NAVEGAÇÃO SEGREGADA EM ENXAMES DE ROBÔS

FABRÍCIO RODRIGUES INÁCIO

NAVEGAÇÃO SEGREGADA EM ENXAMES DE ROBÔS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Luiz Chaimowicz.
Coorientador: Douglas Guimarães Macharet.

Belo Horizonte
Setembro de 2016

Inácio, Fabrício Rodrigues

I35n Navegação Segregada em Enxames de Robôs / Fabrício Rodrigues Inácio. — Belo Horizonte, 2016.
xxvi, 64 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação

Orientador: Luiz Chaimowicz.
Coorientador: Douglas Guimarães Macharet.

1. Computação - Teses. 2. Robótica. 3. Optimal Reciprocal Collision Avoidance. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*82.9(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Navegação segregada em enxames de robôs

FABRÍCIO RODRIGUES INÁCIO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. LUIZ CHAIMOWICZ - Orientador
Departamento de Ciência da Computação - UFMG

PROF. DOUGLAS GUIMARÃES MACHARET - Coorientador
Departamento de Ciência da Computação - UFMG

PROFA. GINA MAIRA BARBOSA DE OLIVEIRA
Faculdade de Ciência da Computação - UFU

PROF. LUCIANO CUNHA DE ARAÚJO PIMENTA
Departamento de Engenharia Eletrônica - UFMG

Belo Horizonte, 30 de setembro de 2016.

Aos meus inestimáveis pais,

Agradecimentos

Deixo aqui o meu eterno agradecimento aos meus pais, que sempre me ofereceram apoio incondicional, em suas mais variadas formas e sem as quais tudo isso seria impossível. Agradeço pelo incentivo e esforço, pelos exemplos, pelo carinho, pela dedicação e preocupação que possibilitaram a mim, chegar tão longe. Se hoje tenho minhas mãos macias e a mente povoada de conhecimento, tudo devo às mãos calejadas desses dois. Mesmo que eu dedicasse todos os dias da minha vida para agradecê-los, pouco eu teria feito diante de tudo que já passaram e fizeram para me ver chegar aqui. Dedico meu amor e gratidão a vocês que são meus exemplos de vida.

Agradeço à minha irmã, Jaqueline Rodrigues, por todos os anos de dedicação, companheirismo e amizade incondicional.

Aos meus eternos amigos, Hugo Rainner, Murillo Germano e Dênis Elias, que me presentearam com a mais pura e sincera amizade que já experimentei.

À Lorena, pela dedicação e preocupação incontestáveis e pela paciência em dividir minha atenção com este trabalho.

Aos avós, tios e primos que sempre me viram como o nerd estudioso da família.

A todos os professores que já passaram pela minha vida, em especial aos meus orientadores Luiz Chaimowicz e Douglas Macharet e aos amigos mestres Rosilane Mota e Marcelo Nery. Com o conhecimento que todos vocês se dispuseram a compartilhar, pude construir o que sou hoje.

Aos amigos do vôlei, pelos muitos anos de companheirismo e por sempre me proporcionarem momentos únicos de descontração. Por serem a "desculpa perfeita" para eu sair um pouco do meio dos cálculos e algoritmos e ir me divertir com vocês.

Aos amigos da Oncomed BH, por sempre me incentivarem e dedicarem palavras tão carinhosas sobre minha paixão pelos estudos. Em especial, à melhor equipe de TI já vista em BH, à Carla e à Cibele por me proporcionarem a oportunidade única que permitiu minha aprovação no mestrado.

Aos amigos que fiz na UFMG, Raquel Aoki, Vaux Gomes, Rafael Lima, Victor Nascimento, Adriano Lages, Lucas Rosas, Paulo Bicalho, Thássia Almeida e Jean Freire

pelo apoio e ideias sobre os trabalhos e a dissertação e até mesmo pelas conversas sobre os assuntos mais banais que tornaram a caminhada mais fácil e agradável.

Ao Vinícius, por ceder o código do algoritmo VGVO e ao Rezeck, por tanto me ajudar com os testes reais. Sem a contribuição de vocês, meu esforço seria infinitamente maior.

O momento que vivo agora é fascinante e só existe porque vocês se doaram em silêncio e aceitaram viver comigo o meu sonho. A vocês ofereço a minha vitória e, de coração, o meu muito obrigado.

"É muito melhor arriscar coisas grandiosas,
alcançar triunfos e glórias, mesmo expondo-se a derrota,
do que formar fila com os pobres de espírito que nem gozam muito nem sofrem muito,
porque vivem nessa penumbra cinzenta que não conhece vitória nem derrota."
(Theodore Roosevelt)

Resumo

Um enxame robótico é um tipo particular de sistema multi-robô que emprega um grande número de agentes simples, a fim de executar cooperativamente diferentes tipos de tarefas.

Neste contexto, um tópico que tem recebido muita atenção nos últimos anos é o conceito de segregação, que existe em vários sistemas biológicos, e pode ser útil em muitas tarefas e cenários diferentes. Este conceito é importante, por exemplo, em tarefas que requerem manutenção de robôs com características ou objetivos semelhantes, organizados em grupos coesos, enquanto robôs com características diferentes permanecem separados em seus próprios grupos.

Neste trabalho, propomos uma metodologia descentralizada para navegar grupos heterogêneos de robôs, mantendo a segregação entre os diferentes grupos. Nossa abordagem consiste em estender o algoritmo ORCA (*Optimal Reciprocal Collision Avoidance*) com uma versão modificada dos comportamentos clássicos de flocking. Numerosos experimentos foram realizados em ambientes simulados e forneceram informações estatísticas sobre o desempenho da técnica proposta. Os resultados mostram que os grupos permaneceram coesos durante a navegação em todos os cenários avaliados. Além disso, a metodologia permitiu uma convergência mais rápida do grupo para seu objetivo quando em comparação com os algoritmos avaliados.

Palavras-chave: Enxames de robôs, navegação, segregação, ORCA, *flocking*.

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, which exists in several biological systems, and can be useful in many different tasks and scenarios. 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 separate 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 (Optimal Reciprocal Collision Avoidance) algorithm with a modified version of the classical flocking behaviors. Numerous trials that were carried out in simulated environments provide statistical insight on the performance of the proposed technique. The results show that the groups remained cohesive during navigation in all evaluated scenarios. Furthermore, the methodology allowed for a faster convergence of the group to the goal when compared to the evaluated algorithms.

Keywords: Robotic swarms, navigation, segregation, ORCA, flocking.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathcal{R}_i$	agente i.
$\boldsymbol{p}_i$	posição $\langle x, y \rangle$ do agente i.
θ_i	ângulo formado por um vetor unitário que aponta na direção de movimento do agente i no espaço de configuração global.
q_i	pose $\langle x, y, \theta \rangle$ do agente i.
$\boldsymbol{v}_i$	velocidade do agente i.
$\mathcal{N}_i$	conjunto formado por todos os agentes situados na área delimitada pelo raio de visão do agente i.
$\mathcal{N}_i^+$	conjunto formado pelos agentes situados na área delimitada pelo raio de visão do agente i e que pertencem ao mesmo grupo que ele.
$\mathcal{N}_i^-$	conjunto formado pelos agentes situados na área delimitada pelo raio de visão do agente i e que pertencem a grupos diferentes do grupo do agente i.
λ	raio da vizinhança do agente i.
CC_B^A	conjunto de todas as velocidades relativas entre os agentes A e B que levam a uma colisão entre eles.
$VO_{A\|B}^{\tau}$	conjunto de velocidades absolutas do agente A que levam a uma colisão com o agente B dentro de um intervalo de tempo τ.
$CA_{A\|B}^{\tau}$	conjunto de velocidades anti-colisão do agente A em relação ao agente B dentro de um intervalo de tempo τ.
$ORCA_{A\|B}^{\tau}$	conjunto de todas as velocidades do agente A que garantem uma navegação livre de colisão com o agente B dentro de um intervalo de tempo τ.
$\boldsymbol{v}_{\text{coesão}}$	força potencial artificial que mantém o agente próximo dos outros agentes do seu próprio grupo.
$\boldsymbol{v}_{\text{separação}}$	força potencial artificial que mantém o agente afastado dos outros agentes presentes em sua vizinhança.
$\boldsymbol{v}_{\text{alinhamento}}$	força potencial artificial que mantém o agente na mesma direção que os outros agentes do seu próprio grupo.
$\boldsymbol{v}_{\text{objetivo}}$	força potencial artificial que atrai o agente para seu objetivo.
$\boldsymbol{v}_{\text{flock}}$	força potencial artificial que tenta manter o grupo de agentes unidos e se deslocando na mesma direção.

$\boldsymbol{v}_{\text{aux}}$	força potencial artificial que faz o agente seguir outro agente de seu grupo ou se deslocar para a direita, dependendo do estado que o agente esteja.
$\boldsymbol{v}_i^{\text{opt}}$	velocidade de otimização do agente i.
$\boldsymbol{v}_{\text{pref}}$	velocidade preferencial do agente i.

Sumário

Capítulo 1

Introdução

Durante as últimas décadas, a Robótica experimentou um grande avanço tecnológico e vem se tornando uma alternativa viável para a execução de tarefas nas quais um homem teria um desempenho inferior a um robô, como por exemplo em linhas de montagem industriais, ou ainda, em situações que ofereçam riscos para seres humanos [Dias et al., 2006]. Devido a esse avanço e à redução dos custos de se desenvolver sistemas robóticos, várias abordagens começaram a ser desenvolvidas e uma área que tem recebido bastante atenção são os sistemas multi-robôs, em especial os chamados enxames de robôs, também conhecidos como *swarms*.

Os enxames são sistemas formados por uma grande quantidade de robôs que apresentam um comportamento coletivo que emerge das interações entre seus agentes e o ambiente no qual estão inseridos [Şahin, 2004]. Essa abordagem tem inspiração nas colônias de insetos sociais, como abelhas e formigas, e em outras comunidades de animais nas quais um grande número de indivíduos cooperam entre si para realizar determinadas tarefas de interesse comum, que dificilmente poderiam ser executadas por indivíduos isolados [Traniello & Rosengaus, 1997; Parker, 2000]. Devido ao grande número de agentes existentes nos enxames e ao seu custo de produção, os robôs que formam o enxame possuem baixo poder de processamento e capacidades de comunicação e percepção limitadas, ou seja, são mais simples que os utilizados em aplicações que empregam um único robô ou um pequeno grupo deles [Mohan & Ponnambalam, 2009].

Assim como acontece nos sistemas biológicos, normalmente não existe uma entidade responsável por coordenar o enxame de robôs de forma centralizada. Os comportamentos isolados de cada indivíduo levam a um comportamento emergente que busca alcançar o objetivo do enxame [Cao et al., 1997; Mohan & Ponnambalam, 2009]. Um exemplo desse comportamento emergente em sistemas biológicos é o transporte de

alimentos por formigas. Nessa situação, as formigas precisam cooperar entre si para exercer uma força muito maior do que a que uma única formiga é capaz de exercer para carregar o alimento. Para cumprir o objetivo comum, cada indivíduo exerce uma força em determinada direção, essas ações individuais são observadas pelas demais formigas do grupo e, de forma descentralizada, produzem um comportamento emergente que faz com que o alimento seja levado para a colônia [Şahin, 2004].

Como destacado por Şahin [2004]; Bahgeci & Sahin [2005] e Dorigo et al. [2013], a utilização de enxames de robôs confere ao sistema três importantes características, sendo:

- Robustez: esta primeira característica está relacionada à tolerância a falhas, ou seja, o sistema é capaz de continuar a execução de suas tarefas mesmo quando ocorrem falhas individuais. Essa característica pode ser relacionada a uma série de fatores. Primeiramente, com a redundância presente nos enxames, caso um robô específico deixe de funcionar, como o número de agentes no sistema é grande, a execução das tarefas designadas ao enxame não é prejudicada por essa falha individual. O segundo fator relacionado à robustez é o fato de, geralmente, não existir uma entidade que controle o enxame de forma centralizada. Caso essa entidade exista, qualquer falha relacionada a ela afeta todo o enxame, o que pode fazer com que os agentes deixem de executar suas tarefas de forma adequada. O terceiro fator é a simplicidade dos agentes que compõem o enxame: como o conjunto de ações que eles são capazes de executar é limitado e formado por ações relativamente simples, é menos provável que um agente execute suas ações de forma incorreta. O último fator diz respeito à percepção dos agentes, com o emprego de múltiplos agentes e mesmo com a percepção limitada que cada um deles possui, é possível que, a partir da cooperação entre os agentes, a informação captada pelo enxame seja mais completa e fiel ao ambiente no qual o enxame opera.

- Flexibilidade: o sistema é capaz de gerar soluções para diferentes tipos de tarefas. Assim como acontece nas colônias de insetos sociais, os enxames podem ser compostos por indivíduos que exerçam tarefas específicas, ou seja, podem existir indivíduos responsáveis pela captação de informação, outros responsáveis pelo transporte de objetos e assim por diante. Diante dessa possibilidade de execução de múltiplas tarefas, os agentes devem ser capazes de se organizar para realizar as tarefas atribuídas ao enxame de forma cooperativa, cada um executando as ações que lhes são permitidas.

- Escalabilidade: essa característica está diretamente relacionada à capacidade do enxame se adaptar às variações no seu tamanho. Isso quer dizer que os algoritmos responsáveis pela coordenação do enxame devem garantir o funcionamento do sistema, sendo perturbado o mínimo possível pela variação no número de agentes que o compõem.

Diante da vasta gama de aplicações que podem utilizar os enxames de robôs, Cao et al. [1997] e Şahin [2004] classificaram algumas delas em grupos que representam o objetivo de mais alto nível atribuído ao enxame:

- Agregação: os agentes iniciam a execução da tarefa em posições arbitrárias e precisam se organizar/reunir em determinada posição;

- Automontagem: os robôs devem se encaixar uns aos outros de maneira coordenada, formando entidades robóticas fisicamente conectadas;

- Coleta ou *Foraging*: os agentes devem se deslocar no ambiente buscando objetos específicos, coletando-os e os levando a alguma posição específica;

- Controle de Tráfego: os agentes devem se organizar com o intuito de evitar que um grande número deles se desloque para o mesmo local, evitando um congestionamento de robôs e problemas de navegação pela influência de um robô no caminho calculado pelos outros;

- Construção Auto-organizada: os robôs precisam cooperar para transportar objetos e montar estruturas específicas com os objetos transportados.

- Construção de Padrões: os robôs devem se distribuir com o objetivo de formar padrões geométricos específicos;

- Dispersão: agentes que inicialmente se encontram próximos uns dos outros devem se posicionar de tal forma que a distância entre eles seja maximizada;

- Movimento Conectado: quando existe uma conexão física entre os agentes, eles precisam planejar movimentos que respeitem as restrições impostas pelas conexões existentes entre eles;

- Transporte Coordenado: agentes que necessitam transportar grandes objetos precisam agir de forma coordenada.

Além desses grupos, pode-se trabalhar com a ideia de segregação de grupos. Nesse tipo de tarefa, o objetivo é fazer com que grupos heterogêneos de agentes se dissociem uns dos outros, como mostrado na Figura 1.1. Esse objetivo pode ser importante para manter robôs com características ou objetivos semelhantes organizados em grupos coesos, enquanto grupos com características ou objetivos diferentes se mantêm separados uns dos outros [Santos et al., 2014b].

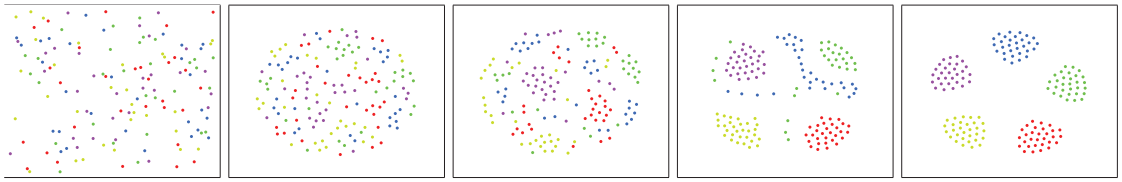


Figura 1.1: Segregação de cinco grupos heterogêneos de robôs [Santos et al., 2014b].

Diante do que foi dito, fica clara a necessidade de se desenvolver algoritmos que consigam controlar um enxame na realização de diferentes tipos de tarefas.

1.1 Motivação

Se locomover com segurança pelo ambiente é um princípio básico da Robótica Móvel. Para atender a esse requisito, vários pesquisadores propuseram algoritmos específicos para navegação com desvio de colisões [Fiorini & Shiller, 1993; Guo & Parker, 2002; Masehian & Katebi, 2007; Gal et al., 2009; van den Berg et al., 2011]. Com o uso de enxames e o compartilhamento do ambiente, é possível que as trajetórias que devem ser seguidas por cada robô se interceptem em algum momento. O aumento no número de agentes no sistema faz com que o problema fique ainda mais complicado devido à maior probabilidade de um agente entrar em rota de colisão com os outros. Por isso, é necessário desenvolver estratégias que evitem que um robô interfira de forma negativa no comportamento dos demais robôs do enxame. Além disso, é necessário desenvolver meios para que os robôs possam navegar em segurança pelo ambiente.

Devido à grande quantidade de agentes presentes no ambiente e à influência que cada um deles pode exercer sobre os outros, sendo considerados como obstáculos móveis, as trajetórias calculadas pelo algoritmo responsável pela navegação dos agentes podem ser desnecessariamente longas, o que leva a um gasto excessivo de tempo e bateria para que os agentes alcancem seus objetivos. Com a aplicação de enxames em tarefas cada vez mais complexas, nem sempre é possível implementar todas as habilidades de sensoriamento e atuação em um único tipo de agente. Devido a essa restrição,

pesquisadores e organizações vêm trabalhando em aplicações que empregam robôs que possuam características ou até mesmo objetivos distintos [Dorigo et al., 2004, 2013].

Nos últimos anos, novas aplicações práticas para os enxames vêm surgindo. Vassev et al. [2012] descreve o *ANTS* (*Autonomous NanoTechnology Swarm*), um novo conceito que propõe o uso de milhares de pequenas naves espaciais (de aproximadamente 1 kg) para exploração de asteroides. De acordo com o projeto, essas naves funcionariam como uma rede de sensores, fazendo observações dos asteroides e analisando suas composições. Em Jaimes et al. [2008] é descrito um modelo composto por aviões autônomos de asa fixa e enxames semi-autônomos de quadri-rotores que trabalham em conjunto para fazer a vigilância de uma determinada área. Outras aplicações para os enxames envolvem ainda mineração de dados [Martens et al., 2011], uso de nanorrobôs para o tratamento de tumores [Lewis & Bekey, 1992; Lenaghan et al., 2013], controle de multidões em simulações de computação gráfica [Jackson, 2003; ping Chen & yin Lin, 2009; Forster, 2013], dentre várias outras.

Alguns tipos específicos de tarefas podem levar a uma divisão do enxame em grupos distintos de agentes. Uma situação onde isso pode ser interessante é a coleta de recursos pelos agentes. Nessa tarefa, o enxame pode ser formado por agentes de tipos diferentes, sendo cada tipo especializado na coleta de um recurso específico. Para que o enxame possa solucionar o problema para o qual foi designado, pode-se separar cada tipo de agente em um grupo específico e enviar os grupos para regiões onde os recursos mais adequados para cada um deles sejam mais abundantes. Quando existem grupos distintos de robôs que não devem se misturar durante a execução de suas tarefas, além de garantir uma navegação segura para cada um dos agentes, ainda é preciso adicionar técnicas que permitam que os robôs executem suas tarefas sem ter sua eficiência degradada por essa restrição. Nesses cenários, os robôs pertencentes a um mesmo grupo devem se manter próximos uns dos outros, o que aumenta a chance de colisão entre eles. Além disso, as manobras de desvio executadas pelos agentes devem respeitar à restrição da segregação dos grupos. Apesar dos desafios que surgem com a utilização de enxames que devam manter a segregação de grupos durante a navegação, esse tipo de abordagem pode se mostrar como uma boa alternativa para esse problema, pois as trajetórias executadas pelos grupos podem ser mais eficientes do que as observadas quando cada agente navega sem levar em conta os agentes do seu próprio tipo.

Diversos trabalhos já foram realizados para solucionar, de forma isolada, os problemas citados acima Kumar et al. [2010]; Chen et al. [2012]; Santos et al. [2014b]; Ferreira Filho & Pimenta [2015]. Porém, são escassas as abordagens que buscam soluções em cenários nos quais seja preciso tratar as colisões entre os agentes mantendo a segregação dos grupos existentes no enxame durante sua navegação. Um trabalho

desenvolvido com esse propósito é apresentado em Santos et al. [2014a]. A abordagem proposta pelos autores utiliza a ideia de grupos virtuais e os conceitos de *Velocity Obstacles* [Fiorini & Shiller, 1998] para manter a segregação de grupos distintos de agentes durante sua navegação. Porém, o algoritmo proposto por esse trabalho encontra dificuldades em manter um bom desempenho enquanto o número de agentes compartilhando o mesmo ambiente de trabalho aumenta.

1.2 Definição do Problema

Durante os últimos anos, diversos algoritmos foram propostos com o intuito de evitar colisões em sistemas formados por uma grande quantidade de agentes. Trabalhos como os apresentados em Fiorini & Shiller [1998], van den Berg et al. [2008] e Wilkie et al. [2009] utilizam a técnica conhecida como *Velocity Obstacles* (VO), ou alguma variação dela, para fazer com que os robôs sejam capazes de desviar de obstáculos durante a execução das tarefas as quais foram designados.

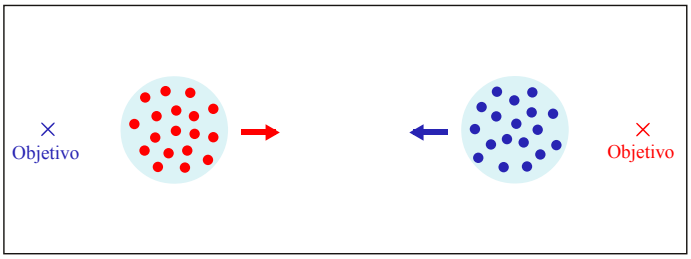
Durante a resolução de determinados problemas, podemos trabalhar com a ideia de navegação segregada. Nesse tipo de tarefa, o objetivo é manter grupos de agentes que possuam características ou objetivos distintos separados uns dos outros enquanto navegam pelo ambiente [Santos et al., 2014a; Ferreira Filho & Pimenta, 2015], como mostrado na Figura 1.2.

Dito isso, o problema de navegação segregada pode ser definido como:

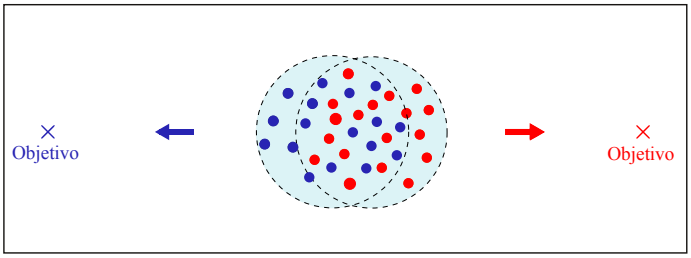
Dado um enxame de robôs composto por n grupos heterogêneos que já encontram-se segregados e que devem se deslocar para uma região específica do seu ambiente de trabalho, deve-se navegar cada grupo em direção a seu respectivo objetivo mantendo a segregação entre cada um deles.

1.3 Objetivos e Contribuições

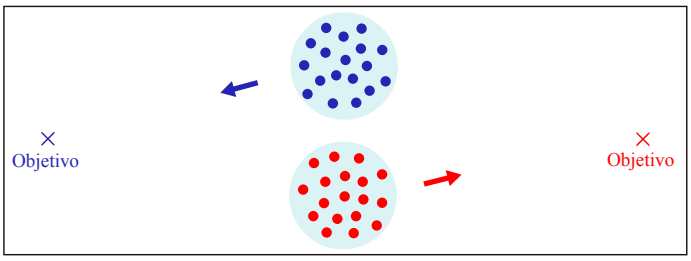
O desenvolvimento de algoritmos eficientes que consigam manter vários grupos de robôs separados durante a execução das tarefas se mostra como um importante campo de pesquisa. Por isso, o intuito desse trabalho é desenvolver um algoritmo capaz de manter a segregação de grupos heterogêneos de robôs em cenários nos quais existam restrições que limitem, por exemplo, a comunicação entre os agentes ou a área de percepção de cada um deles, reduzindo o tempo necessário para que cada robô alcance seu objetivo.



(a) Cenário no qual dois grupos de agentes possuem objetivos opostos.



(b) Navegação sem controle de segregação.



(c) Navegação com controle de segregação.

Figura 1.2: Navegação de grupos de agentes com objetivos distintos.

Para atingir esse objetivo, uma variação do algoritmo VO – *Velocity Obstacles*, conhecida como ORCA (*Optimal Reciprocal Velocity Obstacles*), foi utilizada para evitar colisões entre os agentes e uma abordagem baseada no conceito de *flocking*, proposto por Reynolds [1987], foi desenvolvida para manter os grupos de robôs segregados. O algoritmo proposto, chamado de *FL-ORCA–Flocked Optimal Reciprocal Velocity Obstacle*, utiliza as regras definidas por Reynolds [1987] e uma máquina de estados finitos para calcular uma velocidade que permita que o agente permaneça próximo dos demais agentes do seu grupo ao mesmo tempo em que faz com que ele mantenha-se afastado de grupos diferentes do seu. Após o cálculo dessa velocidade, ela é passada como velocidade preferencial para o algoritmo *Optimal Reciprocal Collision Avoidance* (ORCA), responsável pela navegação e pelo controle de colisão, para que a nova velocidade que o agente deve desenvolver seja definida.

Para avaliar o algoritmo proposto, foram realizados testes em cenários com quan-

tidades variadas de agentes e grupos. A partir da análise desses testes, avaliou-se o desempenho do algoritmo em relação ao tamanho do enxame e do número de grupos que o compõe. Também foram realizados testes para comparar o algoritmo proposto com o algoritmo *Virtual Group Velocity Obstacles* (VGVO), proposto por Santos et al. [2014a], e com técnicas que façam apenas a navegação com controle de colisões, sem se preocupar com a restrição de segregação dos grupos. Para possibilitar uma análise quantitativa dos algoritmos utilizados nos experimentos, definiu-se uma nova métrica que avalia a navegação segregada de grupos distintos de agentes a partir da distância entre agentes de um mesmo grupo. Além dos experimentos simulados, também foram realizados experimentos com robôs reais como prova de conceito para investigar o comportamento da abordagem proposta quando submetida às restrições e incertezas que o uso de agentes reais traz ao problema e demonstrar a sua viabilidade de uso.

Após a execução dos experimentos, foi possível concluir que o algoritmo proposto é uma abordagem viável para as aplicações nas quais a segregação dos grupos deva ser preservada. O algoritmo apresentou resultados melhores que os outros dois algoritmos avaliados na maioria dos cenários utilizados, sendo superado, em relação ao tempo de execução, somente em um cenário pelo algoritmo ORCA. Vale ressaltar que o ORCA não garante a segregação dos agentes, o que mostra que a abordagem desenvolvida não causou a degradação da qualidade das trajetórias a partir da imposição da segregação de grupos.

Dentre as principais contribuições alcançadas com este trabalho, podemos destacar o desenvolvimento de uma abordagem eficiente para a manutenção de segregação de grupos distintos de agentes com uma navegação livre de colisões e a criação de uma nova métrica para avaliar o nível de segregação de grupos distintos de agentes.

Durante o desenvolvimento dessa dissertação, o artigo *United we move: Decentralized segregated robotic swarm navigation* [Inácio et al., 2016] foi aceito para publicação no *13th International Symposium on Distributed Autonomous Robotic Systems*.

1.4 Organização

Este documento está organizado da seguinte forma: o Capítulo 2 apresenta trabalhos relacionados, apresentando de forma geral a área de enxames de robôs e citando trabalhos que lidam com a navegação de robôs móveis, o controle de colisões entre agentes e com a segregação de grupos em enxames. O Capítulo 3 descreve o algoritmo proposto, apresentando as premissas que foram consideradas e detalhando o funcionamento do algoritmo. O Capítulo 4 apresenta os cenários e os resultados obtidos da experimen-

tação do algoritmo proposto em simulação e em ambiente real. Por fim, o Capítulo 5 mostra as conclusões do trabalho realizado e possíveis trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

2.1 Sistemas Multi-robô

Um robô autônomo pode ser definido como um dispositivo capaz de realizar, sem ajuda humana, um determinado conjunto de ações com o intuito de alcançar os objetivos a ele atribuídos. Normalmente, os robôs são projetados para realizar um conjunto limitado de ações, e precisam definir uma sequência finita dessas ações para atingir seus objetivos. Buscando aumentar a eficiência e aplicabilidade dos robôs para solucionar problemas complexos, vem se tornando cada vez mais frequente a utilização de sistemas formados por uma grande quantidade de robôs [Cao et al., 1997; Şahin, 2004; Dias et al., 2006].

O uso de múltiplos robôs confere ao sistema algumas características que podem melhorar sua eficiência e aplicabilidade. Com o aumento do número de agentes, a divisão do trabalho, normalmente, aumenta a eficiência do sistema. Além disso, com o uso de vários robôs é possível solucionar uma gama de tarefas dificilmente resolvidas por um único agente [Dorigo et al., 2004]. Porém, quando se decide utilizar um enxame para a resolução de determinado problema, uma das primeiras decisões que deve ser tomada é se o enxame será homogêneo ou não [Barca & Sekercioglu, 2013]. Um enxame homogêneo é formado somente por robôs de um mesmo tipo, como, por exemplo, os sistemas formados pelos robôs e-puck. Por outro lado, os enxames heterogêneos são formados por robôs de diferentes tipos, como enxames formados por quadri-rotores e robôs terrestres, por exemplo. Essa decisão deve ser baseada nos tipos de problemas que o enxame poderá encontrar em seu ambiente de trabalho.

2.2 Navegação e controle de colisões

Para solucionar qualquer tipo de problema que exija deslocamento entre sua posição inicial e seu objetivo, um robô móvel deve ser capaz de encontrar caminhos pelos quais possa se movimentar. Além de planejar um caminho pelo qual ele possa se deslocar, o tamanho do caminho calculado e o tempo necessário para percorrê-lo são fatores relevantes que devem ser levados em consideração durante a resolução do problema.

Para solucionar o problema do planejamento de caminhos, um robô móvel pode utilizar métodos que são classificados como planejadores deliberativos. Nesse tipo de abordagem são feitas duas considerações básicas:

- o agente tem conhecimento de todo seu ambiente de trabalho - a forma e a posição de todos os obstáculos são representadas através de um mapa.
- o ambiente no qual o agente se encontra é estático - nenhum obstáculo ou outro robô se move pelo ambiente.

O método dos grafos de visibilidade [Lozano-Pérez & Wesley, 1979] trata dos ambientes nos quais existam obstáculos poligonais e propõe que os vértices dos obstáculos sejam considerados como vértices de um grafo e que sejam criadas arestas entre vértices que pertençam à mesma borda de um obstáculo e também entre vértices que permitam a criação de uma aresta que não intercepte nenhum outro obstáculo. A partir desse grafo é possível calcular o menor caminho que permita que o robô se desloque da sua posição inicial até seu objetivo. Essa solução garante que sempre que existir um caminho que leve o robô até seu objetivo, ele será encontrado. Porém, existe a desvantagem de o agente se locomover muito próximo aos obstáculos do ambiente, além da quantidade de arestas do grafo ser grande. Outra solução de planejamento deliberativo são os diagramas de Voronoi generalizados Takahashi & Schilling [1989]. Nesse método, o plano bidimensional sobre o qual o robô se desloca é decomposto em células cujas bordas são definidas como o conjunto de pontos equidistantes aos obstáculos que estão posicionados ao seu redor, como mostrado na figura 2.1. Apesar de proporcionar trajetórias seguras para os agentes, o uso dos diagramas de Voronoi não garante que as trajetórias geradas sejam ótimas, possibilitando que os agentes executem trajetórias maiores que as necessárias para a realização de suas tarefas Takahashi & Schilling [1989].

Por considerarem o ambiente estático, as abordagens deliberativas não são capazes de se adaptar aos problemas dinâmicos que podem surgir durante a execução da trajetória. Outra limitação dessa abordagem é a necessidade de se conhecer todo o ambiente, o que limita o uso dos algoritmos desse tipo.

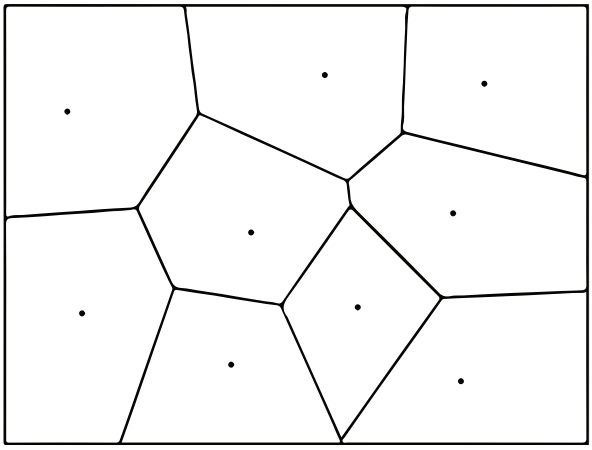


Figura 2.1: Diagrama de Voronoi de um ambiente no qual cada ponto representa um obstáculo.

Com o intuito de contornar os problemas acima mencionados, várias estratégias de navegação reativa também podem ser encontradas na literatura, tais como *Vector Field Histogram* [Borenstein & Koren, 1991] e *Dynamic Window Approach* [Fox et al., 1997]. Nesse tipo de abordagem, o agente navega pelo ambiente e corrige sua trajetória sempre que um obstáculo é percebido por ele. Com isso, os algoritmos reativos podem ser utilizados em cenários dinâmicos e não existe a necessidade de se ter um mapa que represente o ambiente. Outra técnica de navegação reativa são os chamados campos potenciais artificiais [Khatib, 1986]. Nesse método, o robô é considerado como uma partícula que sofre influência de campo potencial artificial que é construído de modo que o agente seja atraído pelo seu objetivo e repelido pelos obstáculos presentes no ambiente. Essa abordagem pode ser afetada pelo problema dos mínimos locais, nos casos nos quais a função utilizada para gerar o campo potencial possua mínimos locais, o agente pode se deslocar para um desses mínimos e ficar preso nessa posição [Koren & Borenstein, 1991], como mostrado na Figura 2.2.

Pensando-se em aplicações nas quais sejam utilizadas grandes quantidades de agentes, trabalhos como Bruni et al. [2013]; Vrba et al. [2007] e Chang et al. [2003] foram desenvolvidos. Uma classe de algoritmos reativos que vem sendo bastante estudada nos últimos anos é a que engloba os métodos de navegação com controle de colisão baseados no espaço de velocidades dos agentes [Fiorini & Shiller, 1998; van den Berg et al., 2008; Snape et al., 2009; Gal et al., 2009; van den Berg et al., 2010; Snape et al., 2011]. Nesses métodos, um agente utiliza as velocidades dos outros agentes existentes em sua vizinhança para inferir as trajetórias desses agentes e calcular uma velocidade que o permita se deslocar sem entrar em rota de colisão com os demais robôs. Essas técnicas serão descritas nas próximas seções.

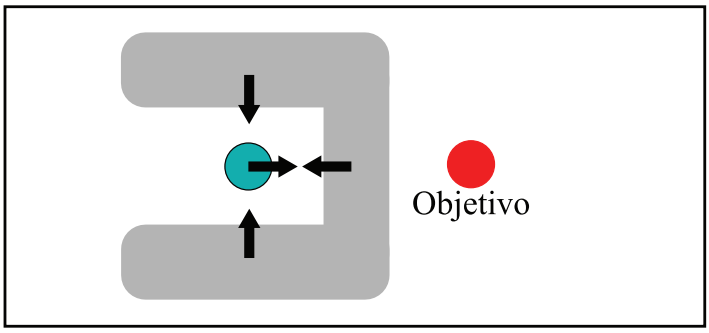


Figura 2.2: Cenário de um mínimo local. Todas as forças potenciais que estão atuando sobre o agente se anulam, fazendo com que o agente fique preso nessa posição.

2.2.1 Velocity Obstacle

Uma abordagem bastante estudada nos últimos anos é a chamada *Velocity Obstacle* (VO). Proposta por Fiorini & Shiller [1998], essa técnica considera os demais agentes presentes no ambiente como obstáculos móveis e trabalha no espaço de velocidades dos agentes para calcular caminhos livres de colisão. Para isso, assume-se que as posições e velocidades dos agentes são conhecidas ou podem ser inferidas a partir das observações feitas por eles. Com esses dados, os agentes calculam o conjunto VO de todas as velocidades que, se aplicadas aos agentes, levam a uma colisão entre eles. Feito isso, escolhe-se o vetor velocidade, não pertencente ao conjunto VO calculado, para se aplicar aos agentes e garantir uma navegação livre de colisão.

Mais formalmente, considere dois agentes $\mathcal{R}_A$ e $\mathcal{R}_B$ com raios r_A e r_B e posições $\boldsymbol{p}_A$ e $\boldsymbol{p}_B$, respectivamente, como representado na Figura 2.3. Neste exemplo, o agente $\mathcal{R}_A$ considera $\mathcal{R}_B$ como um obstáculo móvel e calcula as velocidades que garantem um caminho livre de colisão com esse agente. Para calcular o conjunto VO, é necessário mapear $\mathcal{R}_B$ no espaço de configuração de $\mathcal{R}_A$. Para isso, reduz-se $\mathcal{R}_A$ ao ponto $\hat{\mathcal{R}}_A$ e transforma-se $\mathcal{R}_B$ em $\hat{\mathcal{R}}_B$ aumentando-se o raio de $\mathcal{R}_B$ para r_A+r_B. Feito isso, define-se o Cone de Colisão, CC_B^A, como o conjunto de todas as velocidades relativas entre $\hat{\mathcal{R}}_A$ e $\hat{\mathcal{R}}_B$ que levam a uma colisão entre eles:

$$CC_B^A = \{v_{A,B} | \lambda_{A,B} \bigcap \hat{\mathcal{R}}_B \neq 0\}, \tag{2.1}$$

onde $v_{A,B}$ é a velocidade relativa de $\hat{\mathcal{R}}_A$ com respeito a $\hat{\mathcal{R}}_B$, $v_{A,B} = v_A - v_B$, e $\lambda_{A,B}$ é a reta dada pelo vetor suporte $v_{A,B}$ passando por $\hat{\mathcal{R}}_A$.

Esse cone, como mostrado na Figura 2.4, é uma região com ápice em $\hat{\mathcal{R}}_A$(origem do espaço de velocidades de $\mathcal{R}_A$), limitada pelas tangentes λ_f e λ_r de $\hat{\mathcal{R}}_A$ para $\hat{\mathcal{R}}_B$.

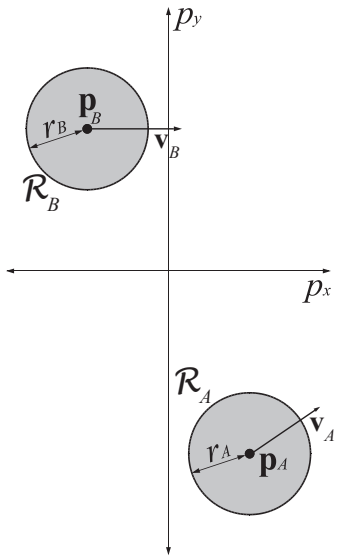


Figura 2.3: Configuração dos agentes $\mathcal{R}_A$ e $\mathcal{R}_B$.

Qualquer velocidade que esteja entre essas duas tangentes, se aplicadas a $\mathcal{R}_A$ e se $\mathcal{R}_B$ mantiver sua velocidade e forma constantes, irá causar uma colisão entre esses dois agentes. O algoritmo utiliza um parâmetro τ, mínimo período de tempo no qual os agentes não devem colidir entre si, para truncar o Cone de Colisão com um arco do círculo de raio $(r_A + r_B)/\tau$ com centro em $(\boldsymbol{p}_B - \boldsymbol{p}_A)/\tau$. Como o conjunto CC_B^A é formado por velocidades relativas de $\mathcal{R}_A$ em relação a $\mathcal{R}_B$, é interessante que essas velocidades relativas sejam transformadas em velocidades absolutas de $\mathcal{R}_A$, definindo assim o conjunto $VO^{\tau}_{A|B}(v_B)$ representado na Figura 2.5. Para fazer essa transformação, basta adicionar a velocidade $\boldsymbol{v}_B$ de $\mathcal{R}_B$ a cada velocidade existente no cone induzido por $\mathcal{R}_B$, ou seja

$$VO^{\tau}_{A|B}(v_B) = CC_B^A \oplus v_B, \tag{2.2}$$

onde $\oplus$ representa o operador da soma vetorial de Minkowski, definido formalmente como $A \oplus B = \{a + b \mid a \in A, b \in B\}$. Dessa forma torna-se possível calcular caminhos livres de colisão em ambientes com múltiplos agentes/obstáculos a partir da união dos conjuntos VO induzidos por cada agente/obstáculo presente na vizinhança de $\mathcal{R}_A$.

Um comportamento indesejado que pode afetar os resultados apresentados pelo algoritmo VO é a oscilação nas trajetórias executadas pelos robôs. Suponha que dois agentes $\mathcal{R}_A$ e $\mathcal{R}_B$ estejam se deslocando um na direção do outro com velocidades $\boldsymbol{v}_A$ e $\boldsymbol{v}_B$, respectivamente, e que $v_A \in VO^{\tau}_{A|B}(v_B)$ e $v_B \in VO^{\tau}_{B|A}(v_A)$. Digamos que, ao detectar a possível colisão com o agente $\mathcal{R}_B$, o agente $\mathcal{R}_A$ escolha a velocidade v'_A de tal modo que $v'_A \notin VO^{\tau}_{A|B}(v_B)$. De maneira análoga, $\mathcal{R}_B$ escolhe v'_B como sua nova velocidade para evitar a colisão com $\mathcal{R}_A$ sendo que $v'_B \notin VO^{\tau}_{B|A}(v_A)$. Porém, as antigas

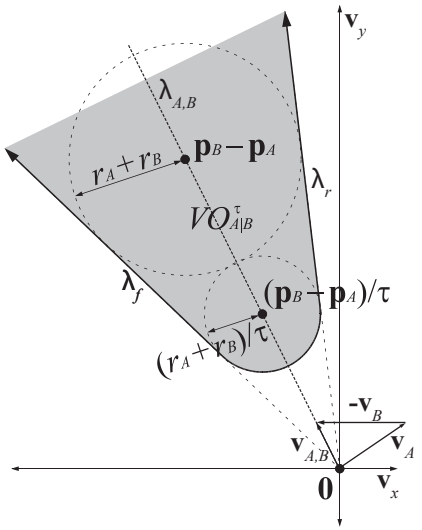


Figura 2.4: Conjunto CC_B^A das velocidades relativas de $\mathcal{R}_A$ que causam colisão com o agente $\mathcal{R}_B$.

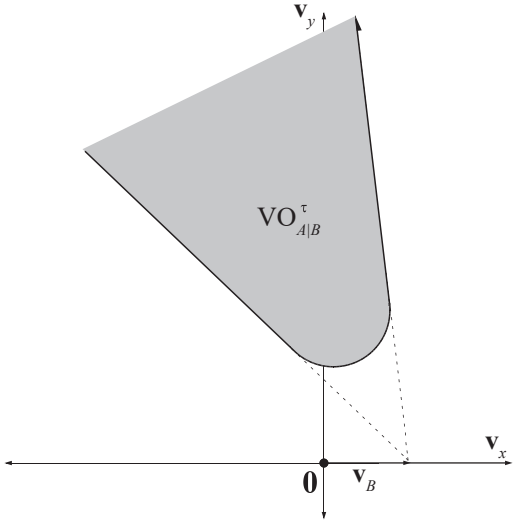


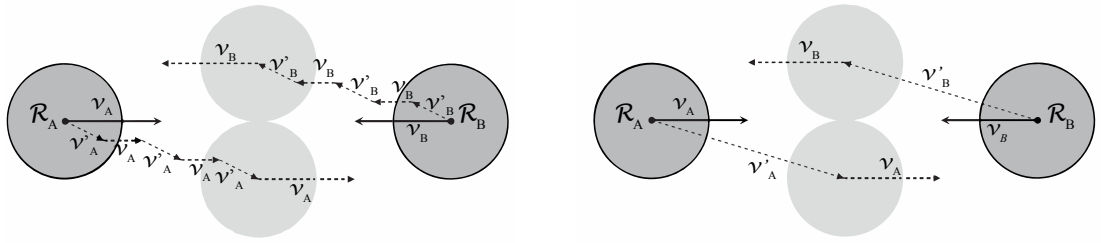
Figura 2.5: Conjunto $VO^{\tau}_{A|B}(v_B)$ das velocidades absolutas de $\mathcal{R}_A$ que causam colisão com o agente $\mathcal{R}_B$.

velocidades v_A e v_B, caso levem os agentes na direção dos seus respectivos objetivos, serão preferidas por eles. Assim, os agentes $\mathcal{R}_A$ e $\mathcal{R}_B$ irão escolher novamente as velocidades v_A e v_B como suas respectivas velocidades. No entanto, isso fará com que os agentes entrem novamente em rota de colisão, forçando-os a escolher novas velocidades, que poderão ser novamente v'_A e v'_B. Essa situação leva ao cenário ilustrado na Figura 2.6a, é possível notar que as trajetórias executadas pelos agentes sofrem oscilações indesejadas.

Para contornar o problema da oscilação, van den Berg et al. [2008] propõem uma variação do algoritmo VO chamada de *Reciprocal Velocity Obstacle* (RVO). Nessa abordagem, assume-se que todos os agentes presentes no ambiente utilizam estratégias similares para o controle de colisão e, para percorrer um caminho livre de colisões, ao invés de o agente calcular uma velocidade que esteja fora do conjunto VO induzido pelo outro agente, ele calcula sua nova velocidade como a média entre sua velocidade atual e uma velocidade que esteja fora do conjunto VO induzido pelo outro agente, dessa forma:

$$RVO^{\tau}_{A|B}(v_B, v_A) = \{v'_A \mid 2v'_A - v_A \in VO^{\tau}_{A|B}(v_B)\}. \tag{2.3}$$

Utilizando essa nova abordagem, é possível que os agentes executem trajetórias mais suaves e livres de oscilações, como as ilustradas na Figura 2.6b.



(a) Trajetória com oscilação.

(b) Trajetória sem oscilação.

Figura 2.6: Trajetórias executadas por dois agentes que possuem velocidades preferenciais opostas e entram em rota de colisão. (a) Trajetória calculada pelo algoritmo VO. (b) Trajetória calculada pelo algoritmo RVO.

Para estender o algoritmo VO para agentes não-holonômicos do tipo carro, Wilkie et al. [2009] propõem o algoritmo chamado de *Generalized Velocity Obstacles* (GVO), uma abordagem baseada no VO que incorpora as restrições cinemáticas e dinâmicas desse tipo de agente para calcular trajetórias livres de colisão. Outra variação do algoritmo VO é a técnica apresentada em Snape et al. [2011], conhecida como *Hybrid Reciprocal Velocty Obstacles* (HRVO). Nesse trabalho, os autores utilizam uma com-

binação dos algoritmos VO e RVO para calcular as novas velocidades que os agentes devem assumir. Vários outros trabalhos utilizaram o conceito dos *Velocity Obstacles* como base e propuseram extensões para essa técnica buscando melhorar seu desempenho e ampliando as possibilidades de uso da mesma [Snape et al., 2009; van den Berg et al., 2010, 2011; Snape et al., 2012; Alonso-Mora et al., 2010].

2.2.2 *Optimal Reciprocal Colision Avoidance* - ORCA

Uma outra extensão do algoritmo *Velocity Obstacles* é conhecida como ORCA (*Optimal Reciprocal Collision Avoidance*). Apresentada em van den Berg et al. [2011], essa abordagem calcula, para cada outro agente presente no ambiente, um semiplano que contém as velocidades que garantem um caminho livre de colisões entre os agentes. Feito isso, o agente seleciona a velocidade ótima que esteja na interseção de todos os semiplanos calculados, o que pode ser feito eficientemente resolvendo-se um sistema linear de pequena dimensão. Essa abordagem assume que todos os agentes presentes no ambiente utilizam a mesma estratégia para evitar colisões, o que permite que cada agente tenha metade da responsabilidade em garantir que colisões não ocorram.

Num cenário composto por dois agentes, $\mathcal{R}_A$ e $\mathcal{R}_B$, sendo $\boldsymbol{v}_A$ e $\boldsymbol{v}_B$ as velocidades atuais de $\mathcal{R}_A$ e $\mathcal{R}_B$, respectivamente, pela definição do *velocity obstacle*, se $\boldsymbol{v}_A - \boldsymbol{v}_B \in VO^{\tau}_{A|B}$, $\mathcal{R}_A$ e $\mathcal{R}_B$ irão colidir antes do tempo mínimo τ caso eles mantenham suas velocidades constantes. Consequentemente, caso $\boldsymbol{v}_A - \boldsymbol{v}_B \notin VO^{\tau}_{A|B}$ os dois agentes podem se mover de forma segura por, no mínimo, um tempo τ.

Definindo-se um conjunto de velocidades V_B, se $v_B \in V_B$ e $v_A \notin VO^{\tau}_{A|B} \oplus V_B$, então $\mathcal{R}_A$ e $\mathcal{R}_B$ não irão colidir durante o intervalo de tempo τ. Com isso, é possível definir o conjunto de velocidades anti-colisão (*collision-avoiding velocities*) $CA^{\tau}_{A|B}(V_B)$, mostrado na Figura 2.7, para $\mathcal{R}_A$ dado que $\mathcal{R}_B$ assume uma velocidade presente em V_B:

$$CA^{\tau}_{A|B}(V_B) = \{v \mid v \notin VO^{\tau}_{A|B} \oplus V_B\}. \tag{2.4}$$

Dois conjuntos de velocidades, V_A e V_B, são chamados de reciprocamente anti-colisão se $V_A \subseteq CA^{\tau}_{A|B}(V_B)$ e $V_B \subseteq CA^{\tau}_{B|A}(V_A)$. Caso $V_A = CA^{\tau}_{A|B}(V_B)$ e $V_B = CA^{\tau}_{B|A}(V_A)$ os conjuntos V_A e V_B são chamados reciprocamente máximos (*reciprocally maximal*).

O princípio estabelecido pelo algoritmo ORCA é definir conjuntos V_A e V_B, reciprocamente anti-colisão e máximos, para os agentes $\mathcal{R}_A$ e $\mathcal{R}_B$, respectivamente, garantindo que nenhuma colisão ocorra entre eles num intervalo de tempo τ, ou seja, $CA^{\tau}_{A|B}(V_B) = V_A$ e $CA^{\tau}_{B|A}(V_A) = V_B$. Além de definir esses conjuntos, como há vários

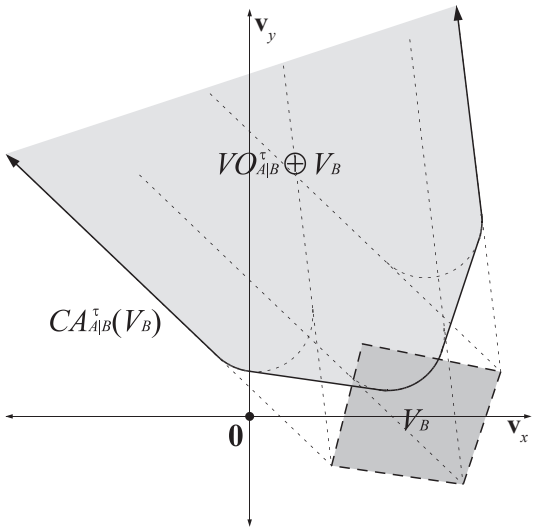


Figura 2.7: Conjunto $CA^{\tau}_{A|B}(V_B)$.

possíveis pares de conjuntos que obedeçam a essas restrições, o algoritmo estabelece que os conjuntos V_A e V_B devem ser o par que maximiza o total de velocidades permitidas próximas às velocidades de otimização v_A^{opt} para $\mathcal{R}_A$ e v_B^{opt} para $\mathcal{R}_B$. Essas velocidades de otimização podem ser interpretadas como as velocidades que causam a menor oscilação nas trajetórias executadas pelos agentes, ou seja, suas velocidades atuais. Esses conjuntos são denominados $ORCA^{\tau}_{A|B}$ para $\mathcal{R}_A$ e $ORCA^{\tau}_{B|A}$ para $\mathcal{R}_B$.

Os conjuntos $ORCA^{\tau}_{A|B}$ e $ORCA^{\tau}_{B|A}$ podem ser definidos geometricamente da seguinte forma: supondo que $\mathcal{R}_A$ e $\mathcal{R}_B$ assumam, respectivamente, as velocidade v_A^{opt} e v_B^{opt} e que isso faça com que eles entrem em rota de colisão. Define-se um vetor $\mathbf{u}$ do ponto $v_A^{opt} - v_B^{opt}$ até o ponto mais próximo pertencente à borda do conjunto $VO^{\tau}_{A|B}$, define-se também a normal externa n no ponto $(v_A^{opt} - v_B^{opt}) + u$. Feito isso, e dada a responsabilidade compartilhada pelos agentes em evitar colisões, o conjunto $ORCA^{\tau}_{A|B}$ pode ser definido como o semiplano perpendicular a n começando pelo ponto $v_A^{opt} + \frac{1}{2}u$, como mostrado na Figura 2.8.

2.3 Segregação de grupos em enxames de robôs

Uma característica presente nas comunidades de insetos sociais é a existência de papéis específicos que segregam os indivíduos da comunidade em grupos distintos. Cada grupo de indivíduos exerce atividades específicas com o intuito de promover a perpetuação da colônia. Alguns indivíduos são responsáveis por coletar alimentos, alguns cuidam da construção e manutenção da colônia, outros se empenham em defendê-la, dentre outras atividades [Oster & Wilson, 1978; Traniello & Rosengaus, 1997].

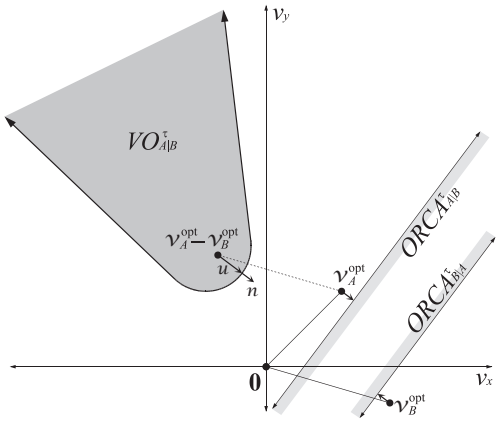


Figura 2.8: Semiplanos de velocidades permitidas para os agentes $\mathcal{R}_A$ e $\mathcal{R}_B$.

Em aplicações que utilizam enxames de robôs, pode ser interessante manter grupos distintos de agentes, designando tarefas específicas para cada grupo existente. Groß et al. [2009] apresenta uma abordagem que permite segregar agentes através de um algoritmo inspirado no chamado *Brazil nut effect*, que utiliza vibrações para segregar partículas de diferentes tamanhos. A técnica apresentada por esse trabalho utiliza três comportamentos reativos distintos para segregar os agentes. A combinação desses três comportamentos faz com que os agentes se organizem em estruturas anelares que levam em consideração o tamanho dos agentes, aqueles que possuírem maior tamanho, ficam em anéis mais externos, enquanto os agentes menores se organizam nos anéis mais internos. Em Kumar et al. [2010] foi desenvolvido um algoritmo capaz de segregar tipos distintos de agentes aplicando diferentes potenciais sobre os robôs de acordo com o tipo de agentes que estão em suas vizinhanças. Esse trabalho apresentou resultados satisfatórios, porém o algoritmo foi proposto para segregar somente dois tipos de agentes, o que restringe seu uso, já que em aplicações que utilizem grandes enxames de robôs pode ser interessante usar um número considerável de grupos distintos que não devam se misturar.

Santos et al. [2014b] apresenta uma abordagem capaz de segregar grupos de robôs heterogêneos utilizando o conceito de potenciais diferenciais. Para calcular o potencial que deve ser utilizado, cada agente avalia se seus vizinhos são do seu próprio tipo ou de um tipo diferente. Com isso, é possível calcular as forçar artificiais que deverão ser aplicadas em cada um deles para que agentes de um mesmo tipo se aproximem e se mantenham unidos, enquanto os de tipos distintos tendem a se manter afastados uns dos outros. As principais limitações dessa técnica são a necessidade de percepção

global e um número balanceado de agentes em todos os grupos.

Mais recentemente, Ferreira Filho & Pimenta [2015] propôs um método que utiliza abstrações para representar os grupos de agentes e faz uso de uma função potencial artificial para separá-los. Diferentemente de outros trabalhos sobre a segregação de enxames, nesse último trabalho, é matematicamente garantido que o sistema sempre converge para um estado em que vários grupos diferentes são segregados.

2.4 Navegação Segregada

Em ambientes nos quais existam grupos distintos de agentes, a utilização de estratégias que busquem uma navegação segregada pode ser interessante para melhorar o desempenho do sistema ou para respeitar restrições impostas ao mesmo. Com a utilização de grandes grupos de robôs num mesmo ambiente, a probabilidade desses grupos possuírem trajetórias que se interceptem em algum momento aumenta, e isso pode levar a situações indesejadas como congestionamentos e casos nos quais um robô, ou um pequeno grupo deles, fica roadeado por agentes de outros grupos e não consegue sair dessa situação.

Marcolino & Chaimowicz [2009] propõe uma abordagem que permite que grupos de agentes evitem regiões de congestionamento. Essa abordagem utiliza troca de mensagens entre os agentes de um mesmo grupo e regras de tráfego pré-estabelecidas para que cada agente execute um manobra que permita que todo o grupo se desloque numa mesma direção evitando a região com maior concentração de robôs.

Já o trabalho apresentado em Santos & Chaimowicz [2011] utiliza abstrações, que podem ser vistas como elipses, para representar cada grupo de agentes. O algoritmo desenvolvido utiliza as elipses abstratas para manter unidos os agentes que estejam em seu interior, enquanto repele os agentes que estejam fora da área compreendida pela elipse.

He & van den Berg [2013] apresenta uma abordagem na qual um agente agrupa os demais agentes presentes em sua vizinhança de acordo com suas posições e velocidades, considerando-os como uma entidade única que pode ser vista como um obstáculo móvel. Feito isso, calcula-se um envoltório convexo para cada grupo definido anteriormente com o intuito de calcular o conjunto VO de cada grupo e definir uma velocidade que permita que o agente desvie dos grupos ao invés de atravessá-los. Por se tratar de uma técnica que permite que um único agente desvie de regiões onde exista um congestionamento de agentes, essa abordagem não deve ser considerada como uma técnica de navegação segregada, porém ela pode ser utilizada como fonte de inspiração para

outras técnicas que busquem esse objetivo. Uma abordagem semelhante é apresentada em Santos et al. [2014a], nesse trabalho o objetivo é fazer com que grupos distintos de agentes naveguem pelo ambiente se mantendo unidos durante toda a navegação. O algoritmo VGVO, apresentado nesse trabalho, propõe que cada agente utilize uma abstração para representar cada grupo diferente do seu que esteja presente em sua vizinhança. Essa abstração pode ser vista como um envoltório convexo que abranja todos os agentes pertencentes a um mesmo grupo. Feito isso, o agente passa a considerar o envoltório convexo como um obstáculo móvel e utiliza o algoritmo RVO (*Reciprocal Velocity Obstacles*) [van den Berg et al., 2008] em conjunto com comportamentos de *flocking* para calcular as velocidades que permitam uma navegação segura entre tais obstáculos mantendo a segregação de grupos distintos.

Os trabalhos desenvolvidos por Santos et al. [2014a], Santos et al. [2014b] e Ferreira Filho & Pimenta [2015] são algoritmos que dependem de abstrações para representar cada grupo de agentes e/ou de agentes que possuam visão global do seu ambiente de trabalho para manter a segregação dos grupos nele existentes. Diferentemente dessas abordagens, o FL-ORCA pode ser caracterizado como um algoritmo descentralizado que é capaz de apresentar resultados satisfatórios, como será mostrado na seção de experimentos, mesmo com percepção local do ambiente.

2.5 Contextualização da abordagem proposta

A abordagem proposta nesse trabalho tem o intuito de realizar a navegação segregada de grupos distintos de agentes que possuam sensoriamento local do seu ambiente de trabalho. O algoritmo é baseado nos conceitos de *flocking* e utiliza o algoritmo ORCA para garantir que não ocorram colisões entre os agentes, sejam eles pertencentes a um mesmo grupo ou não. Assim como todos os demais trabalhos relacionados na Tabela 2.1, o FL-ORCA é um algoritmo descentralizado e reativo.

O uso de abstrações para representar os grupos de agentes permite que eles se mantenham afastados uns dos outros mantendo a segregação durante a navegação dos agentes. Porém, a necessidade de cada agente calcular o modelo abstrato que represente seu grupo exige que eles tenham um alto poder de processamento, além de possuir uma grande capacidade de percepção. Dito isto, destaca-se o fato de o FL-ORCA não utilizar abstrações para representar os grupos de robôs, cada agente calcula somente um somatório de componentes vetoriais para definir as suas velocidades preferenciais. Devido ao fato de não se utilizar abstrações para representar os grupos de robôs, torna-se possível também a utilização de agentes que possuam uma capacidade de percepção

mais reduzida, sem comprometer de forma drástica a eficiência do algoritmo.

Outra característica que faz do FL-ORCA uma abordagem robusta é a utilização do ORCA como algoritmo de controle de colisões. Com o seu uso, os agentes passam a ter metade da responsabilidade por evitar colisões, o que permite que eles executem trajetórias seguras e livres de oscilações indesejadas.

Destaca-se ainda, a capacidade de utilização do FL-ORCA em cenários nos quais os grupos não sejam totalmente balanceados. Os trabalhos desenvolvidos previamente por outros autores pressupõem que os grupos existentes no ambiente tenham tamanhos equilibrados, o que pode restringir o uso dessas abordagens em cenários nos quais não exista a garantia de que essa restrição seja respeitada.

Por fim, foi possível constatar que o algoritmo FL-ORCA obtém resultados melhores que outras abordagens como será demonstrado no Capítulo 4.

Abordagem	**Percepção**	**Estratégia**	**Aplicação**
Groß et al. [2009]	Local	Abstração	Segregação
Kumar et al. [2010]	Global	Campos Potenciais	Segregação
Santos et al. [2014b]	Global	Campos Potenciais	Segregação
Ferreira Filho & Pimenta [2015]	Local	Abstração	Segregação
Marcolino & Chaimowicz [2009]	Local	Comunicação + Regras de tráfego	Navegação Segregada
Santos & Chaimowicz [2011]	Local	Abstração	Navegação Segregada
Santos et al. [2014a]	Local*	Flocking	Navegação Segregada
FL-ORCA	Local	Flocking	Navegação Segregada

Tabela 2.1: Características dos trabalhos relacionados e do algoritmo proposto.

* A qualidade dos resultados alcançados pelo algoritmo é diretamente proporcional ao raio de visão do agente.

Capítulo 3

Metodologia

Como foi mencionado no Capítulo 1, o objetivo deste trabalho é manter grupos distintos de robôs segregados enquanto eles navegam por um ambiente de trabalho compartilhado. Para isso, foi proposto o FL-ORCA, um algoritmo de navegação reativa baseado nos conceitos dos campos potenciais artificiais e nos algoritmos de *flocking* [Reynolds, 1987] e ORCA [van den Berg et al., 2011]. Neste algoritmo, um agente percebe o ambiente no qual está inserido e classifica os outros agentes que estejam próximos a ele em dois grupos: (i) agentes do seu próprio grupo e (ii) agentes de um grupo diferente do seu. De acordo com o tipo de cada agente presente em sua vizinhança, o agente calcula um somatório de componentes vetoriais com o intuito de definir uma velocidade que é utilizada como velocidade preferencial pelo ORCA. Com isso, é possível que cada agente calcule as velocidades que o mantém próximo dos demais agentes do seu grupo e o afastam de agentes de grupos diferentes. A Figura 3.1 mostra o processo que o algoritmo segue para definir a nova velocidade que o agente deve desenvolver.

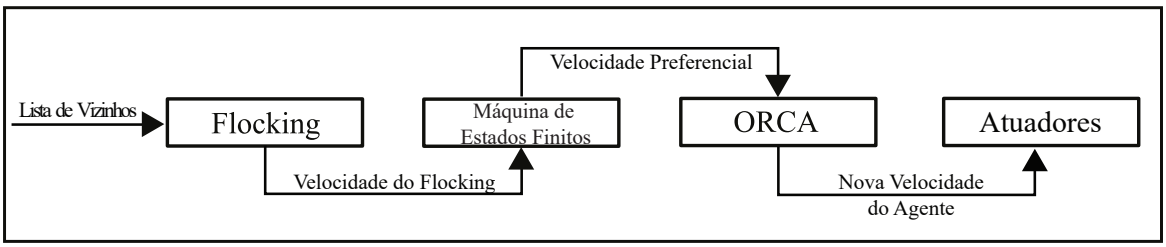


Figura 3.1: Representação esquemática do funcionamento do algoritmo FL-ORCA.

Neste trabalho, são considerados cenários nos quais um enxame $\boldsymbol{\mathcal{R}} = \{\mathcal{R}_1, ..., \mathcal{R}_\eta\}$ de η robôs precisa navegar em um ambiente bidimensional. Cada robô é representado pela sua pose $q_i = \langle p_i, \theta_i \rangle$, com o seguinte modelo cinemático:

$$\dot{q}_i = u_i. \tag{3.1}$$

Os robôs considerados são holonômicos e podem se movimentar em qualquer direção dada pelo vetor velocidade $\mathbf{u}_i$. Dessa forma, $p_i =< x, y >$ é a posição do robô $\mathcal{R}_i$ enquanto θ_i representa o ângulo formado por um vetor unitário que aponta na direção de movimento de $\mathcal{R}_i$ no espaço de configuração global.

Os enxames utilizados nesse trabalho são formados por grupos (tipos) distintos de robôs e podem ser representados pela partição $\Gamma = \{\Gamma_1, ..., \Gamma_m\}$, onde cada Γ_k contém todos os agentes do grupo k. Assume-se que $\forall j, k : j \neq k \rightarrow \Gamma_j \cap \Gamma_k = \emptyset$, ou seja, cada robô pertence a um único grupo.

3.1 O Algoritmo *Flocking*

Para manter a união de um grupo enquanto ele navega, foram usados os comportamentos clássicos propostos, por Reynolds [1987], no algoritmo de *flocking*. Esses comportamentos são obtidos a partir da aplicação de três regras simples que atuam sobre cada agente individualmente e serão descritas a seguir.

Para que as regras do *flocking* possam ser utilizadas, define-se a vizinhança de um robô. Neste trabalho, essa vizinhança está diretamente relacionada com a área de sensoriamento do robô. Durante sua navegação, o robô $\mathcal{R}_i$ considera sua vizinhança $\mathcal{N}_i$ como uma região circular de raio λ em volta da sua posição atual. Dentro de sua vizinhança, podem ser percebidos agentes do mesmo grupo de $\mathcal{R}_i$ ou agentes de grupos diferentes. Portanto, define-se $\mathcal{N}_i^+ = \{\mathcal{R}_j : \|\boldsymbol{p}_j - \boldsymbol{p}_i\| \leq \lambda \wedge \mathcal{R}_j, \mathcal{R}_i \in \Gamma_k\}$ como o conjunto de todos os robôs do mesmo grupo que $\mathcal{R}_i$ atualmente localizados em sua vizinhança. De forma análoga, $\mathcal{N}_i^- = \{\mathcal{R}_j : \|\boldsymbol{p}_j - \boldsymbol{p}_i\| \leq \lambda \wedge (\mathcal{R}_j \in \Gamma_u \wedge \mathcal{R}_i \in \Gamma_k, u \neq k)\}$ consiste no conjunto de todos os robôs pertencentes a qualquer grupo que seja diferente do grupo de $\mathcal{R}_i$ e que esteja localizado dentro da região definida como vizinhança de $\mathcal{R}_i$. O tamanho da vizinhança de um agente tem influência significativa sobre a qualidade dos resultados alcançados pelo algoritmo. A utilização de uma grande área de percepção aumenta a quantidade de informação disponível para os cálculos, o que pode permitir uma otimização nas trajetórias executadas por cada robô. Por outro lado, o uso de áreas de percepção maiores pode levar a um aumento significativo do processamento necessário para se definir as velocidades permitidas para cada agente.

A primeira regra do algoritmo de *flocking* está relacionada com a coesão dos agentes e é utilizada para mantê-los próximos uns dos outros. Para que a coesão de um grupo seja mantida, um agente $\mathcal{R}_i$ calcula o ponto médio das posições de todos os

agentes pertencentes a $\mathcal{N}_i^+$. Após isso, uma força vetorial é aplicada sobre $\mathcal{R}_i$ fazendo com que ele tenda a se deslocar para a posição calculada. Formalmente:

$$\boldsymbol{v}_{\text{coesão}} = \left(\frac{1}{|\mathcal{N}_i^+|} \sum_{j \in \mathcal{N}_i^+} \boldsymbol{p}_j \right) - \boldsymbol{p}_i. \tag{3.2}$$

A segunda regra foi originalmente proposta para manter uma distância segura entre os agentes. Apesar de obter resultados significativos, essa regra não garante que colisões não irão ocorrer. Porém, como será mostrado mais adiante, neste trabalho utiliza-se o algoritmo ORCA para garantir que não ocorram colisões durante a navegação dos agentes. Por isso, essa regra de separação foi adaptada para manter grupos diferentes afastados uns dos outros. Dessa forma, um agente calcula a posição que respeita uma distância mínima de cada um dos agentes de um grupo diferente do seu que estejam presentes em sua vizinhança, ou seja:

$$\boldsymbol{v}_{\text{separação}} = \sum_{j \in \mathcal{N}_i^-} (\boldsymbol{p}_i - \boldsymbol{p}_j). \tag{3.3}$$

O algoritmo ORCA, quando utilizado como estratégia de navegação com controle de colisão, é capaz de evitar que os agentes escolham velocidades que façam com que as trajetórias executadas apresentem o comportamento oscilatório existente no algoritmo VO. Porém, quando se utiliza o ORCA em conjunto com o *flocking* para manter a segregação de grupos, essa característica fica comprometida. Após alguns experimentos, foi possível notar que os grupos apresentavam um comportamento bastante parecido quando as configurações iniciais de cada grupo eram semelhantes, o que provocava um espelhamento de movimento dos grupos. Nesse tipo de situação, os grupos de agentes tendem a escolher trajetórias muito parecidas, fazendo com que um grupo bloqueie o caminho do outro. Esse comportamento leva a um aumento indesejado no tempo necessário para que cada grupo navegue até seu objetivo, degradando assim o desempenho do enxame. Para contornar esse problema, foi inserida uma componente aleatória Δ que representa no máximo 10% da componente de separação do agente. Dessa forma temos:

$$\boldsymbol{v}_{\text{separação}} = \sum_{j \in \mathcal{N}_i^-} (\boldsymbol{p}_i - \boldsymbol{p}_j) + \Delta. \tag{3.4}$$

Por fim, a terceira regra avalia o alinhamento de agentes de um mesmo grupo. Para que essa regra seja respeitada, um agente calcula a média da orientação espacial

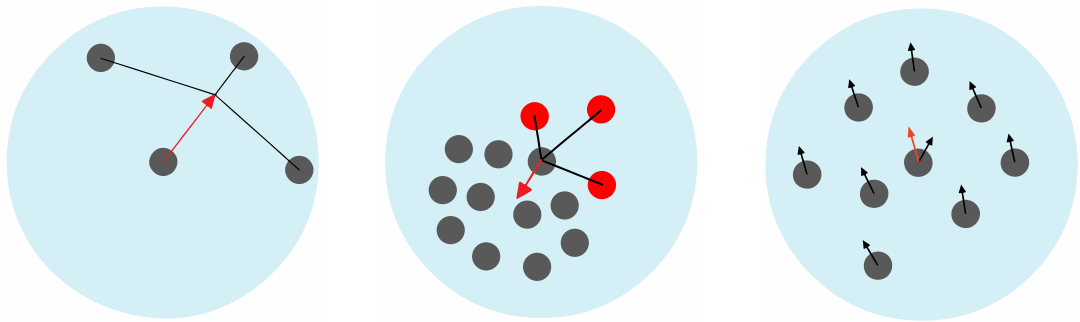
de todos os agentes do seu grupo que possam ser percebidos por ele. Depois disso, uma força é aplicada sobre o agente para que ele se alinhe com o valor médio calculado, o que pode ser representado formalmente como:

$$\boldsymbol{v}_{\text{alinhamento}} = \frac{1}{|\mathcal{N}_i^+|} \sum_{j \in \mathcal{N}_i^+} \theta_j. \tag{3.5}$$

A partir da combinação das três regras apresentadas, calcula-se a força resultante $\boldsymbol{v}_{\text{flock}}$ que será usada na fase de controle. Essa força resultante é definida como:

$$\boldsymbol{v}_{\text{flock}} = k_c \cdot \boldsymbol{v}_{\text{coesão}} + k_s \cdot \boldsymbol{v}_{\text{separação}} + k_d \cdot \boldsymbol{v}_{\text{alinhamento}}, \tag{3.6}$$

dessa forma, $\boldsymbol{v}_{\text{flock}}$ é composta por uma soma ponderada dos três vetores calculados pelas regras do *flocking* e as constantes k_c, k_s e k_a são determinadas empiricamente. A Figura 3.2 ilustra essas regras. Após o cálculo de $\boldsymbol{v}_{\text{flock}}$, esse valor é utilizado de acordo com o estado atual do agente para calcular a velocidade que é utilizada como velocidade preferencial pelo ORCA, como será discutido na próxima seção.



(a) Coesão (b) Separação (c) Alinhamento

Figura 3.2: Regras do algoritmo *flocking* (modificado). A seta vermelha representa a força que deve ser aplicada ao agente para que ele respeite as regras do algoritmo.

3.2 Determinação da Velocidade Preferencial

Durante sua navegação, um agente pode se deparar com diferentes situações. Por exemplo, ele pode se ver rodeado somente por agentes do seu próprio grupo, pode se encontrar numa posição na qual não exista nenhum outro agente entre ele e seu objetivo ou pode ainda estar próximo de agentes pertencentes a um grupo diferente do seu. Ao

se deparar com cada uma dessas situações, o agente precisa se comportar de uma forma adequada para manter-se próximo dos agentes do seu próprio tipo, enquanto evita se misturar a agentes de tipos diferentes do seu, preservando assim a segregação de grupos no ambiente de trabalho.

Foi proposta uma abordagem que pudesse se ajustar às diferentes situações que um agente enfrenta enquanto navega. Nessa abordagem, a velocidade preferencial de cada robô é composta basicamente por três componentes:

- uma componente que atrai o robô para a posição do objetivo atribuído ao seu grupo;
- a componente relacionada ao *flocking* que é calculada pela equação 3.6 e;
- uma componente auxiliar que é adicionada dependendo da situação na qual o agente se encontra.

Assim, essa velocidade preferencial é dada por:

$$v_{pref} = \alpha.v_{objetivo} + \beta.v_{flock} + \gamma.v_{aux}. \tag{3.7}$$

As possíveis situações nas quais um agente pode se encontrar durante sua navegação foram representadas como um dos estados de uma máquina de estados finitos, sendo estes estados definidos como:

• **Grupo Único:** Este primeiro estado está ativo sempre que um agente não percebe nenhum outro robô de um grupo diferente do seu em sua vizinhança, como mostrado na Figura 3.3 . Neste caso, o agente pode se mover diretamente para seu objetivo enquanto mantém a coesão com seu grupo. Para que esse comportamento seja alcançado, as constantes são ajustadas de tal modo que $\alpha \gg \beta$ e $\gamma = 0$.

• **Visão Livre:** O agente entra nesse estado sempre que ele consegue detectar a presença de um ou mais agentes pertencentes a um grupo que não seja o seu, mas ainda possui visão livre até o seu objetivo, como mostrado na Figura 3.4. Mais especificamente, quando não existe nenhum outro agente em um setor do seu campo de visão a partir da sua posição atual em direção ao seu objetivo, como mostrado na Figura 3.5. Neste caso, o agente ainda se desloca em direção ao seu objetivo, porém ele passa a dar maior importância à componente relacionada ao *flocking*. Assim, as constantes são ajustadas de forma que $\alpha > \beta$ e $\gamma = 0$.

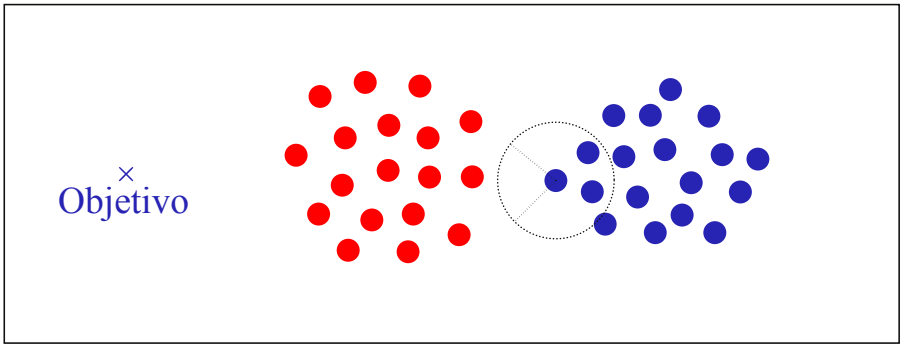


Figura 3.3: Cenário no qual o agente não percebe nenhum agente de um grupo diferente do seu.

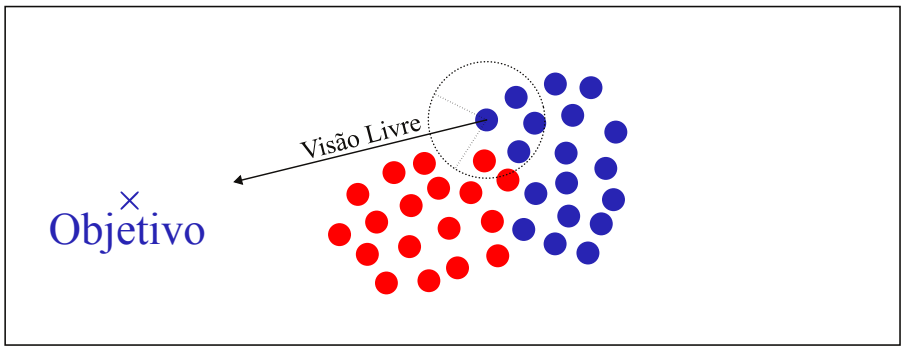


Figura 3.4: Cenário no qual o agente possui visão livre para o seu objetivo.

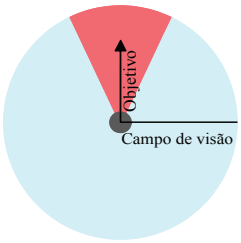


Figura 3.5: O setor vermelho é avaliado pelo robô na busca por outros agentes.

• **Seguidor:** Caso o agente não tenha uma visão livre para o objetivo mas identifique um agente vizinho, pertencente ao seu próprio grupo, que esteja em um dos estados apresentados anteriormente, como mostrado na Figura 3.6, o agente entra no estado seguidor, passando a seguir o agente que possui visão livre para o objetivo, mas ainda mantendo o comportamento de *flocking*. Neste caso, a componente v_{aux} é ajustada para induzir o agente a se mover em direção a este vizinho. Dessa forma, $\alpha = 0$ e $\beta < \gamma$.

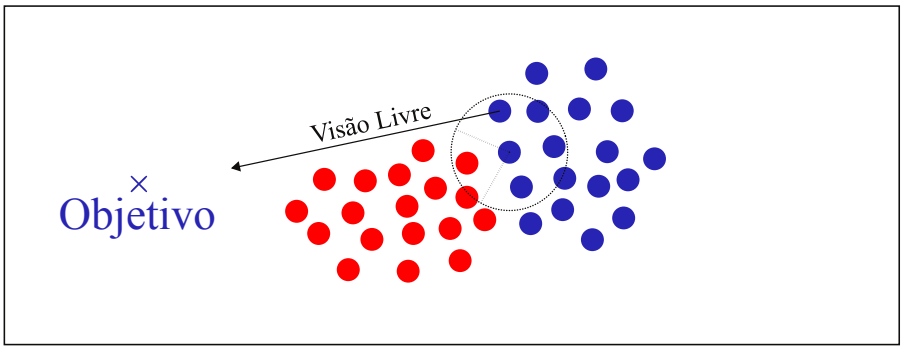


Figura 3.6: Cenário no qual o agente não possui visão livre para o seu objetivo, mas um de seus vizinhos possui.

• **Vire a direita:** Finalmente, se o agente não estiver em nenhuma das situações descritas anteriormente, ele provavelmente tem um cenário de congestionamento em sua frente, como mostrado na Figura 3.7, e é desejável que ele consiga evitar essa situação. Dessa forma, uma regra de controle de tráfego é imposta fazendo com que o robô se mova perpendicularmente à direção do seu objetivo. Para isso, v_{aux} é definida na direção perpendicular a $v_{objetivo}$. Com esse comportamento, os agentes tentam contornar essas áreas congestionadas ao invés de tentar atravessá-las. Nessa situação, as constantes são definidas para balancear essa componente perpendicular e a componente do *flocking*: $\alpha = 0$ e $\beta \simeq \gamma$.

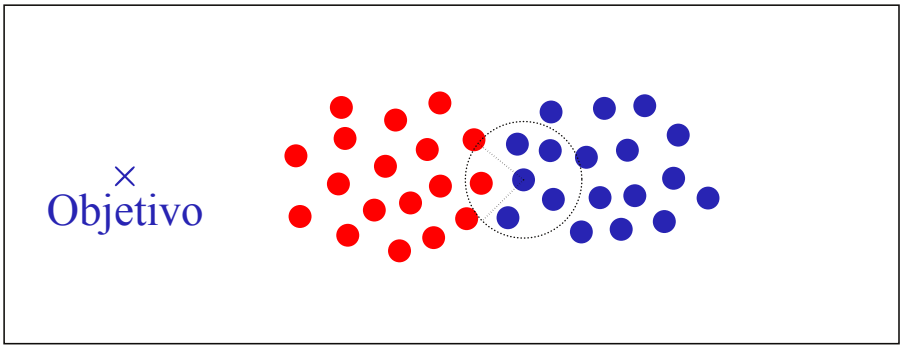


Figura 3.7: Cenário no qual o agente e seus vizinhos não possuem visão livre para o objetivo.

As transições entre cada um dos estados apresentados são controladas pela máquina de estados finitos apresentada na Figura 3.8.

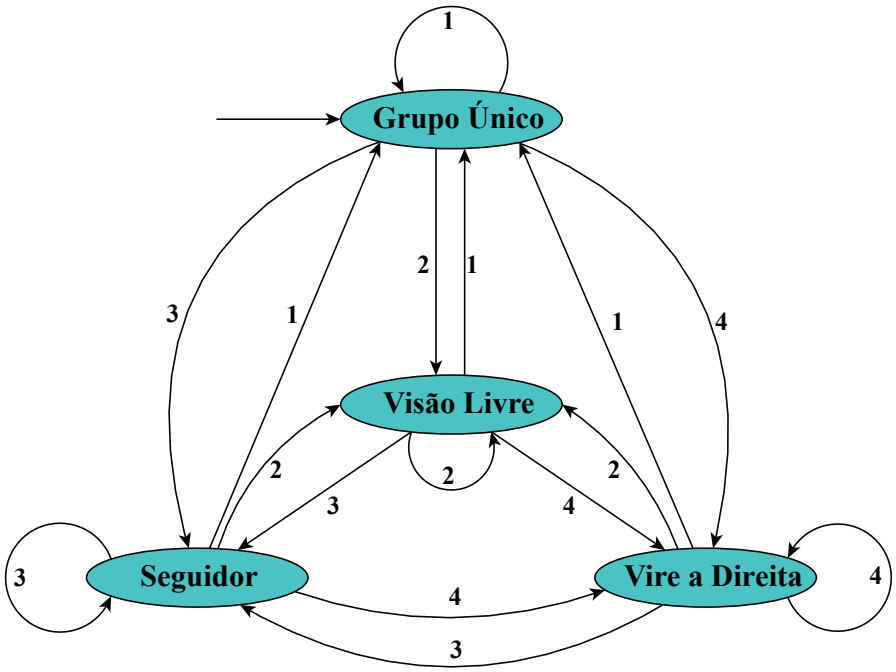


Figura 3.8: Máquina de estados finitos descrevendo as possíveis situações observadas por um robô.

3.3 O Algoritmo FL-ORCA

Como foi dito anteriormente, o algoritmo ORCA é uma estratégia de navegação baseada em velocidade que foi inspirado no conceito de *velocity obstacles* [Fiorini & Shiller, 1998]. Esse algoritmo assume que um agente tem acesso à posição e velocidade atuais de todos os outros agentes existentes em sua vizinhança, ou pode inferir essas informações com base em suas observações. Assume-se também que outros agentes presentes no ambiente utilizam a mesma abordagem para evitar colisões. Com isso, um indivíduo tem metade da responsabilidade pelos cálculos necessários para garantir que colisões não ocorram.

Um conceito importante na execução do ORCA é a chamada velocidade preferencial. Esse valor é utilizado como referência para determinar a velocidade que será aplicada ao robô durante a navegação. Consequentemente, o valor pertencente ao semi-

plano que mais se aproxime a essa velocidade preferencial é calculado por um algoritmo de programação linear e passado como a nova velocidade que o robô deve assumir.

O algoritmo FL-ORCA consiste em utilizar a informação capturada pelo agente para definir qual o estado atual no qual ele se encontra e então calcular uma velocidade na tentativa de manter os grupos existentes no ambiente segregados. Dessa forma, a velocidade calculada é passada para o algoritmo ORCA como a velocidade preferencial do agente. Como os controladores são definidos com o intuito de manter os agentes unidos em seus próprios grupos ao mesmo tempo em que mantém grupos distintos afastados uns dos outros, as velocidades calculadas pelo algoritmo FL-ORCA são as mais próximas possíveis dos valores que buscam manter os grupos segregados. Esse cálculo é executado de forma contínua até que o agente alcance seu objetivo, sendo o intervalo de tempo entre cada iteração limitado somente pelo tempo necessário para que o agente faça a leitura dos dados do ambiente e processe as informações obtidas. Vale ressaltar que não existe nenhum tipo de sincronização ou centralização do algoritmo, cada agente executa-o de forma totalmente independente e toma suas próprias decisões com base apenas no que é percebido e calculado por ele. Dessa forma o algoritmo proposto pode ser descrito como mostrado a seguir:

Algoritmo 1: FL-ORCA.

Entrada: $\mathcal{N}_i$ //Lista de vizinhos do agente i.

Saída: u //Nova velocidade que o agente i deve desenvolver.

1 **início**
2 $\boldsymbol{v}_{\text{coesão}} \leftarrow$ calcular coesão conforme Eq. 3.2;
3 $\boldsymbol{v}_{\text{separação}} \leftarrow$ calcular separação conforme Eq. 3.4;
4 $\boldsymbol{v}_{\text{alinhamento}} \leftarrow$ calcular alinhamento conforme Eq. 3.5;
5 Determinar estado atual do agente i de acordo com a máquina de estados finitos da Figura 3.8;
6 $\boldsymbol{v}_{\text{flock}} \leftarrow$ calcular *flocking* conforme Eq. 3.6;
7 $v_{pref} \leftarrow$ calcular velocidade preferencial conforme Eq. 3.7 ;
8 $u_i \leftarrow \text{ORCA}(v_{pref})$;
9 **retorna** u
10 **fim**

Capítulo 4

Experimentos

Neste capítulo, são apresentados os experimentos simulados e reais que foram desenvolvidos para investigar a eficiência e a viabilidade do uso do FL-ORCA como um algoritmo de navegação e manutenção da segregação de grupos heterogêneos de robôs, bem como a análise dos dados obtidos nesses experimentos.

Para evidenciar a eficiência do algoritmo proposto neste trabalho, o algoritmo FL-ORCA foi comparado com os algoritmos ORCA e VGVO. Foram criados nove cenários com configurações iniciais distintas e cada um deles foi utilizado nas execuções dos três algoritmos. A partir dessas execuções é possível fazer uma comparação entre o tempo necessário para que os agentes alcancem seus objetivos e a qualidade do algoritmo em relação à manutenção da segregação dos grupos de robôs. Mesmo que o algoritmo ORCA não tenha como objetivo a manutenção da segregação dos grupos, é interessante que seus resultados nos experimentos sejam considerados para possibilitar uma análise de como essa restrição afeta o comportamento dos agentes. Além das simulações mencionadas acima, também foram realizados experimentos com grupos de robôs reais para demonstrar a aplicabilidade do algoritmo.

Um parâmetro que tem influência direta sobre os algoritmos reativos é a área que um agente consegue observar para tomar suas decisões. Por isso, para investigar o efeito que esse parâmetro exerce sobre os algoritmos utilizados nos experimentos, cada um deles foi executado com dois valores distintos de raio de visão dos agentes. Nas simulações, considerou-se o uso de agentes com raios de visão de 3 e 10 metros, sendo que o corpo do agente mede 35 centímetros de raio e o ambiente tem uma área de aproximadamente 10 metros quadrados.

Pelo fato dos algoritmos não serem determinísticos, cada cenário foi utilizado em cem execuções distintas e os resultados obtidos em cada uma delas foram compilados para gerar os dados e análises apresentadas nas próximas seções.

4.1 Métricas

Para avaliar a eficiência dos algoritmos em relação à manutenção da segregação dos grupos de agentes foram utilizadas duas métricas. Primeiramente, utilizou-se a métrica definida por Kumar et al. [2010] para avaliar a distância média entre os agentes. Devido ao fato dessa métrica ter sido proposta, originalmente, para avaliar cenários nos quais existam apenas dois grupos distintos de agentes, uma segunda métrica, que avalia a conectividade dos agentes, foi proposta. Além disso, considerou-se importante estabelecer uma comparação de desempenho entre os algoritmos. Para isso, o tempo total necessário para que todos os agentes alcançassem seus objetivos foi utilizado como a terceira métrica avaliada. Com essa análise busca-se investigar o impacto que a restrição de se manter a segregação dos grupos durante sua navegação exerce sobre o desempenho dos algoritmos avaliados.

4.1.1 Distância média entre agentes

A métrica proposta por Kumar et al. [2010] define que dois grupos distintos de agentes são considerados segregados quando a distância média entre agentes de um mesmo grupo é menor que a distância média entre agentes de grupos distintos. Formalmente, dois grupos de agentes A e B são ditos segregados se

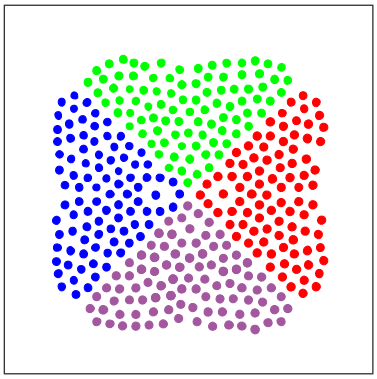
$$(d_{AA} < d_{AB}) \quad \wedge \quad (d_{BB} < d_{AB}), \tag{4.1}$$

onde d_{XY} é a distância média entre agentes dos tipos X e Y.

Como mencionado anteriormente, essa métrica foi originalmente definida para medir a segregação entre, somente, dois grupos. Em cenários nos quais existe uma maior quantidade, a utilização dessa métrica não é suficiente para afirmar se os grupos estão ou não segregados. A Figura 4.1 (a) apresenta uma situação na qual os agentes estão visivelmente segregados, porém os valores de distância média entre os agentes, apresentados na Figura 4.1 (b), não respeitam a restrição definida na Equação 4.1. Apesar dessa limitação, essa métrica pode ser utilizada como um indicador da segregação, principalmente, na fase de ajuste dos parâmetros utilizados para ponderar as componentes vetoriais que formam a velocidade preferencial dos agentes.

4.1.2 Conectividade dos agentes

Como a métrica mostrada acima apresenta limitações quando o número de grupos no enxame é maior que dois, uma segunda métrica foi proposta para avaliar os algoritmos



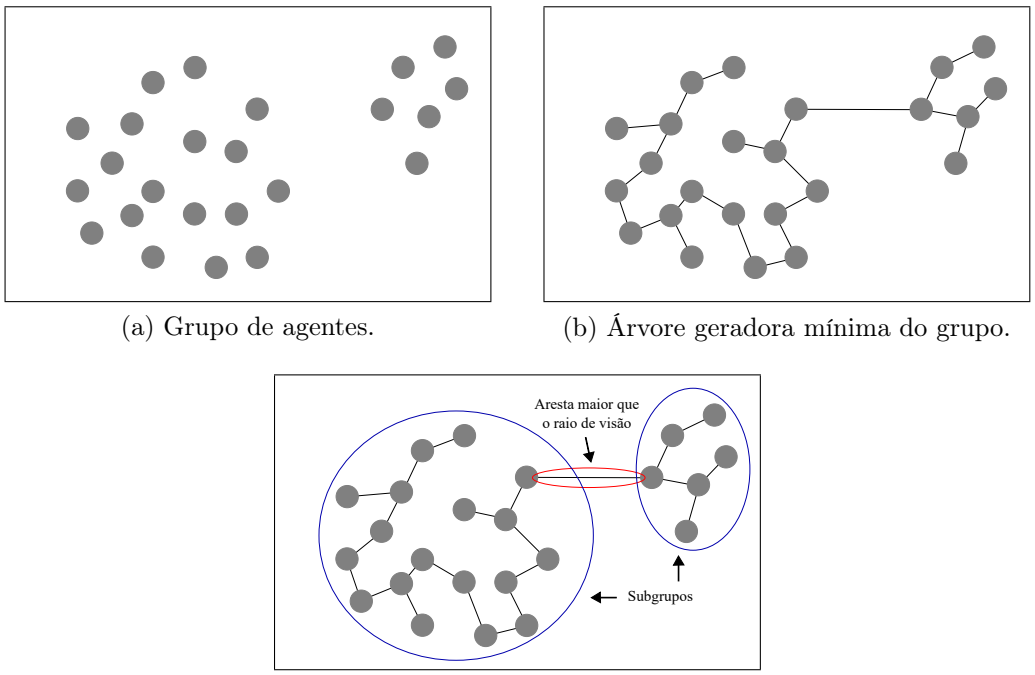
(a) Cenário com grupos segregados.

	15,59892	19,82472	14,65471	14,64562
	19,82472	15,58618	14,66869	14,66551
	14,65471	14,66869	15,5601	19,83299
	14,64562	14,66551	19,83299	15,55976

(b) Distâncias médias entre os agentes.

Figura 4.1: Situação na qual os grupos estão segregados, porém não obedecem a restrição de distância média.

em relação à conectividade entre agentes de um mesmo grupo. Esta métrica define que um agente está conectado a outro agente do seu grupo se a distância entre eles é menor do que seu raio de visão, ou seja, um agente $\mathcal{R}_A$ está conectado a um agente $\mathcal{R}_B$ se e somente se $\mathcal{R}_B \in \mathcal{N}_A^+$. Para calcular a conectividade de um grupo, foi utilizada uma modelagem que propõe que cada agente de um grupo representa um vértice em um grafo completo não direcionado, a Figura 4.2(a) apresenta um grupo de agentes que será utilizado como exemplo. Com a representação do grupo de agentes como um grafo completo, é possível utilizar algum algoritmo para criar uma árvore geradora mínima que represente esse grafo (Figura 4.2(b)), nesse trabalho utilizou-se o Algoritmo de Prim para gerar tal árvore. Feito isso, verifica-se a existência de alguma aresta maior que o raio de visão do agente, caso exista alguma aresta com essa característica, os vértices ligados por ela representam dois agentes que não estão conectados. A Figura 4.2(c) ilustra a situação mencionada, é possível observar que existem dois subgrupos ligados por uma aresta maior que o raio de visão dos agentes. Dessa forma, foi possível calcular o tempo que os agentes permaneceram conectados durante toda a sua navegação. Essa métrica está diretamente relacionada à capacidade de um grupo se manter unido, quando todos os agentes de um grupo se mantém conectados durante sua navegação a probabilidade de que a segregação dos grupos seja perdida é menor. Além disso, se a restrição da navegação segregada for violada, é mais provável que os grupos consigam reestabelecer a segregação nos casos nos quais os agentes pertencentes a um mesmo grupo estejam conectados uns aos outros.



(a) Grupo de agentes.

(b) Árvore geradora mínima do grupo.

(c) Subgrupos afastados.

Figura 4.2: Situação na qual um grupo está dividido em dois subgrupos.

4.2 Experimentos Simulados

Os experimentos realizados foram divididos em 3 grupos. No primeiro grupo, o algoritmo foi avaliado considerando cenários com um número fixo de grupos e aumentando-se o número de agentes em cada um deles. Em seguida, o número de agentes em cada grupo foi fixado e variou-se o número de grupos. Por fim, o algoritmo foi avaliado em cenários nos quais o número de agentes fosse diferente em cada grupo existente no ambiente.Inicialmente, os experimentos foram realizados com o simulador *Stage* presente no framework *ROS* [Quigley et al., 2009], um arcabouço bastante utilizado para programação e simulações com robôs. Porém, como o algoritmo proposto tem como objetivo manter grandes grupos de agentes segregados, ao se aumentar a quantidade de agentes nas simulações observou-se uma perda significativa de desempenho devido ao uso do *Stage*. Por isso, foi desenvolvida uma interface gráfica com a biblioteca *OpenCV* para substituir o simulador nativo do *ROS*.

4.2.1 Avaliação da distância média entre agentes

A partir da análise da Figura 4.3, nota-se que o algoritmo ORCA violou a restrição proposta pela métrica das distâncias médias entre agentes por aproximadamente 23

segundos, o que representa 38,33% do tempo total necessário para que os grupos alcançassem seus objetivos. Observa-se que as distância entre os agentes do grupo A se tornaram bem maiores que as registradas entre esses mesmos agentes e os agentes pertencentes aos demais grupos, o que demonstra uma mistura entre os grupos.

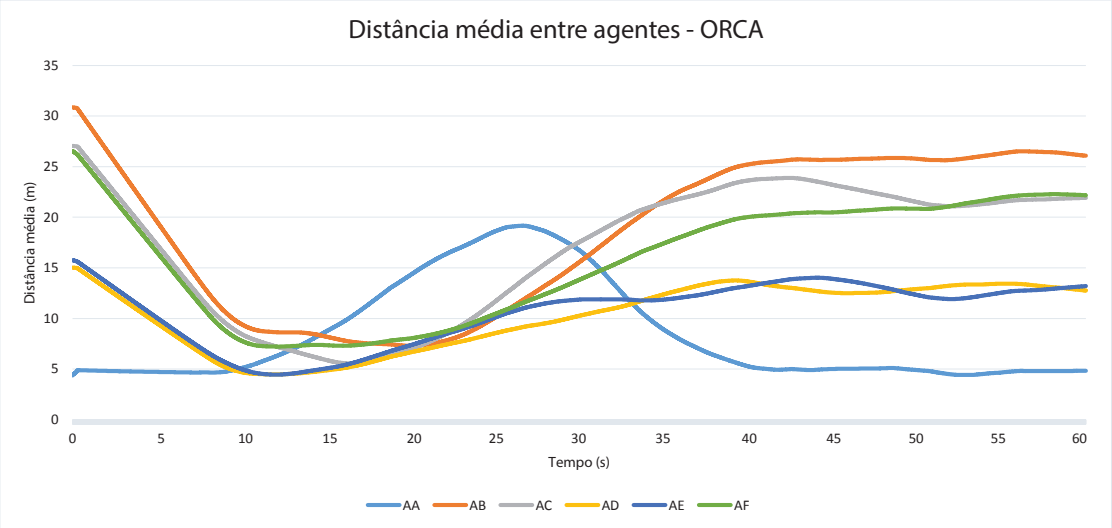


Figura 4.3: Distâncias médias entre agentes durante a navegação com a utilização do algoritmo ORCA.

O comportamento do algoritmo VGVO, ilustrado na Figura 4.4, apresentou dois momentos nos quais a restrição de distância entre agentes foi violada, os intervalos entre 69 e 86 segundos e entre 151 e 175, totalizando 41 segundos ou 15,24% do tempo total de navegação. Porém, é possível perceber que os valores das distâncias entre agentes permaneceram muito próximos entre esses dois intervalos, o que pode significar uma violação da segregação dos grupos não detectada pela métrica em questão, pois o esperado é que os valores médios de distância se afastem de forma contínua logo após os grupos voltarem a respeitar a restrição de segregação. Assim sendo, se considerarmos o espaço de tempo que vai de 69 a 175 segundos de execução, temos um intervalo de 106 segundos, o que representa 39,4% do tempo total navegado.

Já na execução do algoritmo FL-ORCA, os agentes mantiveram distâncias médias que não respeitaram a restrição por 14 segundos, o que corresponde a 17,5% do tempo total da execução, como pode ser vistou na Figura 4.5.

Essa seção avalia as distâncias médias entre os agentes durante a execução dos três algoritmos utilizados nos experimentos em um cenário composto por seis grupos de 30 agentes. Ressalta-se o fato de que o comportamento apresentado acima foi verificado também em todos os outros cenários avaliados na fase de experimentação

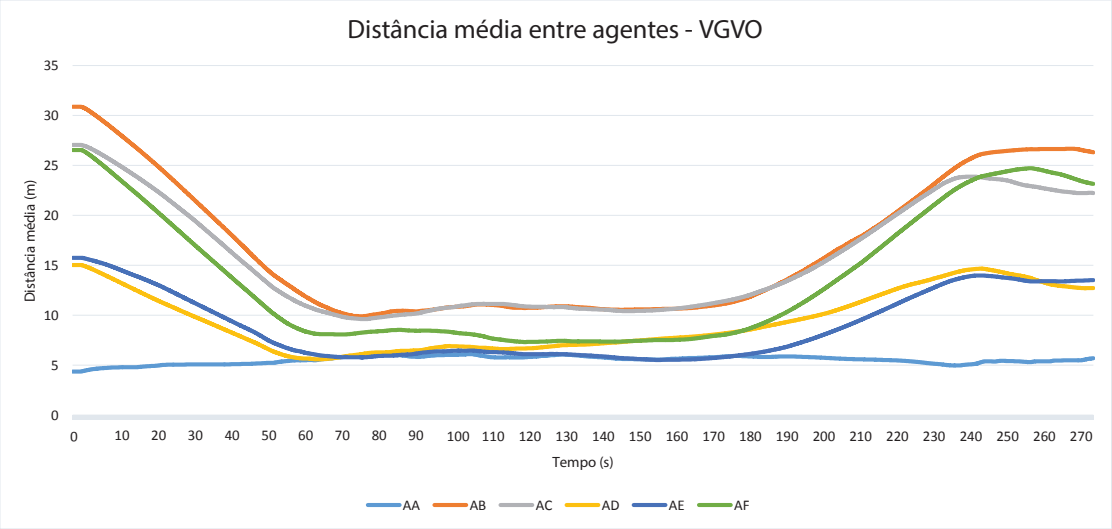


Figura 4.4: Distâncias médias entre agentes durante a navegação com a utilização do algoritmo VGVO.

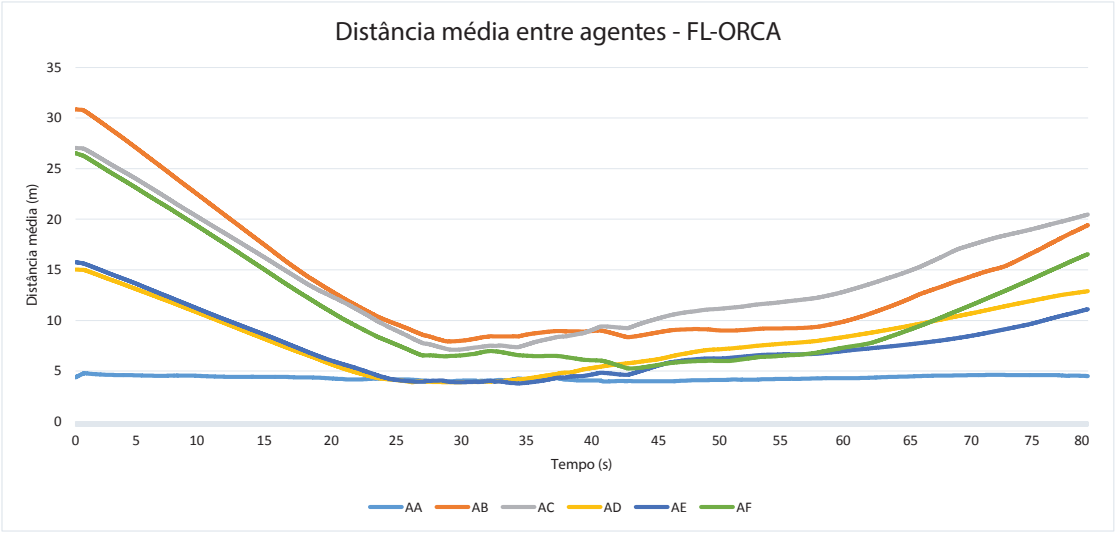


Figura 4.5: Distâncias médias entre agentes durante a navegação com a utilização do algoritmo FL-ORCA.

com excessão do cenário no qual existem grupos de tamanhos distintos, já que uma análise de distâncias médias em grupos de tamanhos diferentes não faria sentido.

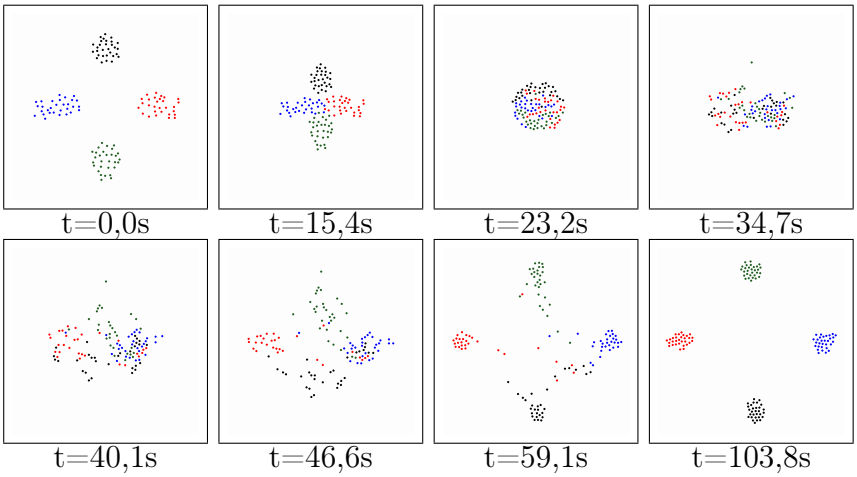
4.2.2 Quantidade variável de agentes

O primeiro grupo de experimentos contemplou cenários com enxames formados por grupos de mesmo tamanho e um número fixo de grupos, variando-se apenas a quantidade de agentes em cada cenário. A Figura 4.6 ilustra uma execução dos 3 algoritmos

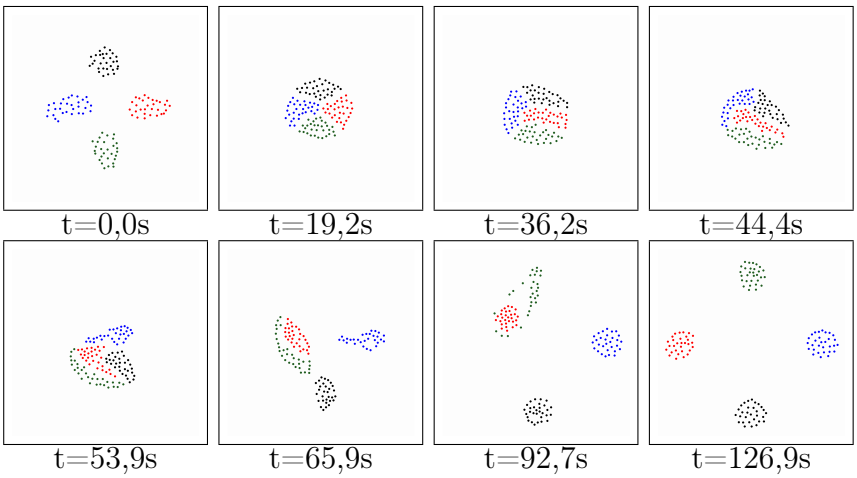
utilizados nos experimentos num cenário composto por 4 grupos de 30 agentes. Como pode ser visto, o algoritmo ORCA não foi capaz de manter a segregação dos grupos durante a navegação, o que já era esperado devido ao fato desse algoritmo não ter sido desenvolvido com esse intuito. O algoritmo VGVO, apesar de ter sido desenvolvido para manter a segregação enquanto os agentes navegam, também não foi capaz de cumprir esse objetivo. Essa incapacidade está relacionada com o raio de visão utilizado nas execuções mostradas, que foi de 3 metros. Quando utilizado o valor de 10 metros para o raio de visão do robô, na maioria da vezes, o algoritmo VGVO, assim como o FL-ORCA, conseguiu manter a segregação durante toda a navegação dos agentes.

As Figuras 4.7 e 4.8 mostram os tempos médios de execução dos algoritmos para agentes com raio de visão de 3 e 10 metros, respectivamente. É possível observar que com o aumento do raio de visão o tempo médio necessário para que os agentes alcançassem seus objetivos utilizando o algoritmo ORCA diminui, o que pode ser justificado pelo fato de cada agente possuir mais informação para tomar suas decisões. Com mais informação disponível e por não ser obrigado a escolher caminhos por onde todo o seu grupo possa navegar em segurança, um agente é capaz de prever situações de congestionamento e contorná-las, individualmente, com antecedência, o que leva à geração de trajetórias mais eficientes. Em contrapartida, os tempos de execução dos algoritmos VGVO e FL-ORCA aumentaram quando o raio de visão utilizado foi maior. Isso é justificado pelo fato de cada agente considerar um número maior de vizinhos, tanto nos cálculos relativos às componentes do *flocking* quanto nos cálculos dos semiplanos definidos para o controle de colisão, o que aumenta o processamento necessário para definição das novas velocidades. Além disso, um agente é obrigado a escolher velocidades menores que o mantenham próximo dessa quantidade maior de vizinhos e permitam que o grupo se movimente mantendo-se unido. No caso do FL-ORCA o maior número de vizinhos faz com que a componente do *flocking* relacionada à coesão do grupo aumente e, consequentemente, que a força resultante que atrai o agente até o seu objetivo diminua. Dessa forma, como os algoritmos VGVO e FL-ORCA passam a selecionar velocidades menores, o tempo necessário para que os agentes alcancem seus objetivos aumenta.

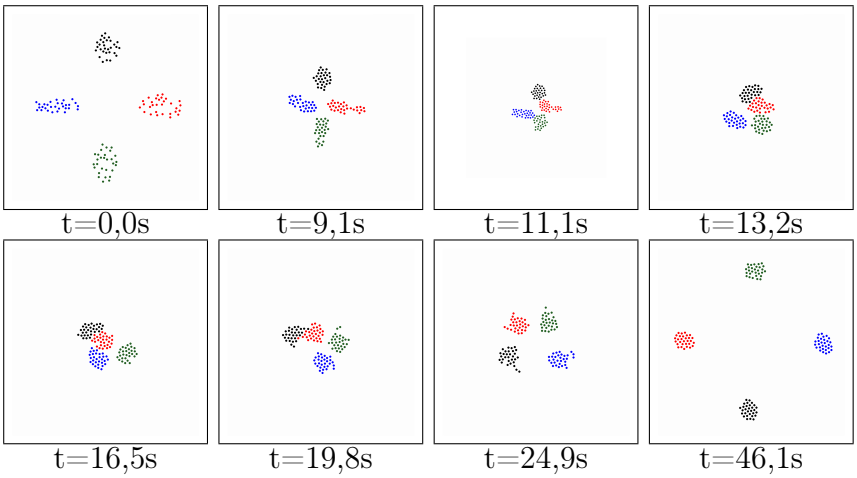
É possível notar que o algoritmo FL-ORCA se mostrou mais eficiente que os demais algoritmos em todos os cenários utilizados. Além disso, destaca-se o fato de a diferença entre os tempos de execução dos algoritmos tornar-se mais expressiva à medida que o número de agentes em cada grupo aumenta. Nota-se também que a taxa de crescimento do tempo de execução do algoritmo FL-ORCA é bem menor que a dos outros dois algoritmos, especialmente em relação ao VGVO.



(a) Execução do algoritmo ORCA.



(b) Execução do algoritmo VGVO.



(c) Execução do algoritmo FL-ORCA.

Figura 4.6: Execução dos algoritmos em um cenário composto por 4 grupos de 30 agentes com 3 metros de raio de visão.

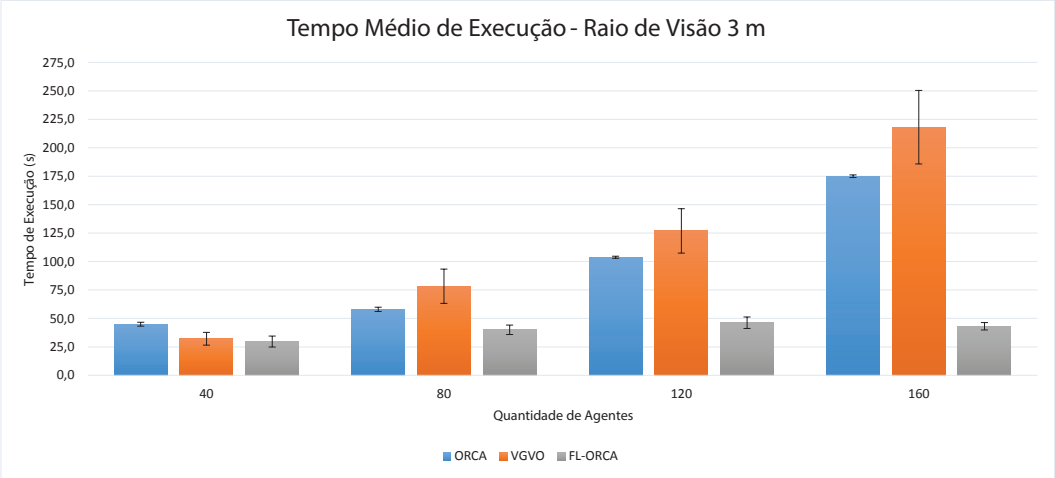


Figura 4.7: Tempos médios de execução dos algoritmos para enxames formados por agentes com raio de visão de 3 metros.

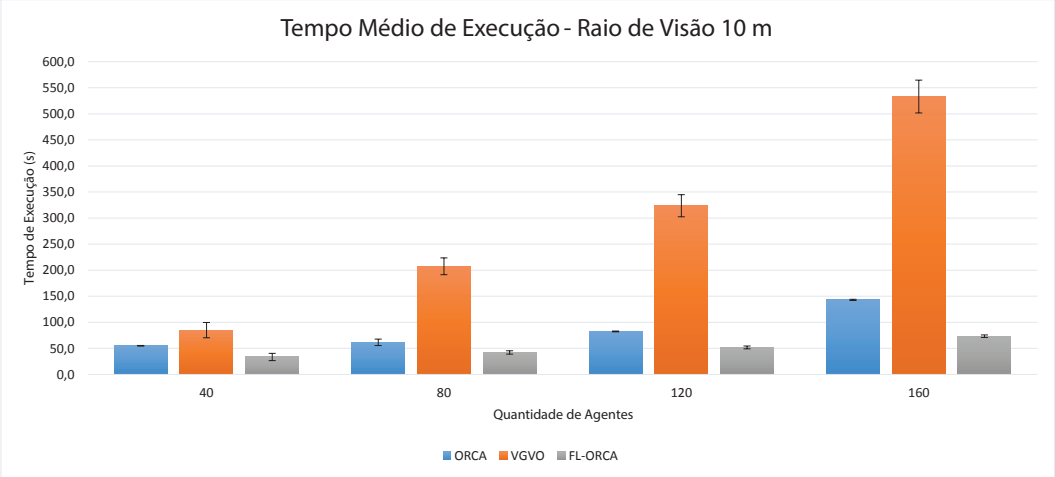


Figura 4.8: Tempos médios de execução dos algoritmos para enxames formados por agentes com raio de visão de 10 metros.

As Figuras 4.9 e 4.10 apresentam os dados de conectividade dos agentes. Por não possuir nenhuma regra que busque manter a segregação e, consequentemente, a conectividade dos grupos, o algoritmo ORCA apresentou os piores resultados nessa métrica. Esse algoritmo foi capaz de manter a conectividade de todos os agentes durante menos de 40% do tempo total de navegação, quando utilizado o raio de visão de 3 metros, em todos os cenários utilizados. Já os algoritmos VGVO e FL-ORCA obtiveram resultados superiores a 75% em todos os cenários. Nota-se que aumento do raio de visão possibilitou que esses dois algoritmos alcançassem resultados próximos, ou iguais, a 100% de conectividade nos 4 cenários. Porém, é importante ressaltar que

os resultados do algoritmo FL-ORCA se mostraram maiores ou iguais aos do VGVO em todos os cenários, independentemente do raio de visão utilizado nos agentes.

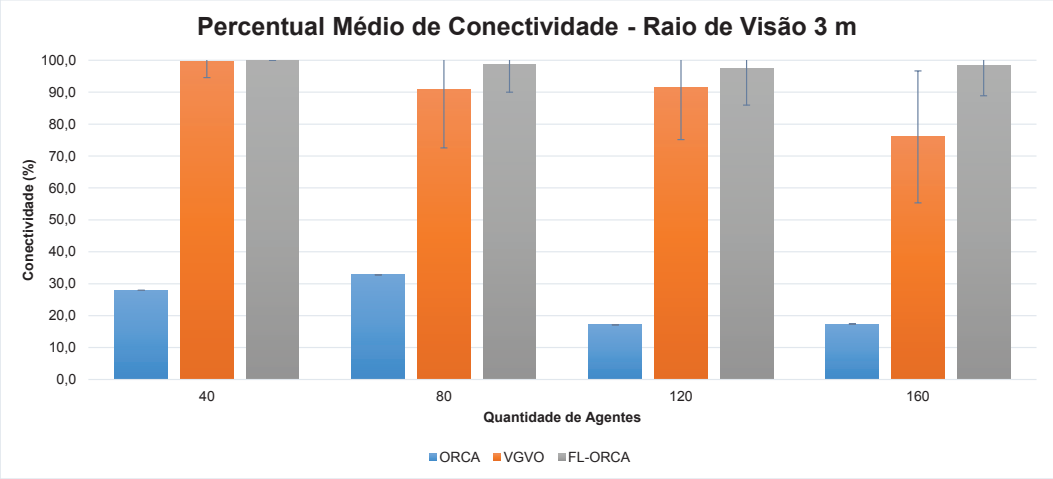


Figura 4.9: Conectividade média dos grupos em enxames formados por agentes com raio de visão de 3 metros.

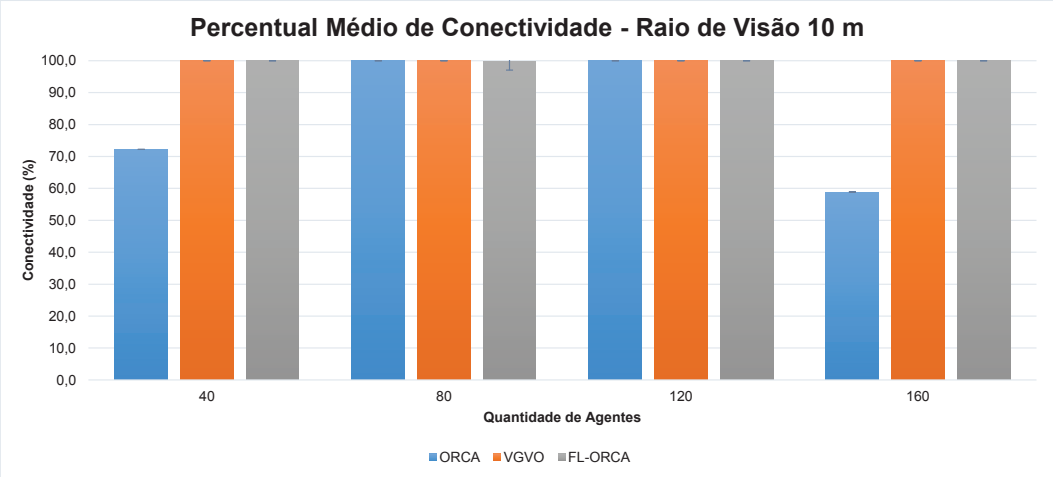


Figura 4.10: Conectividade média dos grupos em enxames formados por agentes com raio de visão de 10 metros.

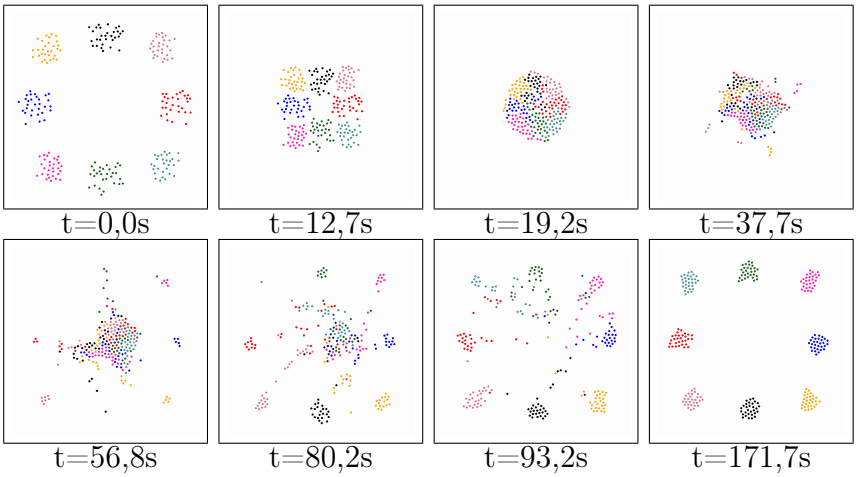
4.2.3 Quantidade variável de grupos

Para investigar o impacto do número de grupos nos algoritmos avaliados, criou-se 4 cenários com diferentes quantidades de grupos em cada um deles. Em todos esses cenários foram utilizados grupos formados por 30 agentes. A Figura 4.11 exibe um exemplo do comportamento observado durante a execução dos 3 algoritmos utilizados nos experimentos num cenário com um enxame formado por 8 grupos. Novamente,

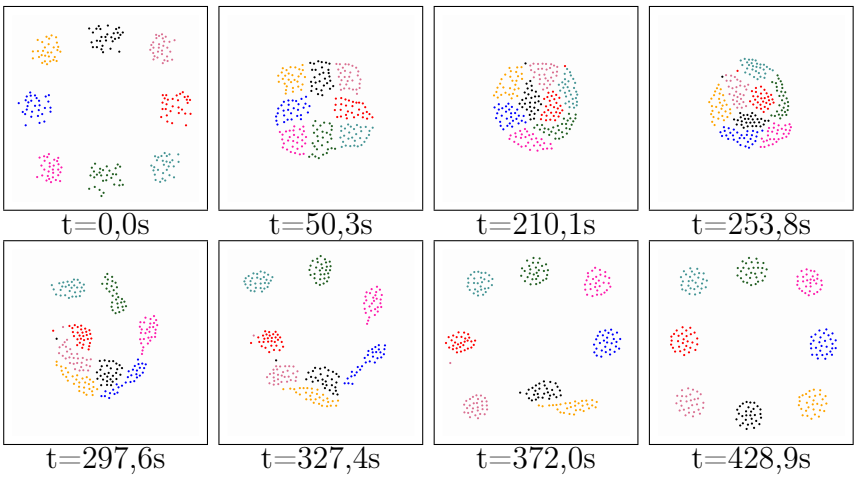
notou-se a incapacidade do algoritmo ORCA em manter a segregação dos grupos, permitindo uma grande fusão entre os grupos que formavam o enxame. O algoritmo VGVO também não foi capaz de manter os grupos segregados. Porém, somente 3 agentes, sendo cada um de um grupo diferente, se separaram de seus grupos durante o experimento ilustrado na referida figura. Por fim, o FL-ORCA conseguiu executar a navegação de todos os grupos sem permitir que os grupos se misturassem durante o experimento.

Com a utilização de várias quantidades de grupos os resultados encontrados foram semelhantes aos obtidos nos experimentos mostrados na seção anterior. Como mostrado nas Figuras 4.12 e 4.13, o algoritmo ORCA alcançou melhores resultados quando utilizado o raio de visão de 10 metros, além de apresentar uma taxa de crescimento do tempo médio de execução bem menor nessa situação. O FL-ORCA apresentou resultados melhores que os outros dois algoritmos em todos os cenários nos quais os agentes possuíam 3 metros de raio de visão, chegando a gastar cerca de 20% do tempo médio necessário pra que o VGVO alcançasse os objetivos dados aos grupos quando o enxame estava dividido em 8 grupos. Com agentes que possuíam raio de visão de 10 metros, o FL-ORCA obteve resultados bem parecidos com os do ORCA, sendo melhor nos cenários com 4 e 6 grupos e um pouco pior nos outros dois cenários. As diferenças entre os tempos médios de execução do FL-ORCA e do VGVO ficaram ainda mais evidentes quando o agente era capaz de enxergar 10 metros ao seu redor, chegando a ser, em média, quase sete vezes mais rápido que o concorrente nos cenários com 8 grupos. Outro fato que podemos destacar, é o desvio padrão apresentado pelos algoritmos. Enquanto o ORCA e o FL-ORCA apresentaram valores baixos de desvio padrão, o algoritmo VGVO chegou a ter um desvio de 181,2 segundos em um cenário no qual o tempo médio de execução foi de 271,1 segundos. Essa variação específica foi causada por uma simulação na qual o algoritmo levou mais de 2000 segundos para que os grupos alcançassem seus objetivos. É possível notar que o desvio padrão do algoritmo VGVO foi significativamente maior que os obtidos com os outros dois algoritmos em praticamente todos os cenários analisados, esse comportamento se deve ao fato de que, quando ocorre uma mistura entre grupos que estejam navegando com o VGVO, uma pequena quantidade de agentes de um determinado grupo fica bloqueado, ou até mesmo rodeado, por agentes de outros grupos. Nessas situações, os agentes que se encontram em menor número apresentam dificuldade em calcular uma trajetória que contorne o grupo que os impede de se deslocar em direção ao seu objetivo fazendo o tempo de navegação aumentar consideravelmente.

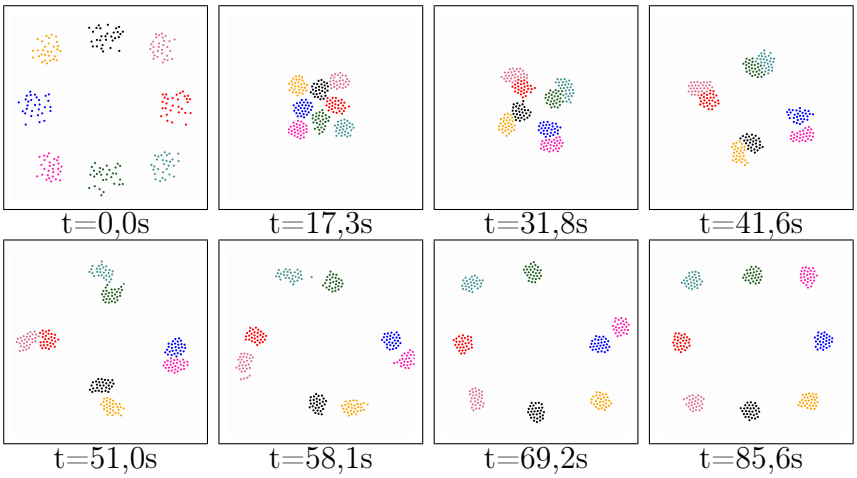
As Figuras 4.14 e 4.15 mostram os resultados de conectividade média para os cenários com diferentes quantidades de grupos. Observa-se que, para os três algorit-



(a) Execução do algoritmo ORCA.



(b) Execução do algoritmo VGVO.



(c) Execução do algoritmo FL-ORCA.

Figura 4.11: Execução dos algoritmos em um cenário composto por 8 grupos de 30 agentes com 3 metros de raio de visão.

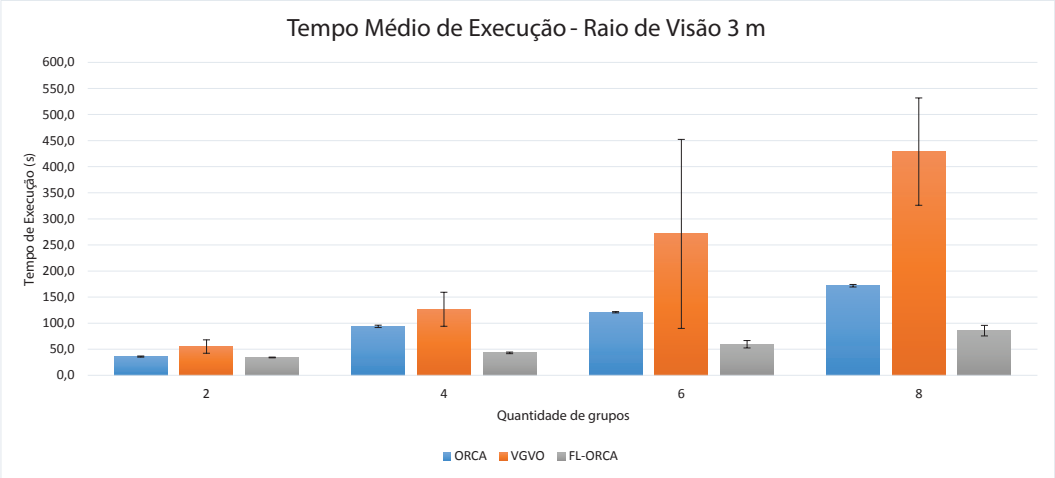


Figura 4.12: Tempos médios de execução dos algoritmos para enxames formados por agentes com raio de visão de 3 metros.

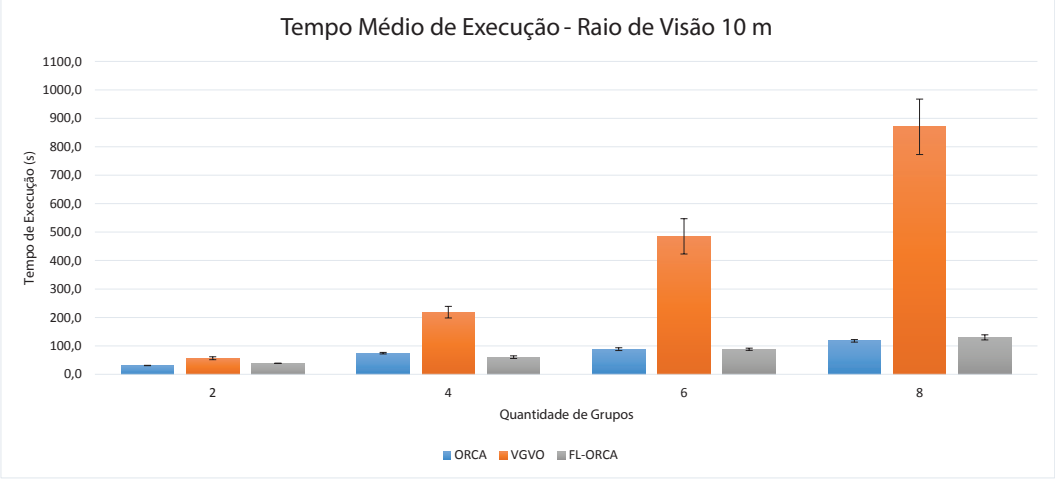


Figura 4.13: Tempos médios de execução dos algoritmos para enxames formados por agentes com raio de visão de 10 metros.

mos utilizados, o aumento do número de grupos fez com que o percentual médio de conectividade diminuísse quando o raio de visão utilizado foi de 3 metros. Por outro lado, quando o raio utilizado foi de 10 metros, os percentuais dos três algoritmos aumentaram em relação às execuções com o raio menor, sendo que os algoritmos VGVO e FL-ORCA alcançaram médias de 100% em todos os cenários testados com o maior raio de visão.

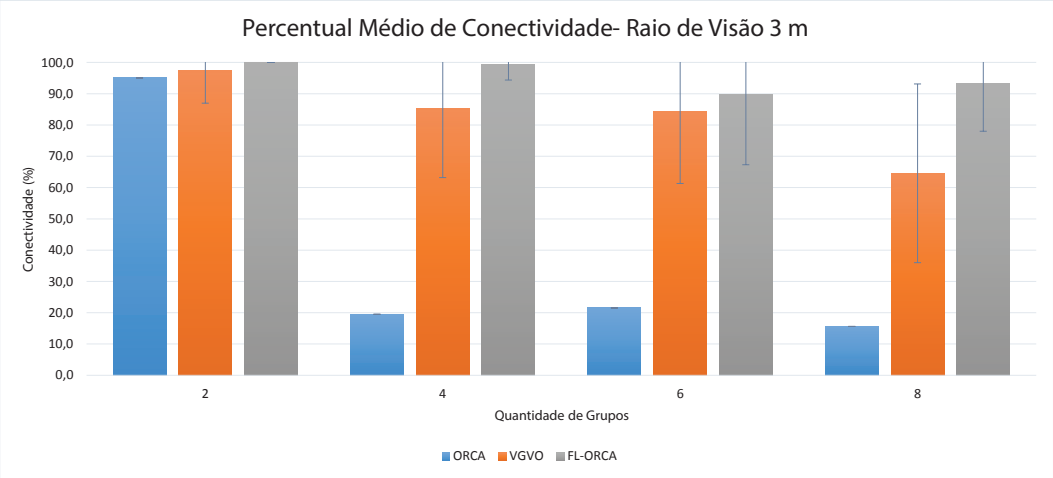


Figura 4.14: Conectividade média dos grupos em enxames formados por agentes com raio de visão de 3 metros.

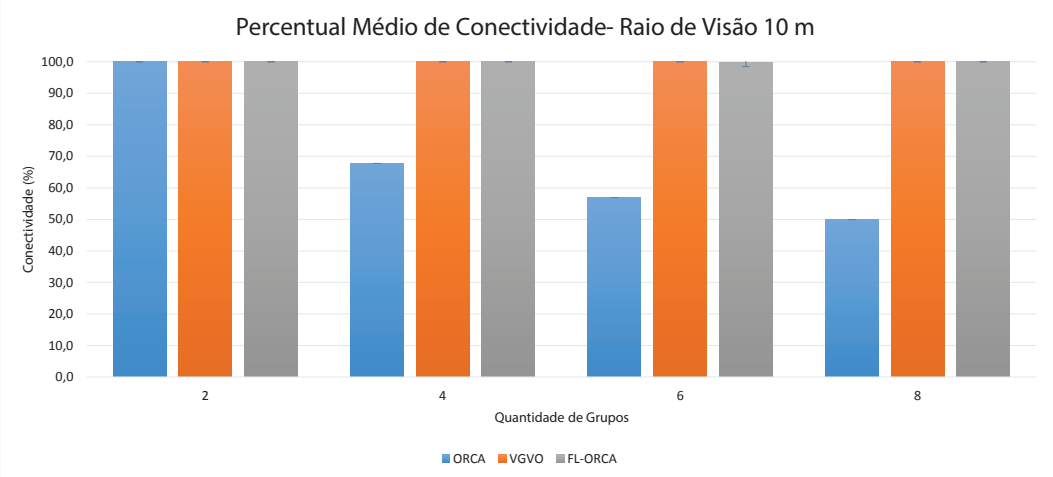
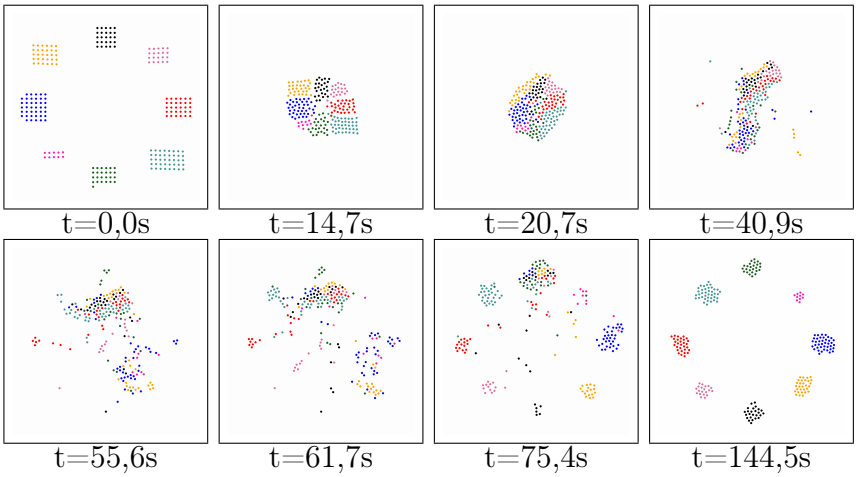


Figura 4.15: Conectividade média dos grupos em enxames formados por agentes com raio de visão de 10 metros.

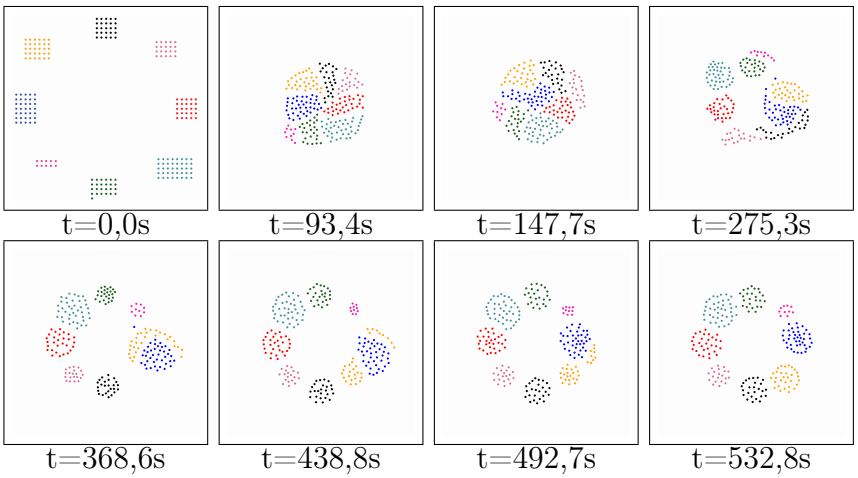
4.3 Quantidade variável de agentes em cada grupo

Uma característica que difere o FL-ORCA de outros algoritmos é a capacidade de manter a segregação de grupos de tamanhos diferentes. Normalmente os algoritmos propostos para esse fim supõem que o enxame seja divido em grupos de mesmo tamanho, o que pode restringir o seu uso em ambientes reais onde essa restrição não seja obedecida. A Figura 4.16 ilustra uma execução dos algoritmos em um cenário no qual os grupos têm tamanhos 10, 20, 25, 30 e 40 agentes. Ao contrário do ORCA, o algoritmo FL-ORCA manteve os grupos de agentes segregados durante toda a sua navegação, comprovando a possibilidade de seu uso em cenários dessa natureza.

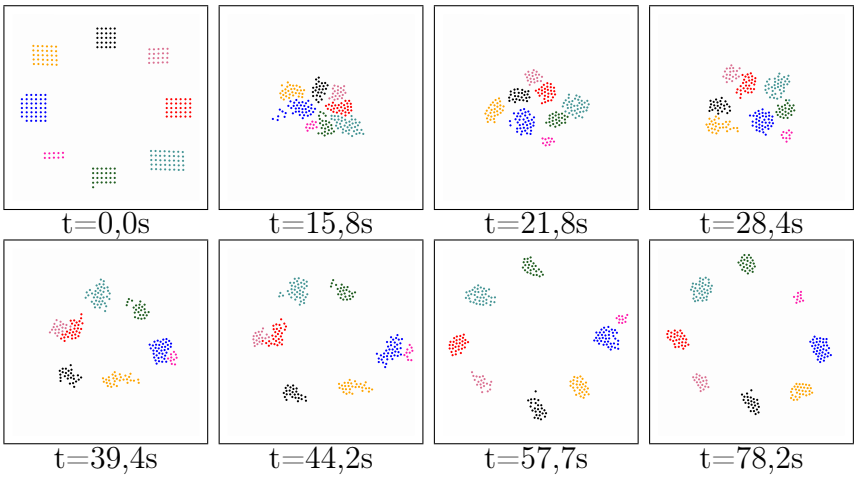
Os tempos médios de execução do FL-ORCA também foram melhores que os



(a) Execução do algoritmo ORCA.



(b) Execução do algoritmo VGVO.



(c) Execução do algoritmo FL-ORCA.

Figura 4.16: Execução dos algoritmos em um cenário composto por 8 grupos com número variado de agentes com 3 metros de raio de visão.

alcançados pelo ORCA no cenário com grupos de tamanhos diferentes, conforme mostra a Figura 4.17. Como os agentes controlados pelo FL-ORCA executam manobras para contornar o congestionamento que se forma no centro do cenário com o intuito de manter os grupos unidos, as trajetórias executadas por eles apresentam menos conflitos entre os grupos levando a uma diminuição do tempo necessário para finalizar a navegação.

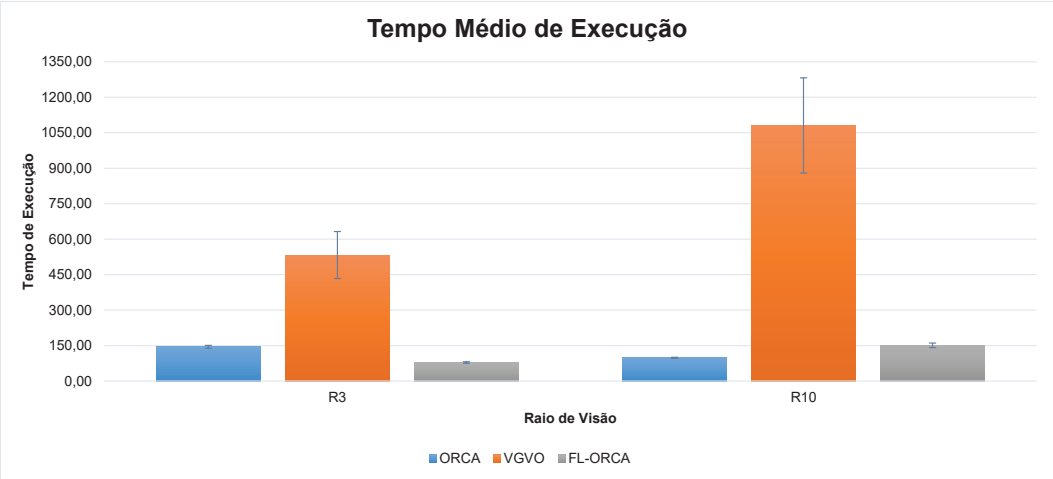


Figura 4.17: Tempos médios de execução dos algoritmos.

A Figura 4.18 apresenta os resultados de conectividade dos grupos para o cenário avaliado. É possível notar que por não existir uma regra explícita que tenta manter a segregação de grupos no algoritmo ORCA, os resultados alcançados por ele são bastante baixos, alcançando um resultado de 15,01% quando o raio de visão dos agentes foi definido como 3 metros, e esse valor chegou a 62,74% quando utilizado um raio de visão de 10 metros. Como esperado, os algoritmos VGVO e FL-ORCA alcançaram resultados melhores que o ORCA. Porém, o algoritmo VGVO se mostrou mais sensível à variação do raio de visão dos agentes. Nos experimentos nos quais esse parâmetro foi definido como 3 metros o VGVO obteve um resultado pouco superior a 60%, já o FL-ORCA conseguiu um resultado de 100% independente do raio de visão utilizado.

Após avaliar todas as simulações feitas, uma análise conjunta dos dados de tempo médio de execução e percentual de conectividade dos agentes nos permite concluir que os valores para o raio de visão devem ser escolhidos pensando-se em qual o comportamento esperado do sistema, caso seja necessário garantir altos níveis de conectividade, sacrificando-se a eficiência do sistema em relação ao tempo médio de execução, deve-se usar valores maiores para o raio de visão. Já se o quesito mais importante for a garantia de tempos de execução menores, permitindo-se que a restrição de segregação

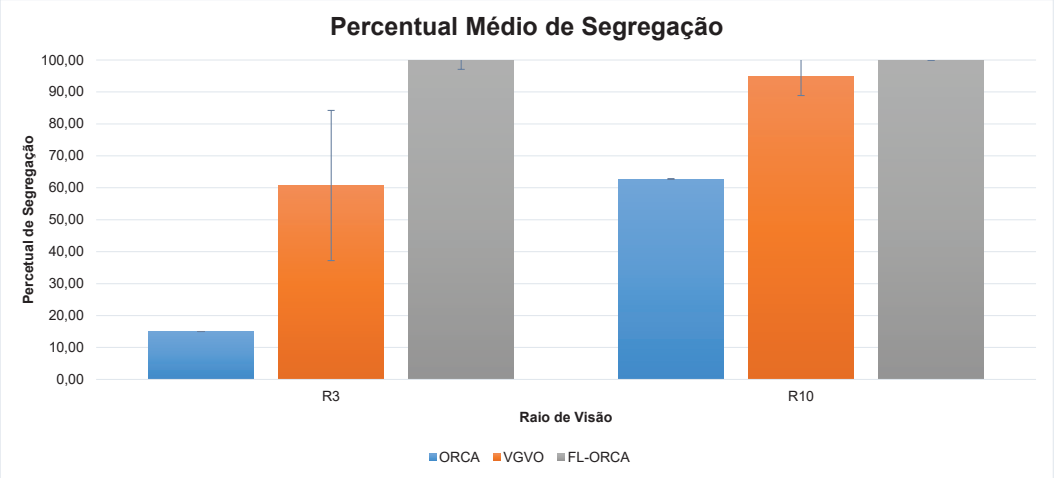


Figura 4.18: Conectividade média dos grupos.

dos grupos seja violada em alguns momentos, o raio de visão dos agentes deve ser mais curto.

4.4 Experimentos Reais

Para validar a abordagem desenvolvida e demonstrar sua viabilidade em cenários reais, foram realizados experimentos como prova de conceito utilizando-se robôs reais. Esse tipo de experimento é importante para avaliar o impacto que as incertezas encontradas nas aplicações reais, tais como dados obtidos através de sensores ou ações executadas pelos atuadores dos robôs, têm sobre o algoritmo proposto. A Figura 4.19 apresenta um robô *e-puck* [Mondada et al., 2009], que é robô diferencial de pequeno porte muito usado em projetos que utilizem enxames de robôs [Santos & Chaimowicz, 2011; Chen et al., 2012; Santos et al., 2014a; Santos & Chaimowicz, 2014]. Os robôs *e-puck* possuem oito sensores infravermelhos que podem ser utilizados para sensoriamento de proximidade e uma interface *Bluetooth* que permite que um robô se comunique com outros robôs ou com um computador. Os experimentos foram realizados com seis robôs controlados através do *framework* ROS.

Como dito anteriormente, os algoritmos utilizados nesse trabalho precisam ter acesso às posições dos agentes. Apesar dos *e-pucks* possírem sensores infravermelhos que podem ser utilizados para sensoriamento de proximidade, foi necessária a utilização de outro arcabouço que permitisse o acesso às posições de cada agente devido às limitações que os sensores nativos dos robôs apresentam. Para isso, foi utilizado o arcabouço para localização de robôs desenvolvido por Garcia et al. [2007]. Esse arcabouço utiliza marcadores individuais para identificar cada agente e um conjunto de câmeras

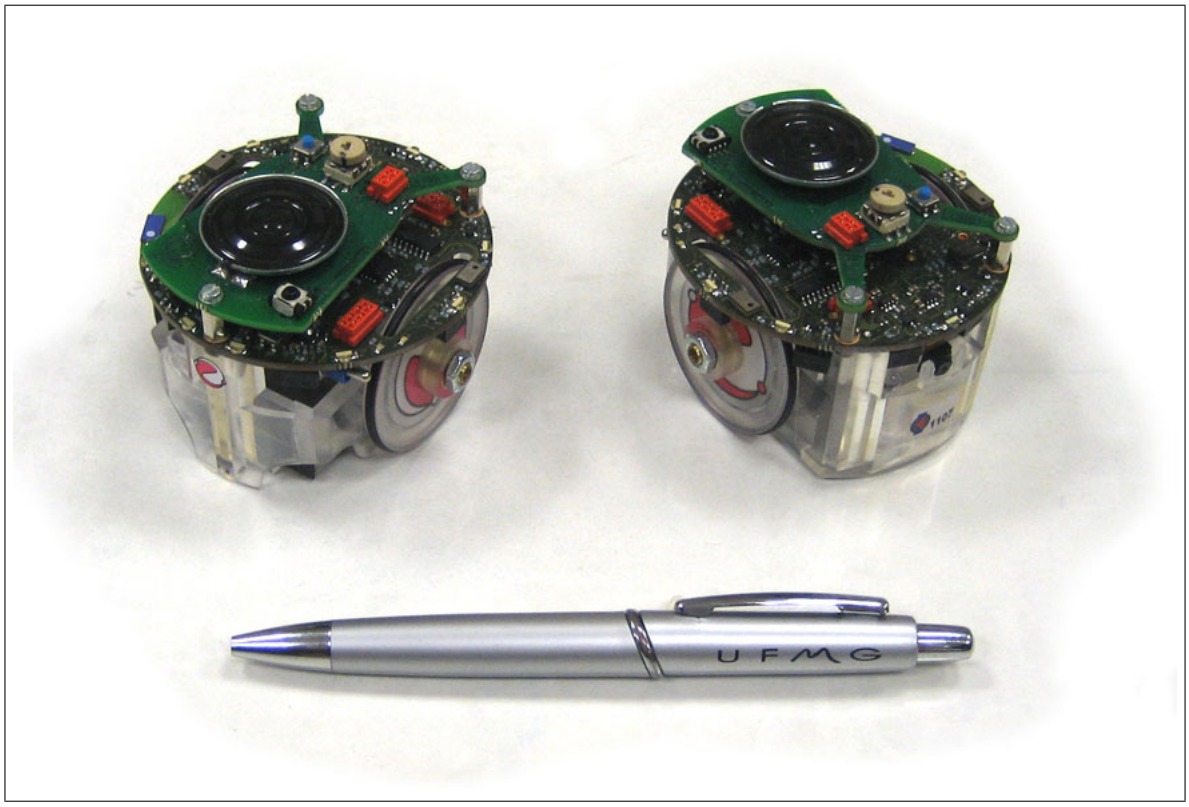
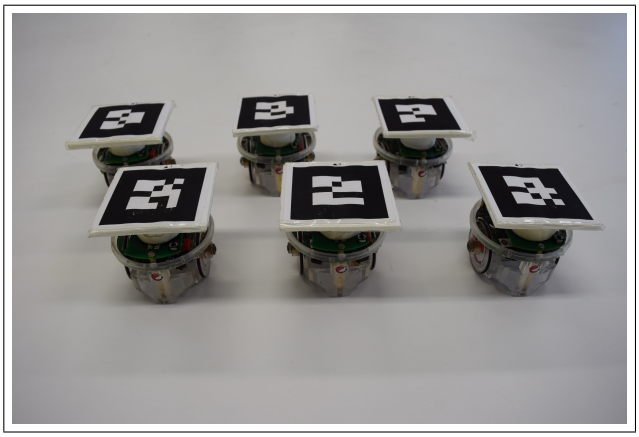


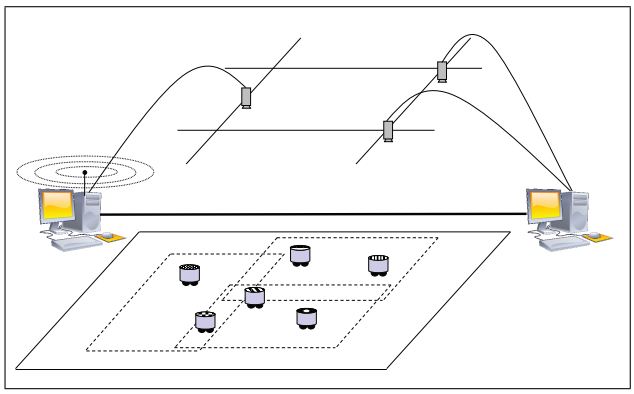
Figura 4.19: Robôs *e-puck*.

posicionadas sobre o ambiente onde os experimentos são executados, como apresentado na Figura 4.20. Para a execução dos experimentos, as entradas de controle para cada agente são calculadas de acordo com o algoritmo proposto com base nas posições capturadas pelas câmeras e transmitidas para o enxame através das conexões *Blueto-oth*. O algoritmo proposto foi, originalmente, desenvolvido para robôs holonômicos, por isso, após o cálculo das velocidades que cada agente deveria desenvolver utilizou-se a abordagem apresentada por De Luca et al. [2000] para converter essas velocidades em valores que obedecessem às restrições não-holonômicas dos robôs *e-puck*.

É importante ressaltar que, devido à quantidade limitada de robôs disponível, foram avaliados cenários formados por 2 ou 3 grupos. Por isso, os experimentos executados com os *e-pucks* têm o intuito de ser apenas uma prova de conceito da abordagem desenvolvida. Nos experimentos com dois grupos distintos, cada grupo era formado por 3 ou 4 agentes. Já nos experimentos com 3 grupos, cada um deles possuía 2 robôs. Além desses experimentos, utilizou-se também um cenário no qual os grupos possuíam tamanhos distintos, sendo cada grupo formado por 1, 2 ou 4 agentes. Como mencionado anteriormente, foi necessário converter as velocidades calculadas para cada agente para valores que respeitassem as restrições não-holonômicas dos robôs *e-puck*. Com a



(a) Conjunto de robôs *e-puck* com os identificadores utilizados para localização.



(b) Diagrama do arcabouço de localização utilizado nos experimentos. [Garcia et al., 2007]

Figura 4.20: Robôs *e-puck* e arcabouço de localização utilizados para execução dos experimentos.

conversão do vetor velocidade para velocidades lineares e angulares, os robôs apresentaram uma oscilação indesejada na sua orientação. A velocidade na qual os robôs giravam em seus próprios eixos para corrigir a sua orientação era muito alta, isso fazia com que o robô girasse mais do que o necessário para corrigir a sua orientação. Para corrigir esse problema foi necessário limitar a velocidade máxima dos agentes. Dessa forma, foi definido que cada robô não deveria ultrapassar a velocidade de 0,05 m/s.

A Figura 4.21 apresenta alguns *snapshots* de uma execução do algoritmo FL-ORCA em um cenário formado por dois grupos de 3 agentes cada. É possível notar que os robôs apresentam comportamentos similares aos observados durante as simulações. Ao detectar um grupo diferente do seu, os agentes passam a utilizar as entradas de controle que os levam a se deslocar para a direita ou a seguir outros agentes que possuam visão livre para o objetivo do grupo. Nesse experimento o tempo total de navegação foi de aproximadamente 13 minutos e durante toda a execução os agentes

se mantiveram segregados.

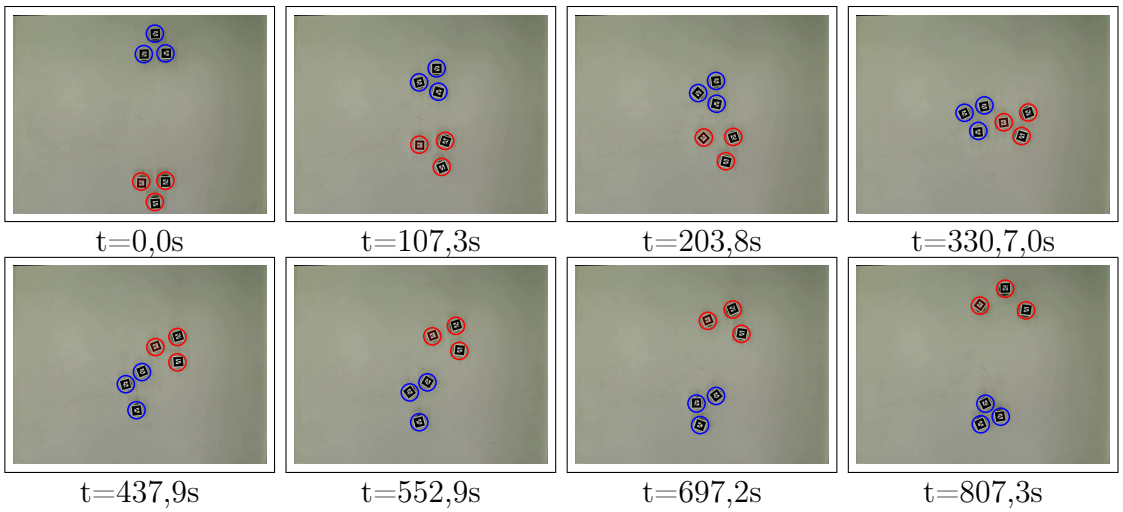


Figura 4.21: Execução do algoritmo FL-ORCA com 6 robôs *e-puck* divididos em 2 grupos de 3 agentes.

A Figura 4.22 apresenta uma execução do FL-ORCA com 3 grupos de dois agentes. Assim como no experimento anterior, é possível notar que os agentes conseguiram navegar pelo ambiente mantendo a segregação dos grupos. Após chegarem à região central do ambiente, os agentes começaram a se delocar para a direita até alcançarem posições de onde tivessem visão livre para seus respectivos objetivos.

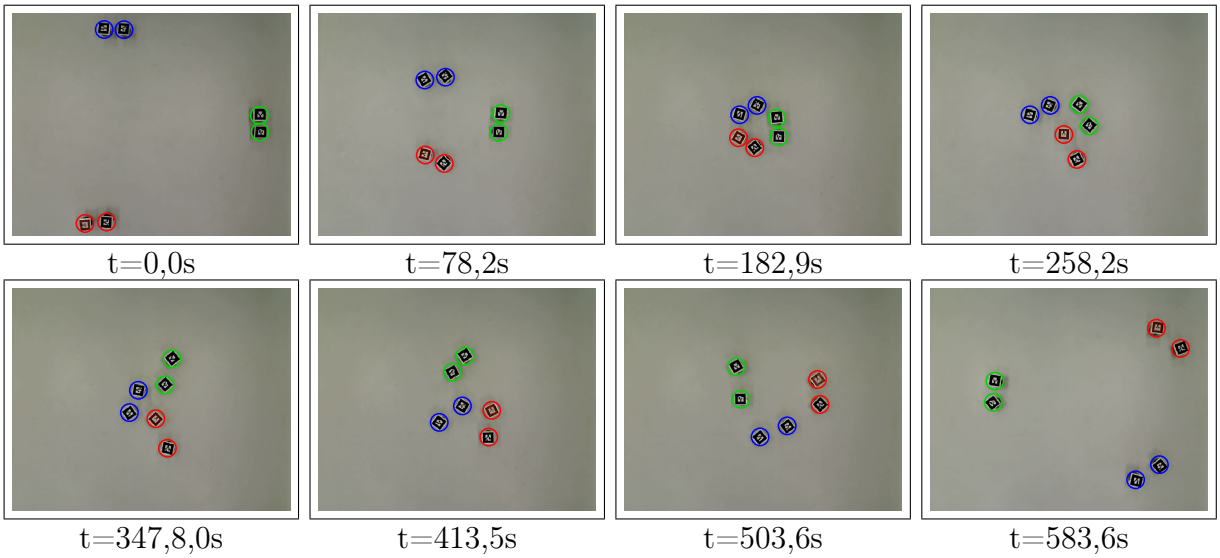


Figura 4.22: Execução do algoritmo FL-ORCA com robôs *e-puck* divididos em 3 grupos de 2 agentes.

Capítulo 5

Conclusão e Trabalhos Futuros

O uso de sistemas multi-robôs em diversas situações pode trazer vantagens sobre sistemas que utilizam um único robô. Nesse contexto, enxames de robôs são sistemas que empregam uma grande quantidade de agentes simples para executar diferentes tipos de tarefas.

Por ser um fenômeno natural comumente utilizado como mecanismo de ordenação em vários sistemas biológicos, e por ser útil em diversos tipos de tarefas e cenários, a segregação é um fenômeno que têm sido bastante estudado recentemente. Por isso, o principal objetivo desse trabalho foi desenvolver uma abordagem capaz de manter agentes pertencentes a um mesmo grupo próximos uns aos outros, enquanto se mantém afastados de outros grupos de agentes durante toda a sua navegação.

Para solucionar esse problema, foi proposta uma metodologia descentralizada com o intuito de executar a navegação de grupos heterogêneos de robôs enquanto mantém a segregação entre eles. Essa metodologia consiste em estender o algoritmo ORCA [van den Berg et al., 2011] com uma versão modificada dos comportamentos clássicos de *flocking* [Reynolds, 1987]. Com o uso das regras do *flocking*, foi possível calcular uma velocidade que mantém a segregação entre os agentes e esta velocidade foi utilizada como preferencial pelo ORCA garantindo trajetórias livres de colisões entre os agentes.

Para avaliar a aplicabilidade do algoritmo proposto, foi executada uma série de experimentos em diversos cenários distintos. Os resultados obtidos com os experimentos mostraram que o algoritmo proposto é uma alternativa efetiva para manter diferentes grupos de agentes segregados durante sua navegação em um ambiente compartilhado. Quando comparado com o algoritmo VGVO, o algoritmo FL-ORCA se mostrou menos sensível ao raio de visão dos agentes, o que pode ser destacado como uma característica importante já que em ambientes reais os sensores dos robôs costumam ser bastante limitados. Além disso, os resultados encontrados com o algoritmo

proposto foram melhores que os obtidos pelos outros dois algoritmos usados nos experimentos. O algoritmo foi capaz de calcular trajetórias qua demandam menos tempo de navegação a partir da redução de conflitos entre as trajetórias de grupos diferentes, reduzindo o tempo necessário para conclusão da navegação e aumentando o nível de segregação entre os grupos quando comparado com outros algoritmos.

Devido à natureza do problema da navegação segregada, acredita-se que o algoritmo FL-ORCA pode ser utilizado como uma estratégia para a resolução de subtarefas em missões mais complexas, como controle de formação em enxames heterogêneos, coleta de recursos por grupos distintos de agentes, transporte cooperativo de objetos, dentre outras. O algoritmo pode ser utilizado também em outros tipos de problemas como inteligência artificial para agentes virtuais em jogos digitais ou animações gráficas.

Apesar da abordagem proposta se mostrar como uma boa alternativa para o problema de navegação segregada, existem ainda alguns pontos que podem ser estudados com o intuito de melhorar os resultados obtidos com o uso do algoritmo ou ainda expandir suas possibilidades de uso. Possíveis trabalhos futuros incluem o desenvolvimento e avaliação de novas métricas para a segregação de grupos distintos, como avaliar a área de interseção entre os envoltórios convexos formados pelos agentes de cada grupo ou avaliar se uma versão determinística do algoritmo FL-ORCA oferece ganho em relação ao que foi proposto nesse trabalho. Além disso, investigar outras funções potenciais que possam ser utilizadas na definição das componentes do *flocking* para agentes com um raio de visão ainda menor do que o utilizado nesse trabalho, permitiria a utilização do FL-ORCA em sistemas mais fiéis às aplicações reais, nas quais a área de percepção dos agentes é bastante reduzida e susceptível a ruídos.

Referências Bibliográficas

Alonso-Mora, J.; Breitenmoser, A.; Rufli, M.; Beardsley, P. A. & Siegwart, R. (2010). Optimal reciprocal collision avoidance for multiple non-holonomic robots. Em *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pp. 203--216. Springer.

Bahgeci, E. & Sahin, E. (2005). Evolving aggregation behaviors for swarm robotic systems: a systematic case study. Em *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS)*, pp. 333–340.

Barca, J. C. & Sekercioglu, Y. A. (2013). Swarm robotics reviewed. *Robotica*, 31:345--359. ISSN 1469-8668.

Borenstein, J. & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288. ISSN 1042-296X.

Bruni, L.; Colombo, A. & Vecchio, D. D. (2013). Robust multi-agent collision avoidance through scheduling. Em *52nd IEEE Conference on Decision and Control*, pp. 3944–3950. ISSN 0191-2216.

Cao, Y. U.; Fukunaga, A. S. & Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7--27.

Chang, D. E.; Shadden, S. C.; Marsden, J. E. & Olfati-Saber, R. (2003). Collision avoidance for multiple agent systems. Em *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pp. 539–543 Vol.1. ISSN 0191-2216.

Chen, J.; Gauci, M.; Price, M. J. & Groß, R. (2012). Segregation in swarms of e-puck robots based on the brazil nut effect. Em *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pp. 163--170, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

De Luca, A.; Oriolo, G. & Vendittelli, M. (2000). Stabilization of the unicycle via dynamic feedback linearization. Em *6th IFAC Symposium on Robot Control*, pp. 397--402.

Dias, M. B.; Zlot, R.; Kalra, N. & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270. ISSN 0018-9219.

Dorigo, M.; Floreano, D.; Gambardella, L. M.; Mondada, F.; Nolfi, S.; Baaboura, T.; Birattari, M.; Bonani, M.; Brambilla, M.; Brutschy, A.; Burnier, D.; Campo, A.; Christensen, A. L.; Decugniere, A.; Caro, G. D.; Ducatelle, F.; Ferrante, E.; Forster, A.; Gonzales, J. M.; Guzzi, J.; Longchamp, V.; Magnenat, S.; Mathews, N.; de Oca, M. M.; O'Grady, R.; Pinciroli, C.; Pini, G.; Retornaz, P.; Roberts, J.; Sperati, V.; Stirling, T.; Stranieri, A.; Stutzle, T.; Trianni, V.; Tuci, E.; Turgut, A. E. & Vaussard, F. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71. ISSN 1070-9932.

Dorigo, M.; Trianni, V.; Şahin, E.; Groß, R.; Labella, T. H.; Baldassarre, G.; Nolfi, S.; Deneubourg, J.-L.; Mondada, F.; Floreano, D. et al. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2-3):223--245.

Ferreira Filho, E. B. & Pimenta, L. C. A. (2015). Segregating multiple groups of heterogeneous units in robot swarms using abstractions. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 401--406.

Fiorini, P. & Shiller, Z. (1993). Motion planning in dynamic environments using the relative velocity paradigm. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 560--565. IEEE.

Fiorini, P. & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760--772.

Forster, M. (2013). World war z. Paramount Pictures. Filme (116 min).

Fox, D.; Burgard, W.; Thrun, S. et al. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23--33.

Gal, O.; Shiller, Z. & Rimon, E. (2009). Efficient and safe on-line motion planning in dynamic environments. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 88--93. IEEE.

Garcia, R. F.; Shiroma, P. M.; Chaimowicz, L. & Campos, M. F. (2007). Um arcabouço para a localização de enxames de robôs. *Proc. of VIII SBAI*.

Groß, R.; Magnenat, S. & Mondada, F. (2009). Segregation in swarms of mobile robots based on the brazil nut effect. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 4349--4356. IEEE.

Guo, Y. & Parker, L. E. (2002). A distributed and optimal motion planning approach for multiple mobile robots. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pp. 2612--2619. IEEE.

He, L. & van den Berg, J. (2013). Meso-scale planning for multi-agent navigation. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2839--2844. IEEE.

Inácio, F. R.; Macharet, D. G. & Chaimowicz, L. (2016). United we move: Decentralized segregated robotic swarm navigation. Em *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics. Springer.

Jackson, P. (2003). The lord of the rings: The return of the king. New Line Cinema. Filme (201 min).

Jaimes, A.; Kota, S. & Gomez, J. (2008). An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav (s). Em *System of Systems Engineering, 2008. SoSE'08. IEEE International Conference on*, pp. 1--6. IEEE.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90--98.

Koren, Y. & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1398--1404. IEEE.

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

Lenaghan, S. C.; Wang, Y.; Xi, N.; Fukuda, T.; Tarn, T.; Hamel, W. R. & Zhang, M. (2013). Grand challenges in bioengineered nanorobotics for cancer therapy. *Biomedical Engineering, IEEE Transactions on*, 60(3):667--673.

Lewis, M. A. & Bekey, G. A. (1992). The behavioral self-organization of nanorobots using local rules. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, volume 2, pp. 1333–1338. ISSN 1.

Lozano-Pérez, T. & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560--570.

Marcolino, L. S. & Chaimowicz, L. (2009). Traffic control for a swarm of robots: Avoiding group conflicts. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 1949--1954. IEEE.

Martens, D.; Baesens, B. & Fawcett, T. (2011). Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1--42.

Masehian, E. & Katebi, Y. (2007). Robot motion planning in dynamic environments with moving obstacles and target. *International Journal of Mechanical Systems Science and Engineering*, 1(1):20--25.

Mohan, Y. & Ponnambalam, S. (2009). An extensive review of research in swarm robotics. Em *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 140--145. IEEE.

Mondada, F.; Bonani, M.; Raemy, X.; Pugh, J.; Cianci, C.; Klaptocz, A.; Magnenat, S.; Zufferey, J.-C.; Floreano, D. & Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. Em *In Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pp. 59--65.

Oster, G. F. & Wilson, E. O. (1978). *Caste and ecology in the social insects*. Princeton University Press.

Parker, L. E. (2000). *Current State of the Art in Distributed Autonomous Mobile Robotics*, pp. 3--12. Springer Japan, Tokyo.

ping Chen, Y. & yin Lin, Y. (2009). Controlling the movement of crowds in computer graphics by using the mechanism of particle swarm optimization. *Applied Soft Comput*, 9(3):1170--1176.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R. & Ng, A. Y. (2009). Ros: an open-source robot operating system. Em *ICRA workshop on open source software*, volume 3, p. 5.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. Em *ACM Siggraph Computer Graphics*, volume 21, pp. 25--34. ACM.

Şahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. Em Sahin, E. & Spears, W. M., editores, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pp. 10--20. Springer.

Santos, V. G.; Campos, M. F. M. & Chaimowicz, L. (2014a). *On Segregative Behaviors Using Flocking and Velocity Obstacles*, pp. 121--133. Springer Berlin Heidelberg, Berlin, Heidelberg.

Santos, V. G. & Chaimowicz, L. (2011). Hierarchical congestion control for robotic swarms. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 4372--4377. IEEE.

Santos, V. G. & Chaimowicz, L. (2014). Cohesion and segregation in swarm navigation. *Robotica*, 32(02):209--223.

Santos, V. G.; Pimenta, L. C. A. & Chaimowicz, L. (2014b). Segregation of multiple heterogeneous units in a robotic swarm. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1112–1117. ISSN 1050-4729.

Snape, J.; Guy, S. J.; Vembar, D.; Lake, A.; Lin, M. C. & Manocha11, D. (2012). Reciprocal collision avoidance and navigation for video games. Em *Game Developers Conference, San Francisco*.

Snape, J.; v. d. Berg, J.; Guy, S. J. & Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706. ISSN 1552-3098.

Snape, J.; Van den Berg, J.; Guy, S. J. & Manocha, D. (2009). Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 5917--5922. IEEE.

Takahashi, O. & Schilling, R. J. (1989). Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on Robotics and Automation*, 5(2):143–150. ISSN 1042-296X.

Traniello, J. F. & Rosengaus, R. B. (1997). Ecology, evolution and division of labour in social insects. *Animal Behaviour*, 53(1):209--213.

van den Berg, J.; Guy, S. J.; Lin, M. & Manocha, D. (2010). Optimal reciprocal collision avoidance for multi-agent navigation. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

van den Berg, J.; Guy, S. J.; Lin, M. & Manocha, D. (2011). *Reciprocal n-Body Collision Avoidance*, pp. 3--19. Springer Berlin Heidelberg, Berlin, Heidelberg.

van den Berg, J.; Lin, M. & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928--1935. IEEE.

Vassev, E.; Sterritt, R.; Rouff, C. & Hinchey, M. (2012). Swarm technology at nasa: building resilient systems. *IT Professional*, (2):36--42.

Vrba, P.; Mařík, V.; Přeučil, L.; Kulich, M. & Šišlák, D. (2007). Collision avoidance algorithms: Multi-agent approach. Em *Holonic and Multi-Agent Systems for Manufacturing*, pp. 348--360. Springer.

Wilkie, D.; Van den Berg, J. & Manocha, D. (2009). Generalized velocity obstacles. Em *Proceedings of the IEEE International Conference on IntelligentRobots and Systems (IROS)*, pp. 5573--5578. IEEE.