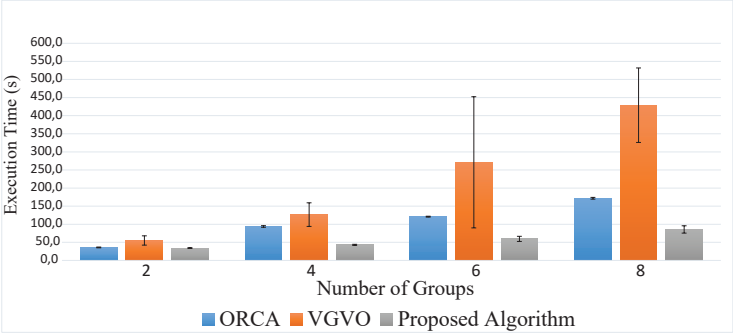


Fig. 5: Average execution time of the algorithm with different number of groups.

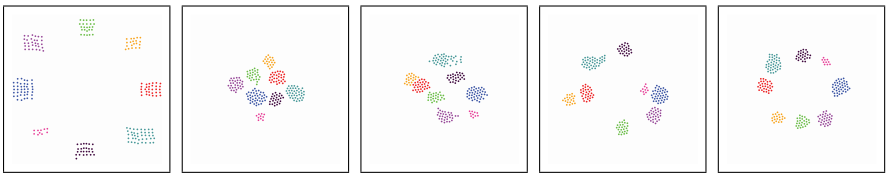


Fig. 6: Execution of the proposed algorithm in a scenario with 220 agents distributed in 8 groups of different sizes.

occurs because at least one of the agents on the pink group could directly see the goal and, consequently, the other agents began to follow it.

Finally, real experiments were conducted indoors using six e-puck robots. These proof-of-concept experiments are important in order to show the feasibility of the algorithm in real scenarios, where uncertainties caused by sensing and actuation errors may have a great impact on the results.

In these experiments, we used a swarm localization framework based on an overhead camera and fiduciary markers for estimating robot's pose, orientation. Also, as the e-puck's IR sensors have a very small range, we implemented a virtual sensor based on the localization system to detect neighboring agents. To account for non-holonomic constraints, input velocities were transformed following the approach presented in [1].

The Figures 7 and 8 show snapshots from executions of the proposed algorithm with two and three groups of robots (we overlay colored circles to highlight the different groups). We can visually inspect that the behaviors obtained with the real robots is pretty similar to the simulation results, i.e., robots maintain cohesion and segregation during navigation. Despite not showing the graphs, we observed that average distances follow the trend shown in Figure 3: the average distance between robots in the same group is always less than the average distance among robots in

different groups. These proof of concept experiments indicate that the algorithm can work well to coordinate groups of real robots, allowing them to navigate while maintaining a segregative behavior in an efficient way.

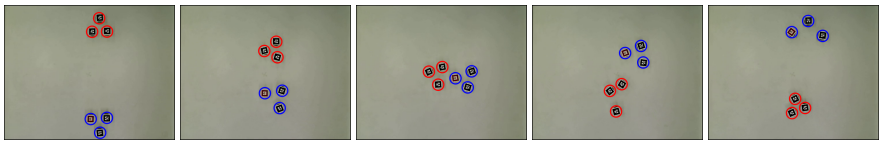


Fig. 7: Execution of the proposed algorithm in a scenario with 6 e-puck robots distributed in 2 groups.

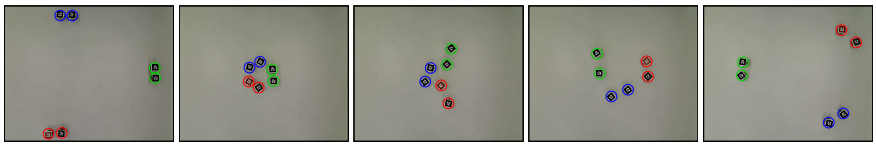


Fig. 8: Execution of the proposed algorithm in a scenario with 6 e-puck robots distributed in 3 groups.

5 Conclusion

In this paper we proposed a decentralized methodology to navigate heterogeneous groups of robots maintaining segregation among different groups. Our approach consists of extending the ORCA algorithm with a modified version of the classical flocking behaviors: using local sensors, each robot is able to sense its neighborhood and, using a variation of the flocking rules, compute the velocity that will guide it to the goal, while keeping it close to its group. This velocity is used as the preferred velocity by the ORCA algorithm, which is responsible for navigating the robot while avoiding other agents.

Several experiments were performed to evaluate the proposed methodology and the results showed that our algorithm is an effective alternative to maintain different robots groups segregated whilst they navigate on a shared environment. By choosing better trajectories and avoiding congested areas, it achieved a better performance when compared to other state-of-the-art algorithms.

As in ORCA and VGVO, in our methodology the robots must be able to infer the position and velocities of the neighboring robots. Moreover, robots should infer the state (considering the FSM) and orientation of their neighbors and also know

the position of a common goal for the group. This can be considered a limitation of the proposed methodology, mainly considering that swarms of robots are normally comprised of simple robots with limited sensing capabilities.

Future research directions include the evaluation of different segregation metrics, such as the intersection area of the convex hull formed by the agents and a possible network connectivity inside the group. We also intend to consider standard metrics of separation used in cluster analysis, for example, the distance between centroids of the groups weighted by their variance. More experiments should also be performed using a larger number of real robots to better analyze the behavior of the algorithm in real scenarios.

References

1. De Luca, A., Oriolo, G., Vendittelli, M.: Stabilization of the unicycle via dynamic feedback linearization. In: 6th IFAC Symp. on Robot Control, pp. 397–402 (2000)
2. Egerstedt, M., Hu, X.: Formation constrained multi-agent control. In: Robotics and Automation, 2001. ICRA'01. IEEE International Conference on, vol. 4, pp. 3961–3966 vol.4 (2001)
3. Ferreira Filho, E.B., Pimenta, L.C.A.: Segregating Multiple Groups of Heterogeneous Units in Robot Swarms using Abstractions. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 401–406 (2015)
4. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research **17**(7), 760–772 (1998)
5. Groß, R., Magnenat, S., Mondada, F.: Segregation in swarms of mobile robots based on the brazil nut effect. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 4349–4356 (2009)
6. He, L., van den Berg, J.: Meso-scale planning for multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2013)
7. Kumar, M., Garg, D.P., Kumar, V.: Segregation of heterogeneous units in a swarm of robotic agents. Automatic Control, IEEE Transactions on **55**(3), 743–748 (2010)
8. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: ACM Siggraph Computer Graphics, vol. 21, pp. 25–34. ACM (1987)
9. Şahin, E.: Swarm robotics: From sources of inspiration to domains of application. In: Swarm robotics, pp. 10–20. Springer (2005)
10. Santos, V.G., Campos, M.F., Chaimowicz, L.: On segregative behaviors using flocking and velocity obstacles. In: Distributed Autonomous Robotic Systems, pp. 121–133. Springer (2014)
11. Santos, V.G., Pimenta, L.C., Chaimowicz, L., et al.: Segregation of multiple heterogeneous units in a robotic swarm. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 1112–1117. IEEE (2014)
12. Snape, J., Guy, S.J., Vembar, D., Lake, A., Lin, M.C., Manocha11, D.: Reciprocal collision avoidance and navigation for video games. In: Game Developers Conf., San Francisco (2012)
13. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Optimal reciprocal collision avoidance for multi-agent navigation. In: Proc. of the IEEE International Conference on Robotics and Automation, Anchorage (AK), USA (2010)
14. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)
15. Wilkie, D., Van den Berg, J., Manocha, D.: Generalized velocity obstacles. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 5573–5578. IEEE (2009)