Universidade Federal de Mato Grosso do Sul - UFMS
Faculdade de Computação - FACOM
Programa de Pós-Graduação em Computação Aplicada

Uma solução ao problema da dispersão de pássaros baseada em Enxame de Robôs utilizando o campo potencial de *Lennard-Jones*

Alexsandro Procópio da Silva

Área de Concentração: Sistemas Embarcados e Robótica

Curso: Mestrado Profissional em Computação Aplicada

Professores: Dr. Reginaldo Inosoja da Silva Filho (orientador)
Dr. Fabrício Augusto Rodrigues (coorientador)

Campo Grande-MS, 30 de junho de 2015.

Uma solução ao problema da dispersão de pássaros baseada em Enxame de Robôs utilizando o campo potencial de *Lennard-Jones*

Dissertação defendida por Alexsandro Procópio da Silva perante à comissão julgadora no Programa de Mestrado Stricto Sensu em Computação Aplicada da Faculdade de Computação, mantido pela Universidade Federal de Mato Grosso do Sul, como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

(Área de Concentração: Sistemas Embarcados e Robótica).

Campo Grande-MS, 30 de junho de 2015.

Banca Examinadora:

- Prof. Dr. Reginaldo Inosoja da Silva Filho (orientador) (CPPP-UFMS)
- Prof. Dr. Fabrício Augusto Rodrigues (coorientador) (CPPP-UFMS)
- Prof. Dr. Amaury Antônio de Castro Junior (CPPP-UFMS)
- Prof. Dr. Jonathan Andrade Silva (CPPP-UFMS)

A minha formação como profissional não seria concretizada sem a ajuda de minha mãe, que, no decorrer da minha vida, proporcionou, além de extenso carinho e amor, os conhecimentos da integridade, da perseverança, necessários para o meu desenvolvimento como ser humano. Por essa razão, dedico a você, minha imensa gratidão.

À Deus, dedico o meu agradecimento maior, o qual sempre me prospera com incontáveis bençãos.

Um agradecimento especial aos meus queridos irmãos Alexandre, Maria Emília e Alexander pelo amor, admiração, respeito e pelo incansável apoio durante o desenvolvimento deste trabalho, pela paciência e compreensão, fundamentais nesta trajetória. À todos vocês, meu muito obrigado.

Alexsandro Procópio da Silva

Agradecimentos

Esta dissertação é resultado de muito esforço e dedicação, e não poderia deixar de registrar aqui minha sincera gratidão a todos que estiveram ao meu lado durante o desenvolvimento deste trabalho.

Ao Prof. Dr. Reginaldo Inosoja da Silva Filho, pelo seu permanente apoio e atenção dispensada no decorrer de todo o processo de orientação. A sua disponibilidade irrestrita com a realização de reuniões em finais de semanas e feriados. A sua exigência e crítica foram fundamentais para a contribuição do meu crescimento enquanto pesquisador.

Ao coorientador Prof. Dr. Fabrício Augusto Rodrigues, o qual contribuiu para viabilização desta pesquisa. Obrigado pelos ensinamentos, atenção e dedicação ao longo deste período.

Ao Prof. Dr. Amaury Antônio de Castro Junior por me aconselhar a procurar o Prof. Dr. Reginaldo para ser meu orientador.

Aos Professores Drs. Vagner Pedrotti, Maria Istela Cagnin Machado e Debora Maria Barroso Paiva os quais acreditaram em meu potencial e me deram a motivação inicial com as cartas de recomendação para aceitação ao Programa de Mestrado.

Aos Professores integrantes do Colegiado do Curso que me acolheram tão bem durante esses dois anos como representante discente.

Aos Coordenadores do Curso, os Professores Drs. Luciano Gonda e Debora Maria Barroso Paiva pela atenção, sabedoria e profissionalismo demonstrados.

Agradeço a todos os professores do Programa de Mestrado Profissional, amigos e funcionários da Universidade de Mato Grosso do Sul, pelo auxílio e atenção ao longo destes meses.

Obrigado aos colegas de mestrado, em especial aos amigos: Vanessa, Fabrício, Joelmo, Roni e Marcel pela amizade então demonstrada, que, com uma perfeita mistura de dever e descontração, me ajudaram a cumprir minhas obrigações acadêmicas. Valeu turma pelos inesquecíveis churrascos para aliviar a tensão das provas e trabalhos.

Em especial quero agradecer a Daiane pelas aulas ministradas sobre Inteligência Artificial, que mesmo sem remuneração, esteve sempre disponível. Também quero fazer menção aos melhores parceiros de grupo que já tive, os amigos Evandro Luís Souza Falheiros e José Neto Gonçalves.

Também não poderia esquecer do amigo Virmerson B. Santos que várias vezes cedeu a sala de aula de sua empresa *Hightech* para nosso grupo de estudo se reunir.

Por fim, deixo aqui minha sincera gratidão a todas as pessoas que, direta ou indiretamente, contribuíram para a concretização desse sonho.

Novamente, obrigado Deus!

"Quão melhor é adquirir a sabedoria do que o ouro!
Quão mais excelente é adquirir a prudência do que a prata!"
(Bíblia Sagrada, Provérbios 16:16)

Abstract

The Swarm aproach has attracted the attention of the scientific community in applications systems due to control and coordination capacity of multi-robots [23]. The goal from this work is to use the fundamentals concepts and properties for the Robot Swarm to propose a new solution for dispersion of birds in agricultural areas. The solution is to employ a group of mobile robots for monitoring this agricultural area. When infringed access this site by birds, the robots perform procedures to disperse them in the region. Experiments were conducted in a 3D simulation environment with different contexts and the results are discussed in this work. The idea is that in the future this approach will be used in farm areas to minimize the damage caused by birds action.

Keywords: *Swarm robots, Self-organization, Potential Fields, Agriculture, Damage caused by bird.*

Resumo

O uso da abordagem de Enxame tem atraído a atenção da comunidade científica nas aplicações de sistemas devido a capacidade de coordenação e controle de multi-robôs [23]. O objetivo principal desse trabalho consiste em utilizar os principais conceitos e propriedades referentes ao Enxame de Robôs para propor uma solução alternativa para dispersão de pássaros em áreas agrícolas. A solução consiste em empregar um grupo de robôs móveis para monitoramento de uma área agrícola. Quando violado o acesso desse local por pássaros, os robôs executam procedimentos para dispersá-los da região. Foram realizados experimentos em um ambiente de simulação 3D com diferentes contextos e os resultados são discutidos nesse trabalho. Acredita-se, que no futuro, o emprego dessa abordagem na agricultura poderá minimizar os danos provocados pelos pássaros na vegetação cultivada.

Palavras-chaves: *Enxame de Robôs, Auto-organização, Campos Potenciais, Agricultura, Danos com Pássaro.*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Em 2012, o Departamento de Agricultura dos Estados Unidos (USDA) apresentou uma pesquisa [12] referente a danos com pássaros nas culturas de frutas em cinco estados americanos (Califórnia, Michigan, *New York*, Oregon e Washington). A pesquisa abordou produtores envolvidos com o cultivo de uvas para vinho, cerejas (doces e ácidas), mirtilos e maçãs. Entre os objetivos da pesquisa patrocinada pelo USDA destacaram-se o cálculo dos valores monetários da perda dos produtores com pássaros, os benefícios do emprego da ADP e a estimativa do impacto econômico dos danos em termos de mudanças na produção e empregos. A pesquisa concluiu que a média anual do impacto econômico dos danos com pássaros em 2011, relacionados com uvas para vinho, foi de US$ 126 milhões com a perda de 1.648 postos de trabalho. Também estimou uma perda de 32% da produção, se os produtores em todos os cinco estados não fizerem atentamente a ADP. Constatou ainda que a ADP previne entre US$ 421 milhões e US$ 444 milhões em perdas para os produtores de uva na receita dos cinco estados. Finalmente, afirmou que não administrar os danos com pássaros causaria uma perda de US$ 762 milhões de dólares na combinação da produção dos cinco estados e resultaria em cerca de 9.500 vagas de empregos perdidos.

Na Austrália, prejuízo provocado por pássaros para a produção hortícola é estimado em quase US$ 300 milhões por ano, sendo conhecidas mais de 60 espécies de aves responsáveis por esses danos. Essas espécies possuem diferenças marcantes em estratégias de alimentação e padrões de movimentos que influenciam a natureza, a época e a gravidade dos danos que causam [68].

No Brasil, o Anu, da espécie *(Agelaius ruficapillus)*, traz danos à cultura do arroz irrigado no Rio Grande do Sul. Ele arranca plântulas na fase inicial da cultura, o que pode reduzir aproximadamente em 25 a 60 por cento a população de plantas. Lavouras de arroz pré-germinado são as preferidas dessa espécie de pássaro, principalmente as mais próximas dos bosques. Ele também ataca durante a fase de maturação dos grãos, podendo causar perdas de produtividade superiores a 1250 kg por hectare [26]

O crescente desmatamento provocado pelo ser humano, em busca de moradia ou para expansão agrícola, podem ocasionar impactos nos *habitats* dos pássaros, resultando na migração de algumas espécies para regiões diferentes em busca de alimentos (brotos, frutos e sementes) e abrigo. Essa situação acarreta uma interação mais frequente dos pássaros com as áreas de cultivo [45] [31].

1.2 Definição do Problema e Justificativa

Danos com pássaros é um problema enfrentado por muitos agricultores. A dispersão dos pássaros nas áreas agrícolas não é um problema trivial de se resolver. Várias técnicas são empregadas pela comunidade agrícola para afastar pássaros. Embora sejam tradicionais na prática, suas eficácias podem não coincidirem com os rendimentos desejados [55]. Entre as técnicas de dispersão de pássaros [65] destacam-se: por meio do som [27], [38], [55], [61], [60], [68], pelo emprego da luz [27], [38], [39], [55], [61], [60], [68], por meio de predadores [47], [18], [27], [13], [60], [68], por meio de modificação do habitat [55], [13], [60], pelo emprego do sistema de exclusão da compensação [27], [68], [60], pelo emprego de repelentes químicos [11], [36] e pelo uso de avicidas [13], [60].

A justificativa para este trabalho é de que maioria das técnicas empregadas para dispersão de pássaros possuírem algum tipo de limitação (efetividade ou custo) e permitirem que os pássaros com o tempo se habituem com algumas das técnicas, passando a ignorá-las. Dentro desse contexto, é necessário estudar novas técnicas para minimizar os danos provocados por pássaros.

A principal motivação é a observação de que as propriedades e características da abordagem de Enxame [1] de Robôs (ER) [4] poderem proporcionar um aporte para resolver o problema da dispersão de pássaros. Pesquisando na literatura não foram encontrados relatos do emprego de enxame de robôs para resolverem este tipo de problema. A abordagem de ER [70] permite soluções algorítmicas baseadas nos comportamentos de animais sociais, possibilitando:

- sistemas distribuídos com interações entre os robôs autônomos;
- robôs com controle auto-organizado e descentralizado;
- divisão e distribuição do trabalho, com alocação de tarefas;
- interações indiretas entre cada robô e o meio ambiente.

1.3 Objetivo e Hipótese

O propósito desse trabalho é demonstrar que a estratégia de controle de robôs, por meio de campos potenciais de *Lennard-Jones* [34], pode ser empregada para dispersar pássaros de áreas agrícolas com o uso de múltiplos robôs empregando a abordagem de ER. Para a formulação da proposta da solução foi estabelecida a seguinte hipótese de pesquisa: "É possível utilizar métodos baseados em enxame de robôs com campos potenciais de *Lennard-Jones* para dispersão de pássaros de maneira eficiente, com baixo custo e aumentando a produtividade da lavoura?". Em relação aos pássaros, pesquisou-se a resposta para o seguinte questionamento: "Quais são as técnicas atualmente empregadas para dispersão de pássaros na agricultura?".

1.4 Contribuições

O estudo científico sobre ER permite soluções interessantes que podem automatizar as operações de desconcentração de pássaros em áreas agrícolas. Acredita-se que mediante o uso da metodologia

[1] O termo Enxame é utilizado de forma genérica para se referir a qualquer coleção estruturada de robôs capazes de interagir [59].

proposta, o problema de dispersão de pássaros seja minimizado, reduzindo a perda da safra.

Uma contribuição dessa dissertação foi a publicação do artigo "*Swarm robotics:* comportamento adaptativo aplicado ao problema de dispersão de pássaros em áreas agrícolas" [65], em 2014, no VIII *Workshop* de Tecnologia Adaptativa, na Escola Politécnica da Universidade de São Paulo, demonstrando que a estratégia de controle de robôs, por meio de campos potenciais de *Lennard-Jones*, pode ser empregada para dispersar pássaros de áreas agrícolas com o uso de múltiplos robôs empregando a abordagem de ER.

Além disso, com base na pesquisa apresentada, pretende-se, que se abra um leque de aplicações para uso futuro, dentre elas:

- A utilização para varredura de uma determinada área, com o objetivo de localizar pessoas desaparecidas (por exemplo em uma floresta), vítimas de catástrofes (por exemplo *Tsunami* e furacões), aviões desaparecidos em alto-mar.
- O uso do enxame de robôs para vigilância de área para evitar o acesso de pássaros em aeroportos, ou de outros animais que possam interferir na produção agrícola, por exemplo, ratos, formigas e gafanhotos.
- Aplicações na área de segurança pública para monitoria e contenção de pessoas estranhas em locais restritos.

1.5 Material e Métodos

A simulação computacional foi escolhida como processo metodológico e os experimentos realizados no simulador 3D ARGoS [52]. Os programas da simulação foram escritos na linguagem C++ para implementar os movimentos do pássaro. Nos experimentos simulou-se uma área agrícola de 144 m^2 na qual um pássaro voa em diferentes trajetórias (retilínea, semicircular e circular), variando a velocidade. Essas trajetórias foram escolhidas por representarem as trajetórias de voo de alguns pássaros ou quando combinadas resultarem em trajetórias mais complexas. Um enxame de robôs é inserido na área agrícola com o objetivo de detectar o pássaro e dispersá-lo da referida área.

Os experimentos foram agrupados em nove cenários com o objetivo de verificar se o enxame de robôs consegue efetuar o cerco do pássaro durante o voo e quais as reações comportamentais ocorridas no enxame diante de determinadas variações (velocidade do voo do pássaro, quantidade de robôs e trajetória do voo do pássaro). Um cenário é composto de duas partes. A primeira parte é representada pelo número de robôs, tipo de trajetória do voo do pássaro, a distância do voo e o fator de interpolação. A segunda parte do cenário corresponde aos dados coletados durante a execução dos experimentos. É composta pelos tempos iniciais e finais do experimento, pela duração do voo, a velocidade do voo, os tempos gastos para o enxame realizar o primeiro e o último cerco, a capacidade do enxame em cercar o pássaro em pleno voo e se o enxame sofre divisões em pequenos grupos durante o processo de cerco. Cada cenário foi executado 16 vezes fixando para cada um o tipo de trajetória, o número de robôs e a distância do voo, variando apenas o fator de interpolação, resultando em 144 experimentos diferentes.

Além disso, verificou se cada experimento apresentava as características inerentes à abordagem de Enxame, entre elas: autonomia, conhecimento limitado do local, controlador central, robustez, flexibilidade, escalabilidade e simplicidade.

1.6 Organização da Dissertação

A escrita desta dissertação está organizada em cinco capítulos.

1. O Capítulo 2 aborda a fundamentação teórica, sendo dividido em quatro Seções:
   - Na Seção 2.1 são mencionadas as técnicas tradicionais para dispersão de pássaros: por meio de som, pelo emprego de luz, por meio de predadores, por meio de modificações no habitat, pelo emprego do sistema de exclusão de compensação, pelo emprego de repelentes químicos e pelo uso de avicidas.
   - Na Seção 2.2 são citados os conceitos básicos sobre Enxame de Robôs, entre eles a origem do termo Enxame, suas características e propriedades.
   - Na Seção 2.3 são abordadas as estratégias de controle dos robôs, tratando sobre o planejamento de caminhos e o número de diferentes representações do espaço de configurações, entre elas Mapas de Caminhos, Decomposição em Células e Método de Campo Potencial.
   - Na Seção 2.4 trata sobre o Padrão de Cerco com Enxame usando o campo potencial de *Lennard-Jones*.
2. O Capítulo 3 trata sobre os materiais e métodos empregados, sendo dividido em sete Seções:
   - Na Seção 3.1 são determinadas as características do conceito de ER;
   - Na Seção 3.2 é determinada como será a coesão do ER;
   - Na Seção 3.3 é delimitado o comportamento dos robôs diante do pássaro;
   - Na Seção 3.4 é definida a dinâmica dos experimentos;
   - Na Seção 3.5 é definindo como serão realizados os experimentos e descreve o simulador ARGoS;
   - Na Seção 3.6 trata sobre as características do robô escolhido para os experimentos;
   - Na Seção 3.7 explica o ambiente computacional utilizado nas simulações e menciona as limitações da simulação.
3. No Capítulo 4 são apresentados os experimentos realizados e os seus resultados, sendo dividido em três Seções:
   - Na Seção 4.1 é realizado um experimento inicial para testes;
   - Na Seção 4.2 são executados novos experimentos com base na experiência adquirida com o experimento inicial e apresentados os resultados obtidos.
   - Na Seção 4.3 é realizada a avaliação do comportamento do durante durante a realização dos experimentos.
4. O Capítulo 5 aborda a conclusão desse trabalho, sendo dividido em duas Seções:
   - Na Seção 5.1 trata-se sobre as contribuições desse estudo realizado;
   - Na Seção 5.2 menciona as possibilidades de pesquisas futuras.

Capítulo 2

Fundamentos Teóricos

2.1 Técnicas Tradicionais para Dispersão de Pássaros

Nessa seção serão abordadas as técnicas de dispersão ou deslocalização mais usuais utilizadas pelos agricultores para reduzirem os danos provocados por espécies de pássaros. Embora os pássaros sejam fundamentais para os ecossistemas nos quais estão inseridos, sua concentração em áreas agrícolas pode afetar negativamente as culturas, quer seja pela contaminação de grãos quer seja pela escassez de alguns tipos de alimentos, em certos períodos de tempos.

Uma das reações de um pássaro ao ter medo é voar. Cada vez que esse fato ocorre devido a sua curiosidade, o pássaro tenta reunir informações sobre o estímulo que o assusta, acumulando informações suficientes para saber se o estímulo representa uma ameaça real. Caso tal estímulo não represente uma ameaça, o pássaro se habitua [27] a ele, passando a ignorá-lo. A partir desses levantamentos o tempo necessário para o pássaro se habituar pode variar, dependendo de um conjunto de fatores, incluindo a espécie, o habitat circundante, a regularidade e o tipo de ruído. A habituação é o fator que mais limita a eficácia das técnicas empregadas para assustar o pássaro [68].

Como resultado das pesquisas realizadas na literatura, verificou-se basicamente oito grupos de abordagens [27] [65] utilizadas para desconcentrar pássaros, que serão apresentadas a seguir:

- Por meio do som;
- Pelo emprego de luz;
- Por meio de predadores;
- Por meio de modificação do habitat;
- Pelo emprego do sistema de exclusão de compensação;
- Pelo emprego de repelentes químicos;
- Pelo uso de avicidas.

2.1.1 Por Meio de Som

Essa abordagem consiste em assustar o pássaro pelo uso de dispositivos acústicos [61] [27]. Entre esses dispositivos, verificou-se a utilização de equipamentos eletrônicos que reproduzam os sons de pássaros [55]. O instinto natural dos pássaros é evitar o perigo [38]. Inicialmente, um determinado som é gravado e, posteriormente, quando o pássaro invade o perímetro não permitido, esse som é exposto para que o pássaro ouça e se assuste. Esse som gravado pode representar:

1. O mesmo som emitido por um pássaro quando ele avista um predador ou algo que lhe ameace. A emissão de chamadas de socorro de aves predadores mantém as aves distantes das culturas [55] [18] [27].
2. O som emitido por um predador, com o objetivo de imitar a presença da ave predadora no local [61] [27].

Os equipamentos eletrônicos (Fig. 2.1) também podem reproduzir sons não-biológicos, como ruídos eletrônicos [61], para que o pássaro se assuste de súbito [68]. Esses sons podem ser gerados em frequência de ultrassom, imperceptíveis ao ser humano.

Figura 2.1: Equipamentos para reprodução de som. Fonte: TRACEY, 2007, [68].

Verifica-se também o uso de pessoas contratadas para caminharem em meio ao campo, fazendo ruído, com latas e outros objetos metálicos [55].

Para tentar evitar que os pássaros se habituem com o som, os dispositivos eletrônicos não devem emiti-lo do mesmo ponto de origem, com o mesmo curto período de tempo e na mesma intensidade (volume). Seu uso a longo prazo tem sido provado ser ineficaz devido a habituação das aves a certos estímulos [38].

O uso de dispositivos explosivos agrícolas [55] [60] [61] [27] ("bombinhas") podem criar sons que assustem os pássaros. Cuidados devem ser tomados sobre a frequência de queima e de mudança de posições e direções para evitar que o pássaro fique habituado. Os dispositivos explosivos envolvem risco de incêndio.

Outra solução envolve o uso das munições denominadas **bolachas *shell***, as quais são disparadas por uma arma calibre 12, gerando grande estrondo (Fig. 2.2) [61] [27]. Elas são úteis em uma variedade de situações devido a longa distância que o projetil percorre. Essa técnica exige a manutenção constante do cano da arma, entre outros mecanismos, devido a sua característica corrosiva. Além disso, é necessária uma proteção adequada aos olhos para a pessoa que efetua os disparos. Um efeito colateral dessa técnica é que pode machucar os pássaros.

Figura 2.2: Biscoitos *Shell* são disparados de uma espingarda de calibre 12. Fonte: GORENZEL, 2008, [27].

Outra forma comum de produção de som é o uso de armas de fogo com munição real [60]. Exemplos mais recorrentes são o emprego de espingardas ou rifles. Embora sejam mais acessíveis, são menos eficazes e envolvem a questão da segurança ao utilizá-las em áreas urbanas [60].

Outro método é o uso de equipamentos mecânicos (Fig. 2.3) que produzem ruídos simulando disparos de arma de fogo, através do uso de gás propano ou acetileno [60] [11] [18] [27]. Esses equipamentos reproduzem o som de uma explosão extremamente alta, de forma intermitente que excede a explosão de uma espingarda de calibre 12. A suposição é que as aves vão associar a explosão com tiros e fugir da área [61]. Devido aos diferentes tipos de configuração, o mecanismo de disparo pode ser acionado por meio de controle remoto ou por sensores que detectam a presença dos pássaros. As desvantagens do uso do canhão de gás são [68] [27]:

- A emissão de gases na atmosfera.
- O incômodo gerado pelo barulho dos disparos.
- As alterações do intervalo dos disparos para os pássaros não se acostumarem.
- A necessidade de instalar vários equipamentos mecânicos, em função da extensão da área rural e do nível de concentração de pássaros, para cobrir toda a região.
- A necessidade de elevar esses dispositivos acima do nível da vegetação envolvente, pois os ruídos provocados acima dela no ar, na altura dos pássaros, são mais eficazes do que aqueles produzidos rente ao chão.

Figura 2.3: Um canhão de propano instalado em uma agrícola. Fonte: GORENZEL, 2008, [27].

2.1.2 Pelo Emprego de Luz

Essa abordagem compreende em assustar o pássaro por meio do uso de dispositivos que emitem ou refletem a luz [27]. Nesse contexto, constatou-se diversos objetos introduzidos no meio da cultura, pendurados na vegetação ou em hastes, para refletirem a luz solar de forma aleatória, quando o objeto é movimentado pelo vento [39]. Entre os objetos encontrados destacam-se: espelhos refletores, fitas reflexivas [55] [13], CDs (*compact discs*), entre outros materiais brilhantes [60] (Fig. 2.4).

Figura 2.4: Balões reflexivos inflados com gás hélio são suscetíveis a danos por ventos superiores a 15 milhas por hora. Fonte: GORENZEL, 2008, [27] .

Outras soluções utilizam equipamentos eletrônicos que são afixados no meio da plantação, emitindo flashes de luz, com alta luminosidade, de tempo em tempo.

A principal limitação dessa abordagem é que [68]:

(a) Os pássaros se habituam muito rápido com os feixes luminosos e acabam se depositando nas regiões opostas às exposições dos feixes luminosos.

(b) Os ventos fortes e outras intempéries podem quebrar ou danificar os objetos.

2.1.3 Por Meio de Predadores

Essa abordagem consiste em assustar um pássaro pelo emprego de predadores naturais ou outros dispositivos artificiais que possuem características de um predador. O uso de predadores naturais, como as aves de rapinas, entre elas, o falcão-de-coleira *(Falco femoralis)*, o falcão-peregrino *(Falco peregrinus)* e o gavião-asa-de-telha *(Parabuteo unicinctus)*, são utilizados para sobrevoarem o local, periodicamente, inibindo a presença de pássaros [47]. Esses predadores quando devidamente treinados e monitorados por especialistas, podem capturar os pássaros e trazê-los em suas garras, para serem presos em gaiolas e, posteriormente, soltos em áreas distantes.

Os problemas dessa técnica são: o alto custo do investimento quando comparado com outras técnicas, a falta de garantia de que os pássaros retirados não retornarão ao mesmo território, o ferimento ocasionado na perseguição e captura pelas aves de rapina, além da dificuldade em levar os pássaros para áreas distantes [68].

Foram verificados casos em que são utilizados predadores artificiais com características de um ser humano denominados popularmente espantalhos [27] [13] (Fig. 2.5). Esses podem ter o formato de um ser humano [18], ou de alguma ave de rapina [60]. Para maximizar os resultados dessa abordagem o espantalho deve [27]:

(a) Ser o mais realista possível e posicionado em lugares com boa visibilidade.

(b) Possuir algum tipo de movimento (mobilidade), não serem totalmente estáticos.

(c) Ser movido com frequência para novos locais em torno da cultura para ajudar a evitar a habituação.

(d) Ser usado em conjunto com outros métodos de controle, por exemplo, dispositivos acústicos.

(e) Ser inserido no local antes dos pássaros desenvolverem um hábito alimentar na cultura.

Figura 2.5: O uso de um espantalho em uma área agrícola. Fonte: GORENZEL, 2008, [27].

Também verificou-se bonecos com características de algum predador natural de aves [55], por exemplo, o uso de pipas em formato e cores de aves predadoras (falcões ou gaviões) [18]. Essas pipas são geralmente suspensas a uma determinada altura sobre a vegetação. O problema é que os pássaros, com o

passar do tempo, percebem que o falso predador não se move e ficam a uma distância segura dele. Existe a necessidade da mudança contínua do local desses predadores artificiais para que essa situação não ocorra. Também constatou-se o uso de balões cheios de gás hélio pendurados no meio da plantação, com cores diferenciadas e com desenhos de olhos [60] [18] [27]. Por fim, verificou-se o uso de aeronaves. Os pássaros desaparecem em revoada quando ouvem o zumbido das hélices ou por verem o objeto voando [27].

2.1.4 Por Meio de Modificações no Habitat

A abordagem referente às modificações de habitat incluem simples atividades que podem torná-lo menos atraente para os pássaros. Por exemplo, criar pertubações aos locais de nidificação das aves (destruição manual dos ninhos) [55] [13] [60] ou efetuar a poda da vegetação para remover a tampa de proteção, ou seja, retirar os galhos e as folhas que permitem que o pássaro se sinta seguro e faça moradia [60]. Isso pode desencorajar os pássaros a também utilizá-la como poleiro. Essa abordagem, em alguns casos, produz um efeito mais duradouro do que outros métodos e é menos caro a longo prazo. A desvantagem dessa técnica é que não pode ser empregada em culturas que não permite a realização de poda, por exemplo, a soja, arroz e milho.

2.1.5 Pelo Emprego do Sistema de Exclusão de Compensação

Essa abordagem consiste em cobrir toda a vegetação usando redes, plástico ou qualquer tipo de material (Fig. 2.6) que impeçam fisicamente os pássaros de acessarem a cultura. É uma das formas mais eficazes de combate às aves. Não utiliza dispositivos acústicos e nem produtos químicos. Essa solução é geralmente empregada apenas em culturas de alto valor, que compense o investimento na implantação e na manutenção dos equipamentos e materiais usados para cobrir a cultura [68] [60]. É impraticável para grandes áreas e é praticamente imóvel uma vez instalado [27].

Figura 2.6: Rede cobrindo a vegetação. Fonte: TRACEY, 2007, [68].

2.1.6 Pelo Emprego de Repelentes Químicos

Nessa abordagem são empregados repelentes químicos [36], os quais são substâncias normalmente pulverizadas sobre as culturas para impedirem a presença de pássaros. Essas substâncias alteram o sabor, o cheiro e a cor, fazendo com que a área agrícola perca sua atratividade para os pássaros. O problema é que essas substâncias podem deixar resíduos nos alimentos, tornando-os inadequados para o consumo humano [11] quando aplicados em excesso.

2.1.7 Pelo Uso de Avicidas

Nessa abordagem, sob autorização do Governo, em alguns raros casos, são utilizadas de forma restrita algumas substâncias químicas conhecidas como avicidas [60] [13]. Essas substâncias (Fig. 2.7) são dispostas em forma de iscas e tem a função de eliminar os pássaros. Os principais dilemas dessa abordagem são: a resistência a sua utilização, seus impactos

Figura 2.7: Preparação de avicida. Fonte: TRACEY, 2007, [68].

2.2 Enxame de Robôs: Origem, Características e Propriedades

Nessa seção é apresentado o resultado de uma pesquisa discorrendo sobre a origem do termo Enxame, seu significado, bem como a influência exercida pelo estudo dos insetos sociais dentro do contexto da Computação. Também são abordadas as características e propriedades básicas inerentes à Enxame de Robôs.

2.2.1 Origem do Termo Enxame

A origem do termo Enxame vem da Biologia, inspirado no comportamento de animais sociais, aparecendo na Computação com diferentes nomes, entre eles: *Swarm [4], Swarming [1], Swarm Intelligence [6], Swarm Optimization [3], Swarm Engineering [32] e Swarm Robotics [70].*

Esse termo sofreu essas variações devido a uma ampla gama de estudos de otimização, afastando-se de seu contexto na robótica. Apesar dessas diferentes nomenclaturas, a expressão recorrente na literatura na área de Robótica é Enxame de Robôs, a qual aborda a coordenação de um grande número de robôs, inspirando-se na observação do comportamento de animais sociais, entre eles formigas, cupins, vespas, abelhas, bandos de pássaros, cardumes [59].

Descrito de um modo mais formal, Enxame de Robôs (ER) é o estudo de como um número relativamente grande de robôs pode ser fisicamente incorporado, de forma que um comportamento coletivo emerja a partir das interações locais entre cada integrante e entre os integrantes e o meio ambiente [59]. Os insetos sociais são muito conhecidos pela capacidade de coordenarem suas ações em grupo, realizando tarefas que estão além da capacidade de um único indivíduo. Um exemplo clássico é o transporte de grandes animais (presas) realizado em conjunto por várias formigas [59].

Em 1988, Gerardo Beni introduziu o termo Enxame no contexto da robótica [4]. Gerardo procurava uma extensão ao conceito de autômatos celulares, uma vez que nesta época os robôs não estavam operando de forma síncrona e nem sequencialmente, apesar de haver certa interação dinâmica entre eles. Assim, Gerardo acabou usando o termo Enxame dentro do contexto das linhas de montagens que utilizavam robôs. Ele considerou que a escolha do termo foi necessária, pois, ao invés de um único sistema central projetado para executar uma atividade, teriam vários elementos de menor complexidade, montados a partir de componentes simples para execução da mesma atividade. Tal processo poderia ser implementado em larga escala, além de suportar pertubações de vários tipos (falhas ou ruídos), trazendo, assim, uma maior confiabilidade [4].

2.2.2 Histórico da Influência dos Estudos de Insetos Sociais na Abordagem do Enxame de Robôs

Konrad Lorenz (1903-1989) apresentou os primeiros estudos científicos [37] sobre o fenômeno denominado *imprinting*, definindo-o como uma forma de comportamento instintivo nos animais chamado como padrão fixo de ação. Esse padrão foi definido como a resposta que um organismo emite em função de um **estímulo** específico. Como exemplo, temos os filhotes de certas aves, que adotam como pais os primeiros seres vivos que vêem.

Em 1953, E. O. Wison [30] influenciado pelo trabalho de Lorenz apresentou suas pesquisas sobre o comportamento da sociedade das formigas. Ele propôs que essa sociedade seria explicada e entendida por simples padrões fixos de ação, os quais seriam resultantes de estímulos provocados por substâncias químicas (feromônios). Tais substâncias seriam produzidas por glândulas nas formigas.

Nesse trabalho mencionou que as formigas emitem feromônios específicos para se comunicarem. Também foram identificados os produtos químicos que os compõem e quais as glândulas que os emitem. Além disso, foram identificadas as respostas da ação para cada tipo de feromônio, permitindo entender os comportamentos do grupo das formigas [30].

Em 1979, Hofstadter [29], também afirmou em seu trabalho que existia algum tipo de comunicação entre as formigas. Afirmou que essa comunicação seria mínima, mas suficiente para que elas não vagassem ao acaso, mas realizando suas atividades em grupos, com um grande número de integrantes. Por exemplo, os cupins realizam a construção de grandes obras, sem necessidade de serem coordenados por um líder (um arquiteto). Através dos estímulos provocados pelos feromônios, cada cupim se move de forma descentralizada, depositando no caso uma substância umedecida em determinados locais, e através destes movimentos aleatórios vai construindo os pilares da cúpula [33].

Nesse sentido, ressalta-se que os cupins têm um sistema de construção auto-organizado, não havendo um controle central. Cada integrante responde apenas aos estímulos do meio ambiente, provocado pelo tipo e a quantidade de feromônio presente.

Grande parte da comunicação entre os insetos sociais é realizada indiretamente através de um método que o entomologista Grassé, em 1959, intitulou de **estigmergia** [28]. Nesse contexto, uma mudança no ambiente proporciona uma sugestão que pode interferir no comportamento de outros insetos. Em 1991, Deneubourg [14] e seu grupo de pesquisa demostraram que a formação de um cemitério de formigas é gerado por simples regras, sem nenhuma coordenação central. Eles observaram que algumas espécies de formigas, como a *Pheidole pallidula*, possuem a característica de empilharem corpos resultando na limpeza do formigueiro. Inicialmente, as formigas formam vários grupos distintos de corpos (várias pilhas). Nesses grupos são constantemente adicionados e removidos corpos, e com o tempo, apenas os grupos de corpos que cresceram mais rapidamente prevalecem [33].

Em 1987, foi apresentada por Reynolds Craig [56] uma simulação sobre o comportamento de aves em bando em um espaço tridimensional, baseada em três tipos de forças locais: anticolisão (se afastar antes de bater em um outro), velocidade (tentar ir sobre a mesma velocidade que seus vizinhos no rebanho) e centralização (tentar mover em direção ao centro do rebanho). Este estudo passou a fornecer o algoritmo para animação de gados e rebanhos em filmes.

Em 1992 e 1995, foram apresentadas pesquisas envolvendo modelos matemáticos sobre a dinâmica do comportamento das formigas [10] [42]. Esse estudo demonstrou que quando o alimento é colocado a uma certa distância a partir do ninho, com dois caminhos de comprimento desigual que conduz a ele, as formigas acabarão com o enxame seguindo o caminho mais curto.

Em 1997, Marcos Dorigo [15] [16] e seu grupo de pesquisa apresentaram uma solução para otimizar o problema do caixeiro viajante, introduzindo os conceitos de feromônio e baseando-se na observação de que as formigas vão encontrar o caminho mais curto em torno de um obstáculo que separa seu ninho de um alvo.

A motivação principal para as pesquisas sobre a coordenação dos movimentos desses insetos sociais é obter conhecimento sobre tais comportamentos e, a partir desse conhecimento, projetar regras simples e descentralizadas que possam ser desenvolvidas e implantadas em robôs.

2.2.3 Enxame de Robôs: Características Básicas

Em 1994, Millonas [42], em seus estudos sobre comportamentos coletivos de animais propôs, algumas características que, ao longo do tempo, serviram como base para inspirar as pesquisas aplicadas no contexto da abordagem ER, são elas:

– **proximidade** - os indivíduos devem ser capazes de interagir, levando em consideração o tempo e o espaço, tais como as colônias de abelhas exploram regularmente diversas áreas para encontrar fontes mais rentáveis de néctar. Cada abelha opera com informações limitadas sobre as fontes de alimento da colônia, mas juntas elas geram uma resposta coerente (comportamento emergente) sobre as diferentes fontes de alimentação [62]. (Veja a Tabela 2.1).

Tabela 2.1 Comportamento Emergente. Fonte: Adaptado de TRIANNI, 2008, [69].

Controlador (Central)	Controlador (Enxame)
Algumas tarefas para serem alcançadas necessitam da sequência ordenada de subtarefas (comportamento global e previsível).	O comportamento global surge como resultado das interações locais entre cada robô e o meio. O comportamento global não é codificado por regras.

– **qualidade** - os indivíduos devem ser capazes não só de interagir, mas de avaliar seus comportamentos;

– **diversidade** - permite ao sistema reagir a situações inesperadas. Para isto o grupo não aloca todos os seus recursos de uma só vez, eles são alocados ao longo do tempo por meio de modos seguros, prevenindo-se de repentinas alterações ambientais;

– **estabilidade** - nem todas as variações ambientais devem afetar o comportamento do grupo;

– **adaptabilidade** - é a capacidade de adequação à variações ambientais, de forma equilibrada, mantendo a estabilidade do sistema. A mudança comportamental ocorrerá quando for necessário o investimento de energia. Nos cupins, quando há uma fragmentação do habitat, o instinto de sobrevivência da espécie inicia a recolonização e a expansão da colônia para outras regiões do meio ambiente [21].

Em 2005, Sahin [59] apresentou uma pesquisa na qual foram abordadas, entre outros assuntos, três propriedades funcionais que motivam o estudo de ER, inspiradas na análise das características comportamentais de animais sociais. São elas: a robustez, a flexibilidade e a escalabilidade. Por **robustez**, entende-se que o sistema seja capaz de continuar a funcionar, apesar das falhas nos indivíduos ou perturbações no ambiente. Ao tentar proteger uma planta que está sendo atacada por formigas, podemos perceber que esta não é uma tarefa trivial. A robustez pode ser atribuída à vários fatores:

(a) **a redundância no sistema**, ou seja, qualquer perda ou avaria de um indivíduo pode ser compensado por outro. Isto faz com que os indivíduos sejam dispensáveis;

(b) **a coordenação descentralizada**, quer dizer, a perda de uma determinada parte do sistema não irá impedir o funcionamento do todo. A coordenação é uma propriedade emergente do sistema. (Veja a Tabela 2.2.)

Tabela 2.2 Descentralização. Fonte: Adaptado de TRIANNI, 2008, [69].

Controlador (Central)	Controlador (Enxame)
A decisão é tomada por um controlador central e repassada a cada robô.	Cada robô é responsável pelas suas decisões, tomadas de forma independente (autonomia).
O controlador central é complexo (com amplo conhecimento do contexto).	Cada robô é homogêneo e simples (conhecimento limitado).

(c) **a simplicidade dos indivíduos**, isto é, em comparação com um único sistema complexo que poderia realizar a mesma tarefa, nessa abordagem os indivíduos seriam mais simples, tornando-se menos propensos a falhas.

Por **flexibilidade**, entende-se que o sistema se adapte para oferecer diferentes estratégias de coordenação, diante das mudanças de contexto do ambiente. Voltando ao exemplo das formigas, elas são

capazes de responder imediatamente as novas condições ambientais [69]. Elas se adaptam para buscar alimentos (uso de feromônios), para arrastar presas para o ninho (recrutamento) e para ultrapassar obstáculos (se aglomerando). (Veja a Tabela 2.3.)

Tabela 2.3 Flexibilidade e robustez. Fonte: Adaptado de TRIANNI, 2008, [69].

Controlador (Central)	Controlador (Enxame)
Se o controlador central falhar, os robôs também falham.	A falha de alguns indivíduos até certo ponto, não chega a comprometer o sistema (robustez).
Os robôs trabalham segundo as informações centrais repassadas.	O sistema é flexível, podendo se adaptar de acordo com novas situações, reagindo a eventos sem precisar ser previamente programado. A estigmergia é muito usada nesse contexto.

Por **escalabilidade**, entende-se que um sistema opere em uma ampla gama de tamanhos de grupos. Isto é, os mecanismos de coordenação que asseguram o funcionamento do enxame devem ser relativamente flexíveis por mudanças nos tamanhos dos grupos.

Além destas propriedades funcionais, Sahin 2005 [59] em sua pesquisa também propôs alguns critérios para que uma dada aplicação fosse enquadrada como uma abordagem de ER:

(a) **auto-organização**: a coordenação de um ER deve permitir que cada indivíduo seja capaz de interagir fisicamente com o ambiente e com outros indivíduos de forma descentralizada;

(b) **homogeneidade**: os indivíduos que compõem o enxame devem ter funções semelhantes;

(c) **simplicidade**: cada indivíduo deve ser simples, em relação as suas capacidades individuais para a realização de uma tarefa e não em relação a sua capacidade de *hardware* ou *software*;

(d) **estigmergia**: os indivíduos devem possuir uma interação local, uma forma de comunicação limitada. Essa propriedade reduz a sobrecarga de informações trocadas entre os robôs. Em geral, cada integrante do enxame se comunica apenas com os seus vizinhos mais próximos, obtendo apenas informações locais. (Veja a Tabela 2.4.)

Tabela 2.4 Localidade e Estigmergia. Fonte: Adaptado de TRIANNI, 2008, [69].

Controlador (Central)	Controlador (Enxame)
O controlador central pode se comunicar com todos os robôs.	Cada robô tem a habilidade de comunicação limitada (em geral, apenas com seus vizinhos mais próximos).
Comunicação é direta (explícita).	Os robôs podem usar informações disponíveis no meio ambiente (estigmergia) reduzindo a complexidade do sistema de controle.

Em 2008, Trianni [69] propôs que, para um sistema ser considerado auto-organizado ele deve ser descentralizado, flexível, robusto e local, conforme a Tabela 2.5 . Ele também apresentou um estudo sobre as comparações entre as diferenças de um sistema com um controle central e um sistema com a abordagem de Enxame. As Tabelas 2.1, 2.2, 2.3 e 2.4 apresentadas anteriormente nessa Seção referem-se a esse estudo.

Tabela 2.5 Propriedades de um Sistema Auto-organizado. Fonte: Adaptado de TRIANNI, 2008, [69].

Propriedade	Descrição
Descentralização	Todos os robôs são autônomos: não há um líder que comande a organização do sistema.
Flexibilidade e robustez	O sistema tem a tendência natural para manter a sua organização: a) É resistente a alterações ambientais e perturbações externas. b) Possui uma elevada redundância de componentes individuais
Localidade	Cada robô se baseia apenas em informações locais e interage localmente com os outros robôs do sistema, e o seu comportamento pode ser modelado com regras simples.

2.3 Estratégias de Navegação de Robôs

A estratégia de navegação dos robôs está diretamente ligada a navegação, a qual está relacionada com a capacidade de um robô mover-se da sua posição original até um objetivo baseando-se em

informações parciais sobre sua posição e sobre o ambiente. Para isso, o robô precisa ter duas habilidades: o planejamento de caminho e a capacidade de desvios de obstáculos.

Planejamento de caminho

O planejamento de caminhos ou planejamento de trajetórias se refere a determinar um caminho em uma espaço de configurações entre a configuração inicial do robô e a configuração final, de modo que o robô não colida com nenhum obstáculo [17]. De acordo com a Figura 2.8, um robô pode ser representado como um ponto em um espaço de configurações [64], onde cada ponto especifica a posição, a orientação e os ângulos das articulações do robô.

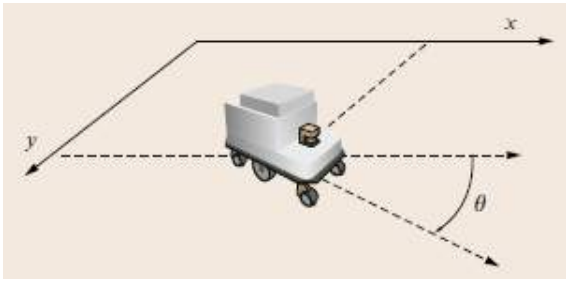

Figura 2.8: Representação do robô com as coordenadas de posição (x,y) e orientação θ. Fonte: SICILIANO, 2008, [63].

O espaço de configurações contém as trajetórias possíveis e possui duas componentes: o espaço de todas as configurações que um robô pode atingir (espaço livre) e o espaço de configurações inacessíveis (espaço ocupado).

Outra definição para o problema de planejamento de caminho menciona que ele consiste em encontrar um caminho contínuo, livre de colisões, para um objeto se mover de uma posição e orientações iniciais para uma posição e orientações finais (objetivo) [63], buscando estas configurações (posições) dentro do espaço de configuração [58] [20], onde:

- O caminho contínuo: representa a trajetória percorrida pelo robô dentro do espaço livre.
- Objetivo: representa a posição final que o objeto móvel deve atingir.
- Colisões: representam os obstáculos existentes no espaço de configuração.

O espaço de configurações pode ser representado pela seguintes metodologias: Mapas de Caminhos [57], Decomposição em Células [64] [63] e Método de Campo Potencial [2] [63] [17], conforme descritas na Tabela 2.6.

Tabela 2.6 Descrição das metodologias para representar o espaço livre de configurações. Fonte: Diversos autores, [57] [64] [63] [2] [17].

Tipo de metodologia	Forma de representação do espaço livre
Mapas de Caminhos	O espaço livre é representado por meio de um grafo.
Decomposição em Células	O espaço livre é dividido em regiões simples (células).
Campos Potenciais	O espaço livre é definido através de uma analogia de campos positivos (atração) e campos negativos (repulsão).

2.3.1 Mapas de Caminhos

Os Mapas de Caminhos [57] representam a estrutura do espaço livre através de um conjunto de segmentos interligados formando um grafo. Procura-se um caminho no grafo, utilizando algoritmos de busca. Os métodos clássicos que adotam essa abordagem são os grafos de Visibilidade [17] e os diagramas de *Voronoi* [43].

Grafo de Visibilidade

O grafo de visibilidade [25] é uma técnica que produz um caminho de tamanho mínimo da posição inicial até a posição final. O grafo de visibilidade $G = (V, A)$, é definido tal que o conjunto de vértices V é composto da união de todos os vértices dos obstáculos no meio, como também do ponto inicial As arestas A do grafo conectam todos os vértices que são visíveis um ao outro, isto é a linha reta conecta eles não intersectando nenhum obstáculo. Todo o segmento de reta que estiver inteiramente na região do espaço livre é adicionada ao grafo. Depois de construído o grafo ele será um subconjunto dos vértices dos obstáculos e dos nós iniciais e finais, com arestas que conectam diretamente um vértice ao outro diretamente sem colidir com nenhum obstáculo [17] Os passos para a construção do Grafo de Visibilidade são:

- Passo 1: Criar um grafo vazio chamado G.
- Passo 2: Adicionar o ROBÔ (posição inicial) como um vértice em G.
- Passo 3:Adicionar o DESTINO (posição final) como um vértice em G.
- Passo 4: Adicionar todos os vértices de todos os polígonos do ambiente (obstáculos) em uma lista.
- Passo 5: Conectar todos os vértices do polígono em G de acordo com suas bordas.
- Passo 6: **Para cada vértice V na lista:**
  1. Tente ligar V ao ROBÔ (posição inicial) e ao DESTINO (posição final) . Se a linha reta entre eles não cruzar nenhum polígono, adicione esta linha como uma aresta em G.
  2. Para todos os outros vértices em todos os outros polígono, tente ligá-lo a V utilizando uma linha reta. Se a linha não cruza qualquer polígono, adicione a linha como uma aresta em G.
- Passo 7: Agora, execute um algoritmo no Grafo (Dijkstra ou A *) no Grafo de Visibilidade para encontrar o caminho mais curto entre o ROBÔ (posição inicial) e o DESTINO (posição final) .

As imagens contidas nas Figuras 2.9, 2.10, 2.11, 2.12 e 2.13 ilustram as etapas executadas para encontrar um caminho por meio do grafo de visibilidade.

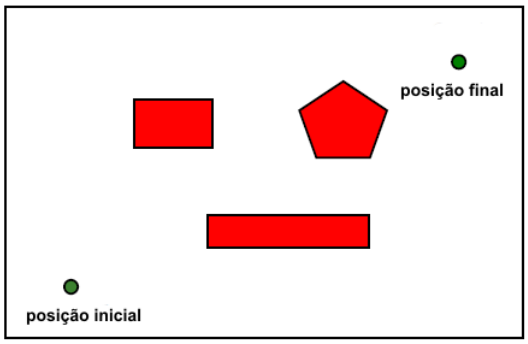

Figura 2.9: Ilustra a posição inicial do robô, o destino (posição final) e os obstáculos (polígonos) . Fonte: CHAIMOWICZ, 2014, [9]

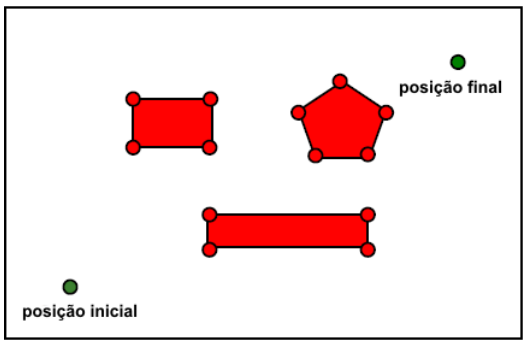

Figura 2.10: Cria um grafo vazio chamado G que contém os nós posição inicial e posição final. Adiciona todos os vértices de todos os polígonos (obstáculos) em uma lista. Fonte: CHAIMOWICZ, 2014, [9]

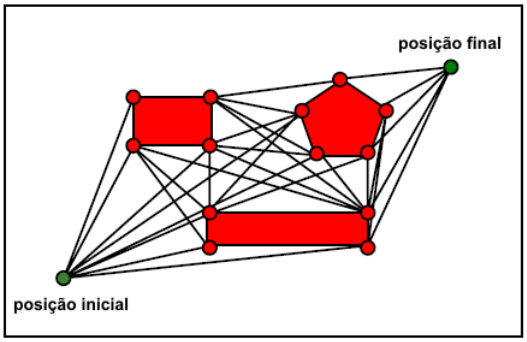

Figura 2.11: Execução dos passos números 5 e 6 do algoritmo. Fonte: CHAIMOWICZ, 2014, [9]

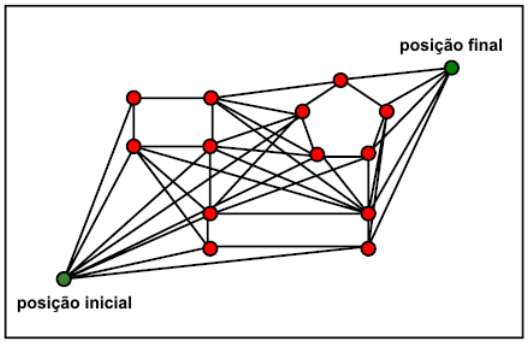

Figura 2.12: Grafo de Visibilidade construído. Fonte: CHAIMOWICZ, 2014, [9]

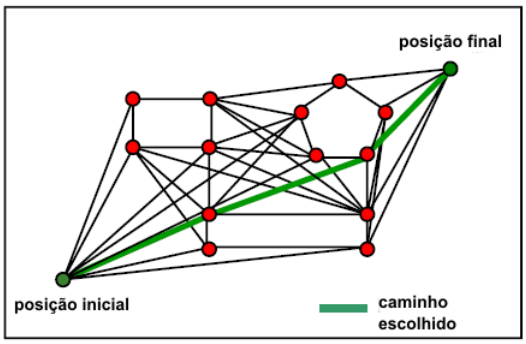

Figura 2.13: Execução de um algoritmo (Dijkstra ou A *) no Grafo de Visibilidade para encontrar o caminho mais curto entre o ROBÔ (posição inicial) e o DESTINO (posição final). Fonte: CHAIMOWICZ, 2014, [9]

As desvantagens da representação do espaço livre por meio do grafo de visibilidade são:

1. O tamanho da representação, o número de arestas e nós aumentam com o número de obstáculos poligonais, podendo ser lento ou ineficiente comparado a outras técnicas quando usado em um ambiente onde a densidade de obstáculos é grande [64].
2. Os caminhos encontrados no grafo podem fazer com que o robô passe muito próximo dos obstáculos, uma vez que os próprios vértices dos obstáculos podem fazer parte dos caminhos escolhidos.
3. Para utilizar essa técnica é necessário que o mapa seja completo e bem definido. Em geral, não é possível navegar em um ambiente que possua obstáculos móveis.

Diagrama de *Voronoi*

O Diagrama de *Voronoi* (DV) é uma estrutura geométrica que representa as informações de proximidade sobre uma série de pontos que representam os obstáculos dentro de uma dada região [54]. O DV é construído usando os pontos que são equidistantes de dois ou mais obstáculos. A borda (aresta) de *Voronoi* passa exatamente sobre a mediatriz de dois pontos vizinhos [43]. Os passos para a construção do Diagrama de *Voronoi* são:

- **Passo 1**: Escolhem-se dois pontos distintos A e B. Passa-se um traçado entre eles, formando um segmento de linha AB.
- **Passo 2:** Com o centro do compasso no ponto A traça-se um arco de circunferência com a largura do compasso para aproximadamente dois terços do comprimento do segmento de linha AB. Sem alterar a largura do compasso, desenhar um arco acima e abaixo do segmento de linha AB.

- **Passo 3:** Com o centro do compasso no ponto B traça-se um arco de circunferência com a mesma largura do compasso anteriormente utilizada. Desenha-se um arco acima e abaixo do segmento de linha AB.
- **Passo 4:** Traça uma linha entre os pontos onde os arcos se cruzam. Esta linha é perpendicular à primeira linha e corta-lo (que corta no ponto médio exacto da linha).
- **Passo 5:** Realiza-se os passos 1 a 4 para todos os demais pontos.

As imagens contidas na Figura 2.14 ilustram o traçado de uma mediatriz entre dois pontos.

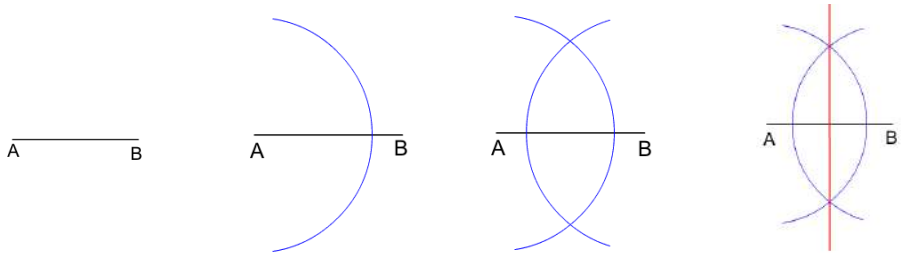

Figura 2.14: Etapas para traçar a mediatriz entre dois pontos.

As imagens das Figuras 2.15, 2.16, 2.17, 2.18, 2.19, 2.20 e 2.21 ilustram o processo de construção do diagrama de *Voronoi*.

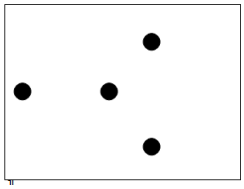

Figura 2.15: Representação dos obstáculos em 4 pontos distintos para construção do DV.

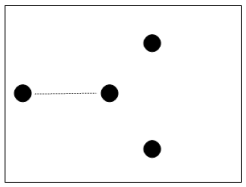

Figura 2.16: Realização do passo 1 para construção do DV.

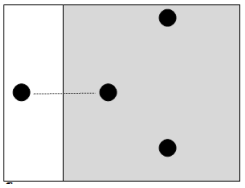

Figura 2.17: Realização dos passos 2 a 4 para construção do DV.

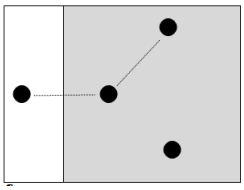

Figura 2.18: Repetição dos passos 1 a 3 para construção do DV.

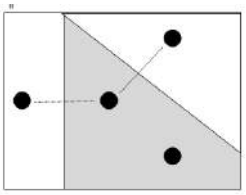

Figura 2.19: Realização do passo 4 para construção do DV.

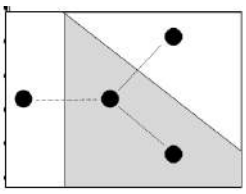

Figura 2.20: Repetição dos passos 1 a 3 para construção do DV.

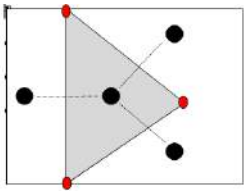

Figura 2.21: Realização do passo 4. Construção do DV finalizada. Fonte: Imagens adaptadas de HESHAM, 2009.

As vantagens do uso dos DV são:

1. Gera um caminho que na maioria dos casos permanece equidistante dos obstáculos, criando um caminho seguro para o robô se locomover [54].
2. O método é prático, pois o robô segue pela bordas (arestas) de *Voronoi* usando regras de controles simples, servindo como auto-estradas [43].

As limitações do uso dos DV são:

1. Essa abordagem tende a maximar a distância entre o robô e os objetos do ambiente [17].
2. Durante o processo de construção do DV, se o robô tiver sensores de curto alcance, isto pode prejudicar a construção do DV com a margem de segurança necessária [64].
3. O principal problema é que o comprimento do caminho é relativamente longo, nem sempre é o mais curto [17] [54].

2.3.2 Decomposição em Células

A técnica de Decomposição em Célula (DC) compreende em dividir o espaço livre em regiões simples e conectadas, chamadas células. Cada célula é representada por um vértice no grafo. Constrói-se então um grafo não-direcionado representando a relação de adjacência entre as células [64].

A construção do grafo é realizada da seguinte maneira. Se um robô pode se mover de uma célula para outra sem que haja colisões, então estas duas células são conectadas por meio de uma aresta. As células que poderiam resultar em colisões entre os obstáculos são removidas da representação, ou seja, a aresta não une estes dois vértices [17]. Assim, o problema do planejamento de caminhos é reduzido no problema de encontrar o menor caminho em um grafo da posição inicial e posição final.

Os métodos baseados em DC podem ser divididos em [48]:

- **Métodos exatos de decomposição em células:** decompõem o espaço livre em um conjunto de células cuja união cobre exatamente o espaço livre. As células possuem a mesma forma, a qual é bastante simples e definida a priori. A desvantagem é que em geral não é possível modelar o ambiente na forma exata como ele é, sendo necessário que algumas aproximações sejam feitas. Isto deve ser feito de uma forma conservadora, ou seja, de maneira que garanta que o robô não colida com os obstáculos. Por este motivo, o espaço livre modelado aproximadamente através das células tem que estar estritamente contido no espaço livre real do robô. Pode decorrer disto que não seja encontrado algum caminho entre duas configurações, embora exista.
- **Métodos aproximados de decomposição em células:** dividem o espaço livre em um conjunto de células de forma predefinida cuja união está estritamente contida no espaço livre. Esse método aproximado, pode-se ajustar a precisão, conforme desejado, mudando, para tanto, apenas o tamanho das células. Esse ajuste está intimamente relacionada à quantidade de espaço e de tempo de execução do planejador de trajetórias.

  A definição do tamanho das células leva em conta que [64]:

  1. células de tamanho grande reduzem o custo de computação (quantidade de memória necessária) e, principalmente, o tempo de execução;
  2. células muito grandes podem causar uma perda de precisão indesejada, fazendo com que não sejam encontradas trajetórias mesmo em situações em que isto é possível.

A principal vantagem do método aproximado sobre o exato é a de fazer a decomposição do ambiente em células através da iteração da mesma por meio computacional, que será simples por causa da forma das células. Assim, o método aproximado geralmente dá origem a sistemas mais simples de planejamento de trajetórias do que os que utilizam o método exato. Uma limitação é que o espaço de busca torna-se mais largo a medida que a resolução da representação do espaço livre for mais refinado [17]. As imagens contidas nas Figuras 2.22, 2.23, 2.24 e 2.25 ilustram as etapas realizadas para encontrar um caminho por meio do método de Decomposição em Células.

Dependendo do algoritmo escolhido para a busca do melhor caminho, podem ser encontrados outros caminhos, conforme ilustrado nas imagens das Figuras 2.26 e 2.27 referentes ao mesmo problema anteriormente mencionado, no qual utilizou a distância euclidiana como métrica.

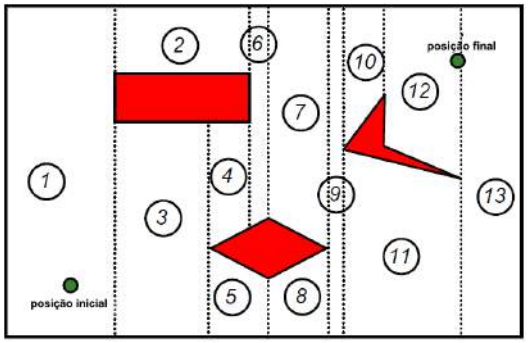

Figura 2.22: Divisão do espaço livre em células de tamanhos não exatos. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

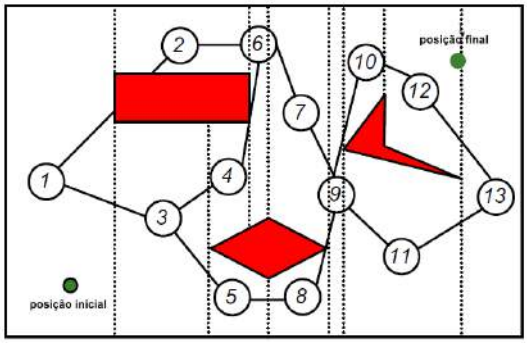

Figura 2.23: Conexão das células para o robô se movimentar sem colisão. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

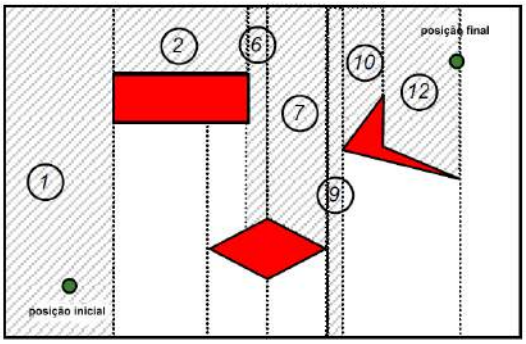

Figura 2.24: Aplicação do algoritmo para a escolha do caminho em DC. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

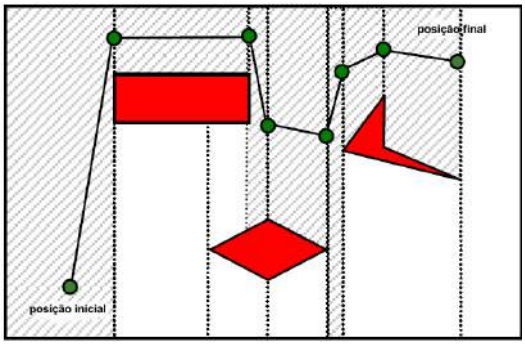

Figura 2.25: Caminho escolhido na técnica de DC. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

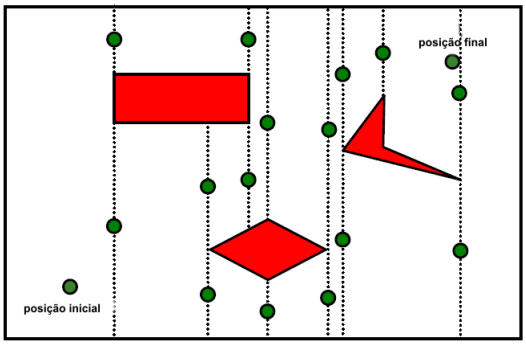

Figura 2.26: Representação das células com a técnica de DC. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

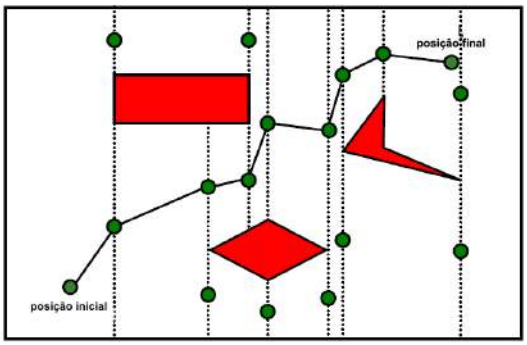

Figura 2.27: Aplicação do algoritmo para escolha do caminho. Fonte: Adaptado de CHAIMOWICZ, 2014, [9].

2.3.3 Campo Potencial

Na técnica denominada Campo Potencial [63] a navegação dos robôs é realizada por meio de campos potenciais, onde uma função matemática é definida sobre o espaço de configuração para o planejamento dos movimentos do robô [58].

O robô é tratado como uma partícula sobre a influência de um campo potencial que é modulado para representar a estrutura do espaço livre. Os obstáculos e o robô possuem a mesma carga e assim, as colisões entre os obstáculos e o robô são evitadas, pela força de repulsão entre eles. Essa força é calculada através do gradiente negativo do campo potencial. A direção de atração do robô ao objetivo é modelada por um campo que evita os obstáculos e conduz o robô até o objetivo [17].

Oussama Khatib [2] definiu um campo potencial no espaço de configurações com pelo menos um ponto de destino e vários obstáculos. O robô no campo potencial é atraído para o ponto de destino, enquanto é repelido por obstáculos no meio ambiente. Em linhas gerais, os campos potenciais possuem duas componentes [57]:

1. **Potencial atrativo**: tem a função de orientar o robô para o alvo, sendo modelado por um campo positivo. Atrativo a longas distâncias e nulo quando próximo do alvo.
2. **Potencial repulsivo**: funciona como uma barreira ao redor do obstáculo que impede a colisão, sendo modelada por um campo negativo. Quanto mais forte, significa que o robô está mais próximo do obstáculo. Ação repulsiva cresce próxima ao obstáculo e nula quando longe do obstáculo.

Descrição Formal do Método

Para descrever esse método, desconsideraremos as dimensões do robô, modelando-o como um ponto e consideraremos a posição do robô através das coordenadas representadas pela posição $q = (x, y)$, onde a posição do obstáculo é $q_{obstaculos} = (x_{obstaculos}, y_{obstaculos})$ e a posição final (objetivo) é $q_{objetivo} = (x_{objetivo}, y_{objetivo})$.

A definição de um campo potencial pode variar dependendo da abordagem ou implementação escolhida. A forma mais comum é definida por Kathib [2]. Em geral, são utilizadas duas componentes [63] para construir o campo potencial $U(q)$, são elas o potencial de atração $U_{atracao}(q)$ e o potencial de repulsão $U_{repulsao}(q)$. O robô fica sujeito ao campo potencial representado na Equação (2.1) e a seguir são demonstrados os campos potencias de atração ($U_{atracao}(q)$) e de repulsão ($U_{repulsao}(q)$) na forma de parábola.

$$U(q) = U_{atracao}(q) + U_{repulsao}(q) \tag{2.1}$$

Cálculo do Campo Potencial de Atração ($U_{atracao}(q)$)

A Equação (2.2) mostra a representação da função potencial de atração em forma de parábola [2]:

$$U_{atracao}(q) = \frac{1}{2}\eta d^2(q, q_{objetivo}) \tag{2.2}$$

onde:

- η representa o coeficiente de atração da função do campo potencial.
- $d(q, q_{objetivo})$ corresponde a distância Euclidiana entre o robô q e a posição de destino $q_{objetivo}$.

A força de atração no robô é calculada pelo gradiente negativo do campo potencial de atração (Equação (2.3)).

$$F_{atracao}(q) = - \bigtriangledown U_{atracao}(q) = -\eta d(q, q_{objetivo}) \tag{2.3}$$

Cálculo do Campo Potencial de Repulsão ($U_{repulsao}(q)$)

O campo potencial de repulsão é o resultado da combinação das forças de repulsão de todos os obstáculos (Equação (2.4))

$$U_{repulsao}(q) = \sum_{i}^{n} U_{repulsao_i}(q) \tag{2.4}$$

onde $U_{repulsao_i}(q)$ representa o potencial de repulsão gerado pelo obstáculo i, onde i é o número de obstáculos que influenciam o meio do robô.

A função de repulsão é representada pelas Equações (2.5) e (2.6).

$$U_{repulsao}(q) = \frac{1}{2}\eta(\frac{1}{d(q, q_{obstaculo})} - \frac{1}{d_0})^2 d(q, q_{objetivo})^n \quad \text{, se } d(q, q_{obstaculo}) < d_0 \tag{2.5}$$

$$U_{repulsao}(q) = 0 \text{ , se } d(q, q_{obstaculo}) > d_0 \tag{2.6}$$

Onde:

- q representa a posição do robô e $q_{obstaculo}$ corresponde a posição obstáculo.
- d_0 é a constante positiva que representa a distância da influência do obstáculo.
- $d(q, q_{obstaculo})$ é a distância entre o robô e o obstáculo.
- η é uma constante ajustável que representa o coeficiente de repulsão da função do campo potencial.

A força de repulsão é o gradiente negativo da função de repulsão (Equações (2.7) e (2.8)).

$$F_{repulsao}(q) = -\bigtriangledown U_{repulsao}(q) = \eta(\frac{1}{d(q, q_{obstaculo})} - \frac{1}{d_0})\frac{(q - q_{obstaculo})}{d^3(q - q_{obstaculo})} \text{ , se } d(q, q_{obstaculo}) < d_0 \tag{2.7}$$

$$F_{repulsao}(q) = 0 \text{ , se } d(q, q_{obstaculo}) > d_0 \tag{2.8}$$

Cálculo do Campo Potencial $(U(q))$

O campo potencial total $U(q)$ é a combinação dos potenciais de atração e de repulsão. A força $F(q)$ que é aplicada no robô para movê-lo para uma posição $q = (x, y)$ é obtida pelo gradiente negativo do campo potencial U, obtendo-se o vetor força a ser seguido pelo robô [64], conforme observa-se na Equação (2.9).

$$F(q) = -\bigtriangledown U(q) = -\bigtriangledown U_{atracao}(q) - \bigtriangledown U_{repulsao}(q) \tag{2.9}$$

onde $\bigtriangledown U(q)$ representa o gradiente negativo do vetor U na posição q.

A força calculada é aplicada diretamente na rodas do robô provocando o movimento. Essa força é definida como a soma de dois vetores de forças (atração e repulsão) (Equação (2.10). A combinação dessas duas forças geram uma força total com magnitude e direção. A cada nova pequena interação o robô permanece sendo guiado pelas forças locais que atuam sobre ele. O robô simplesmente se move na direção fornecida pelo cálculo do gradiente na posição q.

$$F(q) = F_{atracao}(q) + F_{repulsao}(q) \tag{2.10}$$

As imagens contidas nas Figuras 2.28 e 2.29 ilustram a representação do espaço de configurações por meio do campo potencial e com as componentes obstáculo (repulsão) e objetivo (atração). Também demonstram uma trajetória escolhida pelo robô para atingir um objetivo desviando do obstáculo. O tamanho da seta representa a magnitude da força e os ângulos das setas a direção.

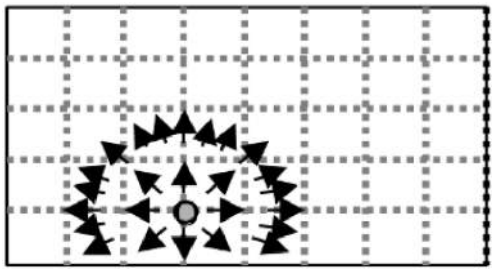

Figura 2.28: Representação de um obstáculo e as forças de repulsão. Fonte: MURPHY, 2000, [43].

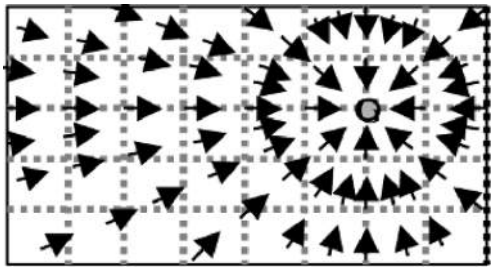

Figura 2.29: Representação de um objetivo e as forças de atração. Fonte: MURPHY, 2000, [43].

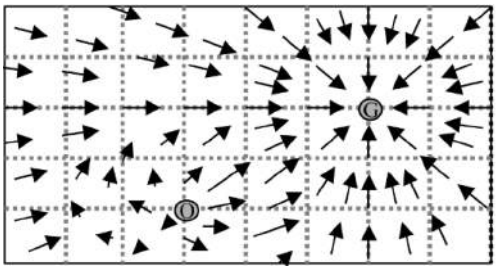

Figura 2.30: Visão geral da combinação das componentes do campo potencial. Fonte: MURPHY, 2000, [43].

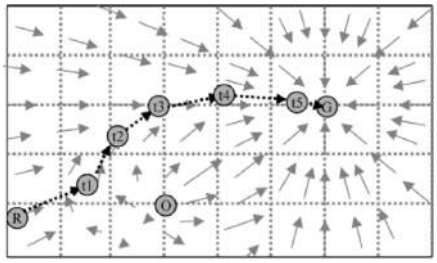

Figura 2.31: Caminho escolhido pelo robô. Fonte: MURPHY, 2000, [43].

Vantagens e Restrições do Uso de Campos Potencias

Os campos potenciais podem ser utilizados para planejamento de trajetórias em tempo real de execução [58]. Eles também possuem baixa complexidade computacional e permitem um comportamento emergente [20]. Algumas limitações desta técnica são:

- Saber como tratar os mínimos locais que podem ocorrer dentro de uma área representada por um campo potencial [17]. Essa questão dos mínimos locais pode gerar armadilhas para o robô, possibilitando que fique preso. Este problema ocorre quando a somatória dos campos é igual a zero. Analisando a Figura 2.28 verifica-se que atrás do obstáculo, os vetores representados por setas têm apenas uma cabeça (direção da seta) e nenhum corpo (comprimento da seta). Isto significa que a força do campo é zero e que, se o robô atingir esse ponto, ele vai parar e não se mover mais. Este é o chamado problema dos **mínimos locais**, porque o campo tem um potencial mínimo, ou vale, que retém o robô [43].
- As forças de atração e repulsão calculadas dependem das posições do obstáculo e do robô. Se a velocidade do robô for alta, de modo que o processamento do cálculo dos vetores não a acompanhe, podem ocorrer falhas [58].

Grande parte das soluções propostas sobre o problema do mínimo local, se baseiam na capacidade do robô conseguir detectar quando ele se encontra em uma região que possa ser um local de mínimo. Essa capacidade de detecção depende de vários fatores entre eles, do algoritmo utilizado e da sensibilidade dos sensores. As abordagens dessas soluções permitem a aplicação de planejadores de caminhos randômicos [63] com algoritmos iterativos aplicando um algoritmo de busca *Hill climbing* [17] como também o uso de obstáculos virtuais. Outra solução, é o emprego de campos potenciais harmônicos, técnica que utiliza funções que não possuem mínimos locais [17].

2.4 Representação de campos potenciais por meio da função potencial de *Lennard-Jones*

Carlo Pinciroli [52] [50] é autor de um método de ER baseado no princípio de formação de padrões. Com um comportamento auto-organizado, tal método se baseia na representação do campo potencial por meio da função potencial de ***Lennard-Jones (LJ)*** [34]. Esta estratégia é utilizada para a navegação dos robôs através da formação de arranjos locais entre robôs vizinhos. Nesta abordagem os resultados mostraram que a distância entre os robôs é mantida em torno de um valor desejado. O cálculo do campo potencial V de *Lennard-Jones* é descrita pela Equação (2.11).

$$V(\rho) = \varepsilon \left((\frac{\delta}{\rho})^{12} - 2(\frac{\delta}{\rho})^{6} \right), \tag{2.11}$$

onde a Tabela 2.7 contém a explicação de cada termo da fórmula.

Tabela 2.7 Descrição das variáveis do cálculo do potencial de LJ. Fonte: CARLO PINCIROLI, 2008 e 2011, [52] [50].

Variável	Descrição)
ρ	Consiste na distância entre os robôs vizinhos.
ε	Consiste no coeficiente que permite aumentar a força de repulsão da função de atração (profundidade do potencial).
δ	Consiste na média da distância entre o diâmetro de dois robôs na qual a força é mínima.

Uma força F pode ser derivada de um campo potencial V de acordo com a Equação (2.12).

$$F(\rho) \quad = \quad - \bigtriangledown V(\rho) = -\frac{12}{\rho}\varepsilon \left((\frac{\delta}{\rho})^{12} - (\frac{\delta}{\rho})^{6} \right) \tag{2.12}$$

Simplificando a Equação (2.12) obtém-se, a Equação (2.13).

$$F(\rho) \quad = \quad - \bigtriangledown V(\rho) = -\frac{4}{\rho}\varepsilon \left((\frac{\delta}{\rho})^{4} - (\frac{\delta}{\rho})^{2} \right) \tag{2.13}$$

A função potencial de *Lennard-Jones (LJ)* permite que os robôs se locomovam de forma coesa. Essa abordagem é denominada de ***flocking*** é um comportamento coletivo amplamente verificado na natureza [41] [46]. Nesse contexto [52] [50] , o *flocking* é um enxame composto por um grupo de robôs móveis terrestres. O método *Flocking* é usado da seguinte forma: assume-se que os robôs para se moverem devem obedecer as coordenadas de um vetor de *flocking* denominado por $\vec{f}$. A expressão completa dessa equação pode ser vista na Equação (2.14).

$$\vec{f} = \vec{p} + \vec{g}, \tag{2.14}$$

onde:

- $\vec{p}$: é o vetor de controle proximal que codifica as regras de atração e repulsão. Para alcançar a coesão, um robô precisa manter uma certa distância de seus vizinhos;
- $\vec{g}$: é o vetor de direção em relação ao alvo, codifica o comportamento dos robôs.

Com base na Equação (2.14), através das leituras dos sensores dos robôs pode-se calcular as distâncias com os robôs vizinhos. Posteriormente, obtêm-se o valor do campo potencial virtual. Aplicando a força derivada do campo potencial de LJ pode-se movimentar os robôs integrantes do enxame. Os resultados são movimentos ordenados permitindo que o enxame se locomova de forma coesa a uma direção comum.

O planejamento da navegação dos robôs com base no potencial de *Lennard-Jones* obedece as seguintes regras:

(a) Se dois robôs estão demasiadamente próximos, irão estar sujeitos a uma força de repulsão, afastando-os um do outro.

(b) Se estiverem muito longe, eles estarão sujeitos a uma força de atração, empurrando-os para perto um do outro.

(c) Se eles estão em uma distância preestabelecida, nenhuma força atuará sobre eles.

A força $F(\rho)$ calculada é empregada na atuação das rodas de cada robô (para frente, para trás, virar). Essa força permite que os robôs se movam em grupo de forma coesa. O algoritmo *Flocking* é subdivido em três passos básicos:

- **Passo 1:** Execução do algoritmo VectorToLigh que obtém a posição do alvo ($pos1(x, y)$).
- **Passo 2:** Execução do algoritmo FlockingToLigh que obtém a posição dos robôs vizinhos ($pos1(x, y)$).
- **Passo 3:** Execução do algoritmo ConfiguraVelocidade que soma as $pos1(x, y)$ e $pos2(x, y)$.

O algoritmo `VectorToLight` calcula um vetor normalizado que aponta para a posição mais próxima do alvo. Este cálculo é realizado com base nas leituras dos sensores de luz do robô. Uma vez que são 24 sensores dispostos igualmente em forma circular em torno do robô, através da posição do sensor no robô, tem-se o ângulo do alvo em relação ao sensor. A sequência de passos do algoritmo é descrita a seguir e a Figura 2.32 apresenta o esquema da disposição dos sensores de luz do robô.

- **Passo 1:** O robô realiza a leitura dos sensores para detectar o alvo.
- **Passo 2:** Para cada sensor que detectou o alvo:
  - **Passo 2.1:** Obtém o valor de leitura do sensor que varia entre $[0, 1]$.
  - **Passo 2.2:** Calcula as posições x e y do alvo detectado, calculando o seno e o cosseno por meio da multiplicação do valor do valor de leitura e do ângulo (posição) do sensor que detectou o robô vizinho.
  - **Passo 2.4:** As posições x e y são armazenadas em uma variável $pos(x, y)$ que vai acumulando o valor das posições.
- **Passo 3:** Normaliza a variável $pos1(x, y)$ dividindo pelo número de robôs vizinhos detectados.

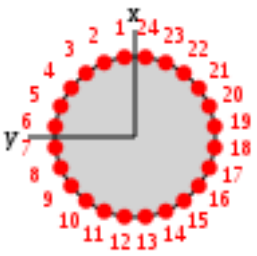

Figura 2.32: Ilustração do esquema da disposição dos sensores de luz do robô, segundo o algoritmo *Flocking*. Fonte: CARLO PINCIROLI, 2014, [49].

O algoritmo `FlockingVector` verifica quais são os robôs vizinhos e calcula as coordenadas x e y. Depois, realiza o cálculo das regras de atração e repulsão, mantendo a coesão do enxame. A sequência de passos é descrita a seguir:

- **Passo 1:** O robô verifica quais são os seus robôs vizinhos.
- **Passo 2:** Para cada robô vizinho detectado:
  - **Passo 2.1:** Calcula a distância o robô e o robô vizinho.

  - **Passo 2.2:** Calcula o campo potencial de *Lennard-Jones* com base na distância do robô vizinho.
  - **Passo 2.3:** Calcula as posições x e y que o robô irá se deslocar, calculando o seno e o cosseno por meio da multiplicação do valor do campo potencial e do ângulo do sensor que detectou o robô vizinho.
  - **Passo 2.4:** As posições x e y são armazenadas em uma variável $pos(x, y)$ que vai acumulando o valor das posições.

- **Passo 3:** Normaliza a variável $pos2(x, y)$ dividindo pelo número de robôs vizinhos detectados.

O algoritmo `ConfiguraVelocidade` recebe um vetor que é a soma das direções de outros 2 vetores (posição do alvo e posição dos robôs vizinhos). O primeiro vetor aponta o mais próximo do alvo e o segundo vetor contém as coordenadas para cada robô manter-se coeso. O robô verifica se a sua trajetória está correta para alcançar o alvo. Caso contrário, ele a corrige. A sequência de passos do algoritmo é descrita a seguir e a Figura 2.33 apresenta o esquema da disposição das rodas do robô.

- **Passo 1:** Soma-se o valor das variáveis $pos1(x, y)$ e $pos2(x, y)$ e adiciona resultado na variável $alvo(x, y)$.
- **Passo 2:** Calcula-se o ângulo formado entre as posições x e y do alvo detectado.
- **Passo 3:** Calcula-se o ângulo formado entre as posições x e y do próprio robô.
- **Passo 4:** O robô verifica se a sua trajetória está correta para alcançar o alvo. Caso contrário, ele a corrige.
  - **Passo 2.3.1:** Menor ou igual a 10°, então o robô continuará seguindo em frente, pois a diferença entre os ângulos (robô e alvo) é muito pequena.
  - **Passo 2.3.2:** Maior que 90°, então as rodas do robô serão configuradas para moverem com velocidades máximas opostas. O alvo pode estar atrás do robô. O robô irá dar uma meia-volta.
  - **Passo 2.3.3:** Maior que 70° e menor que 90° então o robô irá apenas realizar uma suave virada. Ambas as rodas irão mover para frente, mas com a velocidade de uma delas um pouco menor que a outra.
  - **Passo 2.3.4:** Demais situações, o robô permanece seguindo em frente.

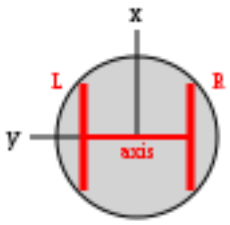

Figura 2.33: Ilustração esquema da disposição das rodas do robô, segundo o algoritmo *Flocking*. Fonte: CARLO PINCIROLI, 2014, [49].

Os códigos fontes do algoritmo *Flocking* constam no Apêndice:

- argos/controllers/footbot/floocking.cpp (ver Código A.1)
- argos/controllers/footbot/floocking.h (ver Código A.2)

Capítulo 3

Materiais e Métodos

Como expectativa pretende-se que o uso da abordagem de ER baseada em campos potenciais de *Lennard-Jones* seja viável no contexto do problema da dispersão de pássaros em áreas agrícolas. Com o objetivo de confirmar essa hipótese, procurou-se elaborar provas por meio de experimentos. Esses experimentos servem de subsídio para as investigações relativas a problemática da prova, permitindo ou não, a comprovação da hipótese inicialmente formulada no item 1.3, do Capítulo 1, dessa dissertação.

Para o levantamento dos dados durante a fase investigativa dos experimentos, optou-se pelo método de simulação computacional, o qual possibilita a coleta de resultados utilizados como meios probatórios dos fatos analisados. A simulação é um método para reproduzir possíveis cenários predeterminados de um ambiente real. A determinação dos cenários compreende a abstração do mundo real, restringindo determinadas variáveis consideradas não importantes a ponto de invalidarem a conclusão dos resultados analisados. A seguir são apresentadas as decisões e as respectivas justificativas nessa fase da pesquisa.

3.1 Determinando as características do Enxame de Robôs

Nesse trabalho o ER terá características físicas idênticas. O enxame será homogêneo e com capacidade de locomoção terrestre. Cada indivíduo realizará as mesmas tarefas. O enxame não terá conhecimento da extensão da área que irá monitorar. O enxame não possuirá capacidade de comunicação. Apenas pelo sensor de presença poderá um robô determinar a sua distância em relação a outro robô e em relação ao pássaro.

3.2 Determinando a Coesão do Enxame de Robôs

Inicialmente, a questão era determinar como um grupo de robôs poderia cobrir fisicamente a área agrícola, de modo que eles pudessem se mover próximos uns dos outros (em grupo), mas não tão perto, a ponto de colidirem e nem tão longe, a ponto de se dispersarem.

Voltando para a natureza com o objetivo de abstrair uma solução para esse problema, verifica-se que alguns animais voam em bando, como exemplo, os estorninhos *(Sturnus vulgaris)*. Esse comportamento animal coletivo na biologia é chamado de *flocking*, onde a ordem global emerge da auto-organização de cada indivíduo do grupo. Nos estorninhos não há nenhum líder no grupo. Os movimentos coletivos são uma consequência única de interações locais entre os indivíduos [8]. Cada indivíduo troca informações com o seu vizinho que, por sua vez, troca com os seus demais vizinhos propagando esse comportamento pelo enxame. Dessa forma, as informações são propagadas para todo o grupo. Entre essas informações, pode-se mencionar o alinhamento e a velocidade que resultam no movimento do bando e no rearranjo dos

indivíduos dentro do grupo, originando formações diferentes durante o voo.

As imagens da Figura 3.1 ilustram que não apenas os estorninhos, mas os insetos sociais também formam padrões espaciais complexos na construção de ninhos por exemplo cupinzeiros, através de mecanismos básicos da auto-organização e estigmergia. Neste contexto, a palavra padrão se refere a um arranjo de objetos que determina um lugar geométrico, uma relação estatística ou matemática. Os insetos são equipados com um sistema sensorial-motor que os habilitam a responderem aos estímulos. A grande variedade de padrões construídos pelos insetos sociais podem ser resultados da variedade de estímulos que eles recebem [67].

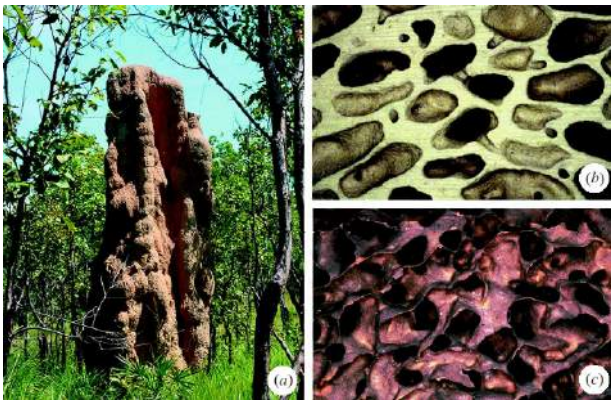

Figura 3.1: Construção de padrões espaciais por insetos sociais. (a) um cupinzeiro construído por *Nasutitermes triodiae*. (b) Uma seção transversal do ninho da espécie *Cubitermes* mostrando a sua estrutura alveolar. (c) Uma seção transversal de um ninho da espécie *Lasius fuliginosus* mostrando a sua estrutura esponjosa. Fonte: SCOTT CAMAZINE & GUY THERAULAZ, 2003, [67].

Como resposta para uma estratégia de controle dos robôs, verificou-se que uma possibilidade seria utilizar essa estratégia do *flocking* dos animais sociais ao voarem em bando, ou das formigas e cupins na formação de padrões espaciais. A ideia é de que o ER se mova para uma determinada direção e que cada robô contribua para a constituição de uma configuração geométrica (um **padrão**), de forma autônoma, o qual seria a estratégia de controle do grupo de robôs.

A abordagem de dispor os robôs geograficamente com base em formação de padrões permite usá-los para monitorar áreas de diferentes tamanhos, possibilitando que os robôs se locomovam de forma coesa (agrupada). As imagens contidas na Figura 3.2 ilustram exemplos de padrões de formação de um grupo de robôs: anel, totalmente conectada, malha, estrela, toroidal e árvore [40].

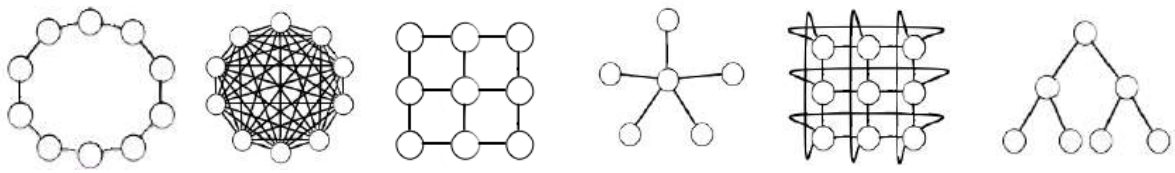

Figura 3.2: Ilustrações de padrões: anel, totalmente conectada, malha, estrela, toroidal e árvore, respectivamente da esquerda para a direita. Fonte: MEDINA, 2009, [40].

Depois de realizada a revisão teórica, a estratégia de planejamento de caminhos dos robôs escolhida foi a de Campos Potenciais com base na função potencial de *Lennard-Jones*, pela sua adequação à geração de trajetórias individuais para cada integrante do ER, tornando-o coeso.

Para implementar essa ideia foi utilizado o algoritmo `Flocking`. Este algoritmo baseia-se na utilização de forças virtuais para a formação de um hexágono. A construção do hexágono é feita através de um círculo de raio R (Fig. 3.3). Assim, quando os robôs estão a uma distância menor de R são repelidos e os robôs que estão a uma distância maior que R são atraídos. Assim como o bando de estorninhos mudam a formação durante o voo, a mesma situação ocorre com o ER durante o processo de monitoramento da área e posterior tentativa de captura do pássaro.

O próximo passo foi definir o comportamento dos robôs diante do pássaro após localizá-lo.

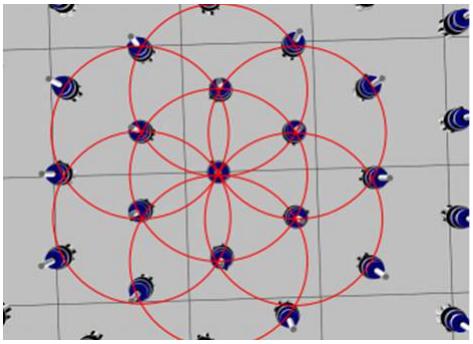

Figura 3.3: Padrão de formação com robôs móveis. Fonte: CARLO PINCIROLI & MANUELE BRAMBILHA, 2014, [51].

3.3 Delimitando o Comportamento dos Robôs Diante do Pássaro

Indivíduos da mesma espécie muitas vezes cooperam em suas caçadas para alimentar a prole, por exempo, os leões [19]. O grupo de animais toma decisões em consenso perseguirem uma presa coletivamente citeCouzin2005. Eles correm atrás de sua presa até que ela fique encurralada. Em geral, a vítima acaba ficando no centro e os demais animais em volta, se aproximando cada vez mais e atacando-a. Esse tipo de comportamento são relatados em Josué Ribeiro (2002) [46] e sua equipe nos quais mencionam que as garças podem unir-se às cegonhas *(Jabiru mycteria e Mycteria americana)* e caçar ativamente em bandos cercando os cardumes de peixes. Eles também descrevem que os biguás *(Phalacrocorax brasilianus)* pescam em bandos de até duzentos indivíduos, cercando as presas para obterem maior sucesso na captura. Outros relatos ainda mencionam o gavião-de-asa-telha *(Parabuteo unicinctus)* caçando em bandos de até seis indivíduos, o que permite cercar e capturar presas maiores, como lebres e coelhos [41].

A ideia desse trabalho é realizar experimentos nos quais os robôs realizam a caça do pássaro, perseguindo por meio de um enxame coeso mediante trabalho em grupo. Nessa situação o pássaro tenderá a ficar cercado pelos robôs. Os robôs não colidem com o pássaro e nem com os próprios robôs integrantes do enxame. O padrão de cerco consiste no pássaro rodeado (cercado) pelos robôs.

3.4 Definindo a Dinâmica dos Experimentos

Considere um enxame de N robôs móveis, sendo $R_1, \cdots, R_N$, deslocando-se em um plano representado no $\mathbb{R}^2$. Denotamos as posições de cada robô R_i (com $1 \leq i \leq N$) por $p_i = [x_i, y_i]^T$. Assume-se que cada robô R_i deve iniciar em posições $p_i = [x_i, y_i]^T$ suficientemente próximos do robô vizinho para garantir a convergência da formação pela função potencial. O problema consiste em projetar as entradas de controle $u_i = [u_{i1}, u_{i2}]^T$ para movimentar cada robô R_i tal que, dada uma determinada área, será verificado se o enxame consegue cercar o pássaro.

A definição da dinâmica dos experimentos engloba:

- O tamanho da área a ser monitorada pelos robôs representando a região agrícola.
- A região de partida dos robôs.
- A representação da dinâmica do pássaro nas coordenadas (x, y) com suas posições inicial e final.
- A definição de qual tipo de robô utilizar.

Foi escolhida uma área quadrada, com 400 m^2. Tais medidas foram escolhidas de modo a possibilitar a visualização das imagens durante a análise das simulações, devido as limitações do simulador ARGoS. A região de partida dos robôs será representado por uma área quadrada de 1 m^2 nas coordenadas (x, y) abaixo especificadas, conforme mostrado na Figura 3.4.

- $x = 4; y = 4$

- $x = 5; y = 5$

Definiu-se que os robôs serão introduzidos em posições aleatórias dentro da região de partida.

Estipulou-se que o pássaro se movimentaria durante a execução dos experimentos na trajetória das posições (x, y) inicial e final, conforme ilustrado na Figura 3.4 .

- inicial: $x = 3; y = 17$
- final: $x = 17; y = 6$

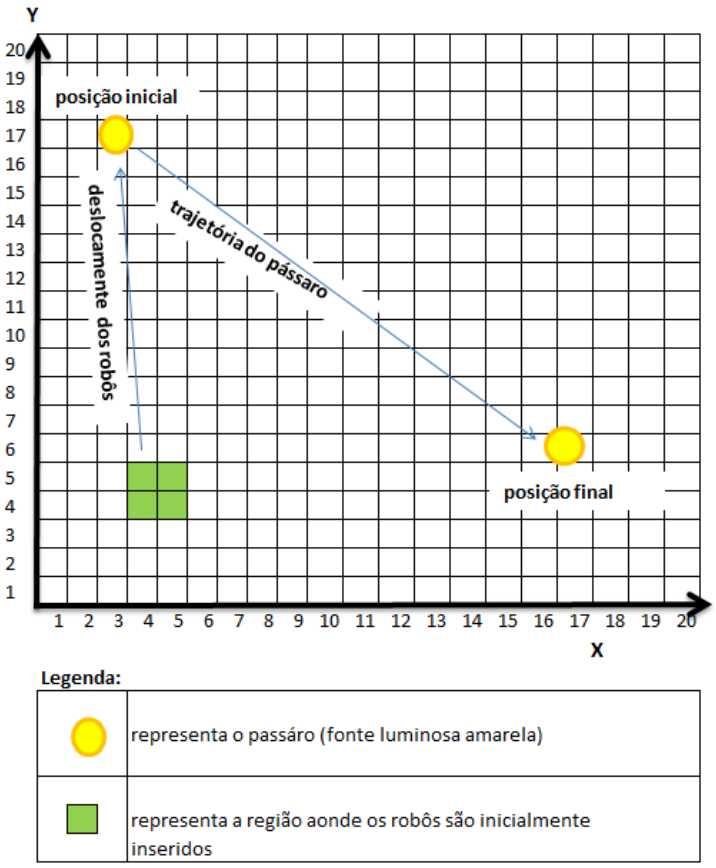

Figura 3.4: Representa a dinâmica do experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Em uma situação real, os pássaros intercalam períodos de voo e repetidamente pousam na vegetação. Nesta simulação, tal fato foi abstraído pela introdução de uma velocidade média na trajetória geométrica do pássaro. Os robôs sucessivamente tentam cercá-lo até retirá-lo da região agrícola. Para isso, o ER deve ter:

- Algum tipo de identificação para que um integrante do conjunto identifique o outro de forma a realizar o cálculo da distância.
- Ter algum sensor que permita localizar um pássaro na área monitorada.
- Um algoritmo para que cada robô realize a leitura dos seus sensores para saber a distância do pássaro e de cada robôs do enxame.
- Estipular uma determinada distância para limitar o número de vizinhos mais próximos que cada robô considerará para o cálculo.

O comportamento do ER será acionado dadas as condições iniciais mais as forças de interação entre os robôs que constituem o enxame. O uso da técnica de campos potenciais para gerar as trajetórias

individuais de cada robô, permitirá observar que o enxame move-se de forma coesa e alinhada. Para evitar colisões, cada robô também deve considerar a posição dos outros robôs para planejar sua trajetória.

Estipulou-se a distância de 1,8m para o cálculo da distância dos n-vizinhos. Com base nessa distância o algorítimo deve iniciar o cálculo do campo potencial de *Lennard-Jones* para todos os robôs. Esse cálculo permite a coesão na realização da tarefa. Assim, cada integrante do ER que antes, estavam muito próximos um dos outros (na região de partida) passam a se repelir e a se atrair, se distanciando uns dos outros.

Em seguida, os sensores dos robôs detectam o pássaro e o enxame se desloca, de forma agrupada, em direção do mesmo. A cada instante de tempo o pássaro vai se movimentando do seu ponto inicial até o seu ponto final e os robôs permanecem cercando-o. O pássaro voa a uma altura constante de 1 metro.

Assim, cada robô, utilizando os campos potenciais, calcula sua trajetória até sua meta, desviando dos demais robôs. A meta estabelecida para os robôs é monitorar área, e caso detecte um pássaro, persegui-lo, cercando-o e dispersando-o da área. Será considerado que os robôs possuem capacidade reativa, desconhecendo o ambiente no qual irão ser introduzidos.

3.5 Definindo como Realizar os Experimentos

Depois de escolhida a estratégia de controle dos robôs no espaço geográfico, a forma de dispersar o pássaro e a forma conceitual da dinâmica dos experimentos, o próximo passo era testar a proposta e verificar a viabilidade. Para isso, inicialmente pensou-se em simular através de robôs reais em uma área real, mas devido a falta de recursos financeiros para a compra de robôs, optou-se pelo uso de ferramentas visuais em 2D ou 3D.

A simulação computacional foi escolhida como processo metodológico por ser uma boa ferramenta para a compreensão dos fenômenos físicos. Justifica-se a aplicação da simulação virtual (SV) como ferramenta de auxílio, por uma série de motivos, entre eles se destacam:

1. A SV é importante para prototipar soluções envolvendo robôs.
2. Um simulador torna possível testar ideias em um ambiente seguro, evitando possíveis danos aos robôs reais, que em geral, possuem um custo significativo [53].
3. Uma experiência envolvendo um grande número de robôs pode ser muito cara, inviabilizando o projeto. Simulações têm um custo bem menor.
4. As simulações podem ser repetidas várias vezes, com todas as variações de parâmetros possíveis.
5. Permite a aplicação dos algoritmos de controle do enxame em um ambiente 3D.

As buscas se concentraram em simuladores gratuitos e com código aberto. Entre eles podemos citar os simuladores *V-REP Rep Pro* [22], *Gazebo* [35] e *Stage* [24]. Durante essa fase, a falta de computador com as configurações mínimas de *hardware* para execução das simulações foi um fator crítico. A grande maioria dos simuladores finalizava a execução quando eram acrescentados novos robôs nos cenários, sendo necessários processadores e placas de vídeos mais potentes. Devido a restrição orçamentária, a pesquisa teve continuidade em busca de um simulador que pudesse contornar essa dificuldade de falta de *hardware*. Durante essas buscas, foi encontrado o simulador ARGoS *(Autonomous Robot Go Swarming)* o qual foi escolhido para realizar os experimentos.

O *software* ARGoS [52] é um simulador 3D, gratuito, específico para simulações voltadas para abordagem de ER. Ele funciona em *Linux* e *Mac OS X* e foi desenvolvido durante o projeto *Swarmanoid*, o qual tem por objetivo o estudo de ferramentas e estratégias de controle de ER. O ARGoS suporta os robôs *swarmanoid*, *foot-bot* e *e-puck* [53] e o código de controle do robô é escrito na linguagem de programação `C++` , usando configurações através de arquivos XML. Durante o desenvolvimento do projeto *Swarmanoid* a flexibilidade e a eficiência foram requisitos essenciais inseridos no simulador ARGoS da seguinte maneira [52]:

1. A **eficiência** foi obtida por meio de uma arquitetura *multithread*, visando à otimização do uso da CPU. Nesse contexto, a eficiência refere-se a capacidade de executar experimentos envolvendo vários robôs no menor tempo possível. Isso permite que a simulação contenha um

número elevado de robôs e um bom funcionamento. Uma característica marcante nesse simulador é que no ambiente 3D simulado, o espaço da simulação pode ser dividido em subespaços, e cada um deles pode ser administrado por diferentes tipos de gerenciadores de leis de física [53];

2. Já a **flexibilidade** foi obtida por meio de uma arquitetura orientada a *plug-ins*, através dos quais é possível obter modelos de robôs, sensores, atuadores, visualizações e outros. Os usuários podem escolher quais *plug-ins* utilizar para cada aspecto da simulação e atribuir recursos apenas aonde interessa [5]. Essa característica permite que o usuário obtenha apenas os recursos necessários para a simulação.

3.6 O Robô Utilizado e suas Características

Uma vez decidido que os experimentos seriam implementados por meio de uma simulação virtual, o próximo passo foi escolher qual o tipo de robô a ser representado na simulação. Verificou-se que o robô disponibilizado pelo ARGoS, o *foot-bot* [7], era apropriado para a realização da experiência no ambiente virtual escolhido. O *foot-bot* possui os seguintes sensores e atuadores:

1. **Sensores de Luz**: o sensor de luz permite que o robô calcule sua distância em relação à outra fonte de luz presente no ambiente [66]. Cada robô tem 24 sensores de luz, igualmente, distribuídos em um anel em torno do seu corpo. Cada leitura do sensor é constituída por um ângulo em radianos e um valor no intervalo [0,1]. O ângulo corresponde ao local onde o sensor está localizado no corpo em relação à parte dianteira do robô que é o local de x. O valor zero corresponde a ausência de luz enquanto valores maiores que 0 significam que a luz foi detectada. O valor aumenta à medida que o robô se aproxima de uma fonte de luz.

2. ***LED*s RGB**: o robô possui 13 *leds* RGB, sendo 12 desses dispostos em um anel em torno do corpo do robô, e um *led* desses está posicionado na parte superior do corpo do robô.

3. **Pinça**: ela permite ao robô se conectar (pegar) a objetos, podendo até transportá-los.

4. **Sensores de Proximidade**: são 24 sensores distribuídos, igualmente, em um anel em torno do corpo do robô. Servem para detectar objetos próximos, possibilitando evitar colisões. Cada um dos sensores retornam uma leitura constituída por um ângulo em radianos e um valor no intervalo [0,1]. O ângulo corresponde ao local onde o sensor está localizado no corpo em relação à parte dianteira do robô, que é o local do eixo x. O valor 0 corresponde a nenhum objeto detectado pelo sensor, enquanto que os valores maiores que 0 significam a detecção. O valor aumenta à medida que o robô se aproxima do objeto.

5. **Sensor de Comunicação**: através do dispositivo de comunicação cada robô pode se comunicar com seus vizinhos. Essa comunicação somente ocorre se eles estiverem na linha direta de visão, sem nenhum objeto entre os dois robôs.

6. **Rodas**: são dois conjuntos de rodas com movimentos de direção e velocidades independentes.

3.7 As limitações e o Ambiente Computacional

A principal limitação desse trabalho foi a falta de recursos financeiros. Não foram considerados nos experimentos o coeficiente de atrito do solo, o tipo de solo (úmido, seco ou arenoso), condições ambientais (chuva ou vento) e terrenos irregulares e inclinados. O simulador ARGoS foi instalado no sistema operacional *Ubuntu* 64 bits, versão 12.04, em um computador com processador *Intel Core* i7 Q 740 @ 1.73 GHz e com 8 GB de memória. Foram executadas algumas simulações básicas e o simulador apresentou um bom funcionamento permitindo verificar a flexibilidade e a eficiência do simulador.

Capítulo 4

Experimentos Efetuados

O próximo passo foi executar os experimentos segundo os critérios definidos no Capítulo 3. Primeiramente, foi realizado um experimento inicial que serviu de base para a coleta de dados e ajuste de parâmetros e, posteriormente, prosseguiu-se com uma série de outros experimentos.

4.1 Experimento Inicial

Foram estipulados 14 robôs para monitorarem a região. No início da simulação, os robôs são dispostos nas coordenadas (x,y) especificadas, as quais representam a região inicial.

Cada robô tem um dispositivo de identificação que foi implementado nas simulações por um *led*, ligado na cor vermelha, localizado na parte superior do corpo do robô.

Na arena também são dispostos outros elementos para compor o cenário, entre eles 35 cilindros estáticos que possuem massa de 0,5 kg e raio de 10 cm, distribuídos em fileiras representando a vegetação. Também foi inserida uma fonte luminosa na cor amarela representando o pássaro se movimentando.

Em seguida, cada robô deve realizar a leitura de seus sensores para verificar se localizam a fonte de luz amarela, a sua distância em relação à fonte e a sua distância em relação a cada um dos seus robôs vizinhos que possuem um *led* aceso na cor vermelha.

4.1.1 Configuração dos Arquivos para Simulação

Para a execução do experimento proposto foi configurado o ambiente do simulador. Foram realizadas várias alterações no arquivo "`Flocking.argos`", conforme Códigos B.3 constante no Apêndice. Esse arquivo contém as seguintes seções básicas:

(a) *Framework*: Definição dos parâmetros internos do simulador, entre eles:
   i. Se o experimento será executado com suporte de *threads*.
   ii. O tempo de duração da experiência em segundos (se definir um valor igual a zero a experiência não tem limite de tempo).
   iii. O comprimento do passo de integração (controla quantas vezes por segundo cada passo da simulação é executado).
   iv. A semente aleatória é opcional e, se não estiver presente o seu valor é definido a partir do relógio interno. Se o valor da semente aleatória é fixo, isso permite obter os mesmos resultados através de muitas repetições do mesmo experimento.

(b) *Controllers*: Contém a lista dos controladores definidos pelo usuário. Cada controlador interage com o ambiente através de sensores e atuadores. Os atuadores modificam o estado do robô e os sensores leem esses estados.

(c) *Arena*: Contém a lista das entidades para adicionar à arena no início da simulação e as suas posições iniciais. Permite especificar quais objetos serão inseridos na arena e como eles serão distribuídos no ambiente, se de forma automática ou em posições fixas previamente determinadas.

(d) *Physics_ engines*: Contêm os motores de física que serão usados e como serão conectados uns aos outros.

(e) *Visualization*: contêm as configurações da visualização das cenas (câmeras).

(f) *Loop_functions*: Contêm as configurações das funções do laço. Essas funções são opcionais e interagem com o experimento em execução alterando o seu comportamento.

Destacam-se as seguintes alterações realizadas no arquivo "`Flocking.argos`":

(a) Entre as linhas de código números 11 a 16, foram definidas as configurações gerais da simulação, por exemplo, a duração do experimento e a semente a aleatória.

(b) Entre as linhas de código números 23 a 42 foram definidos os atuadores e os sensores dos robôs e as respectivas configurações.

(c) Entre as linhas de código números 46 a linha 50 foi inserida a chamada de um novo método, *My _Loop _Functions*, para compor a biblioteca de funções *loop* do simulador ARGoS. Esse método foi implementado para ser invocado em intervalos de tempo, previamente definidos, para simular a movimentação do pássaro. Na simulação o pássaro foi representado por uma fonte de luz na cor amarela. Durante as simulações as posições iniciais e finais do pássaro foram alteradas para verificar o comportamento do grupo de robôs. No Apêndice constam os Códigos B.1 e B.2 nos quais o pássaro se move na plantação da posição p_1 para a posição p_2:

- p_1: $x = 3; y = 17$
- p_2: $x = 17; y = 6$

(d) Na linha de código número 55, foram definidos o comprimento e a largura da arena quadrada (20 m x 20 m) para representar a área agrícola.

(e) Entre as linhas de código números 57 a 62 foram definidas os parâmetros referentes ao pássaro na simulação, entre eles, a posição, a orientação, a cor, e a intensidade do brilho do *led*.

(f) Entre as linhas de código números 67 a 76 foram definidos os parâmetros referentes à distribuição de 35 cilindros na arena para representar a vegetação. Os cilindros são estáticos e foram inicializados os valores referentes ao $raio = 10kg$ e $peso = 0,5kg$. Também foi definido que os cilindros seriam dispostos na arena seguindo o método *grid* (em quadrados) com distância de um *grid* para cada cilindro.

(g) Entre as linhas de código números 98 e 106 foi definido aonde inicialmente o simulador irá lançar o enxame composto por 14 robôs. Essa área foi definida pelas coordenadas (x, y):

- $x = 4; y = 4$
- $x = 5; y = 5$

4.1.2 Resultados do Experimento Inicial

As Figuras 4.1 e 4.2 ilustram o início da execução da simulação. Elas apresentam a arena na qual estão dispostos 14 robôs. Verifica-se que os robôs possuem um *led* aceso na cor vermelha indicando que pertencem ao mesmo ER.

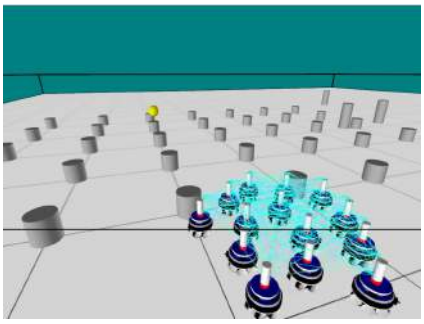

Figura 4.1: Início da simulação. Fonte: Imagens extraídas do simulador ARGoS.

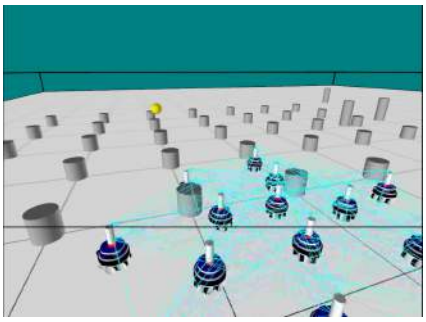

Figura 4.2: Execução do algoritmo *Flocking*. Fonte: Imagens extraídas do simulador ARGoS.

Também é possível verificar a vegetação representada pelos cilindros enfileirados na cor cinza. Inicialmente os robôs se encontram muito próximos um do outro sem nenhum padrão específico. Tal situação está ilustrada na imagem da Figura 4.1.

Dentro da arena e no meio dos cilindros (vegetação), é inserida uma esfera amarela que se desloca acima da superfície a uma altura de 1 metro. Essa esfera possui mobilidade e é identificada pelos sensores de luz dos robôs. Tal esfera, em nossa simulação, representa o pássaro voando no meio da plantação.

Nas Figuras 4.3 e 4.4 o ER realiza o cálculo do campo potencial de *Lennard-Jones* e os robôs que antes estavam muito próximos uns dos outros passam a se repelir e a se atrair gerando um padrão de formação.

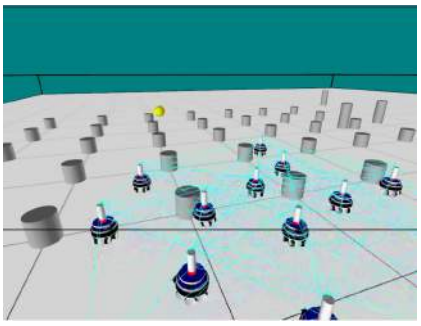

Figura 4.3: Enxame de robôs monitorando a área. Fonte: Imagens extraídas do simulador ARGoS.

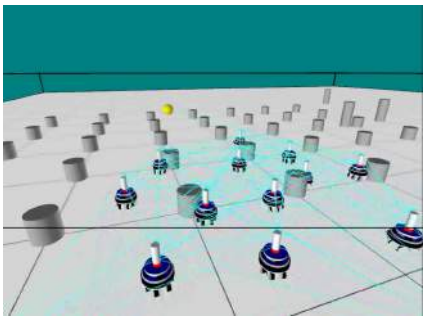

Figura 4.4: Pássaro voando (esfera amarela) no meio da plantação. Fonte: Imagens extraídas do simulador ARGoS.

As Figuras 4.5 e 4.6 representam o momento em que o enxame detecta o pássaro e começa a se movimentar em sua direção, de forma coesa, para cercá-lo. Não importa se apenas um robô detecta o pássaro, ou mais de um consiga detectá-lo, pois, uma vez localizado pelo enxame, o pássaro passa a ser perseguido por ele.

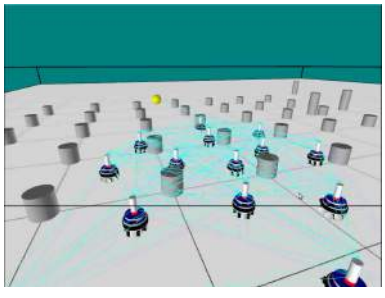

Figura 4.5: Robôs perseguindo o pássaro. Fonte: Imagens extraídas do simulador ARGoS.

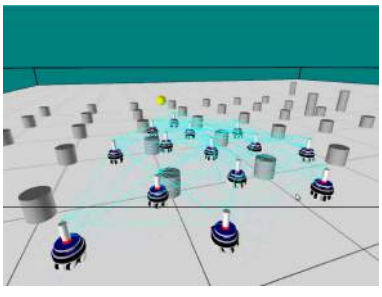

Figura 4.6: Robôs em formação tentando cercar o pássaro. Fonte: Imagens extraídas do simulador ARGoS.

As Figuras 4.7 e 4.8 ilustram os robôs perseguindo o pássaro enquanto ele se move no meio da plantação. O enxame reage adaptativamente às mudanças de trajetória do pássaro mantendo a perseguição ao mesmo e evitando os cilindros em meio à plantação.

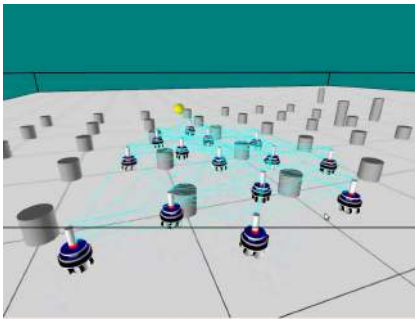

Figura 4.7: Robôs perseguindo o pássaro em movimento. Fonte: Imagens extraídas do simulador ARGoS.

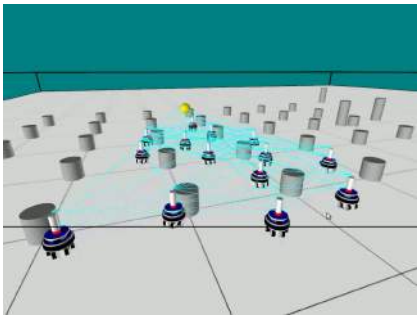

Figura 4.8: Coesão e alinhamento. Fonte: Imagens extraídas do simulador ARGoS.

As Figuras 4.9 e 4.10 ilustram a situação final do experimento mostrando exatamente o padrão de cerco do enxame envolta do pássaro, removendo-o da plantação. Verifica-se que o ER se mantém coeso durante a formação adotada para cercar o pássaro.

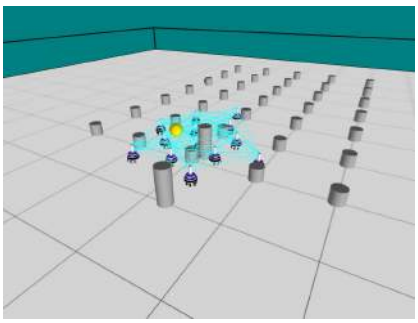

Figura 4.9: Robôs removendo o pássaro da vegetação. Fonte: Imagens extraídas do simulador ARGoS.

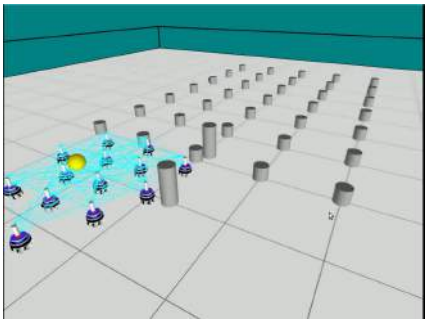

Figura 4.10: Finalização do processo de remoção do pássaro. Fonte: Imagens extraídas do simulador ARGoS.

4.2 Novos Experimentos Realizados

Com base nos resultados obtidos no experimento inicial, aliada à experiência adquirida na utilização do simulador decidiu-se realizar uma série de novos experimentos para verificar o comportamento do enxame frente à diferentes tipos de trajetórias simuladas.

Para realizar as simulações estabeleceu-se uma arena quadrada de 400 m^2. Para que durante os deslocamentos do ER e do pássaro, os robôs não saíssem de dentro dessa área ocasionando erro na execução dos experimentos, foi estipulado que o pássaro voaria apenas numa sub-região quadrada de 144 m^2. Essa sub-região foi posicionada no ponto central da arena. Para fins de simulação, considera-se como a área agrícola a ser monitorada a referida sub-região. A Figura 4.11 ilustra a arena e a área agrícola.

Dentro da área agrícola executou-se experimentos variando o tipo de trajetória (retilínea, semicircular e circular) realizada pelo pássaro durante o voo. Essas trajetórias foram escolhidas por representarem as trajetórias de voo de alguns pássaros ou quando combinadas resultarem em trajetórias mais complexas. O pássaro inicia o voo do ponto inicial (P_i) e vai até ao ponto final (P_f), realizando o deslocamento apenas uma vez, conforme ilustrado nas imagens das Figuras 4.12, 4.13 e 4.14.

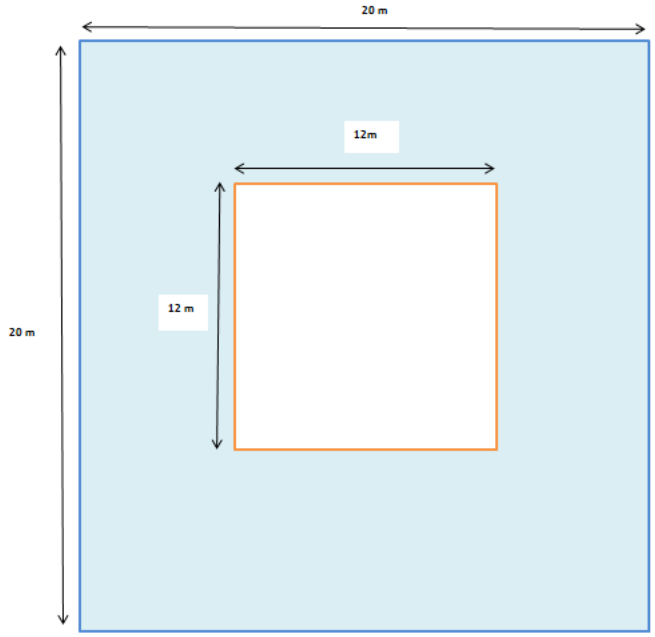

Figura 4.11: Aspecto da arena e da área agrícola. Fonte: Ilustração do próprio autor dessa dissertação.

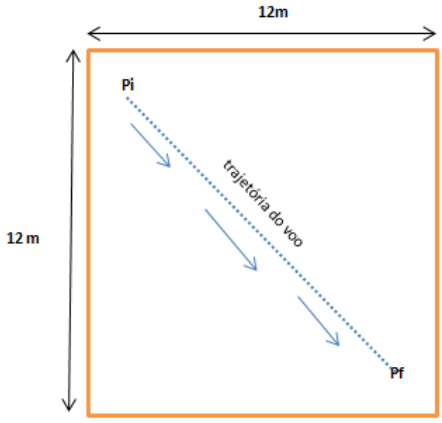

Figura 4.12: Vôo do pássaro em trajetória retilinea.

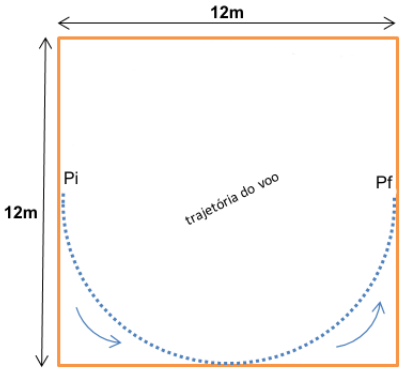

Figura 4.13: Vôo do pássaro em trajetória semicircular.

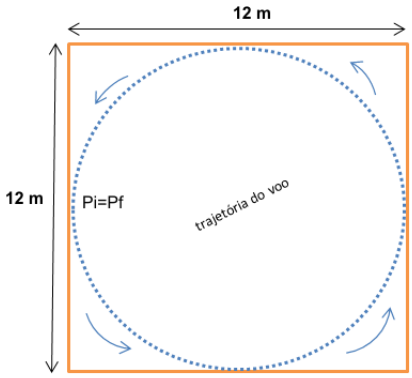

Figura 4.14: Vôo do pássaro em trajetória circular.

As distâncias percorridas pelo pássaro variam para cada tipo de trajetória realizada e foram calculadas da seguinte forma:

a) Para a **trajetória retilínea**: o cálculo da distância (d_r) foi realizado com base na Equação

(4.1), sendo o ponto inicial: $q_i : x_1 = 5m; y_1 = -5m$ e o ponto final: $p_i : x_1 = -5m; y_1 = 5m$.

$$d_r = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} = \sqrt{(5-(-5))^2 + (-5-5)^2} = \sqrt{(10)^2 + (-10)^2} = \sqrt{200} = 14,14m \quad (4.1)$$

b) Para a **trajetória semicircular**: o cálculo da distância (d_s) foi realizado com base na Equação (4.2), sendo $raio = 6m$ e $\pi = 3,14$.

$$\begin{aligned} d_s &= \frac{2 \times \pi \times raio}{2} \\ &= \frac{(2 \times 3,14 \times 6)}{2} \\ &= 18,84m \end{aligned} \quad (4.2)$$

c) Para a **trajetória circular**: cálculo da distância (d_c) foi realizado com base na Equação (4.3), sendo $raio = 6m$ e $\pi = 3,14$.

$$\begin{aligned} d_c &= 2 \times \pi \times raio \\ &= 2 \times 3,14 \times 6 \\ &= 37,68m \end{aligned} \quad (4.3)$$

Para simular a variação da velocidade do voo do pássaro foi necessário definir um fator de interpolação para representar a grandeza de deslocamento entre os conjuntos de pontos $(x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}), ..., (x_{i+n-1}, y_{i+n-1}), (x_{i+n}, y_{i+n})$ que compreendem os pontos inicial e final da trajetória do pássaro. Para isso, foram realizadas 50 simulações. Depois de analisadas, inferiu-se os limites inferiores (pássaro voa muito devagar em relação ao enxame) e superiores (pássaro voa muito rápido em relação ao enxame). Os valores obtidos para o limite inferior foi 10^{-4} e para o limite superior foi 10^{-3}. Realizando o cálculo de progressão aritmética com base na Equação (4.4), obteve-se a razão de crescimento, possibilitando obter cada termo que representa o fator de interpolação. Sendo $a_1 = 10^{-4}$, $a_n = 10^{-3}$ e n=30 (número de fatores de interpolação).

$$\begin{aligned} a_n &= a_1 + (n-1)r \\ 10^{-3} &= 10^{-4} + (30-1)r \\ r &= \frac{10^{-3} - 10^{-4}}{29} \\ r &= 0,0000310 \end{aligned} \quad (4.4)$$

A Tabela 4.1 contém os valores dos fatores de interpolação encontrados para aplicar nos experimentos.

4.2.1 Descrição dos Cenários Simulados

O objetivo dos cenários é verificar se o ER consegue efetuar o cerco do pássaro durante o voo e quais as reações comportamentais ocorridas no enxame diante de determinadas variações (velocidade do voo do pássaro, quantidade de robôs e trajetória do voo do pássaro). As simulações nos cenários abstraídos ocorreram com um pássaro invadindo a área agrícola. Além disso, é necessário analisar se o enxame possui algumas características inerentes à abordagem de ER, para que não seja apenas uma solução que empregue um grupo de robôs, são elas:

1. Se cada integrante do ER é autônomo.
2. Se cada integrante do ER possui um conhecimento limitado do local.

Tabela 4.1 Valores dos fatores de interpolação.

Passo	Fator de interpolação	Passo	Fator de interpolação
1	0,0001000	17	0,0005655
2	0,0001310	18	0,0005966
3	0,0001621	19	0,0006276
5	0,0001931	20	0,0006586
6	0,0002241	21	0,0006897
7	0,0002552	22	0,0007207
8	0,0002862	23	0,0007517
9	0,0003172	24	0,0007828
10	0,0003483	25	0,0008138
11	0,0003793	26	0,0008448
12	0,0004103	27	0,0008759
13	0,0004414	27	0,0009069
14	0,0004724	28	0,0009379
15	0,0005034	29	0,0009690
16	0,0005345	30	0,0010000

3. Se o ER se mantém coeso durante o deslocamento.
4. Se o ER consegue dispersar pássaros sem um controlador central.
5. Se nos experimentos percebe-se a robustez, a flexibilidade e a escalabilidade.
6. Se os robôs executam as mesmas tarefas.
7. Se o ER é auto organizado ou seja, caso seja contenha ao mesmo tempo as propriedades: autônomo, ausência de controle central, localidade e flexibilidade.

Os experimentos foram divididos em 09 cenários nos quais ocorreram as variações da velocidade do voo do pássaro, conforme fatores de interpolação dos passos de números 15 a 30 da Tabela 4.1. Além disso, foram alterados os tipos de trajetória e a quantidade de robôs, conforme expressos na Tabela 4.2.

Tabela 4.2 Cenários simulados.

Cenário	Número de robôs	Tipos de trajetória
1	3	retilínea
2	15	retilínea
3	30	retilínea
4	3	semicircular
5	15	semicircular
6	30	semicircular
7	3	circular
8	15	circular
9	30	circular

O simulador ARGoS não grava arquivos de vídeos dos experimentos. Ele permite que a cada intervalo de tempo seja armazenado uma cópia das imagens que estão sendo visualizadas na janela do simulador, em formato *PNG*. Para contornar essa situação, executou-se a ferramenta *Mencoder* que codificou o vídeo por meio da união dos arquivos *PNG*, convertendo-os em um arquivo de vídeo em formato *AVI*. Segue abaixo o comando utilizado juntamente com a explicação de cada parâmetro.

```
1    # mencoder mf://*.png -mf w=800:h=600:fps=25:type=png -ovc lavc \
2      -lavcopts vcodec=mpeg4:mbd=2:trell -oac copy -o NomeDoArquivo.avi
```

Onde o parâmetro:

- **mf**: É o atributo que estabelece os arquivos de entrada para realização da conversão. Neste caso são todos os arquivos com extensão PNG presentes no diretório corrente, com largura e altura de 800×600. E estabelece que serão convertidos 25 arquivos por segundo.

- **-ovc**: Refere-se ao tipo de filtro a ser utilizado para gerar o vídeo. Neste caso a opção *"lavc"* resulta em um arquivo com a melhor qualidade com arquivos relativamente pequenos.
- **-oac**: Este atributo permite a gravação de sons. Como os arquivos PNG não possuem som, este atributo é irrelevante neste caso.
- **-o**: Determina o nome do arquivo de saída de vídeo.

Para a execução dos experimentos propostos foi novamente configurado o ambiente do simulador. Foram realizadas várias alterações nos arquivos constantes no Apêndice:

- floocking.cpp (ver Código C.2)
- *My_loop_functions.cpp* (ver Código C.1)
- *My_loop_functions.h* (ver Código C.3)

4.2.2 Apresentação dos Resultados dos Experimentos

Os experimentos dos cenários, anteriormente, mencionados, foram simulados no mesmo ambiente computacional utilizado no experimento inicial. As informações inerentes as simulações dos experimentos foram coletadas e analisadas. Serão comentados os resultados separadamente de acordo com o tipo de trajetória simulada. Primeiramente, a retilínea. Em seguida, a semicircular. E por último, a circular. Para cada tipo de trajetória serão apresentadas seis tabelas, sendo as três primeiras com informações básicas e as três últimas com informações complementares.

Cada coluna das tabelas básicas representam as seguintes informações:

- **Experimento**: Representa o número do experimento, variando entre 15 a 30.
- **Fator**: Corresponde ao fator de interpolação utilizado no experimento.
- **Tempo IV**: Representa o tempo inicial do voo do pássaro. Expresso em horas, minutos e segundos.
- **Tempo FV**: Representa o tempo final do voo do pássaro.Expresso em horas, minutos e segundos.
- **Duração**: Representa a duração do vôo do pássaro.Expressa em horas, minutos e segundos.
- **Passo**: Corresponde a quantidade de passos utilizados pelo simulador durante a realização do experimento.
- **Distância**: Representa a distância em metros percorrida pelo pássaro durante o vôo.
- **Velocidade**: Representa a velocidade do vôo do pássaro. Expressa em m/s^{-10}.

Cada coluna das tabelas complementares representam as seguintes informações:

- **Experimento**: Representa o número do experimento, variando entre 15 a 30.
- **Fator**: Corresponde ao fator de interpolação utilizado no experimento.
- **Cerco**: Este campo menciona se durante o voo do pássaro o enxame consegue em algum momento cercá-lo
- **Divide**: Este campo menciona se até cercar o pássaro o enxame se divide em grupo. Apesar de um enxame ser constituído de N integrantes, em alguns casos o exame de subdivide em dois pequenos grupos durante o trajeto, voltando a se reagrupar novamente.
- **Tempo PC**: Tempo gasto pelo enxame para realizar o primeiro cerco. Corresponde ao momento que o pássaro invade a área agrícola e o enxame consegue cercá-lo mesmo enquanto o pássaro está voando. Expresso em minutos e segundos.
- **Tempo UC**: Tempo gasto pelo enxame para realizar o último cerco. Corresponde ao cerco realizado pelo enxame apenas quando o pássaro pousa na vegetação. Expresso em minutos e segundos.

Voo do pássaro em Trajetória Retilínea

As Tabelas 4.3, 4.5 e 4.7 contêm as informações básicas referentes aos experimentos realizados com a trajetória retilínea e as Tabelas 4.4, 4.6 e 4.8 apresentam as informações complementares.

Tabela 4.3 Informações coletadas referentes às simulações do Cenário 1 - trajetória retilínea com enxame com 3 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	21:57:17	23:31:11	01:33:54	18.711	14,14	25,097622
2	16	19:52:32	21:20:25	01:27:53	17.685	14,14	26,815854
3	17	17:56:20	19:21:40	01:25:20	16.763	14,14	27,617188
4	18	15:57:55	17:18:49	01:20:54	15.935	14,14	29,130614
5	19	14:16:42	15:32:23	01:15:41	15.185	14,14	31,138516
6	20	04:42:14	05:52:37	01:10:23	14.501	14,14	33,483306
7	21	03:09:28	04:17:30	01:08:02	13.877	14,14	34,639882
8	22	01:36:43	02:42:09	01:05:26	13.305	14,14	36,016302
9	23	00:13:03	01:15:38	01:02:35	12.776	14,14	37,656458
10	24	22:43:25	23:44:00	01:00:35	12.290	14,14	38,899587
11	25	21:17:43	22:17:14	00:59:31	11.839	14,14	39,596752
12	26	19:59:54	20:56:16	00:56:22	11.418	14,14	41,809580
13	27	18:28:22	19:22:43	00:54:21	11.028	14,14	43,360932
14	28	17:04:33	17:57:49	00:53:16	10.664	14,14	44,242804
15	29	15:45:35	16:37:12	00:51:37	10.321	14,14	45,657088
16	30	13:28:03	14:19:11	00:51:08	10.001	14,14	46,088657
-	-	-	Total	17:56:58	-	-	-

Tabela 4.4 Informações complementares coletadas referentes às simulações do Cenário 1 - trajetória retilínea com enxame com 3 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim	Não	01:03	12:31
2	16	Sim	Não	01:13	11:58
3	17	Sim	Não	01:19	11:25
4	18	Sim	Não	01:21	11:05
5	19	Sim	Não	01:00	10:10
6	20	Sim	Não	00:54	10:00
7	21	Sim	Não	01:04	09:20
8	22	Sim	Não	01:12	09:10
9	23	Sim	Não	01:13	08:32
10	24	Sim	Não	00:59	08:18
11	25	Sim	Não	01:04	08:04
12	26	Sim	Não	01:19	07:42
13	27	Sim	Não	01:09	07:40
14	28	Sim	Não	01:08	07:13
15	29	Sim	Não	01:12	06:55
16	30	Sim	Não	00:54	06:45
-	-	-	Total	18:04	2:48:00

Tabela 4.5 Informações coletadas referentes às simulações do Cenário 2 - trajetória retilínea com enxame com 15 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	00:02:30	01:48:25	01:45:55	18.711	14,14	22,25020
2	16	02:41:48	04:24:13	01:42:25	17.685	14,14	23,01058
3	17	13:07:04	14:45:15	01:38:11	16.763	14,14	24,00272
4	18	15:25:49	16:57:25	01:31:36	15.935	14,14	25,72780
5	19	16:15:57	17:42:30	01:26:33	15.185	14,14	27,22896
6	20	14:12:49	15:35:57	01:23:08	14.501	14,14	28,34804
7	21	12:09:38	13:40:44	01:31:06	13.877	14,14	25,86901
8	22	11:56:39	13:12:03	01:15:24	13.305	14,14	31,25553
9	23	01:56:26	03:06:38	01:10:12	12.776	14,14	33,57075
10	24	10:37:47	11:50:09	01:12:22	12.290	14,14	32,56564
11	25	02:53:04	04:01:19	01:08:15	11.839	14,14	34,52991
12	26	00:38:29	01:42:11	01:03:42	11.418	14,14	36,99634
13	27	23:05:02	00:06:51	01:01:49	11.028	14,14	38,12348
14	28	21:40:14	22:41:01	01:00:47	10.664	14,14	38,77159
15	29	20:21:45	21:19:48	00:58:03	10.321	14,14	40,59719
16	30	00:23:50	01:20:47	00:56:57	10.001	14,14	41,38133
-	-	-	Total	20:46:25	-	-	-

Tabela 4.6 Informações complementares coletadas referentes às simulações do Cenário 2 - trajetória retilínea com enxame com 15 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim	Não	01:41	12:32
2	16	Sim	Não	01:59	12:27
3	17	Sim	Sim, em 10 e 5	02:05	11:37
4	18	Sim	Não	02:20	11:42
5	19	Sim	Não	02:08	10:20
6	20	Sim	Não	02:14	10:14
7	21	Sim	Sim,em 11 e 4	02:05	09:37
8	22	Sim	Não	02:00	09:19
9	23	Sim	Não	02:03	09:00
10	24	Sim	Não	02:10	08:19
11	25	Sim	Não	01:58	08:04
12	26	Sim	Sim, em 9 e 6	01:54	08:02
13	27	Sim	Sim, em 10 e 5	01:49	07:37
14	28	Sim	Sim, em 10 e 5	01:39	07:23
15	29	Sim	Sim, em 8 e 7	01:33	07:05
16	30	Sim	Não	01:32	06:59
-	-	-	Total	31:10	2:30:17

Tabela 4.7 Informações coletadas referentes às simulações do Cenário 3 retilínea com enxame com 30 integrantes..

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	15:06:35	16:55:54	01:49:19	18.711	14,14	21,558164
2	16	12:50:46	14:33:03	01:42:17	17.685	14,14	23,040574
3	17	17:06:12	18:42:31	01:36:19	16.763	14,14	24,467901
4	18	19:24:58	20:58:26	01:33:28	15.935	14,14	25,213980
5	19	21:35:25	23:04:14	01:28:49	15.185	14,14	26,534059
6	20	10:16:18	11:41:31	01:25:13	14.501	14,14	27,654997
7	21	23:40:12	01:00:22	01:20:10	13.877	14,14	29,397089
8	22	22:43:31	00:01:24	01:17:53	13.305	14,14	30,258934
9	23	21:00:25	22:15:27	01:15:02	12.776	14,14	31,408263
10	24	19:19:39	20:31:21	01:11:42	12.290	14,14	32,868433
11	25	17:34:39	18:43:40	01:09:01	11.839	14,14	34,146341
12	26	15:31:56	16:38:59	01:07:03	11.418	14,14	35,147900
13	27	13:56:28	15:00:51	01:04:23	11.028	14,14	36,603676
14	28	21:28:07	22:30:10	01:02:03	10.664	14,14	37,980124
15	29	18:35:32	19:33:31	00:57:59	10.321	14,14	40,643863
16	30	18:14:10	19:09:26	00:55:16	10.001	14,14	42,641737
-	-	-	Total	20:55:57	-	-	-

Tabela 4.8 Informações complementares coletadas referentes às simulações do Cenário 3 - trajetória retilínea com enxame com 30 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim	Não	01:27	12:24
2	16	Sim	Não	01:24	11:59
3	17	Sim	Não	01:20	11:26
4	18	Sim	Não	01:15	11:06
5	19	Sim	Não	01:00	10:11
6	20	Sim	Não	00:53	09:51
7	21	Sim	Não	01:04	09:20
8	22	Sim	Não	01:11	09:08
9	23	Sim	Não	01:12	08:31
10	24	Sim	Não	00:59	08:17
11	25	Sim	Não	01:04	08:01
12	26	Sim	Não	01:11	07:42
13	27	Sim	Não	01:10	07:27
14	28	Sim	Não	01:14	07:20
15	29	Sim	Não	01:12	06:55
16	30	Sim	Não	00:54	06:54
-	-	-	Total	18:30	2:26:32

O gráfico da Figura 4.15 ilustra a quantidade de passos de tempo de cada experimento realizado com o pássaro voando em trajetória retilínea. A cada passo de tempo o motor de física 3D do simulador ARGoS executa cálculos para configurar suas representações (x, y, z), e em seguida, atualizar o espaço 3D de representação visual. De acordo com esse gráfico, a quantidade de passos de tempo calculada durante os experimentos com o pássaro voando em trajetória retilínea foram iguais para cada experimento que tenha o mesmo fator de interpolação, independente da quantidade de robôs do enxame ($n = 3; n = 15; n = 30$). Percebe-se também que de acordo com o que foi planejado anteriormente, aumentando-se o valor do fator de interpolação diminui-se a quantidade de passos de tempo, permitindo assim alterar a velocidade de deslocamento do pássaro.

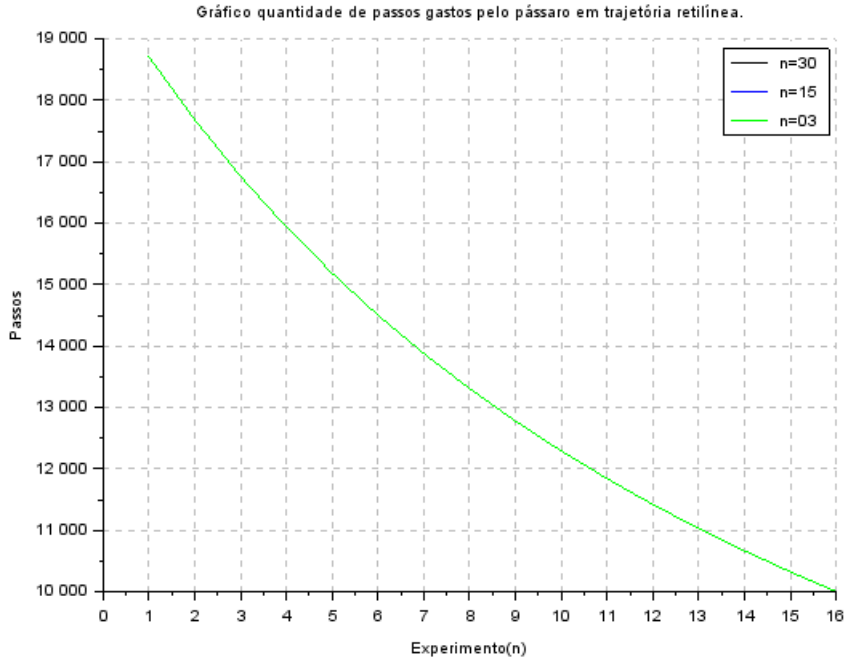

Figura 4.15: O gráfico ilustra a quantidade de tempos de passos em função do fator de interpolação de cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Analisando os gráficos das Figuras 4.16 e 4.17, conclui-se que quando o enxame tem:

- **3 robôs:** A velocidade do voo do pássaro é mais rápida entre os pontos iniciais e finais, em relação aos enxames com 15 e 30 integrantes. Além disso, a velocidade tende a crescer ao aumentar o fator de interpolação. O enxame permanece coeso durante as atividades de monitoramento ou cerco (Ver Tabela 4.4).
- **15 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a sofrer algumas oscilações entre os pontos iniciais e finais. O enxame se divide em alguns momentos durante as atividades de monitoramento ou cerco. Apesar dessa divisão, o enxame tende a se reagrupar novamente (Ver Tabela 4.6).
- **30 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a crescer entre os pontos iniciais e finais. O enxame não se divide durante as atividades de monitoramento ou cerco. (Ver Tabela 4.8).

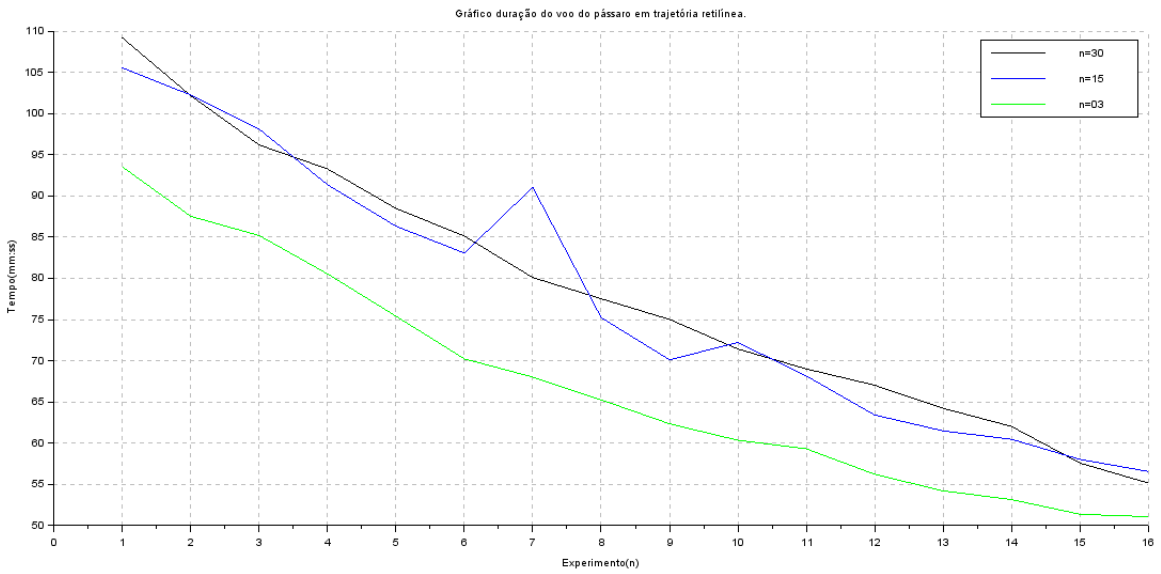

Figura 4.16: O gráfico ilustra a duração do voo em trajetória retilínea para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

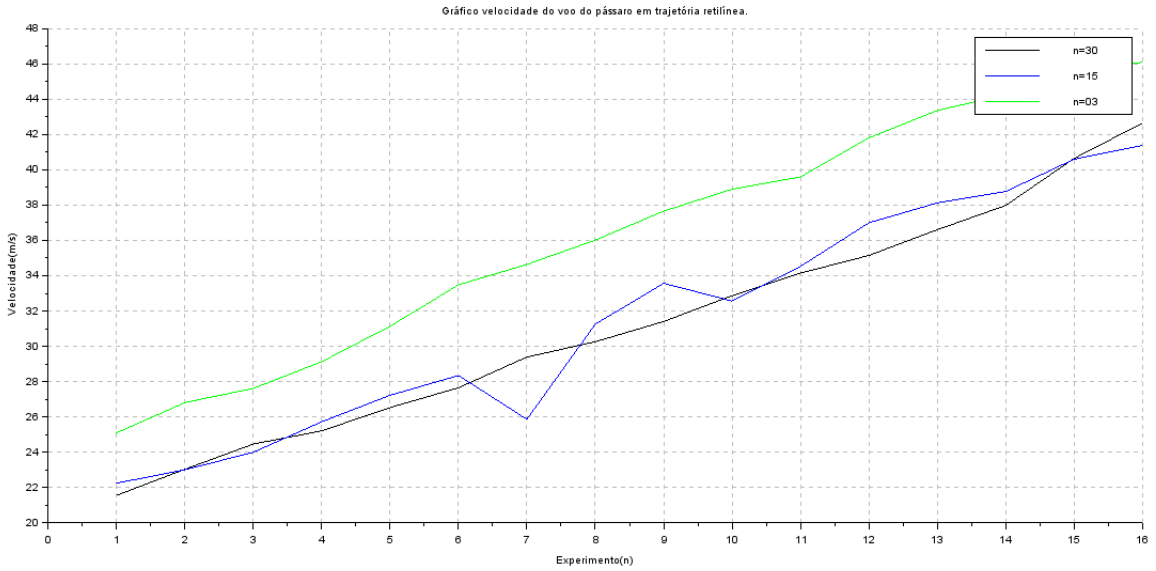

Figura 4.17: O gráfico ilustra a velocidade do voo em trajetória retilínea para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

O gráfico da Figura 4.18 ilustra o tempo gasto pelo enxame para cercar o pássaro pela primeira vez para cada experimento. Nas simulações com a trajetória retilínea verificou-se que em todos os experimentos o enxame conseguiu cercar o pássaro durante o vôo independente da quantidade de robôs e da velocidade. Apesar disso o tempo gasto pelo enxame com 15 integrantes foi superior aos demais (n=3 e n=15).

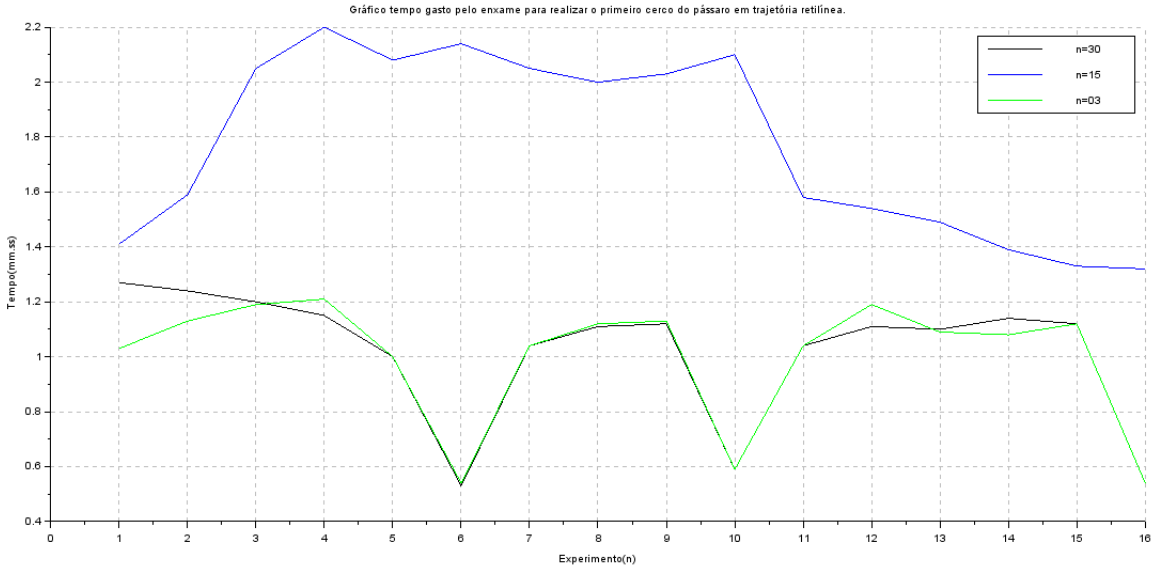

Figura 4.18: O gráfico ilustra o tempo gasto pelo enxame para cercar o pássaro pela primeira vez para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

O gráfico da Figura 4.19 ilustra o tempo gasto pelo enxame para realizar o cerco quando o pássaro pousa. Verifica-se que o tempo gasto pelo enxame com 30 integrantes é menor do que os demais (n=3 e n=15), embora o enxame com 3 integrantes tenha os mesmos valores em alguns momentos. Já o enxame com 15 robôs tende a demorar mais tempo para cercar o pássaro.

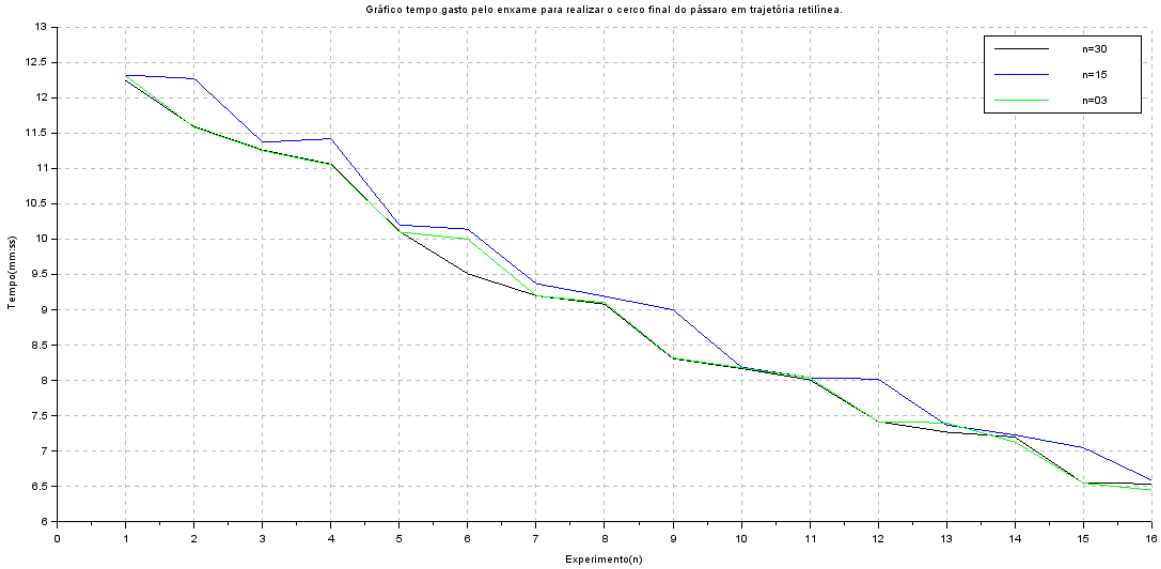

Figura 4.19: O gráfico ilustra o tempo gasto pelo enxame para cercar o pássaro pela última vez para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Com base nas informações anteriormente mencionadas das Tabelas 4.3, 4.5 e 4.7, o tempo total gasto durante os 48 experimentos realizados para a trajetória retilínea do voo do pássaro, sem considerar o tempo de processamento das imagens em vídeos, foi de 59 horas, 39 minutos e 20 segundos, distribuídos em:

- **Cenário 1 (3 robôs):** com 17 horas, 56 minutos e 58 segundos.
- **Cenário 2 (15 robôs):** com 20 horas, 46 minutos e 25 segundos.
- **Cenário 3 (30 robôs):** com 20 horas , 55 minutos e 57 segundos.

A Tabela 4.9 contem o resumo das principais características da abordagem de ER presentes em cada cenário referentes aos experimentos simulando um pássaro voando em trajetória retilínea. A análise foi efetuada com base nos códigos fontes e nos vídeos examinados.

Tabela 4.9 Características da abordagem de Enxame presentes nos experimentos realizados.

Cenário	Autônomo	Localidade	Coesão	ACC	Robustez	Flexibilidade	Escalabilidade	Similaridade	Auto-organizado
Cenário 1	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 2	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 3	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Onde as colunas:

- **ACC:** Representa se o enxame consegue dispersar o pássaro sem um controlador central.
- **Coesão:** Refere-se a conectividade dos integrantes do enxame durante o deslocamento com o intuito de verificar se é coeso.
- **Localidade:** Verifica se cada integrante do enxame tem um conhecimento limitado do local (extensão da área e número de integrantes do enxame).

Voo do pássaro em Trajetória Semicircular

As Tabelas 4.10, 4.12 e 4.14 contêm as informações básicas referentes aos experimentos realizados com a trajetória semicircular e as Tabelas 4.11, 4.13 e 4.15 apresentam as informações complementares.

Tabela 4.10 Informações coletadas referentes às simulações do Cenário 4 - trajetória semicircular com enxame com 3 integrantes..

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	23:22:03	23:53:10	00:31:07	5.875	18,84	100,910552
2	16	22:39:55	23:09:31	00:29:36	5.553	18,84	106,081081
3	17	21:57:28	22:25:12	00:27:44	5.264	18,84	113,221154
4	18	21:19:26	21:45:54	00:26:28	5.004	18,84	118,639798
5	19	18:15:19	18:40:25	00:25:06	4.768	18,84	125,099602
6	20	17:28:37	17:53:24	00:24:47	4.553	18,84	126,698050
7	21	16:55:36	17:19:15	00:23:39	4.357	18,84	132,769556
8	22	16:21:32	16:44:12	00:22:40	4.178	18,84	138,529412
9	23	15:48:51	16:10:36	00:21:45	4.012	18,84	144,367816
10	24	15:16:03	15:37:01	00:20:58	3.859	18,84	149,761526
11	25	14:46:55	15:06:53	00:19:58	3.717	18,84	157,262104
12	26	14:14:32	14:33:28	00:18:56	3.585	18,84	165,845070
13	27	00:52:55	01:11:15	00:18:20	3.463	18,84	171,272727
14	28	00:24:03	00:41:54	00:17:51	3.348	18,84	175,910364
15	29	23:56:17	00:13:56	00:17:39	3.241	18,84	177,903683
16	30	23:27:33	23:44:54	00:17:21	3.141	18,84	180,979827
-	-	-	Total	06:03:55	-	-	-

Tabela 4.11 Informações complementares coletadas referentes às simulações do Cenário 4 - trajetória semicircular com enxame com 3 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Não	Não	-	00:04:10
2	16	Não	Não	-	00:03:58
3	17	Não	Não	-	00:03:42
4	18	Não	Não	-	00:03:30
5	19	Não	Não	-	00:03:25
6	20	Não	Não	-	00:03:14
7	21	Não	Não	-	00:02:59
8	22	Não	Não	-	00:03:07
9	23	Não	Não	-	00:02:53
10	24	Não	Não	-	00:02:46
11	25	Não	Não	-	00:02:38
12	26	Não	Sim, em 2 grupos: 1 e 2 integrantes	-	00:02:29
13	27	Não	Não	-	00:02:30
14	28	Não	Não	-	00:02:33
15	29	Não	Sim, em 2 grupos: 1 e 2 integrantes	-	00:02:20
16	30	Não	Não	-	00:02:13
-	-	-	-	Total	00:48:27

Tabela 4.12 Informações coletadas referentes às simulações do Cenário 5 - trajetória semicircular com enxame com 15 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	12:12:49	12:43:17	00:30:28	5.875	18,84	103,06346
2	16	13:49:02	14:10:42	00:21:40	5.553	18,84	144,92308
3	17	14:17:44	14:45:42	00:27:58	5.264	18,84	112,27652
4	18	14:54:46	15:21:05	00:26:19	5.004	18,84	119,31602
5	19	15:44:13	16:10:21	00:26:08	4.768	18,84	120,15306
6	20	16:24:04	16:49:14	00:25:10	4.553	18,84	124,76821
7	21	17:04:44	17:28:48	00:24:04	4.357	18,84	130,47091
8	22	17:41:30	18:04:38	00:23:08	4.178	18,84	135,73487
9	23	18:17:33	18:40:30	00:22:57	4.012	18,84	136,81917
10	24	18:53:26	19:14:45	00:21:19	3.859	18,84	147,30258
11	25	19:29:40	19:50:00	00:20:20	3.717	18,84	154,42623
12	26	20:05:19	20:24:48	00:19:29	3.585	18,84	161,16339
13	27	20:38:55	20:58:02	00:19:07	3.463	18,84	164,25458
14	28	21:39:44	21:58:51	00:19:07	3.348	18,84	164,25458
15	29	22:23:21	22:41:26	00:18:05	3.241	18,84	173,64055
16	30	22:55:37	23:13:13	00:17:36	3.141	18,84	178,40909
-	-	-	Total	06:02:55	-	-	-

Tabela 4.13 Informações complementares coletadas referentes às simulações do Cenário 5 - trajetória semicircular com enxame com 15 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim, com 4 robôs	Sim, em 2 grupos: 10 e 5 integrantes	00:01:53	00:04:27
2	16	Sim, com 3 robôs	Sim, em 2 grupos: 9 e 6 integrantes	00:03:00	00:04:09
3	17	Sim, com 3 robôs	Sim, em 2 grupos: 10 e 5 integrantes	00:01:21	00:04:02
4	18	Sim, com 3 robôs	Não	00:01:26	00:03:58
5	19	Sim, com 3 robôs	Sim, em 2 grupos: 9 e 6 integrantes	00:01:47	00:03:38
6	20	Sim, com 3 robôs	Não	00:01:24	00:03:43
7	21	Sim, com 4 robôs	Não	00:01:54	00:03:22
8	22	Sim, com 3 robôs	Sim, em 2 grupos: 8 e 7 integrantes	00:01:47	00:03:20
9	23	Não	Sim, em 2 grupos: 10 e 5 integrantes	-	00:03:17
10	24	Não	Sim, em 2 grupos: 9 e 6 integrantes	-	00:03:13
11	25	Não	Não	-	00:03:11
12	26	Não	Não	-	00:03:07
13	27	Não	Não	-	00:02:51
14	28	Não	Não	-	00:02:50
15	29	Não	Não	-	00:02:48
16	30	Não	Não	-	00:02:36
-	-	-	Total	00:14:32	00:54:32

Tabela 4.14 Informações coletadas referentes às simulações do Cenário 6 - trajetória semicircular com enxame com 30 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	15:59:00	16:32:23	00:33:23	5.875	18,84	94,058912
2	16	17:01:03	17:33:13	00:32:10	5.553	18,84	97,616580
3	17	17:54:47	18:25:28	00:30:41	5.264	18,84	102,335687
4	18	18:47:23	19:16:09	00:28:46	5.004	18,84	109,154114
5	19	19:38:21	20:06:12	00:27:51	4.768	18,84	112,746858
6	20	20:25:31	20:51:42	00:26:11	4.553	18,84	119,923616
7	21	23:04:04	23:29:20	00:25:16	4.357	18,84	124,274406
8	22	23:49:41	00:14:04	00:24:23	4.178	18,84	128,776487
9	23	00:35:25	00:58:55	00:23:30	4.012	18,84	133,617021
10	24	01:23:08	01:45:43	00:22:35	3.859	18,84	139,040590
11	25	12:28:51	12:50:21	00:21:30	3.717	18,84	146,046512
12	26	13:14:02	13:34:21	00:20:19	3.585	18,84	154,552912
13	27	13:53:56	14:14:22	00:20:26	3.463	18,84	153,670473
14	28	14:40:29	15:00:11	00:19:42	3.348	18,84	159,390863
15	29	15:20:13	15:39:35	00:19:22	3.241	18,84	162,134251
16	30	15:58:57	16:17:40	00:18:43	3.141	18,84	167,764915
-	-	-	Total	6:34:48	-	-	-

Tabela 4.15 Informações complementares coletadas referentes às simulações do Cenário 6 - trajetória semicircular com enxame com 30 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim, com 4 robôs	Não	00:01:39	00:05:09
2	16	Sim, com 4 robôs	Não	00:01:44	00:04:30
3	17	Sim, com 3 robôs	Não	00:01:51	00:04:25
4	18	Sim, com 3 robôs	Não	00:01:57	00:04:19
5	19	Sim, com 4 robôs	Não	00:01:23	00:03:59
6	20	Sim, com 3 robôs	Não	00:01:17	00:03:54
7	21	Sim, com 3 robôs	Não	00:02:19	00:03:51
8	22	Sim, com 5 robôs	Não	00:02:10	00:03:41
9	23	Sim, com 4 robôs	Sim, em 2 grupos: 07 e 23 integrantes	00:01:40	00:03:42
10	24	Não	Não	-	00:03:14
11	25	Não	Não	-	00:03:35
12	26	Não	Sim, em 2 grupos: 22 e 8 integrantes	-	00:03:13
13	27	Não	Não	-	00:03:08
14	28	Não	Não	-	00:03:02
15	29	Não	Não	-	00:02:58
16	30	Não	Não	-	00:02:48
-	-	-	Total	15:44	00:59:28

O gráfico da Figura 4.20 ilustra a quantidade de passos de tempo em função do fator de interpolação de cada experimento realizado com o pássaro voando em trajetória retilínea. A cada passo de tempo o motor de física 3D do simulador ARGoS executa cálculos para configurar suas representações (x, y, z), e em seguida, atualizar o espaço 3D de representação visual. De acordo com esse gráfico, a quantidade de passos de tempo calculada durante os experimentos com o pássaro voando em trajetória semicircular foram iguais para cada experimento que tenha o mesmo fator de interpolação, independente da quantidade de robôs do enxame ($n = 3; n = 15; n = 30$). Percebe-se também que de acordo com o que foi planejado anteriormente, aumentando-se o valor do fator de interpolação diminui-se a quantidade de passos de tempo, permitindo assim alterar a velocidade de deslocamento do pássaro.

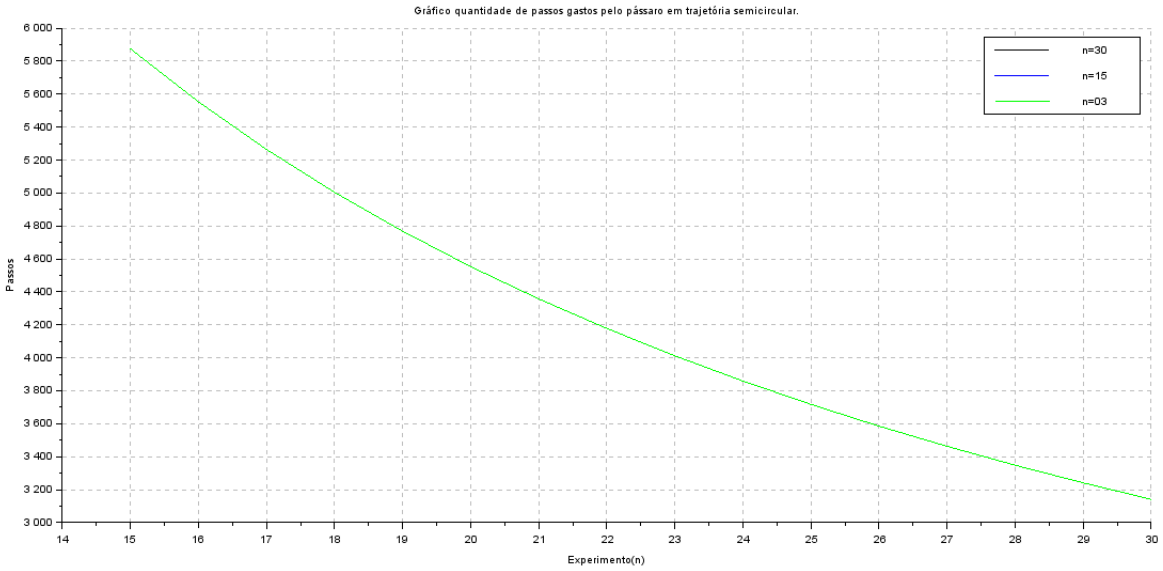

Figura 4.20: O gráfico ilustra a quantidade de tempos de passos em função do fator de interpolação de cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Analisando os gráficos das Figuras 4.21 e 4.22, conclui-se que quando o enxame tem:

- **3 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a crescer entre os pontos iniciais e finais. O enxame se divide em apenas dois experimentos durante as atividades de monitoramento ou cerco. Embora o enxame com 3 robôs só consiga cercar o pássaro depois que ele pousa, o enxame fica constantemente próximo do pássaro tentando cercá-lo, mostrando-se apto a afugentá-lo (Veja Tabela 4.11).
- **15 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a crescer sofrendo algumas oscilações entre os pontos iniciais e finais. O enxame se divide em vários momentos durante as atividades de monitoramento ou cerco. O enxame consegue em 8 experimentos cercar o pássaro com parte de seus integrantes (Veja as Tabelas 4.12 e 4.13).
- **30 robôs:** A velocidade do voo do pássaro é mais lenta entre os pontos iniciais e finais, em relação aos enxames com 3 e 15 integrantes. Além disso, a velocidade tende a crescer ao aumentar o fator de interpolação. O enxame se divide em alguns momentos durante as atividades de monitoramento ou cerco. O enxame consegue em 9 experimentos cercar o pássaro com parte de seus integrantes (Veja Tabela 4.15).

Friza-se que embora o enxame se divida durante a perseguição, ele se une antes iniciar o cerco inicial ou final do pássaro.

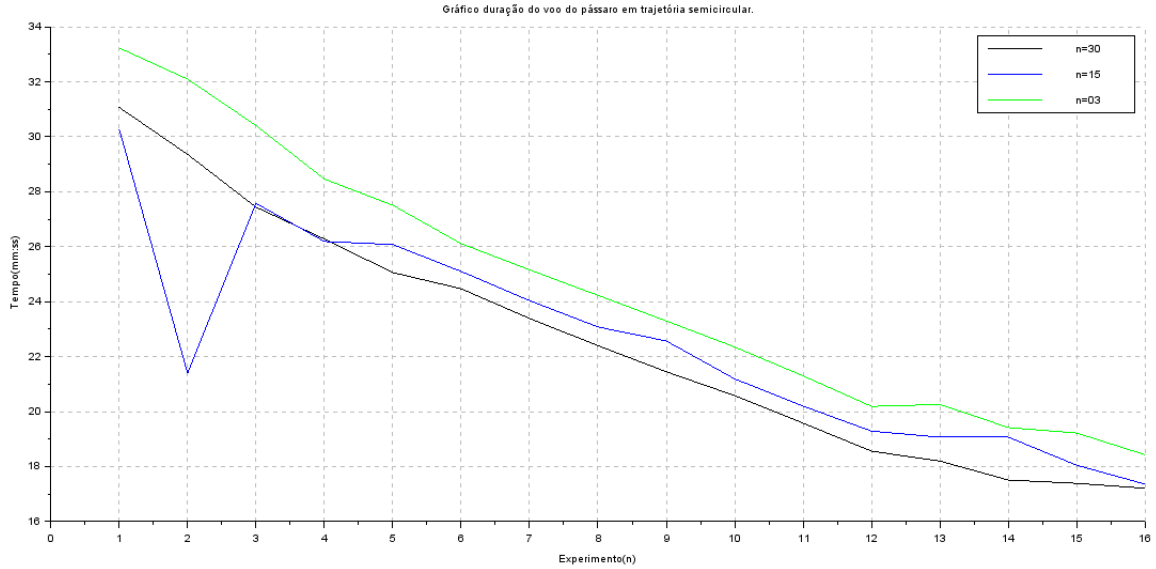

Figura 4.21: O gráfico ilustra a duração do voo em trajetória semicircular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

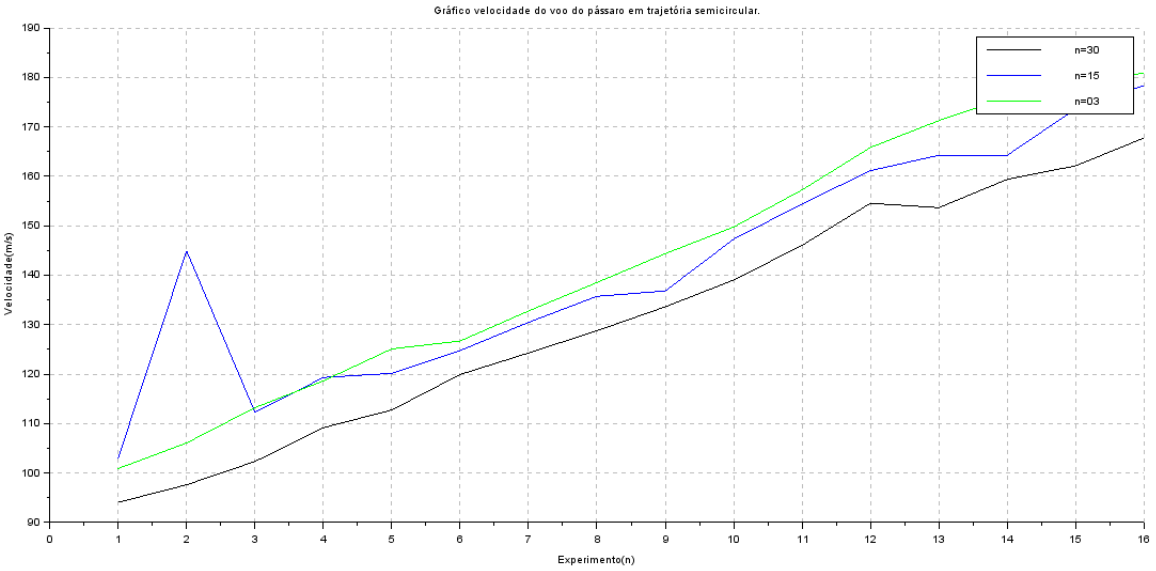

Figura 4.22: O gráfico ilustra a velocidade do voo em trajetória semicircular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

As Tabelas 4.11, 4.13, 4.15, anteriormente citadas, ilustram o tempo gasto pelo enxame para cercar o pássaro pela primeira vez para cada experimento. Nas simulações com a trajetória semicircular verificou-se que em alguns experimentos dependendo da velocidade do pássaro o enxame só o cerca quando ele pousa.

O gráfico da Figura 4.23 ilustra o tempo gasto pelo enxame para realizar o cerco quando o pássaro pousa. Verifica-se que o tempo gasto pelo enxame com 3 integrantes é menor do que os demais (n=30 e n=15).

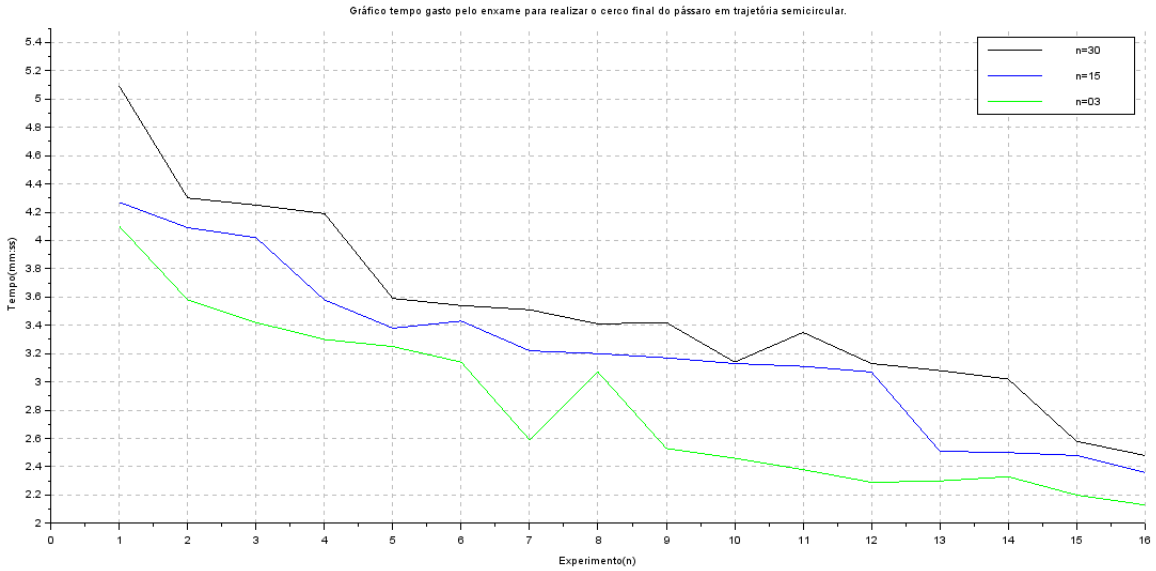

Figura 4.23: O gráfico ilustra o tempo gasto pelo enxame para cercar o pássaro pela última vez para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Com base nas informações anteriormente mencionadas das Tabelas 4.10, 4.12 e 4.14, o tempo total gasto durante os experimentos realizados para a trajetória semicircular do voo do pássaro, sem considerar o tempo de processamento das imagens em vídeos, foi de 18 horas, 36 minutos e 24 segundos, distribuídos em:

- **Cenário 4 (3 robôs):** 6 horas, 3 minutos e 55 segundos.
- **Cenário 5 (15 robôs):** 6 horas, 2 minutos e 55 segundos.
- **Cenário 6 (30 robôs):** 6 horas , 29 minutos e 34 segundos.

A Tabela 4.16 contem o resumo das principais características da abordagem de Enxame presentes em cada cenário referentes aos experimentos simulando um pássaro voando em trajetória semicircular. A análise foi efetuada com base nos códigos fontes e nos vídeos examinados.

Tabela 4.16 Características da abordagem de Enxame presentes nos experimentos realizados.

Cenário	Autônomo	Localidade	Coesão	ACC	Robustez	Flexibilidade	Escalabilidade	Similaridade	Auto-organizado
Cenário 4	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 5	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 6	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Onde as colunas:

- **ACC:** Representa se o enxame consegue dispersar o pássaro sem um controlador central.
- **Coesão:** Refere-se a conectividade dos integrantes do enxame durante o deslocamento com o intuito de verificar se é coeso.
- **Localidade:** Verifica se cada integrante do enxame tem um conhecimento limitado do local (extensão da área e número de integrantes do enxame).

Voo do pássaro em Trajetória Circular

As Tabelas 4.17, 4.19 e 4.21 contêm as informações básicas referentes aos experimentos realizados com a trajetória circular e as Tabelas 4.18, 4.20 e 4.22 apresentam as informações complementares.

Tabela 4.17 Informações coletadas referentes às simulações do Cenário 7 - trajetória circular com enxame com 3 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	18:09:19	19:09:08	00:59:49	11.750	37,68	104,987462
2	16	19:49:22	20:46:13	00:56:51	11.106	37,68	110,466139
3	17	21:05:29	21:59:13	00:53:44	10.527	37,68	116,873449
4	18	22:19:03	23:08:39	00:49:36	10.007	37,68	126,612903
5	19	23:32:41	00:20:21	00:47:40	9.536	37,68	131,748252
6	20	00:38:15	01:25:14	00:46:59	9.106	37,68	133,664420
7	21	01:46:31	02:30:36	00:44:05	8.714	37,68	142,457467
8	22	09:09:08	09:51:18	00:42:10	8.355	37,68	148,932806
9	23	11:59:17	12:40:07	00:40:50	8.023	37,68	153,795918
10	24	13:01:41	13:41:30	00:39:49	7.717	37,68	157,722897
11	25	17:33:25	18:11:27	00:38:02	7.434	37,68	165,118317
12	26	21:09:01	21:59:23	00:50:22	7.170	37,68	124,685639
13	27	20:39:28	21:14:35	00:35:07	6.925	37,68	178,832463
14	28	21:28:49	22:03:24	00:34:35	6.696	37,68	181,590361
15	29	22:17:56	22:51:04	00:33:08	6.481	37,68	189,537223
16	30	10:38:22	11:16:18	00:37:56	6.280	37,68	165,553603
-	-	-	Total	11:50:43	-	-	-

Tabela 4.18 Informações complementares coletadas referentes às simulações do Cenário 7 - trajetória circular com enxame com 3 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim	Não	00:02:25	00:07:56
2	16	Sim	Não	00:02:49	00:07:28
3	17	Sim	Não	00:01:45	00:07:12
4	18	Sim	Não	00:04:37	00:06:48
5	19	Não	Não	-	00:06:25
6	20	Sim	Não	00:02:11	00:06:20
7	21	Não	Não	-	00:06:04
8	22	Não	Não	-	00:05:44
9	23	Não	Não	-	00:05:26
10	24	Não	Não	-	00:05:11
11	25	Não	Não	-	00:05:02
12	26	Não	Sim, em 1 e 2	-	00:04:49
13	27	Não	Não	-	00:04:46
14	28	Não	Não	-	00:04:36
15	29	Não	Sim, em 1 e 2	-	00:04:21
16	30	Não	Não	-	00:04:13
-	-	-	Total	00:13:47	01:32:21

Tabela 4.19 Informações coletadas referentes às simulações do Cenário 8 - trajetória circular com enxame com 15 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	11:26:03	12:28:15	01:02:12	11.750	37,68	100,96463
2	16	09:47:17	10:45:29	00:58:12	11.106	37,68	107,90378
3	17	08:30:05	09:25:26	00:55:21	10.527	37,68	113,45980
4	18	15:43:07	16:37:37	00:54:30	10.007	37,68	115,22936
5	19	21:04:33	21:54:09	00:49:36	9.536	37,68	126,61290
6	20	15:48:16	16:35:33	00:47:17	9.106	37,68	132,81636
7	21	13:17:50	14:23:18	01:05:28	8.714	37,68	95,92668
8	22	13:39:56	14:25:20	00:45:24	8.355	37,68	138,32599
9	23	12:38:05	13:22:05	00:44:00	8.023	37,68	142,72727
10	24	00:46:58	01:26:20	00:39:22	7.717	37,68	159,52583
11	25	23:49:23	00:29:04	00:39:41	7.434	37,68	158,25283
12	26	22:53:41	23:30:03	00:36:22	7.170	37,68	172,68561
13	27	21:13:24	21:48:49	00:35:25	6.925	37,68	177,31765
14	28	20:07:48	20:42:09	00:34:21	6.696	37,68	182,82387
15	29	18:50:18	19:25:58	00:35:40	6.481	37,68	176,07477
16	0	18:02:29	18:35:10	00:32:41	6.280	37,68	192,14686
-	-	-	Total	12:15:32	-	-	-

Tabela 4.20 Informações complementares coletadas referentes às simulações do Cenário 8 - trajetória circular com enxame com 15 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim, com 4	Sim, 9 e 6	00:04:49	00:08:04
2	16	Sim, com 4	Sim, 9 e 6	00:03:35	00:07:43
3	17	Sim, com 3	Não	00:02:40	00:07:28
4	18	Sim, com 4	Não	00:01:28	00:06:59
5	19	Sim, com 4	Sim, 9 e 6	00:02:34	00:06:41
6	20	Sim, com 4	Não	00:04:13	00:06:33
7	21	Sim, com 4	Não	00:05:38	00:06:27
8	22	Sim, com 3	Não	00:01:14	00:05:59
9	23	Sim, com 4	Não	00:05:05	00:05:40
10	24	Sim, com 3	Sim, 9 e 6	00:02:00	00:05:38
11	25	Sim, com 3	Não	00:04:33	00:05:35
12	26	Sim, com 4	Não	00:03:02	00:05:25
13	27	Não	Não	-	00:04:55
14	28	Não	Não	-	00:04:51
15	29	Não	Não	-	00:04:46
16	30	Não	Não	-	00:04:26
-	-	-	Total	00:41:11	01:37:10

Tabela 4.21 Informações coletadas referentes às simulações do Cenário 9 - trajetória circular com enxame com 30 integrantes.

Experimento	Fator	Tempo IV	Tempo FV	Duração	Passo	Distância	Velocidade
1	15	22:55:17	00:00:44	01:05:27	11.750	37,68	95,951108
2	16	00:27:40	01:30:20	01:02:40	11.106	37,68	100,212766
3	17	12:55:24	13:52:55	00:57:31	10.527	37,68	109,185743
4	18	14:47:55	15:40:32	00:52:37	10.007	37,68	119,353817
5	19	21:56:17	22:47:27	00:51:10	9.536	37,68	122,736156
6	20	23:11:33	00:02:10	00:50:37	9.106	37,68	124,069806
7	21	00:23:26	01:10:50	00:47:24	8.714	37,68	132,489451
8	22	14:05:37	14:52:03	00:46:26	8.355	37,68	135,247667
9	23	16:23:44	17:08:24	00:44:40	8.023	37,68	140,597015
10	24	17:31:04	18:13:49	00:42:45	7.717	37,68	146,900585
11	25	18:33:49	19:15:29	00:41:40	7.434	37,68	150,720000
12	26	19:34:12	20:14:58	00:40:46	7.170	37,68	154,047424
13	27	20:36:48	21:16:07	00:39:19	6.925	37,68	159,728699
14	28	22:21:20	22:57:36	00:36:16	6.696	37,68	173,161765
15	29	23:14:36	23:50:06	00:35:30	6.481	37,68	176,901408
16	30	00:11:41	00:46:22	00:34:41	6.280	37,68	181,066795
-	-	-	Total	12:29:29	-	-	-

Tabela 4.22 Informações complementares coletadas referentes às simulações do Cenário 9 - trajetória circular com enxame com 30 integrantes.

Experimento	Fator	Cerco	Divide	Tempo PC	Tempo UC
1	15	Sim, com 5	Sim, 25 e 5	00:01:51	00:08:23
2	16	Sim, com 5	Sim, 15 e 15	00:01:50	00:07:57
3	17	Sim, com 3	Não	00:01:41	00:07:48
4	18	Sim, com 3	Não	00:01:39	00:07:13
5	19	Sim, com 5	Não	00:02:06	00:07:02
6	20	Sim, com 4	Não	00:02:26	00:06:41
7	21	Sim, com 5	Sim, 13 e 17	00:01:54	00:06:26
8	22	Sim, com 3	Não	00:02:13	00:06:11
9	23	Sim, com 3	Sim, 15 e 15	00:02:08	00:05:57
10	24	Não	Não	-	00:05:47
11	25	Sim, com 5	Não	00:02:10	00:05:25
12	26	Não	Não	-	00:05:18
13	27	Não	Sim, 9 e 21	-	00:04:53
14	28	Não	Sim, 25 e 5	-	00:04:51
15	29	Não	Não	-	00:04:50
16	30	Não	Sim, 6 e 24	-	00:04:38

O gráfico da Figura 4.24 ilustra a quantidade de passos de tempo em função do fator de interpolação de cada experimento com o pássaro voando em trajetória circular. A cada passo de tempo o motor de física 3D do simulador ARGoS executa cálculos para configurar suas representações (x, y, z), e em seguida, atualizar o espaço 3D de representação visual. De acordo com esse gráfico, a quantidade de passos de tempo calculada durante os experimentos com o pássaro voando em trajetória circular foram iguais para cada experimento que tenha o mesmo fator de interpolação, independente da quantidade de robôs do enxame ($n = 3; n = 15; n = 30$). Percebe-se também que de acordo com o que foi planejado anteriormente, aumentando-se o valor do fator de interpolação diminui-se a quantidade de passos de tempo, permitindo assim alterar a velocidade de deslocamento do pássaro.

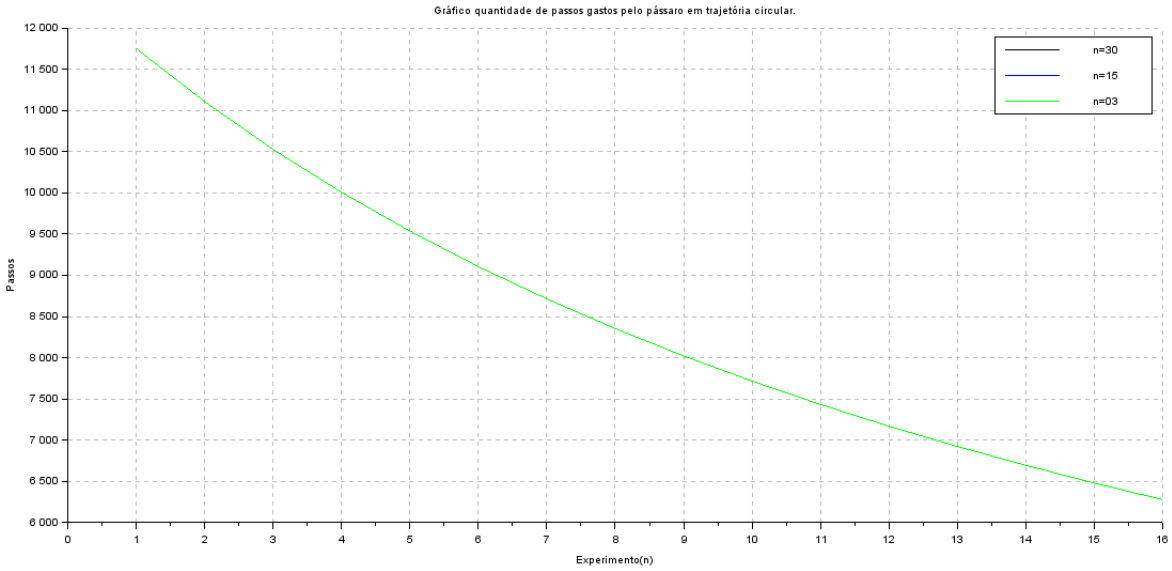

Figura 4.24: O gráfico ilustra a quantidade de tempos de passos em função do fator de interpolação de cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Analisando os gráficos das Figuras 4.25 e 4.26 conclui-se que quando o enxame tem:

- **3 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a sofrer algumas oscilações entre os pontos iniciais e finais. O enxame se divide em alguns momentos (em 2 experimentos) durante as atividades de monitoramento ou cerco (Veja Tabela 4.18).
- **15 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a sofrer algumas oscilações entre os pontos iniciais e finais. O enxame se divide em alguns momentos (em 4 experimentos) durante as atividades de monitoramento ou cerco (Veja Tabela 4.20).
- **30 robôs:** Ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a crescer entre os pontos inicial e final sem alterações bruscas. O enxame se divide em alguns momentos (em 7 experimentos) durante as atividades de monitoramento ou cerco (Veja Tabela 4.22).

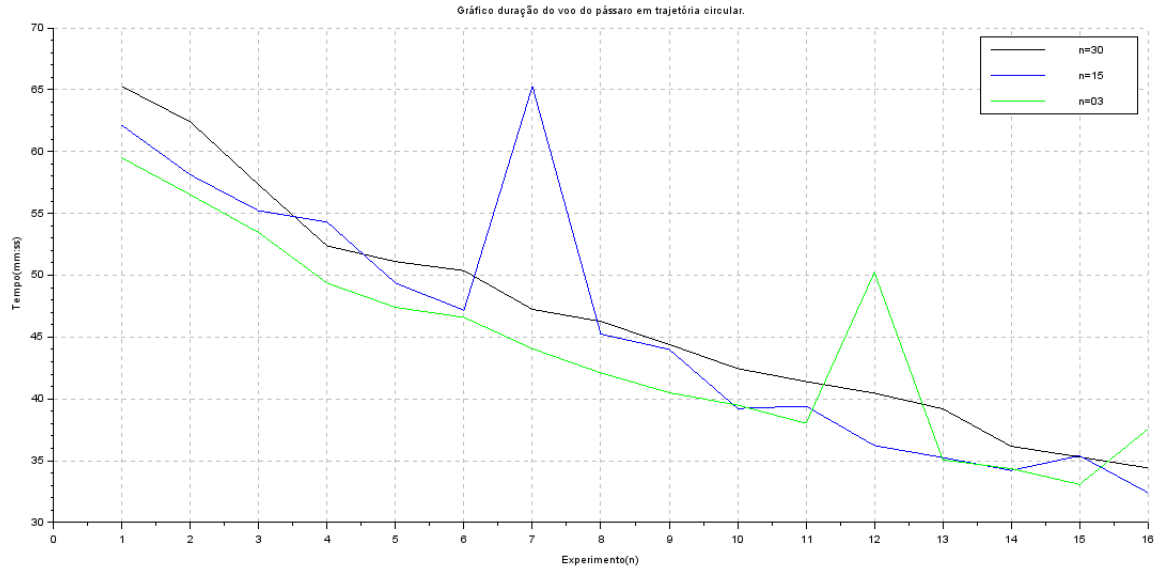

Figura 4.25: O gráfico ilustra a duração do voo em trajetória circular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

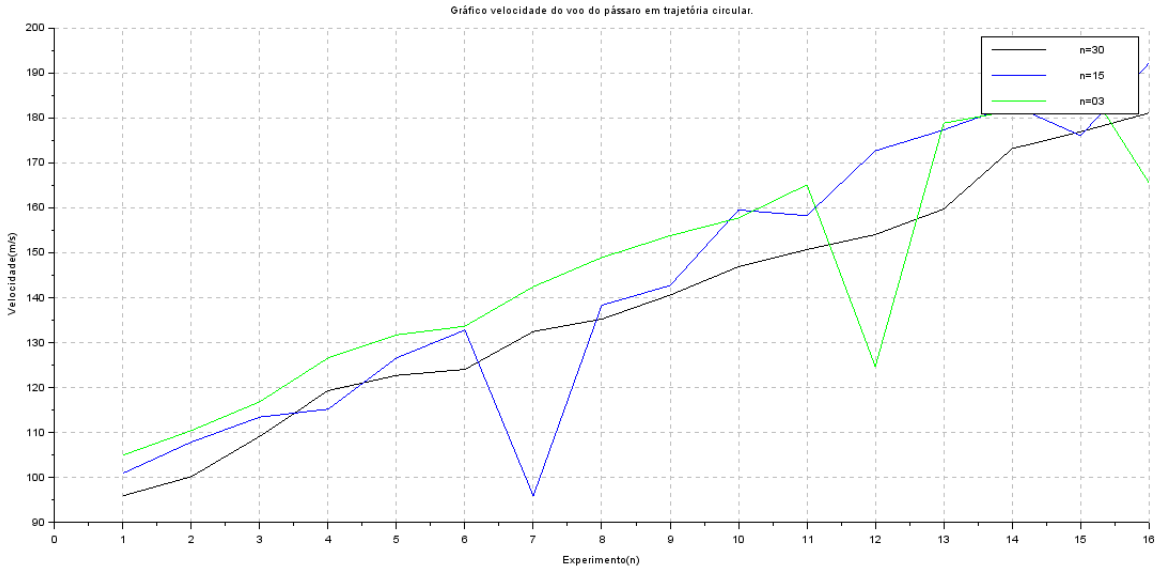

Figura 4.26: O gráfico ilustra a velocidade do voo em trajetória circular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

As Tabelas 4.18, 4.20 e 4.22 anteriormente citadas, ilustram o tempo gasto pelo enxame para cercar o pássaro pela primeira vez para cada experimento. Nas simulações com a trajetória circular verificou-se que em alguns experimentos dependendo da velocidade do pássaro o enxame só o cerca quando ele pousa.

O gráfico da Figura 4.27 ilustra o tempo gasto pelo enxame para realizar o cerco quando o pássaro pousa. Verifica-se que o tempo gasto pelo enxame com 3 integrantes é menor do que os demais (n=30 e n=15). Já o enxame com 30 robôs tende a demorar mais tempo para cercar o pássaro que os demais.

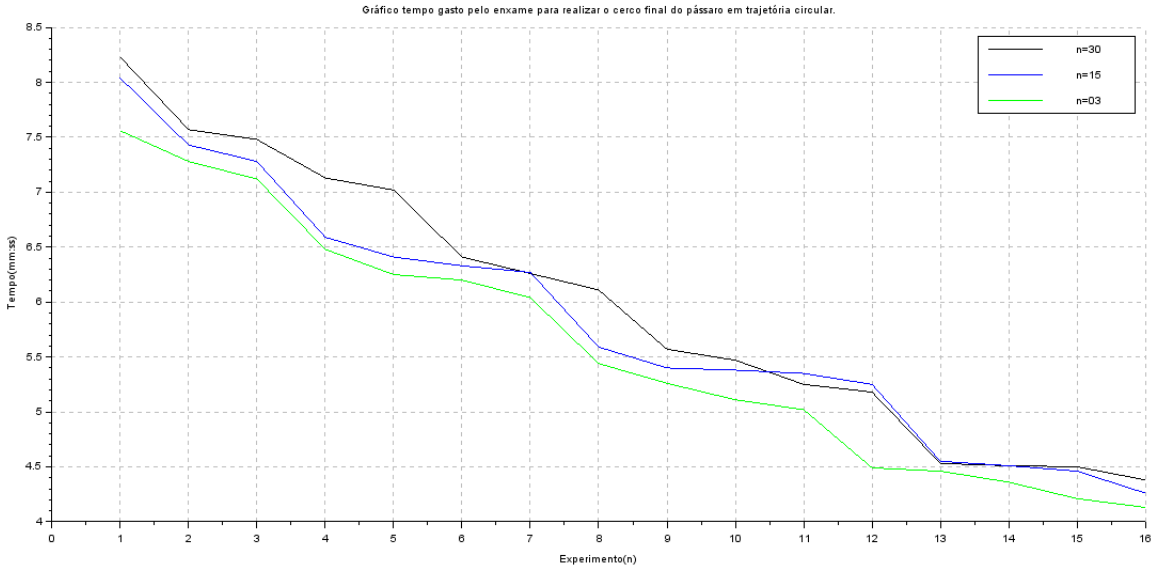

Figura 4.27: O gráfico ilustra o tempo gasto pelo enxame para cercar o pássaro pela última vez para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Com base nas informações anteriormente mencionadas das Tabelas 4.17, 4.19 e 4.21 o tempo total gasto durante os experimentos realizados para a trajetória circular do voo do pássaro, sem considerar o tempo de processamento das imagens em vídeos, foi de 36 horas, 35 minutos e 44 segundos, distribuídos em:

- **Cenário 7 (3 robôs):** 11 horas, 50 minutos e 43 segundos.
- **Cenário 8 (15 robôs):** 12 horas, 15 minutos e 32 segundos.
- **Cenário 9 (30 robôs):** 12 horas , 29 minutos e 29 segundos.

A Tabela 4.23 contem o resumo das principais características da abordagem de Enxame presentes em cada cenário referentes aos experimentos simulando um pássaro voando em trajetória circular. A análise foi efetuada com base nos códigos fontes e nos vídeos examinados.

Tabela 4.23 Características da abordagem de Enxame presentes nos experimentos realizados.

Cenário	Autônomo	Localidade	Coesão	ACC	Robustez	Flexibilidade	Escalabilidade	Similaridade	Auto-organizado
Cenário 7	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 8	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Cenário 9	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Onde as colunas:

- **ACC:** Representa se o enxame consegue dispersar o pássaro sem um controlador central.
- **Coesão:** Refere-se a conectividade dos integrantes do enxame durante o deslocamento com o intuito de verificar se é coeso.
- **Localidade:** Verifica se cada integrante do enxame tem um conhecimento limitado do local (extensão da área e número de integrantes do enxame).

4.2.3 Discussão Por Trajetória

Voo do pássaro em Trajetória Retilínea

Nas simulações com a trajetória retilínea verificou-se que em 100% dos experimentos o enxame conseguiu cercar o pássaro em dois momentos. Primeiro, durante o voo do pássaro antes de chegar no ponto final (P_f). Por último, quando o pássaro pousa (P_f). Sendo que a maior média do tempo de cerco quando o pássaro pousa foi do enxame (n=15) e a menor no enxame (n=30) e Tabela 4.24).

Tabela 4.24 Contém os percentuais sobre a capacidade do enxame para cercar o pássaro em voo e sofrer divisões em trajetória retilínea

Cenário	Trajetória	N robôs	Enxame efetua o cerco? sim (%)	não (%)	Enxame sofre divisões? sim (%)	não (%)	Tempo Médio Cerco (hh:mm:ss)
1	Retilínea	3	100	0	0	100	00:09:11
2	Retilínea	15	100	0	37,50	62,50	00:09:24
3	Retilínea	30	100	0	0	100	00:09:10

Quando o enxame tem **3 robôs** o enxame não se divide durante as atividades de monitoramento ou cerco (Tabela 4.24).

O enxame com **15 robôs** se divide em 37,5% dos experimentos durante as atividades de cerco. Apesar dessa divisão, o enxame tende a se reagrupar novamente (Tabela 4.24). Já o enxame com **30 robôs** não se divide durante as atividades de monitoramento (Tabela 4.24).

Voo do pássaro em Trajetória Semicircular

Embora o enxame com **3 robôs** só consiga cercar o pássaro depois que ele pousa, o enxame fica constantemente próximo do pássaro tentando cercá-lo, mostrando-se apto a afugentá-lo. O enxame se divide em 12,5 % dos experimentos durante as atividades de cerco (Tabela 4.25).

Tabela 4.25 Contém os percentuais sobre a capacidade do enxame para cercar o pássaro em voo e sofrer divisões em trajetória semicircular

Cenário	Trajetória	N robôs	Enxame efetua o cerco? sim (%)	não (%)	Enxame sofre divisões? sim (%)	não (%)	Tempo Médio Cerco (hh:mm:ss)
4	Semicircular	3	0	100	12,50	87,50	00:03:02
5	Semicircular	15	50	50	43,75	56,25	00:03:24
6	Semicircular	30	56,25	43,75	12,50	87,50	00:03:43

O enxame com **15 robôs** se divide em 43,75% dos experimentos durante as atividades de cerco. Apesar dessa divisão, o enxame tende a se reagrupar novamente obtendo 50% de sucesso em efetuar o cerco do pássaro durante o voo (Tabela 4.25) .

O enxame com **30 robôs** se divide em 12,5% dos experimentos durante as operações de cerco. O enxame consegue em 56,25 % dos experimentos cercar o pássaro durante o voo com parte de seus integrantes (Tabela 4.25).

A capacidade de efetuar o cerco do pássaro durante o voo cresce a medida que se aumentam o número de robôs. Sendo mais eficiente o enxame (n=30). As divisões ficaram constantes (12,5%) para os enxames (n=3; n=30), enquanto que para n=30 o percentual foi de 43,75% (Tabela 4.25). A maior média do tempo de cerco quando o pássaro pousa (P_f) foi do enxame (n=30) e a menor no enxame (n=03).

Voo do pássaro em Trajetória Circular

O enxame com **3 robôs** se divide em 12,5% dos experimentos durante as atividades de cerco. Apesar dessa divisão, o enxame tende a se reagrupar novamente (Tabela 4.26) obtendo 31,25% de sucesso em efetuar o cerco do pássaro durante o voo.

Ao aumentar o fator de interpolação no enxame com **15 robôs** a velocidade do voo do pássaro tende a crescer sofrendo algumas oscilações entre os pontos iniciais e finais. O enxame se divide em 25% dos experimentos durante as atividades de cerco. Apesar dessa divisão, o enxame tende a se reagrupar novamente (Tabela 4.26) obtendo 75% de sucesso em efetuar o cerco do pássaro durante o voo.

Já o enxame com **30 robôs** ao aumentar o fator de interpolação, a velocidade do voo do pássaro tende a crescer entre os pontos inicial e final sem alterações bruscas. O enxame se divide em 43,75% dos experimentos durante as operações de cerco. O enxame consegue em 62,50% dos experimentos cercar o pássaro durante o voo com parte de seus integrantes (Tabela 4.26). A maior média do tempo de cerco quando o pássaro pousa (P_f) foi do enxame (n=30) e a menor no enxame (n=3).

Tabela 4.26 Contém os percentuais sobre a capacidade do enxame para cercar o pássaro em voo e sofrer divisões em trajetória circular.

Cenário	Trajetória	N robôs	Enxame efetua o cerco?		Enxame sofre divisões?		Tempo Médio Cerco (hh:mm:ss)
			sim (%)	não (%)	sim (%)	não (%)	
7	Circular	3	31,25	68,75	12,50	87,50	00:05:46
8	Circular	15	75	25	25	75	00:06:04
9	Circular	30	62,50	37,50	43,75	56,25	00:06:12

4.3 Avaliando o Desempenho do *Hardware* Durante a Realização dos Experimentos

Foram coletadas informações com o objetivo de comparar o desempenho do hardware durante a execução de cada experimento simulado. Esta avaliação tem como motivação verificar de que forma as variações de parâmetros (quantidade de indivíduos do enxame, tipo de trajetória e velocidade) dos experimentos podem influenciar no desempenho do *hardware*. O *hardware* testado foi o mesmo ambiente computacional anteriormente citado no Capítulo 3, na Seção 3.7.

O instrumento utilizado para efetuar a medição foi a observação por meio dos dados coletados para o processo de mensuração. Foram consideradas as métricas do tipo recurso para efetuar as medições. A coleta dos dados foi realizada por meio da função `getrusage( )`. Esta função obtém as informações do sistema operacional referentes aos recursos disponibilizados, conforme ilustrado na Figura 4.28.

```
struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size */
    long   ru_ixrss;         /* integral shared memory size */
    long   ru_idrss;         /* integral unshared data size */
    long   ru_isrss;         /* integral unshared stack size */
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
    long   ru_nswap;         /* swaps */
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
    long   ru_msgsnd;        /* IPC messages sent */
    long   ru_msgrcv;        /* IPC messages received */
    long   ru_nsignals;      /* signals received */
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};
```

Figura 4.28: Conjunto de recursos disponibilizados por meio da função getrusage( ).

Depois de analisar estes conjuntos de recursos, decidiu-se que seriam medidas a quantidade de memória e taxa de utilização da CPU.

4.3.1 Memória Utilizada

Verifica-se que a quantidade de memória utilizada pelos processos referentes aos experimentos simulados tende a crescer a medida que o número de integrantes do enxame aumenta. As Tabelas 4.27 e 4.28 contêm os dados referentes a estes experimentos.

Tabela 4.27 Quantidade de memória utilizada nos experimentos de nºs 15 a 22

Trajetória	Nº de robôs	15	16	17	18	19	20	21	22
Circular	3	77.596	77.500	76.972	77.144	77.440	77.536	76.904	77.384
Circular	15	133.412	130.748	127.584	125.316	119.544	118.556	113.516	117.336
Circular	30	244.900	236.192	234.376	216.944	212.008	199.064	204.204	186.436
Semicircular	3	76.636	76.060	75.984	76.048	75.956	76.284	76.536	76.148
Semicircular	15	43.128	43.128	43.116	43.120	43.116	43.132	43.108	43.116
Semicircular	30	179.940	168.920	159.640	159.228	162.740	158.204	167.156	160.504
Retilínea	3	79.120	78.800	78.448	78.760	78.548	77.736	78.300	77.756
Retilínea	15	181.696	173.276	168.504	167.036	158.272	154.432	151.976	148.868
Retilínea	30	393.988	376.168	370.180	344.324	333.380	323.040	319.000	293.040

Tabela 4.28 Continuação da Tabela 4.27- Quantidade de memória utilizada nos experimentos de nºs 23 a 30

Trajetória	Nº de robôs	23	24	25	26	27	28	29	30
Circular	3	76.980	74.420	74.828	74.588	74.528	74.452	76.276	74.892
Circular	15	111.344	109.508	115.660	109.476	105.104	103.652	103.272	107.420
Circular	30	190.720	181.816	181.760	180.348	180.128	173.416	171.544	180.268
Semicircular	3	76.256	75.800	75.800	75.808	75.672	76.016	75.832	76.004
Semicircular	15	43.120	43.108	43.116	43.116	43.120	43.124	43.116	43.128
Semicircular	30	151.252	144.500	148.672	153.596	156.020	139.940	147.392	133.300
Retilínea	3	78.120	77.788	78.016	77.580	77.888	77.408	77.664	77.484
Retilínea	15	145.556	145.992	142.740	140.276	140.440	135.396	131.212	129.632
Retilínea	30	282.568	275.620	266.584	286.408	256.040	254.212	249.432	240.636

Para os enxames com 3, 15 e 30 integrantes, a média aritmética de memória utilizada (em *kilobytes*) é de, respectivamente:

(a) 76.215, 115.716 e 198.383, para a trajetória circular.

(b) 76.053, 43.120 e 155.688, para a trajetória semicircular.

(c) 78.089, 150.957 e 304.039, para a trajetória retilínea.

A média aritmética de memória utilizada (em *kilobytes*) independente da quantidade de integrantes do enxame considerando apenas o tipo de trajetória é de:

(a) 130.104,42 para a trajetória circular.

(b) 91.619,92 para a trajetória semicircular.

(c) 177.694,58 para a trajetória retilínea.

Os gráficos das Figuras 4.29 , 4.30 e 4.31 ilustram a quantidade de memória utilizada nos experimentos realizados referentes aos dados das Tabelas 4.27 e 4.28 anteriormente comentadas.

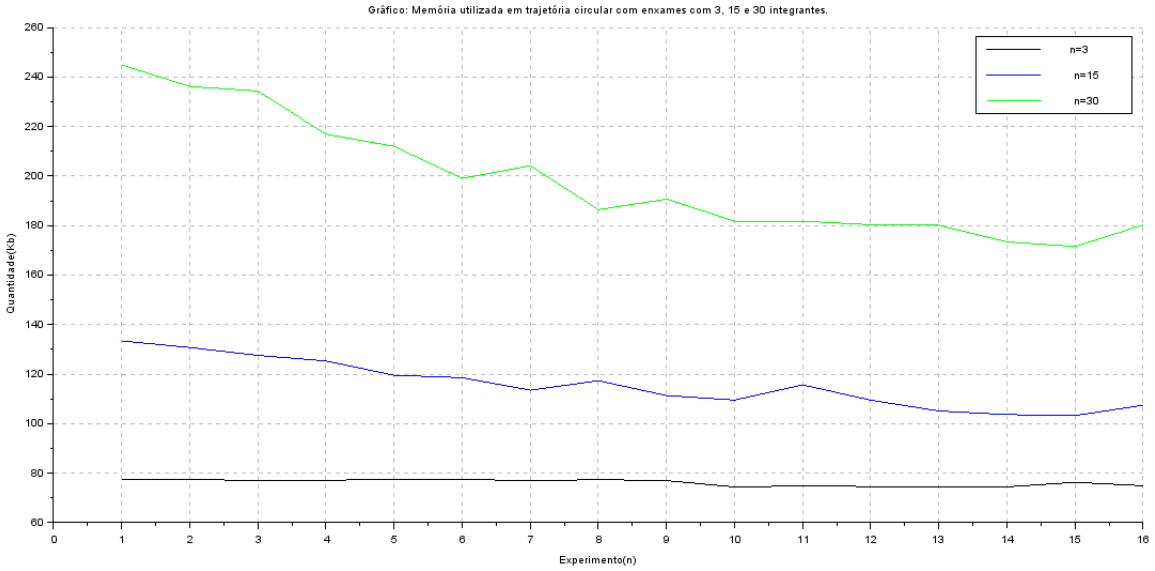

Figura 4.29: O gráfico ilustra a quantidade de memória utilizada em trajetória circular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

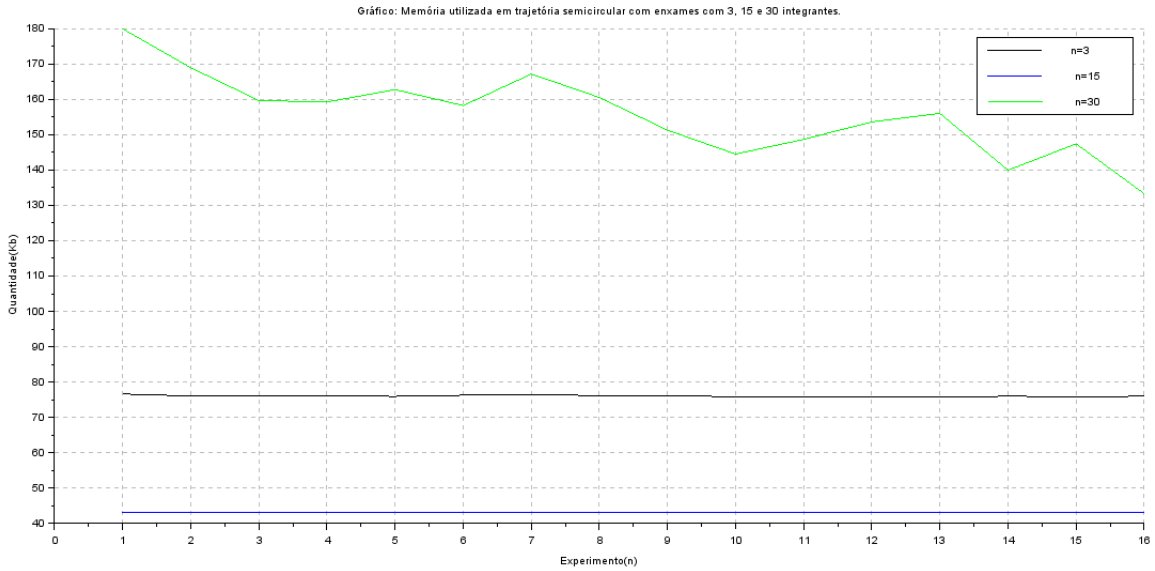

Figura 4.30: O gráfico ilustra a quantidade de memória utilizada em trajetória semicircular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

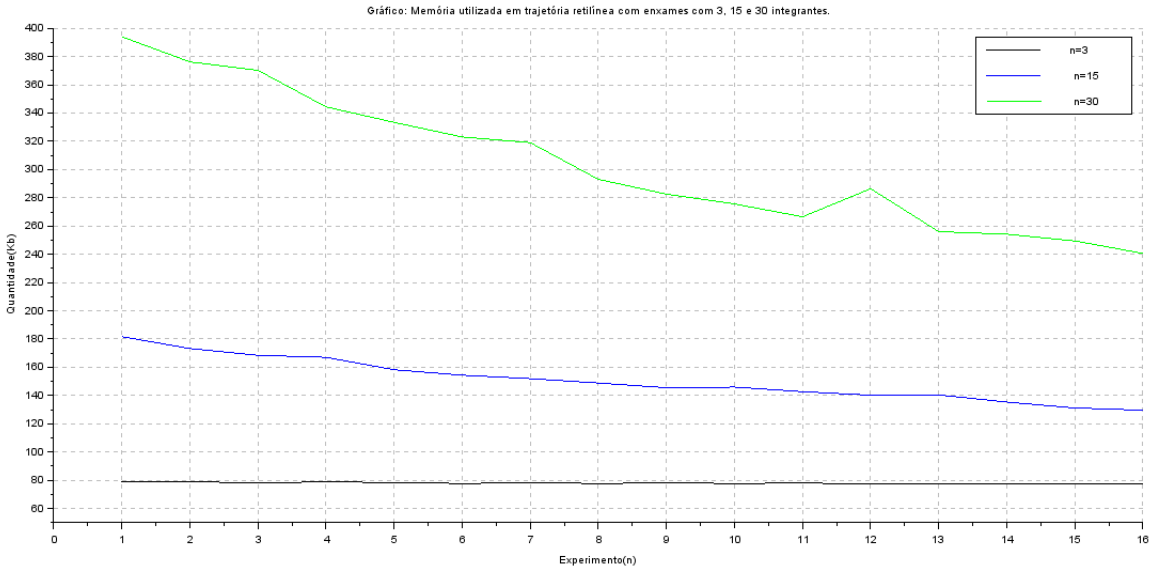

Figura 4.31: O gráfico ilustra a quantidade de memória utilizada em trajetória retilínea para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

4.3.2 Taxa de Utilização da CPU

Verifica-se que a taxa de CPU (0 a 100%) utilizada pelos processos referentes aos experimentos simulados tende a crescer a medida que o número de integrantes do enxame aumenta. As Tabelas 4.29 e 4.30 contêm os dados referentes a estes experimentos.

Tabela 4.29 Quantidade de CPU utilizada nos experimentos de nºs 15 a 22

Trajetória	Nº de robôs	15	16	17	18	19	20	21	22
Circular	3	1,84131	1,83247	1,82518	1,85976	1,93268	1,81227	1,94141	1,85087
Circular	15	14,2566	14,0026	14,4151	14,1493	14,0096	14,1783	14,0549	14,2505
Circular	30	26,7228	26,3292	26,6032	26,061	26,4025	26,4656	26,1871	25,8685
Semicircular	3	1,8928	1,99902	1,89734	2,00109	2,00405	1,88172	2,12766	1,91193
Semicircular	15	4,83226	4,93362	4,84903	4,89061	4,87385	4,8746	4,8994	4,85963
Semicircular	30	8,46009	8,5911	8,71732	8,46289	8,5341	8,51139	8,57075	8,63145
Retilínea	3	1,9783	1,81635	1,94976	1,79818	1,80242	1,99121	1,82671	1,96053
Retilínea	15	4,87184	4,81174	4,8406	4,82806	4,84881	4,89443	4,89244	4,80666
Retilínea	30	8,83433	8,81792	8,71894	8,7581	8,7958	8,77979	8,94527	8,84356

Tabela 4.30 Continuação da Tabela 4.29 - Quantidade de CPU utilizada nos experimentos de nºs 23 a 30

Trajetória	Nº de robôs	23	24	25	26	27	28	29	30
Circular	3	5,21	5,51259	5,57176	5,28073	5,42055	5,39143	5,35967	5,3838
Circular	15	14,0785	14,0991	13,4543	13,9373	14,1163	14,1189	13,6665	14,0054
Circular	30	25,8106	26,07	26,2036	25,53484	25,51277	25,70348	25,51652	25,56266
Semicircular	3	1,96908	1,96144	1,96905	1,98604	2,07111	1,95156	2,12128	2,1049
Semicircular	15	4,94316	4,94454	4,98393	4,83348	4,84909	4,95201	4,98525	5,04859
Semicircular	30	8,47821	8,44471	8,48253	8,46191	8,62879	8,568	8,27247	8,64611
Retilínea	3	1,94283	1,81419	1,82652	1,80393	1,82237	1,99787	1,86628	1,82991
Retilínea	15	4,78739	4,79559	4,90675	4,88739	4,88653	4,85439	4,84305	4,87561
Retilínea	30	8,72614	8,79218	8,78122	8,77381	8,67877	8,75548	8,6835	8,72784

Para os enxames com 3, 15 e 30 integrantes, a taxa média aritmética de CPU utilizada é de, respectivamente:

(a) 3,62%, 14,04% e 26,03%, para a trajetória circular.

(b) 1,99%, 4,90% e 8,52% , para a trajetória semicircular.

(c) 1,87%, 4,85% e 8,77%, para a trajetória retilínea.

A taxa média aritmética de CPU utilizada (em *kilobytes*) independente da quantidade de integrantes do enxame considerando apenas o tipo de trajetória é de:

(a) 14,57% para a trajetória circular.

(b) 5,14% para a trajetória semicircular.

(c) 5,16% para a trajetória retilínea.

Os gráficos das Figuras 4.29 , 4.30 e 4.31 , ilustram a taxa de utilização da CPU utilizada nos experimentos realizados referentes aos dados das Tabelas 4.29 e 4.30 anteriormente comentadas.

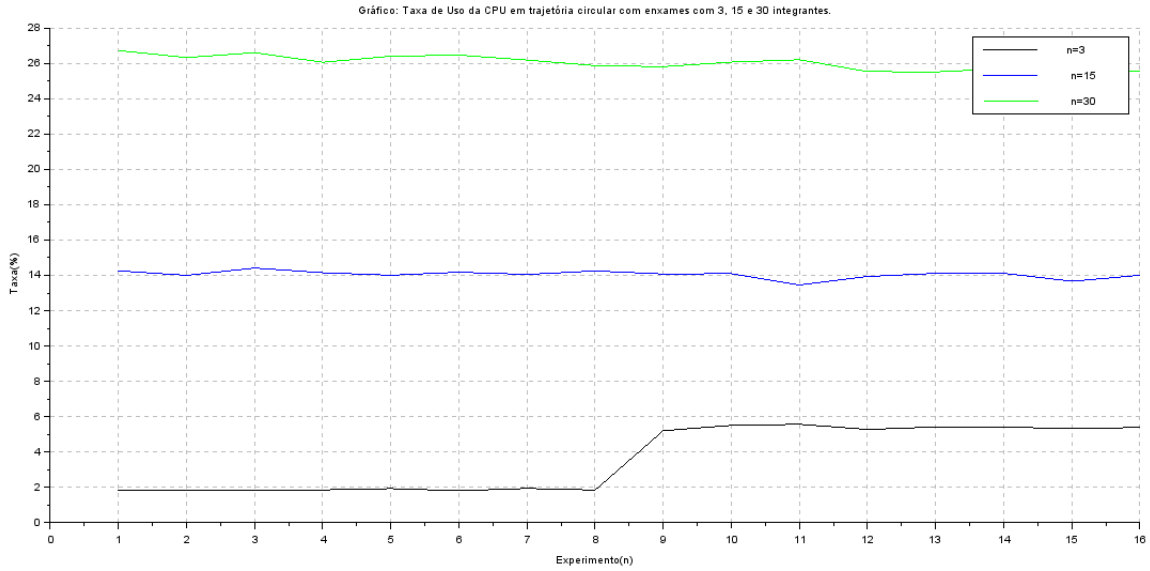

Figura 4.32: O gráfico ilustra a taxa de CPU utilizada em trajetória circular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

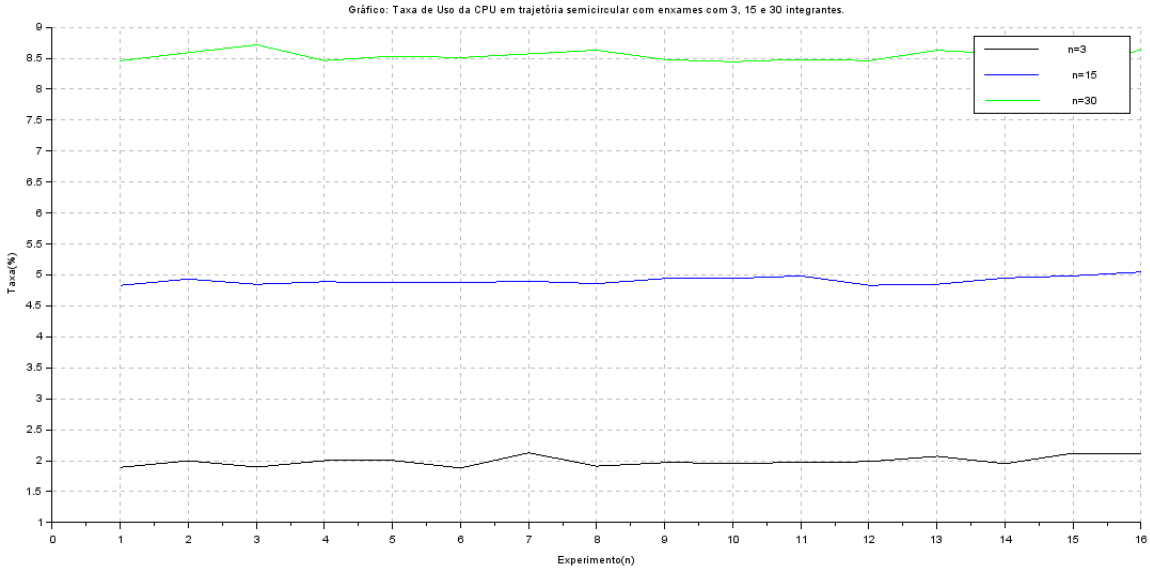

Figura 4.33: O gráfico ilustra a taxa de CPU utilizada em trajetória semicircular para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

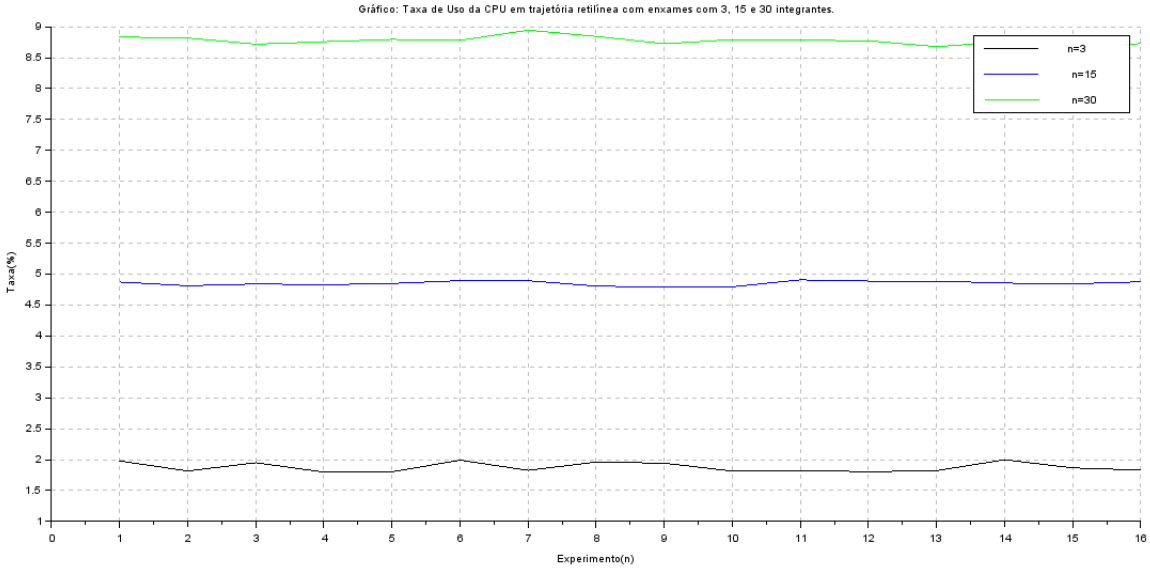

Figura 4.34: O gráfico ilustra a taxa de CPU utilizada em trajetória retilínea para cada experimento. Fonte: Ilustração do próprio autor dessa dissertação.

Capítulo 5

Conclusão

Os avanços tecnológicos aplicados a agricultura, além de aumentarem a produção, reduzem a agressão ao meio ambiente e com isso, o produto pode ficar mais barato e o consumidor ter uma melhora significativa da saúde. Nessa visão, uma das finalidades da aplicação da robótica na agricultura é tentar obter maior lucro, gastando menos com insumos e impactando menos o ambiente.

Cabe recordar as indagações sobre as quais se construiu a pesquisa apresentada neste trabalho. A primeira questionava a possibilidade de obter um método ecológico e alternativo para a deslocalização dos pássaros em plantações. As simulações demonstraram que isso é possível. A nova abordagem não utiliza agentes químicos (repelentes ou avicidas) e não é necessária a realização de poda na plantação.

A próxima questão levantava o problema da habituação dos pássaros às técnicas tradicionais. Na ocasião, observou-se que uma nova técnica só poderia anular a adaptatividade dos pássaros se fosse, ela mesma, adaptativa. Assim, chegou-se à abordagem de ER e, mais especificamente, à implementação da solução baseada em um padrão de cerco mediante o uso da técnica que lança mão da utilização do potencial de *Lennard-Jones*.

5.1 Discussão Geral dos Experimentos

Com base na análise analítica e estatística de cada vídeo gerado nos experimentos realizados foram observadas várias características inerentes a abordagem Enxame, que seguem descritas a seguir:

1. A técnica do potencial de *Lennard-Jones* permitiu criar as trajetórias individuais de cada integrante do enxame, tornando-os **autônomos**. Cada integrante do enxame não sabe a quantidade total de robôs (conhecimento global) pertencentes ao enxame que estão perseguindo o pássaro para a **formação do grupo coeso.** Cada robô sabe apenas quantos são os seus n-vizinhos (conhecimento limitado) que irá considerar para o cálculo do potencial de *Lennard-Jones*, baseado na distância determinada de alcance para detecção dos seus vizinhos.
2. O comportamento global do enxame surge como resultado das **interações locais** de cada sensor de distância dos robôs com o pássaro e com os outros vizinhos. Fica explícito que não há um **controlador central** que determine a sequência ordenada das tarefas para atingir um comportamento esperado.
3. Percebe-se a **robustez** do enxame quando em alguns momentos inesperados ele se divide em pequenos enxames durante a perseguição do pássaro, voltando a se reagrupar. Cada pequeno enxame permanece perseguindo o pássaro. Isso demostra que a **redundância de robôs** no enxame, permite pequenas variações de comportamento de alguns integrantes do enxame não gere uma falha total no enxame.
4. Verifica-se a **flexibilidade** do enxame diante das mudanças de trajetórias do pássaro **(mudanças de contexto de ambiente)** o enxame se adapta de acordo com as novas posições do pássaro, reagindo aos eventos sem precisar ser programado novamente.

5. Nota-se a **escalabilidade** do enxame, pois ao alterar a quantidade de robôs no enxame, os mecanismos de coordenação asseguram o funcionamento de forma **flexível** sem a necessidade de reprogramação.

6. Os robôs que compõem o enxame realizam a mesma tarefa **(homogeneidade)**, ou seja, perseguir o pássaro. Além disso, cada robô tem seus sensores que detectam a distância dos vizinhos e do pássaro e se movem em direção do pássaro e não colidem com os robôs vizinhos. Isto demonstra a **simplicidade** da capacidade individual de cada robô para a realização de uma tarefa.

7. Nos experimentos, verifica-se que cada robô calcula a distância com seus vizinhos por meio dos seus próprios sensores. Cada robô não precisa perguntar qual a distância dos robôs próximos a ele, pois ele mesmo detecta e calcula. Essa interação local é uma forma de **comunicação limitada (estigmergia).** Cada robô obtém informações locais com seus vizinhos mais próximos, não sobrecarregando o enxame com trocas de informações.

Com base na definição de Trianni já apresentada no Capítulo 2, na Subseção 2.2.3, analisando os experimentos, é possível verificar que o sistema é auto-organizado por ser:

1. **Descentralizado**: os robôs são autônomos em seus cálculos de distância, na geração de suas trajetórias, não havendo um controle central.

2. **Flexível**: o enxame tem a tendência natural de manter suas atividades de perseguir o pássaro e formar um padrão de cerco, sendo resistente as alterações das trajetórias do pássaro **(variações ambientais)**.

3. **Local**: cada robô se baseia em apenas informações locais (distância dos vizinhos e do pássaro). O comportamento de cada robô é modelado com regras simples (técnicas de campos potenciais de LJ).

Foi possível a realização de experimentos em um ambiente virtual e se alcançar um entendimento sobre o comportamento do enxame dentro dos contextos dos cenários representados. A simulação representou a linha metodológica e com base nos resultados dos experimentos, demonstrou-se que a aplicação da abordagem enxame é uma boa estratégia de coesão entre os robôs em áreas a serem monitoradas.

Com esse estudo verifica-se a importância em analisar qual o tipo de pássaro que está ocasionando o dano na plantação e verificar a sua velocidade média de voo, para realizar as configurações necessárias nos sensores de velocidade de cada robô, para que o enxame se movimente com velocidades que possam permitir um melhor monitoramento.

Como demonstrado, se a velocidade de voo do pássaro for, muito superior a velocidade de deslocamento do enxame, a tendência é que o enxame cerque-o, apenas quando ele pousar na plantação. Por outro lado, se a velocidade de voo do pássaro for muito inferior a velocidade de deslocamento do enxame, a tendência é que o pássaro seja cercado logo ao invadir plantação, sendo persuadido a sair durante um maior período de tempo.

Percebe-se que os robôs na simulação não possuem uma percepção limitada do ambiente, uma vez que seus sensores estão dispostos de forma circular, não existindo pontos cegos e nem limitações referentes à distância de alcance do sensor de luz.

5.2 Contribuições

Com base nos resultados obtidos por meio da simulação realizada, verificou-se que a aplicação do método do potencial de *Lennard-Jones* é uma boa estratégia para o controle de um ER para o objetivo proposto. Ele permite que o grupo de robôs se mova em forma coesa, tal qual um grupo de insetos sociais faria, mantendo o alinhamento, a atração e a repulsão entre os integrantes do grupo.

Uma contribuição dessa dissertação foi a publicação do artigo "*Swarm robotics:* comportamento adaptativo aplicado ao problema de dispersão de pássaros em áreas agrícolas" [65], em 2014, no

VIII *Workshop* de Tecnologia Adaptativa, na Escola Politécnica da Universidade de São Paulo, demonstrando que a estratégia de controle de robôs, por meio de campos potenciais de *Lennard-Jones*, pode ser empregada para dispersar pássaros de áreas agrícolas com o uso de múltiplos robôs empregando a abordagem de ER.

Essa estratégia de controle possibilita o monitoramento da área por todos os integrantes do enxame simultaneamente, cercando o pássaro mesmo enquanto ele voa de um local para o outro, até dispersá-lo da região agrícola.

Cabe ressaltar que a intenção desse trabalho não foi abordar de forma exaustiva o campo de estudo do enxame em si, nem tão pouco comparar um método com outro, mas aplicá-lo como uma forma de espantar pássaros. Ressaltando que não foram encontrados relatos desse tipo de aplicação na literatura.

Apesar das simulações serem realizadas apenas com robôs terrestres, a abordagem de ER com *Lennard-Jones* pode ser aplicada a robôs com capacidade de voar bastando adaptá-la com as novas habilidades dos robôs.

Dentro das limitações financeiras e do escopo inicial proposto, foi direcionada a aplicação da solução por meio de simulações da abordagem ER com o campo potencial de *Lennard-Jones* para a coesão dos integrantes do enxame.

5.3 Possibilidades de Pesquisas Futuras

Como parte da sequência a ser dada à pesquisa e a esse trabalho, pode-se investigar uma série de outros caminhos interessantes. Um deles consiste em realizar a simulação com mais de um pássaro e estudar as implicações ocasionadas por essa mudança. Outro caminho consiste no uso de robôs reais e a análise do comportamento do enxame em um ambiente agrícola também real.

O estudo dos efeitos na solução proposta quando considerado: o coeficiente de atrito do solo, o deslocamento do ER em várias condições de superfície (úmidas, seca, pedregulho ou arenosa), as condições ambientais (chuva ou vento) e terrenos irregulares e inclinados

Outra questão a ser estudada é a utilização da composição de técnicas de dispersão de pássaros, por exemplo, luz e som, junto com o ER para dificultar a habituação dos pássaros. Outro ponto para futuros trabalhos diz respeito ao uso de quadrotores (veículos aéreos impulsionados por quatro hélices) robóticos, pois ter robôs que voam tal quais os pássaros pode abrir todo um novo leque de possibilidades.

Existem estudos interessantes que também envolvem os aspectos teóricos. Uma análise que será realizada consiste na aplicação dos Autômatos Adaptativos [44] na programação dos robôs que integram o enxame e estudar a composição do comportamento dos Autômatos Adaptativos com a abordagem de ER. Com um sistema hibrido que combinasse as duas abordagens, existe a possibilidade de um ganho significativo de desempenho para a solução apresentada neste trabalho.

Referências Bibliográficas

[1] Abidin, Z., Arshad, M., and Ngah, U. An introduction to swarming robotics: application development trends. *Artificial Intelligence Review 43*, 4 (2015), 501–514.

[2] Ahmed, A. A., Abdalla, T. Y., and Abed, A. A. Path planning of mobile robot by using modified optimized potential field method. *International Journal of Computer Applications 113*, 4 (2015).

[3] Bahgaat, N. K., El-Sayed, M., Hassan, M. M., and Bendary, F. Load frequency control in power system via improving pid controller based on particle swarm optimization and anfis techniques. *Research Methods: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications* (2015), 462.

[4] Beni, G. From swarm intelligence to swarm robotics. *Workshop on Swarm Robotics. 8th International Conference on Simulation of Adaptive Behavior (SAB) 3342* (2004), 1–9.

[5] Bhaumik, A. Open source robotics: Multi-robot simulators. *Linux For You 10*, 2 (2012), 48–50.

[6] Blum, C., and Gross, R. Swarm intelligence in optimization and robotics. In *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 1291–1309.

[7] Bonani, M., Longchamp, V., Magnenat, S., Retornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., and Mondada, F. The marxbot a miniature mobile robot opening new perspectives for the collective-robotic research. *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS* (2010), 4187–4193.

[8] Cavagna, A., Queirós, S. D., Giardina, I., Stefanini, F., and Viale, M. Diffusion of individual birds in starling flocks. *Proceedings of the Royal Society B: Biological Sciences 280*, 1756 (2013), 20122484.

[9] Chaimowicz, L. Curso de robótica móvel. *Instituto de Ciências Exatas - UFMG* (2014).

[10] Chialvo, D. R., and Millonas, M. M. How swarms build cognitive maps. In *The Biology and Technology of Intelligent Autonomous Agents*. Springer, 1995, pp. 439–450.

[11] Clark, L. *Disease Risks· Posed by Wild Birds Associated with Agricultural Landscapes*. University of Nebraska Lincoln, 2014, ch. 7, pp. 140–165.

[12] Cornel, U. H. D. R. U. *The Economic Impact Of Bird Damage to Wine Grapes*. United States Department of Agriculture, USA, 2012.

[13] de Mey, Y., and Demont, M. *Realizing Africa's Rice Promise*. CABI, 2013, ch. 19 Bird Damage to Rice in Africa: Evidence and Control, pp. 241–249.

[14] Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chrétien, L. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animals* (1991), pp. 356–363.

[15] Dorigo, M., and Gambardella, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation 1*, 1 (1997), 53–66.

[16] Dorigo, M., Maniezzo, V., and Colorni, A. Ant system: Optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics 26*, 1 (1996), 29–41.

[17] Dudek, G., and Jenkin, M. *Computational Principles of Mobile Robotics*, vol. 1. Cambridge Universit Press, 2010.

[18] Ejiogu, A. O., and Okoli, V. Bird scaring technologies in rice production: The need for policies prohibiting participation of women and children. *Journal of Food Science and Technology 1*, 3 (june 2012), 31–38.

[19] Fayle, T. M. Moral hazard in ecology. *Frontiers in Ecology and Evolution 3* (January 2015), 2.

[20] Ferrari, F. A new parameterized potential family for path planning algorithms. *International Journal on Artificial Intelligence Tools 18*, 6 (2009), 949–957.

[21] Filho, J. C. M. Dissertação: Modelos computacionais para o processo de forrageamento e facilitação social em cupins. *Universidade Federal de Vicosa* (2007).

[22] Freese, M., Singh, S., Ozaki, F., and Matsuhira, N. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 51–62.

[23] Gazi, V., Fİdan, B., Hanay, Y. S., and Köksal, İ. Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques. *Turkish Journal of Electrical Engineering & Computer Sciences 15*, 2 (2007), 149–168.

[24] Gerkey, B., Vaughan, R. T., and Howard, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics* (2003), vol. 1, pp. 317–323.

[25] Ghosh, S. K., and Roy, B. Some results on point visibility graphs. *Theoretical Computer Science 575* (2015), 17–32.

[26] Gomes, A. d. S., and Magalhães Júnior, A. M. *Arroz Irrigado no Sul do Brasil*. Embrapa Clima Temperado Informação Tecnológica, 2004.

[27] Gorenzel, P., and Salmon, T. *Bird Hazing Manual: Techniques and Strategies for Dispersing Birds from Spill Sites*. UCANR Publications, 2008.

[28] Grassé, P.-P. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux 6*, 1 (1959), 41–80.

[29] Hofstadter, D. R., and Godel, E. B. An eternal golden braid. *New York: Basic Books* (1979).

[30] Hölldobler, B., and Wilson, E. O. *The Ants*. Harvard University Press, 1990.

[31] Jacinto, J. C., Toti, T. P., Guarita, Lucas, R., and Melo, C. Dano em um cultivo de sorgo (sorghum bicolor) causado por aves. In *VIII Congresso de Ecologia do Brasil* (2007).

[32] Kazadi, S., Jin, D., and Li, M. Utilizing abstract phase spaces in swarm design and validation. In *Advances in Swarm and Computational Intelligence*. Springer, 2015, pp. 14–29.

[33] Kennedy, J. *Swarm Intelligence*. Springer, 2006.

[34] Kittel, C. *Introduction to Solid State Physics*. John Wiley Inc., 1976.

[35] Koenig, N., and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (2004), vol. 3, IEEE, pp. 2149–2154.

[36] Linz, G. M., and Homan, H. J. Demonstration of avian control® bird repellent (ai, methyl anthranilate) for managing blackbird damage to ripening sunflower. *Wildlife Damage Management, Internet Center for Publications* (2013).

[37] Lorenz, K. Behind the mirror: A search for a natural history of human knowledge. *New York: Harcourt Brace Jovanovich* (1973).

[38] Marques, A. T., Batalha, H., Rodrigues, S., Costa, H., Pereira, M. J. R., Fonseca, C., Mascarenhas, M., and Bernardino, J. Understanding bird collisions at wind farms: An updated review on the causes and possible mitigation strategies. *Biological Conservation 179* (August 2014), 40–52.

[39] Medeiros, A. R. M. Figueira do plantio ao processamento caseiro. *EMBRAPA-Circular Técnica*, 35 (2002).

[40] Medina, A. J. R., Pulido, G. T., and Ramírez-Torres, J. G. A comparative study of neighborhood topologies for particle swarm optimizers. In *IJCCI* (2009), pp. 152–159.

[41] Menq, W. *Aves de rapina e suas técnicas de caça. Aves de Rapina BR - águias, gaviões, falcões e corujas do Brasil.* 2013.

[42] Millonas, M. M. *Swarm, Phase Transitions, and Colective Intelligence.* C. G. Langton, 1994.

[43] Murphy, R. R. *Introduction to AI Robotics.* MIT Press, 2000.

[44] Neto, J. J. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. *Revista IEEE América Latina 5*, 7 (2007), 1548–0992.

[45] Neto, J. R. S., and Gomes, D. M. Predação de milho por arara-azul-de-lear anodorhynchus leari (bonaparte, 1856) (aves: Psittacidae) em sua área de ocorrência no sertão da bahia, 2007.

[46] Nunes, J. R. S., Oki, Y., Carmignotto, A. P., and Tello, P. G. *Distribuição de Frequência de Habitats por Aves Aquáticas Piscívoras do Lago Camaleão, Ilha da Marchantaria, Amazonas.* Curso de Campo Ecológia da Floresta Amazônica, 2002.

[47] of Agriculture of Government of Austrália, D. *Best Practice Guidelines for Bird Scaring in Orchards Noise and Threatened Species.* Government of Australia, 2009.

[48] Ottoni, G. d. L., and Lages, Walter, F. Navegação de robôs móveis em ambientes desconhecidos utilizando sonares de ultra-som. *Revista Controle e Automoção 14*, 4 (2003), 402–411.

[49] Pinciroli, C. Swarm robotics. `http://iridia.ulb.ac.be/~cpinciroli/extra/h-414/`. Acessado em 19/01/2015.

[50] Pinciroli, C., Birattari, M., Tuci, E., Dorigo, M., del Rey Zapatero, M., Vinko, T., and Izzo, D. Self-organizing and scalable shape formation for a swarm of pico satellites. In *Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on* (2008), IEEE, pp. 57–61.

[51] Pinciroli, C., and Brambilla, M. Swarm intelligence course 2014. `http://iridia.ulb.ac.be/~cpinciroli/extra/h-414/pattern_formation.pdf`. Acessado em 16/01/2015.

[52] Pinciroli, C., Trianni, V., OGrady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G., Ducatelle, F., Stirling, T., Gutierrez, A., Gambardella, L. M., and Dorigo, M. Argos a modular, multi-engine simulator for heterogeneous swarm robotics. *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on 1* (September 2011), 5027–5034.

[53] Pinciroli, C., Trianni, V., OGrady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G., Ducatelle, F., Stirling, T., Gutierrez, A., Gambardella, L. M., and Dorigo, M. Argos a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence 6*, 4 (2012), 271–295.

[54] Raja, P., and Pugazhenthi, S. Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences 7*, 9 (2012), 1314–1320.

[55] Rao, V. V. Management strategies for birds and wild animals in organic crop production. In *The Second International Conference on Bio-Resource and Stress Management* (January 2015), Agricultural University Hyderabad, Ratikanta Maiti Foundation Kolkata, Puspa Publishing House, Kolkata, India, pp. 34–41.

[56] Reynolds, C. W. Flocks, herds, and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 25–34.

[57] Ribeiro, M. I., and Pedro, L. Mottion plannig mobile robotics course. *Instituto Superior Técnico* (2002).

[58] Russell, S., and Norvig, P. *Inteligência Artificial*, vol. 2. Elsevier, 2004.

[59] Sahin, E. Swarm robotics - from sources of inspiration to domains of application. Tech. rep., Middle East Technical University, 2005.

[60] Schroeder, J., and Lee, C. Managing common bird challenges. *NDSU-North Dakota State University* (April 2015), 1–12.

[61] Seamans, T. W., Martin, J. A., and Belant, J. L. *Wildlife in Airport Environments*. 2013, ch. Tactile and Auditory Repellents to Reduce Wildlife Hazards to Aircraft.

[62] Seeley, T. D., Camazine, S., and Sneyd, J. Collective decision-making in honey bees: How colonies choose among nectar sources. *Behavioral Ecology and Sociobiology 28*, 4 (1991), 277–290.

[63] Siciliano, B., and Khatib, O. *Handbook of Robotics*. Springer, 2008.

[64] Siegwart, Roland e Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[65] Silva, A. P., Filho, R. I. d. S., and Augusto, R. F. Swarm robotics: comportamento adaptativo aplicado ao problema de dispersão de pássaros em áreas agrícolas. *Memórias do VIII Workshop de Tecnologia Adaptativa - Escola Politécnica da USP. ISBN:978-85-86686-76-4* (Fevereiro 2014), 111–119.

[66] Stranieri, A., Trianni, V., Ferrante, E., Dorigo, M., and Pinciroli, C. Self-organized flocking with a heterogeneous mobile robot swarm. *Advances in Artificial Life ECAL 1*, 1 (2011), 789–796.

[67] Theraulaz, G., Gautrais, J., Camazine, S., and Deneubourg, J.-L. The formation of spatial patterns in social insects: from simple behaviours to complex structures. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 361*, 1807 (2003), 1263–1282.

[68] Tracey, John e Mary, B. *Managing Bird Damage to Fruit and Other Horticultural Crops*. Bureau of Rural Sciences, 2007.

[69] Trianni, V. *Evolutionary Swarm Robotics: Evolving Self-organising Behaviours in Groups of Autonomous Robots*, vol. 108. Springer, 2008.

[70] Trianni, V., and Campo, A. Fundamental collective behaviors in swarm robotics. In *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 1377–1394.

Apêndice A

Códigos Fontes do Algoritmo *Flocking*

Para o entendimento desse algoritmo ele possui as seguintes definições de implementações:

- Cada robô que pertence ao enxame possui um *led* aceso de cor vermelha que serve como dispositivo de identificação.
- O cálculo da distância dos robôs vizinhos leva em consideração apenas os robôs com *leds* acesos na cor vermelha e que não estejam longes mais do que a distância máxima estabelecida.
- Inicialmente, os robôs são dispostos em uma região em posições aleatórias muito próximas uma das outras. Assim, de imediato os robôs tendem a se repelirem, distanciando um dos outros. A medida que os robôs começam a ficar muito longe, eles tendem a serem atraídos novamente mantendo a coesão.
- Ao detectarem o alvo predeterminado os robôs se locomovem em grupo até cercá-lo. Durante todo o tempo de locomoção, cada robô verifica quais são os robôs ao seu redor (os seus vizinhos) e realiza os ajustes necessários para manter a coesão, entre eles, o cálculo da força do potencial de *Lennard-Jones*.

Todos os códigos fontes do simulador ARGoS encontram-se disponíveis no site https://github.com/ilpincy/argos3. A seguir serão apresentados os códigos fontes do Algoritmo *Flocking*.

Código A.1: Arquivo: *argos/controllers/footbot _flocking/flocking.cpp*

```
1
2  /****************************************************************************
3  * Arquivo: footbot_flocking.cpp
4  ****************************************************************************/
5  *
6  /* Include the controller definition */
7  #include "footbot_flocking.h"
8  /* Function definitions for XML parsing */
9  #include <argos3/core/utility/configuration/argos_configuration.h>
10
11 /****************************************/
12 /****************************************/
13
14 void CFootBotFlocking::SWheelTurningParams::Init(TConfigurationNode& t_node) {
15    try {
16       CDegrees cAngle;
17       GetNodeAttribute(t_node, "hard_turn_angle_threshold", cAngle);
18       HardTurnOnAngleThreshold = ToRadians(cAngle);
19       GetNodeAttribute(t_node, "soft_turn_angle_threshold", cAngle);
20       SoftTurnOnAngleThreshold = ToRadians(cAngle);
21       GetNodeAttribute(t_node, "no_turn_angle_threshold", cAngle);
22       NoTurnAngleThreshold = ToRadians(cAngle);
23       GetNodeAttribute(t_node, "max_speed", MaxSpeed);
24    }
25    catch(CARGoSException& ex) {
26       THROW_ARGOSEXCEPTION_NESTED("Error initializing controller wheel turning parameters.", ex);
27    }
```

```
28 }
29
30 /****************************************/
31 /****************************************/
32
33 void CFootBotFlocking::SFlockingInteractionParams::Init(TConfigurationNode& t_node) {
34    try {
35       GetNodeAttribute(t_node, "target_distance", TargetDistance);
36       GetNodeAttribute(t_node, "gain", Gain);
37       GetNodeAttribute(t_node, "exponent", Exponent);
38    }
39    catch(CARGoSException& ex) {
40       THROW_ARGOSEXCEPTION_NESTED("Error initializing controller flocking parameters.", ex);
41    }
42 }
43
44 /****************************************/
45 /****************************************/
46
47 /*
48  * This function is a generalization of the Lennard-Jones potential
49  */
50 Real CFootBotFlocking::SFlockingInteractionParams::GeneralizedLennardJones(Real f_distance) {
51    Real fNormDistExp = ::pow(TargetDistance / f_distance, Exponent);
52    return -Gain / f_distance * (fNormDistExp * fNormDistExp - fNormDistExp);
53 }
54
55 /****************************************/
56 /****************************************/
57
58 CFootBotFlocking::CFootBotFlocking() :
59    m_pcWheels(NULL),
60    m_pcLight(NULL),
61    m_pcLEDs(NULL),
62    m_pcCamera(NULL) {}
63
64 /****************************************/
65 /****************************************/
66
67 void CFootBotFlocking::Init(TConfigurationNode& t_node) {
68    /*
69     * Get sensor/actuator handles
70     *
71     * The passed string (ex. "differential_steering") corresponds to the XML tag of the
72     * device whose handle we want to have. For a list of allowed values, type at the
73     * command prompt:
74     *
75     * $ argos3 -q actuators
76     *
77     * to have a list of all the possible actuators, or
78     *
79     * $ argos3 -q sensors
80     *
81     * to have a list of all the possible sensors.
82     *
83     * NOTE: ARGoS creates and initializes actuators and sensors internally, on the basis of
84     *       the lists provided the configuration file at the
85     *       <controllers><footbot_diffusion><actuators> and
86     *       <controllers><footbot_diffusion><sensors> sections. If you forgot to
87     *       list a device in the XML and then you request it here, an error occurs.
88     */
89    m_pcWheels = GetActuator<CCI_DifferentialSteeringActuator         >("differential_steering");
90    m_pcLight  = GetSensor  <CCI_FootBotLightSensor                   >("footbot_light");
91    m_pcLEDs   = GetActuator<CCI_LEDsActuator                         >("leds");
92    m_pcCamera = GetSensor  <CCI_ColoredBlobOmnidirectionalCameraSensor>("colored_blob_omnidirectional_camera
          ");
93    /*
94     * Parse the config file
95     */
96    try {
97       /* Wheel turning */
98       m_sWheelTurningParams.Init(GetNode(t_node, "wheel_turning"));
99       /* Flocking-related */
100      m_sFlockingParams.Init(GetNode(t_node, "flocking"));
101   }
102   catch(CARGoSException& ex) {
103      THROW_ARGOSEXCEPTION_NESTED("Error parsing the controller parameters.", ex);
104   }
105
106   /*
107    * Other init stuff
108    */
109   /* Enable camera filtering */
110   m_pcCamera->Enable();
111   /* Set beacon color to all red to be visible for other robots */
112   m_pcLEDs->SetSingleColor(12, CColor::RED);
113 }
114
115 /****************************************/
```

```
116 /****************************************/
117
118 void CFootBotFlocking::ControlStep() {
119    SetWheelSpeedsFromVector(VectorToLight() + FlockingVector());
120 }
121
122 /****************************************/
123 /****************************************/
124
125 CVector2 CFootBotFlocking::VectorToLight() {
126    /* Get light readings */
127    const CCI_FootBotLightSensor::TReadings& tReadings = m_pcLight->GetReadings();
128    /* Calculate a normalized vector that points to the closest light */
129    CVector2 cAccum;
130    for(size_t i = 0; i < tReadings.size(); ++i) {
131       cAccum += CVector2(tReadings[i].Value, tReadings[i].Angle);
132    }
133    if(cAccum.Length() > 0.0f) {
134       /* Make the vector long as 1/4 of the max speed */
135       cAccum.Normalize();
136       cAccum *= 0.25f * m_sWheelTurningParams.MaxSpeed;
137    }
138    return cAccum;
139 }
140
141 /****************************************/
142 /****************************************/
143
144 CVector2 CFootBotFlocking::FlockingVector() {
145    /* Get the camera readings */
146    const CCI_ColoredBlobOmnidirectionalCameraSensor::SReadings& sReadings = m_pcCamera->GetReadings();
147    /* Go through the camera readings to calculate the flocking interaction vector */
148    if(! sReadings.BlobList.empty()) {
149       CVector2 cAccum;
150       Real fLJ;
151       size_t unBlobsSeen = 0;
152
153       for(size_t i = 0; i < sReadings.BlobList.size(); ++i) {
154
155          /*
156           * The camera perceives the light as a yellow blob
157           * The robots have their red beacon on
158           * So, consider only red blobs
159           * In addition: consider only the closest neighbors, to avoid
160           * attraction to the farthest ones. Taking 180% of the target
161           * distance is a good rule of thumb.
162           */
163          if(sReadings.BlobList[i]->Color == CColor::RED &&
164             sReadings.BlobList[i]->Distance < m_sFlockingParams.TargetDistance * 1.80f) {
165             /*
166              * Take the blob distance and angle
167              * With the distance, calculate the Lennard-Jones interaction force
168              * Form a 2D vector with the interaction force and the angle
169              * Sum such vector to the accumulator
170              */
171             /* Calculate LJ */
172             fLJ = m_sFlockingParams.GeneralizedLennardJones(sReadings.BlobList[i]->Distance);
173             /* Sum to accumulator */
174             cAccum += CVector2(fLJ,
175                                sReadings.BlobList[i]->Angle);
176             /* Increment the blobs seen counter */
177             ++unBlobsSeen;
178          }
179       }
180       /* Divide the accumulator by the number of blobs seen */
181       cAccum /= unBlobsSeen;
182       /* Clamp the length of the vector to the max speed */
183       if(cAccum.Length() > m_sWheelTurningParams.MaxSpeed) {
184          cAccum.Normalize();
185          cAccum *= m_sWheelTurningParams.MaxSpeed;
186       }
187       return cAccum;
188    }
189    else {
190       return CVector2();
191    }
192 }
193
194 /****************************************/
195 /****************************************/
196
197 void CFootBotFlocking::SetWheelSpeedsFromVector(const CVector2& c_heading) {
198    /* Get the heading angle */
199    CRadians cHeadingAngle = c_heading.Angle().SignedNormalize();
200    /* Get the length of the heading vector */
201    Real fHeadingLength = c_heading.Length();
202    /* Clamp the speed so that it's not greater than MaxSpeed */
203    Real fBaseAngularWheelSpeed = Min<Real>(fHeadingLength, m_sWheelTurningParams.MaxSpeed);
204
```

```
205    /* Turning state switching conditions */
206    if(Abs(cHeadingAngle) <= m_sWheelTurningParams.NoTurnAngleThreshold) {
207       /* No Turn, heading angle very small */
208       m_sWheelTurningParams.TurningMechanism = SWheelTurningParams::NO_TURN;
209    }
210    else if(Abs(cHeadingAngle) > m_sWheelTurningParams.HardTurnOnAngleThreshold) {
211       /* Hard Turn, heading angle very large */
212       m_sWheelTurningParams.TurningMechanism = SWheelTurningParams::HARD_TURN;
213    }
214    else if(m_sWheelTurningParams.TurningMechanism == SWheelTurningParams::NO_TURN &&
215            Abs(cHeadingAngle) > m_sWheelTurningParams.SoftTurnOnAngleThreshold) {
216       /* Soft Turn, heading angle in between the two cases */
217       m_sWheelTurningParams.TurningMechanism = SWheelTurningParams::SOFT_TURN;
218    }
219
220    /* Wheel speeds based on current turning state */
221    Real fSpeed1, fSpeed2;
222    switch(m_sWheelTurningParams.TurningMechanism) {
223       case SWheelTurningParams::NO_TURN: {
224          /* Just go straight */
225          fSpeed1 = fBaseAngularWheelSpeed;
226          fSpeed2 = fBaseAngularWheelSpeed;
227          break;
228       }
229
230       case SWheelTurningParams::SOFT_TURN: {
231          /* Both wheels go straight, but one is faster than the other */
232          Real fSpeedFactor = (m_sWheelTurningParams.HardTurnOnAngleThreshold - Abs(cHeadingAngle)) /
                 m_sWheelTurningParams.HardTurnOnAngleThreshold;
233          fSpeed1 = fBaseAngularWheelSpeed - fBaseAngularWheelSpeed * (1.0 - fSpeedFactor);
234          fSpeed2 = fBaseAngularWheelSpeed + fBaseAngularWheelSpeed * (1.0 - fSpeedFactor);
235          break;
236       }
237
238       case SWheelTurningParams::HARD_TURN: {
239          /* Opposite wheel speeds */
240          fSpeed1 = -m_sWheelTurningParams.MaxSpeed;
241          fSpeed2 =  m_sWheelTurningParams.MaxSpeed;
242          break;
243       }
244    }
245
246    /* Apply the calculated speeds to the appropriate wheels */
247    Real fLeftWheelSpeed, fRightWheelSpeed;
248    if(cHeadingAngle > CRadians::ZERO) {
249       /* Turn Left */
250       fLeftWheelSpeed  = fSpeed1;
251       fRightWheelSpeed = fSpeed2;
252    }
253    else {
254       /* Turn Right */
255       fLeftWheelSpeed  = fSpeed2;
256       fRightWheelSpeed = fSpeed1;
257    }
258    /* Finally, set the wheel speeds */
259    m_pcWheels->SetLinearVelocity(fLeftWheelSpeed, fRightWheelSpeed);
260 }
261
262 /****************************************/
263 /****************************************/
264
265 /*
266  * This statement notifies ARGoS of the existence of the controller.
267  * It binds the class passed as first argument to the string passed as second argument.
268  * The string is then usable in the XML configuration file to refer to this controller.
269  * When ARGoS reads that string in the XML file, it knows which controller class to instantiate.
270  * See also the XML configuration files for an example of how this is used.
271  */
272 REGISTER_CONTROLLER(CFootBotFlocking, "footbot_flocking_controller")
```

Código A.2: Arquivo: *argos/controllers/footbot _flocking/flocking.h*

```
1  /*
2   * AUTHOR: Carlo Pinciroli <cpinciro@ulb.ac.be>
3   *
4   * An example flocking controller for the foot-bot.
5   *
6   * This controller lets a group of foot-bots flock in an hexagonal lattice towards
7   * a light source placed in the arena. To flock, it exploits a generalization of the
8   * well known Lennard-Jones potential. The parameters of the Lennard-Jones function
9   * were chosen through a simple trial-and-error procedure on its graph.
10  *
11  * This controller is meant to be used with the XML file:
12  *    experiments/flocking.argos
13  */
14
15 #ifndef FOOTBOT_FLOCKING_H
16 #define FOOTBOT_FLOCKING_H
```

```
17
18 /*
19  * Include some necessary headers.
20  */
21 /* Definition of the CCI_Controller class. */
22 #include <argos3/core/control_interface/ci_controller.h>
23 /* Definition of the differential steering actuator */
24 #include <argos3/plugins/robots/generic/control_interface/ci_differential_steering_actuator.h>
25 /* Definition of the LEDs actuator */
26 #include <argos3/plugins/robots/generic/control_interface/ci_leds_actuator.h>
27 /* Definition of the omnidirectional camera sensor */
28 #include <argos3/plugins/robots/generic/control_interface/ci_colored_blob_omnidirectional_camera_sensor.h>
29 /* Definition of the foot-bot light sensor */
30 #include <argos3/plugins/robots/foot-bot/control_interface/ci_footbot_light_sensor.h>
31 /* Vector2 definitions */
32 #include <argos3/core/utility/math/vector2.h>
33
34 /*
35  * All the ARGoS stuff in the 'argos' namespace.
36  * With this statement, you save typing argos:: every time.
37  */
38 using namespace argos;
39
40 /*
41  * A controller is simply an implementation of the CCI_Controller class.
42  */
43 class CFootBotFlocking : public CCI_Controller {
44
45 public:
46
47    /*
48     * The following variables are used as parameters for
49     * turning during navigation. You can set their value
50     * in the <parameters> section of the XML configuration
51     * file, under the
52     * <controllers><footbot_flocking_controller><parameters><wheel_turning>
53     * section.
54     */
55    struct SWheelTurningParams {
56       /*
57        * The turning mechanism.
58        * The robot can be in three different turning states.
59        */
60       enum ETurningMechanism
61       {
62          NO_TURN = 0, // go straight
63          SOFT_TURN,   // both wheels are turning forwards, but at different speeds
64          HARD_TURN    // wheels are turning with opposite speeds
65       } TurningMechanism;
66       /*
67        * Angular thresholds to change turning state.
68        */
69       CRadians HardTurnOnAngleThreshold;
70       CRadians SoftTurnOnAngleThreshold;
71       CRadians NoTurnAngleThreshold;
72       /* Maximum wheel speed */
73       Real MaxSpeed;
74
75       void Init(TConfigurationNode& t_tree);
76    };
77
78    /*
79     * The following variables are used as parameters for
80     * flocking interaction. You can set their value
81     * in the <parameters> section of the XML configuration
82     * file, under the
83     * <controllers><footbot_flocking_controller><parameters><flocking>
84     * section.
85     */
86    struct SFlockingInteractionParams {
87       /* Target robot-robot distance in cm */
88       Real TargetDistance;
89       /* Gain of the Lennard-Jones potential */
90       Real Gain;
91       /* Exponent of the Lennard-Jones potential */
92       Real Exponent;
93
94       void Init(TConfigurationNode& t_node);
95       Real GeneralizedLennardJones(Real f_distance);
96    };
97
98 public:
99
100    /* Class constructor. */
101    CFootBotFlocking();
102
103    /* Class destructor. */
104    virtual ~CFootBotFlocking() {}
105
```

```
106    /*
107     * This function initializes the controller.
108     * The 't_node' variable points to the <parameters> section in the XML file
109     * in the <controllers><footbot_flocking_controller> section.
110     */
111    virtual void Init(TConfigurationNode& t_node);
112
113    /*
114     * This function is called once every time step.
115     * The length of the time step is set in the XML file.
116     */
117    virtual void ControlStep();
118
119    /*
120     * This function resets the controller to its state right after the Init().
121     * It is called when you press the reset button in the GUI.
122     * In this example controller there is no need for resetting anything, so
123     * the function could have been omitted. It's here just for completeness.
124     */
125    virtual void Reset() {}
126
127    /*
128     * Called to cleanup what done by Init() when the experiment finishes.
129     * In this example controller there is no need for clean anything up, so
130     * the function could have been omitted. It's here just for completeness.
131     */
132    virtual void Destroy() {}
133
134 protected:
135
136    /*
137     * Calculates the vector to the closest light.
138     */
139    virtual CVector2 VectorToLight();
140
141    /*
142     * Calculates the flocking interaction vector.
143     */
144    virtual CVector2 FlockingVector();
145
146    /*
147     * Gets a direction vector as input and transforms it into wheel actuation.
148     */
149    void SetWheelSpeedsFromVector(const CVector2& c_heading);
150
151 private:
152
153    /* Pointer to the differential steering actuator */
154    CCI_DifferentialSteeringActuator* m_pcWheels;
155    /* Pointer to the foot-bot light sensor */
156    CCI_FootBotLightSensor* m_pcLight;
157    /* Pointer to the LEDs actuator */
158    CCI_LEDsActuator* m_pcLEDs;
159    /* Pointer to the omnidirectional camera sensor */
160    CCI_ColoredBlobOmnidirectionalCameraSensor* m_pcCamera;
161
162    /* The turning parameters. */
163    SWheelTurningParams m_sWheelTurningParams;
164    /* The flocking interaction parameters. */
165    SFlockingInteractionParams m_sFlockingParams;
166 };
167
168 #endif
```

Código A.3: Arquivo: *argos/controllers/footbot _flocking/CMakeLists.txt*

```
1 add_library(footbot_flocking MODULE footbot_flocking.h footbot_flocking.cpp)
2 target_link_libraries(footbot_flocking
3   argos3core_simulator
4   argos3plugin_simulator_actuators
5   argos3plugin_simulator_footbot
6   argos3plugin_simulator_genericrobot
7   argos3plugin_simulator_sensors)
```

Apêndice B

Códigos Fontes das Simulações Realizadas no Experimento Inicial

Código B.1: Arquivo: *argos/loop_functions/my_loop_functions.h*

```
1
2  /*****************************************************************************
3  * Arquivo: My_loop_functions.h
4  *****************************************************************************/
5  #ifndef MY_LOOP_FUNCTIONS_H
6  #define MY_LOOP_FUNCTIONS_H
7
8  #include <argos3/core/simulator/loop_functions.h>
9  #include <argos3/plugins/simulator/entities/light_entity.h>
10
11 using namespace argos;
12
13 class MyLoopFunctions : public CLoopFunctions {
14
15 public:
16
17    /* This initializes the pointer to light to NULL */
18    MyLoopFunctions();
19
20    /* It is better to add a destructor explicitly */
21    virtual ~MyLoopFunctions() {}
22
23    /* This initializes the pointer to the light entity */
24    virtual void Init(TConfigurationNode& t_node);
25
26    /* This is executed at the end of each time step */
27    virtual void PostStep();
28
29 private:
30
31    /* Returns the position corresponding to the current time step
32     * If the current time step is < un_start_step, it returns c_start_pos;
33     * If the current time step is > un_end_step, it returns c_end_pos;
34     * Otherwise it interpolates the position linearly.
35     */
36    CVector3 LinearInterpolation(const CVector3& c_start_pos,
37                                 const CVector3& c_end_pos,
38                                 UInt32 un_start_step,
39                                 UInt32 un_end_step);
40 private:
41
42    /* The pointer to the light entity */
43    CLightEntity* pcLight;
44 };
45
46 #endif
```

Código B.2: Arquivo: *argos/loop_functions/my_loop_functions.cpp*

```
1
2  /*****************************************************************************
3  * Arquivo: My_loop_functions.cpp
4  *****************************************************************************/
5  #include "my_loop_functions.h"
6
7  MyLoopFunctions::MyLoopFunctions() :
8     pcLight(NULL) {
```

```
}

/****************************************/

void MyLoopFunctions::Init(TConfigurationNode& t_node) {

   // Set the pointer to the light
   // Since the light has been added in the XML file,
   // we can get a pointer to it once and use it for the
   // rest of the experiment
   pcLight = &dynamic_cast<CLightEntity&>(GetSpace().GetEntity("light"));

}

/****************************************/

void MyLoopFunctions::PostStep() {
   // Move o p ssaro (fonte de luz) para uma determinada posicao
   // Move o p ssaro por 10.000 time steps, da p o s i  o inicial (5,1,0.4) para (-1,3,0.4)

//pcLight->SetPosition(5, 1, 0.4);
 /* Check whether a robot is on a food item */

  pcLight->SetPosition(
  LinearInterpolation(
         CVector3(5, 1, 0.4),
         CVector3(-1, 3, 0.4),
         0,
         10000));
}

/****************************************/

CVector3 MyLoopFunctions::LinearInterpolation(const CVector3& c_start_pos,
                                              const CVector3& c_end_pos,
                                              UInt32 un_start_step,
                                              UInt32 un_end_step) {
   if(GetSpace().GetSimulationClock() <= un_start_step) return c_start_pos;
   if(GetSpace().GetSimulationClock() >= un_end_step) return c_end_pos;
   return c_start_pos +                                          // starting position
      (c_end_pos - c_start_pos) / (un_end_step - un_start_step) *  // interpolation factor
      (GetSpace().GetSimulationClock() - un_start_step);           // current step in the range
}

/****************************************/

REGISTER_LOOP_FUNCTIONS(MyLoopFunctions, "my_loop_functions");
```

Código B.3: Arquivo: *argos/experiments/flocking.argos*

```

/ ****************************************************************************
* Arquivo: Flocking.argos
**************************************************************************** /
<?xml version="1.0" ?>
<argos-configuration>

  <!-- ************************* -->
  <!-- * General configuration * -->
  <!-- ************************* -->
  <framework>
    <system threads="1" />
    <experiment length="0"
                ticks_per_second="10"
                random_seed="124" />
  </framework>

  <!-- *************** -->
  <!-- * Controllers * -->
  <!-- *************** -->
  <controllers>

    <footbot_flocking_controller id="ffc"
                                 library="build/controllers/footbot_flocking/libfootbot_flocking.so">
      <actuators>
        <differential_steering implementation="default" />
        <leds                  implementation="default" medium="leds" />
      </actuators>
      <sensors>
        <footbot_light                      implementation="rot_z_only" show_rays="false" />
        <colored_blob_omnidirectional_camera implementation="rot_z_only" medium="leds" show_rays="true" />
      </sensors>
      <params>
        <wheel_turning hard_turn_angle_threshold="90"
                       soft_turn_angle_threshold="70"
                       no_turn_angle_threshold="10"
                       max_speed="10" />
        <flocking target_distance="75"
```

```
                gain="1000"
                exponent="2" />
    </params>
  </footbot_flocking_controller>

</controllers>

<!-- ****************** -->
<!-- * Loop functions * -->
<!-- ****************** -->

<loop_functions library="build/loop_functions/my_loop_functions/libmy_loop_functions.so" label="
     my_loop_functions" />

<!-- ********************** -->
<!-- * Arena configuration * -->
<!-- ********************** -->
<arena size="20,20,1" center="0,0,0.5">

  <light id="light"
         position="5,1,0.4"
         orientation="0,0,0"
         color="yellow"
         intensity="8.0"
         medium="leds" />

  <!-- This is the way to distribute the cylinders in a grid -->
  <!-- (see also the synchronization example) -->
  <!-- This syntax sets the ids of the cylinders to tree_0 tree_1 ... tree_34 -->
  <distribute>
    <position method="grid"
              center="4,2,0"
              distances="1,1,0"
              layout="7,5,1" />
    <orientation method="constant" values="0,0,0" />
    <entity quantity="35" max_trials="1">
      <cylinder id="tree" height="0.2" radius="0.1" mass="0.1" movable="false" />
    </entity>
  </distribute>

  <!-- This is the way to distribute the cylinders randomly -->
  <!-- The attribute 'base_num' below sets the ids of the cylinders to tree_35 tree_36 tree_37 -->
  <distribute>
    <position method="uniform" min="-2,-2,0" max="2,2,0" />
    <orientation method="constant" values="0,0,0" />
    <entity base_num="35" quantity="3" max_trials="100">
      <cylinder id="tree" height="0.5" radius="0.1" movable="false" mass="0.1" color="red"/>
    </entity>
  </distribute>

  <!-- ********************** -->
  <!--
      You can distribute entities randomly. Here, we distribute
      the entities in this way:
      - the position is uniformly distributed on the ground, in the
      square whose corners are (4,4) and (5,5)
      - the orientations are non-zero only when rotating around Z and
      chosen from a gaussian distribution, whose mean is zero degrees
      and standard deviation is 360 degrees.
  -->
  <distribute>
    <position method="uniform" min="4,4,0" max="5,5,0" />
    <orientation method="gaussian" mean="0,0,0" std_dev="360,0,0" />
    <entity quantity="14" max_trials="100">
      <foot-bot id="fb" omnidirectional_camera_aperture="80">
        <controller config="ffc" />
      </foot-bot>
    </entity>
  </distribute>

</arena>

<!-- ******************* -->
<!-- * Physics engines * -->
<!-- ******************* -->
<physics_engines>
  <dynamics2d id="dyn2d" />
</physics_engines>

<!-- ********* -->
<!-- * Media * -->
<!-- ********* -->
<media>
  <led id="leds" />
</media>

<!-- ****************** -->
<!-- * Visualization * -->
<!-- ****************** -->
```

```
127   <visualization>
128     <qt-opengl>
129       <camera>
130         <placement idx="0"
131                    position="5.33539,5.60355,1.06253"
132                    look_at="4.70428,4.92503,0.686605"
133                    lens_focal_length="20" />
134       </camera>
135     </qt-opengl>
136   </visualization>
137 </argos-configuration>
```

Apêndice C

Códigos Fontes Implementados para Execução dos Experimentos

Os códigos fontes a seguir se referem aos códigos fontes implementados para execução dos experimentos simulando as trajetórias retilínea, semicircular e circular do pássaro na lavoura.

Código C.1: Arquivo: *My_loop_functions.cpp*

```
/*****************************************************************************
* Arquivo: My_loop_functions.cpp
*****************************************************************************/

#include "my_loop_functions.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <cmath>
#include <math.h>

/****************************************/
/****************************************/
//Real  x1=0, y1=0, x2=0, y2=0;

MyLoopFunctions::MyLoopFunctions() :
   pcLight(NULL) {
}

/****************************************/
/****************************************/

void MyLoopFunctions::Init(TConfigurationNode& t_node) {

   // Set the pointer to the light
   // Since the light has been added in the XML file,
   // we can get a pointer to it once and use it for the
   // rest of the experiment
   pcLight = &dynamic_cast<CLightEntity&>(GetSpace().GetEntity("light"));

  //x1=-5.25, y1=9.19, x2=0, y2=0;
    x1=4, y1=0;
    rad=0, tempo=0;
/****************************************/
 /**Usuario escolhe o tipo de trajetoria quer simular*******/
    equacao=2; // trajetoria circular
   //equacao=3; //traj. semicircular
  // equacao=4; //traj. senoidal
  //equacao=5; //traj. retilinea
/****************************************/
    imprime=1;
    step=0;//passo da simulacao

    if (equacao == 4) //traj. senoidal
    {
      x1=-4;
      y1=0;
    }

  if (equacao == 5) //traj. retilinea
    {
      x1=5;
      y1=-5;

```

```
    }

}

/****************************************/
/****************************************/

void MyLoopFunctions::PostStep()
{
UInt32 temp=0;
Real start_step=3;
Real end_step;//em quantos passos o passaro se movera?
Real duracao_experimento;
step = step + 1;
Real raio;
Real mov_circular_continuo; //determina se o passaro ficara voando continuamente durante a simulacao.
Real interpolacao;//fator que incrementa a variavel para passaro se mover de um ponto para o outro.
                  //ela determina quanto o objeto varia entre um ponto P1(x1, y1) para o ponto P2(x2, y2).

//---------------------------------------
//---dados de entrada para simulacao-----
//---------------------------------------
//para trajetorias circulares ou semicircunferencia

//---Intervalo de variacao da interpolacao
//---------------------------------------------
//interpolacao=0.0010000; //1
//interpolacao=0.0001000; //1 passaro voa devagar
//interpolacao=0.0001310; //2
//interpolacao=0.0001621; //3
//interpolacao=0.0001931; //4
//interpolacao=0.0002241; //5
//interpolacao=0.0002552; //6
//interpolacao=0.0002862; //7
//interpolacao=0.0003172; //8
//interpolacao=0.0003483; //9 - pulei este
//interpolacao=0.0003793; //10
//interpolacao=0.0004103; //11
//interpolacao=0.0004414; //12
//interpolacao=0.0004724; //13
//interpolacao=0.0005034; //14
interpolacao=0.0005345; //15
//interpolacao=0.0005655; //16
//interpolacao=0.0005966; //17
//interpolacao=0.0006276; //18
//interpolacao=0.0006586; //19
//interpolacao=0.0006897; //20
//interpolacao=0.0007207; //21
//interpolacao=0.0007517; //22
//interpolacao=0.0007828; //23
//interpolacao=0.0008138; //24
//interpolacao=0.0008448; //25
//interpolacao=0.0008759; //26
//interpolacao=0.0009069; //27
//interpolacao=0.0009379; //28
//interpolacao=0.0009690; //29
//interpolacao=0.0010000; //30 passaro voo rapido
//-----------------------------

mov_circular_continuo=0;
raio=6;
//---------------------------------------

 if (equacao == 2) // trajetoria circular
 {

 if (mov_circular_continuo) //o radiano varia entre 0 a 2pi para dar a volta completa
   if (rad > 6.28)
     rad = 0.0;

  x1 = raio * cos(rad);
  y1 = raio * sin(rad);
  rad = rad + interpolacao;

  if (!mov_circular_continuo)
  {
     if (rad <= 6.28)
        pcLight->SetPosition(CVector3(x1, y1, 0.6));
     else if (imprime)
     {
        imprime=0;
        time (&tempo_final);
        printf ("\n Fim do voo e Qtde de passos(trajetoria) %s %d", ctime (&tempo_final), step);
        printf ("\n Qtde de passos(trajetoria): %d", step);
     }
  }
  else
   pcLight->SetPosition(CVector3(x1, y1, 0.6));
}
```

```
144
145
146  if (equacao == 3) // trajetoria semi circular
147  {
148   if (mov_circular_continuo) //o radiano varia entre 0 a 1pi para dar meia volta
149    if (rad > 3.14)
150        rad = 0.0;            //se ja deu meia volta, retorna ao ponto inicial
151
152   x1 = raio * cos(rad);
153   y1 = raio * sin(rad);
154   rad = rad + interpolacao;
155
156   if (!mov_circular_continuo)
157   {
158      if (rad <= 3.14)
159         pcLight->SetPosition(CVector3(x1, y1, 0.6));
160      else if (imprime)
161      {
162         imprime=0;
163         time (&tempo_final);
164
165         printf ("\n Fim do voo e Qtde de passos(trajetoria) %s %d", ctime (&tempo_final), step);
166         printf ("\n Qtde de passos(trajetoria): %d", step);
167      }
168   }
169   else
170    pcLight->SetPosition(CVector3(x1, y1, 0.6));
171  }
172
173  if (equacao == 4) // trajetoria senoidal
174  {
175    Real alfa=1;
176    Real freq=30;
177    Real k=10;
178    Real amplitude =3;
179
180   y1 = amplitude * cos ((k * x1) + (freq * tempo) + alfa);
181   x1 = x1 + 0.0001;
182   tempo=tempo+0.0001;
183
184   if (x1 > 10)
185    x1=0;
186
187   y1=-2* sin(x1) +1;
188   pcLight->SetPosition(CVector3(x1, y1, 0.6));
189
190 }
191
192  if (equacao == 5) // trajetoria retilinea
193  {
194    //Ponto Inicial (x1,y1), coordenadas x1=5 e y1=-5
195    //Ponto Final (x1,y1)-pouso, coordenadas x1=-5 e y1=5
196
197  if (y1 > 5)
198    {
199        if (imprime)
200        {
201         imprime=0;
202         time (&tempo_final);
203
204         printf ("\n Fim do voo e Qtde de passos(trajetoria) %s %d", ctime (&tempo_final), step);
205         printf ("\n Qtde de passos(trajetoria): %d", step);
206        }
207      }
208   else
209   {
210     pcLight->SetPosition(CVector3(x1, y1, 0.6));
211     x1=x1-interpolacao;
212     y1=y1+interpolacao;
213  }
214  }
215
216 if(GetSpace().GetSimulationClock() == 1)
217 {
218   printf ("%s \n", "-------------------------");
219   time (&tempo_inicial);
220   printf ("\n Inicio do voo: %s", ctime (&tempo_inicial));
221 }
222
223 if(GetSpace().GetSimulationClock() == end_step)
224   time (&tempo_final);
225
226 if(GetSpace().GetSimulationClock() == duracao_experimento)
227   time (&tempo_final);
228
229 }
230 /****************************************/
231 CVector3 MyLoopFunctions::LinearInterpolation(const CVector3& c_start_pos,
232                                               const CVector3& c_end_pos,
```

```
                                          Real un_start_step,
                                          Real un_end_step)
{
   if(GetSpace().GetSimulationClock() <= un_start_step) return c_start_pos;
   if(GetSpace().GetSimulationClock() >= un_end_step) return c_end_pos;
   return c_start_pos +                                          // starting position
      (c_end_pos - c_start_pos) / (un_end_step - un_start_step) *  // interpolation factor
      (GetSpace().GetSimulationClock() - un_start_step);           // current step in the range
}

/****************************************/
REGISTER_LOOP_FUNCTIONS(MyLoopFunctions, "my_loop_functions");
```

Código C.2: Arquivo: *Flocking.cpp*

```
/*****************************************************************************
* Arquivo: Flocking.cpp
*****************************************************************************/

<?xml version="1.0" ?>
<argos-configuration>

  <!-- ************************* -->
  <!-- * General configuration * -->
  <!-- ************************* -->
  <framework>
    <system threads="0" />
    <experiment length="0"
                ticks_per_second="10"
                random_seed="124" />
     <profiling file="log_circ_30_robos_26.txt"
               format="human_readable"
     truncate_file="true" />
  </framework>
   <!-- format="human_readable" -->

  <!-- *************** -->
  <!-- * Controllers * -->
  <!-- *************** -->
  <controllers>

    <footbot_flocking_controller id="ffc"
                                 library="build/controllers/footbot_flocking/libfootbot_flocking.so">
      <actuators>
        <differential_steering implementation="default" />
        <leds                  implementation="default" medium="leds" />
      </actuators>
      <sensors>
        <footbot_light                      implementation="rot_z_only" show_rays="false" />
        <colored_blob_omnidirectional_camera implementation="rot_z_only" medium="leds" show_rays="true" />
      </sensors>
      <params>
        <wheel_turning hard_turn_angle_threshold="90"
                       soft_turn_angle_threshold="70"
                       no_turn_angle_threshold="10"
                       max_speed="50" />
        <flocking target_distance="75"
                  gain="10000"
                  exponent="2" />
      </params>
    </footbot_flocking_controller>

  </controllers>

  <!-- ****************** -->
  <!-- * Loop functions * -->
  <!-- ****************** -->

  <loop_functions library="build/loop_functions/my_loop_functions/libmy_loop_functions.so" label="
       my_loop_functions" />

  <!-- ********************** -->
  <!-- * Arena configuration * -->
  <!-- ********************** -->
  <!-- Posicoes iniciais do passaro -->
  <!-- position="6.5,-6.5,0.5" trajetorias circular e semicircular-->
  <!-- position="1,1,0.5" trajetoria retilinea-->
  <arena size="20,20,1" center="0,0,0.5">

    <light id="light"
           position="1,1,0.5"
           orientation="0,0,0"
           color="yellow"
           intensity="30.0"
           medium="leds" />

    <!-- This is the way to distribute the cylinders in a grid -->
    <!-- (see also the synchronization example) -->
```

```
    <!-- This syntax sets the ids of the cylinders to tree_0 tree_1 ... tree_34 -->
    <distribute>
      <position method="grid"
                center="0,0,0"
                distances="1,1,1"
                layout="10,1,1" />

      <orientation method="constant" values="0,0,0" />
      <entity quantity="1" max_trials="1">
        <cylinder id="tree" height="0.2" radius="0.01" mass="0.1" movable="false" />
      </entity>
    </distribute>

    <!-- This is the way to distribute the cylinders randomly -->
    <!-- The attribute 'base_num' below sets the ids of the cylinders to tree_35 tree_36 tree_37 -->
   <!-- <position method="uniform" min="-8,-8,0" max="-6,-6,0" />-->
    <distribute>
      <position method="uniform" min="-2,-2,0" max="2,2,0" />
      <orientation method="constant" values="0,0,0" />
      <entity base_num="5" quantity="0" max_trials="100">
        <cylinder id="tree" height="0.2" radius="0.1" movable="false" mass="0.1" color="red"/>
      </entity>
    </distribute>

    <!-- ********************** -->

    <!--
        You can distribute entities randomly. Here, we distribute
        the entities in this way:
        - the position is uniformly distributed on the ground, in the
        square whose corners are (4,4) and (5,5)
        - the orientations are non-zero only when rotating around Z and
        chosen from a gaussian distribution, whose mean is zero degrees
        and standard deviation is 360 degrees.
    -->
  <!--
   <position method="uniform" min="0,-8,0" max="2,-6,0" />
   posicao inicial dos robos nas trajetorias circular e semicircular
    -->
    <!--
   <position method="uniform"  min="-8,-8,0" max="-6,-6,0" />
    posicao inicial dos robos na trajetoria retilinea
    -->
      <distribute>
       <position method="uniform"  min="-8,-8,0" max="-6,-6,0" />
      <orientation method="gaussian" mean="0,0,0" std_dev="360,0,0" />
      <entity quantity="30" max_trials="100">
        <foot-bot id="fb29" omnidirectional_camera_aperture="80">
          <controller config="ffc" />
        </foot-bot>
      </entity>
    </distribute>
   </arena>

  <!-- ****************** -->
  <!-- * Physics engines * -->
  <!-- ****************** -->
  <physics_engines>
    <dynamics2d id="dyn2d" />
  </physics_engines>

  <!-- ********* -->
  <!-- * Media * -->
  <!-- ********* -->
  <media>
    <led id="leds" />
  </media>

  <!-- ****************** -->
  <!-- * Visualization * -->
  <!-- ****************** -->
  <visualization>
    <qt-opengl>
      <camera>
        <placement idx="0"
                   position="5.33539,5.60355,1.06253"
                   look_at="4.70428,4.92503,0.686605"
                   lens_focal_length="03" />
        <placement idx="1"
                   position="0.1,0.1,2.5"
                   look_at="0,0,0"
                   lens_focal_length="03" />
           <placement idx="2"
                   position="-5,5,2.5"
                   look_at="0,0,0"
                   lens_focal_length="03" />
        </camera>
    </qt-opengl>
  </visualization>
```

```

</argos-configuration>
```

Código C.3: Arquivo: *My_loop_functions.h*

```
/*****************************************************************************
* Arquivo: My_loop_functions.h
*****************************************************************************/

#ifndef MY_LOOP_FUNCTIONS_H
#define MY_LOOP_FUNCTIONS_H

#include <argos3/core/simulator/loop_functions.h>
#include <argos3/plugins/simulator/entities/light_entity.h>

using namespace argos;

class MyLoopFunctions : public CLoopFunctions {

public:

   /* This initializes the pointer to light to NULL */
   MyLoopFunctions();
   /* It is better to add a destructor explicitly */
   virtual ~MyLoopFunctions() {}
   /* This initializes the pointer to the light entity */
   virtual void Init(TConfigurationNode& t_node);
   /* This is executed at the end of each time step */
   virtual void PostStep();

private:

   /* Returns the position corresponding to the current time step
    * If the current time step is < un_start_step, it returns c_start_pos;
    * If the current time step is > un_end_step, it returns c_end_pos;
    * Otherwise it interpolates the position linearly.
    */
   CVector3 LinearInterpolation(const CVector3& c_start_pos,
                                const CVector3& c_end_pos,
                                Real un_start_step,
                                Real un_end_step);
                               // UInt32 un_start_step,
                              //  UInt32 un_end_step);

private:

   /* The pointer to the light entity */
   CLightEntity* pcLight;

      std::string MSG_Alex;
  /*  std::ofstream m_cOutput;*/

/*alex que criou */
   Real  x1, y1, x2, y2, rad, tempo;
   UInt32 equacao, imprime;
   UInt32 step;
   time_t tempo_inicial;
   time_t tempo_final;
};

#endif
```