ELAINE ALVES DA ROCHA

LOCALIZAÇÃO DE OBJETOS UTILIZANDO LEITURA DE INTENSIDADE DE SINAL WIRELESS E ENXAME DE ROBÔS.

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luiz Carlos Pessoa Albini
Coorientador: Prof. Dr. Eduardo Todt

CURITIBA

2015

R672l Rocha, Elaine Alves da
Localização de objetos utilizando leitura de intensidade de sinal wireless e enxame de robôs/ Elaine Alves da Rocha. – Curitiba, 2015.
99 f. : il. color. ; 30 cm.

TeseDissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática, 2015.

Orientador: Luiz Carlos Pessoa Albini – Coorientador: Eduardo Todt.
Bibliografia: p. 66-72.

1. Robôs - Sistemas de controle. 2. Visão de robô. 3. Detecção de sinais. I. Universidade Federal do Paraná. II.Albini, Luiz Carlos Pessoa. III. Todt, Eduardo. IV. Título.

CDD: 629.8932

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, da aluna Elaine Alves da Rocha, avaliamos o trabalho intitulado, "Localização de objetos utilizando leitura de intensidade de sinal wireless e enxame de robôs", cuja defesa foi realizada no dia 14 de agosto de 2015, às 09:00 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela:
(X)**aprovação** da candidata. ( )**reprovação** da candidata.

Curitiba, 14 de agosto de 2015.

Prof. Dr. Luiz Carlos Pessoa Albini
PPGInf - Orientador

Prof. Dr. Eduardo Todt
PPGInf – Coorientador

Prof. Dr. Alessandro Brawerman
SEPT/UFPR – Membro Externo

Prof. Dr. Andrey Ricardo Pimentel
PPGInf – Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

À minha família por sempre me apoiar nas minhas escolhas, se interessar e principalmente me motivar sem se importar com a distância, amo muito vocês. Emily, Melissa e Eloise, vocês são as jóias mais valiosas da titia. A minha motivação está nos olhinhos brilhantes das minhas princesinhas!

Ao meu esposo Ivan Luiz Pedroso Pires, por sempre acreditar e me incentivar nos estudos, no trabalho e na vida. Sem você eu não teria chegado aqui, amo você!

Aos amigos e colegas que ajudaram a deixar os dias mais alegres, proporcionando momentos divertidos mesmo com as dificuldades normais proporcionadas pelos estudos.

Ao IFMT por me dar a oportunidade de estudar e aprimorar o meu conhecimento com afastamento, e principalmente ao pessoal do IFMT Campus Juína pela compreensão e motivação.

A todas as pessoas da UFPR - professores, técnicos administrativos da pós-graduação e funcionários em geral. Aos professores por sempre estarem disponíveis para sanar as dúvidas e em especial ao meu coorientador, professor Todt, pelas indicações e direcionamentos da pesquisa. Aos técnicos pela agilidade nos documentos e acesso as informações. E aos funcionários por proporcionarem aos alunos um ambiente agradável para a realização de pesquisas. Sem vocês nós não estaríamos aqui.

Ao professor Albini, que me orientou na realização dessa pesquisa e mesmo com tantos imprevistos, nunca me abandonou. Professor, o sucesso desse trabalho é resultado da sua paciência, crença, compreensão e sabedoria que sempre acompanharam os meus passos e tropeços na pesquisa científica. Te agradeço sinceramente por ter chegado aqui e sei que é apenas o início.

Muito obrigada !

"Tantos laços, tantas amarras
Os controles, pretensões
Nada adianta se o vento não soprar
Esse vento sob minhas asas
Eu não mando mais em nada
Sei que é alto, mas eu vou pular".

Pitty & Martin

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

.

ARIA - *Advanced Interface for Applications* - Interface Avançada para Aplicações

APF - *Análises de Pontos de Função*

BSSID - *Basic Service Set Identification* - Serviço Básico de Identificação

CPU - *Central Processing Unit* - Unidade Central de Processamento

dBm - *Decibel Milliwatt*

ESSID - *Extended Service Set Identifier* - Serviço Extendido de Identificação

FFD - *Full Function Device* - Dispositivos com Funções Completas

GHz - *Giga Hertz*

GPS - *Global Position System* - Sistema de Posicionamento Global

GSM - *Global System for Mobile Communications* - Sistema Global para Comunicações Móveis

HSDPA - *High-Speed Downlink Packet Access* - Acesso de pacotes de descarga de alta velocidade

IP - *Internet Protocol* - Protocolo de Internet

LAN - *Local Area Network* - Rede de Área Local

LTE - *Long Term Evolution* - Evolução a longo prazo

Mbps - *Megabit por segundo*

MHz - *Mega Hertz*

MRS - *Microsoft Robotics Studio* - Estúdio Robótico Microsoft

RFD - *Reduced Function Device* - Dispositivos com Funções Reduzidas

RSSI - *Received Signal Strength Indicator* - Indicação da Intensidade do Sinal Recebido

ROS - *Robot Operating System* - Sistema Operativo para Robôs

SLAM - *Simultaneous Localization and Mapping* - Localização e Mapeamento Simultâneos

VRM - *Virtual Robot Modeller* - Modelador Virtual para robôs

VRS - *Virtual Robot Simulator* - Simulador Virtual para Robôs

VRT - *Virtual Robot Translator* - Tradutor Virtual para Robôs

WCDMA - *Wideband Code Division Multiple Access* - Acesso Múltiplo por divisão de código em Banda Larga

WiFi - *Wireless Fidelity* - Internet sem Fio Confiável

RESUMO

A aplicação do enxame robótico aliada à comunicação de redes sem fio formam uma excelente ferramenta para possíveis soluções de problemas diversos, tal como identificação e busca de pessoas perdidas equipadas com aparelhos que possuam emissores de sinais. Este trabalho propõe a busca e localização de objetos utilizando sensoriamento de intensidade de sinais e um enxame robótico. São utilizados três métodos de busca: por tentativa e erro, busca sem comunicação e com comunicação. No primeiro método os robôs saem em busca do sinal do objeto perdido por meio de tentativas aleatórias; no segundo método é considerado um dispositivo que obtém a posição do objeto quando detectado; e o último método também obtém a posição do objeto quando detectado e utiliza a comunicação entre os robôs do enxame para a cooperação. Além disso, são considerados ambientes com diferentes áreas bem como diferentes números de robôs que compõem o enxame. São apresentados vários testes para os três métodos e várias combinações de tamanho de áreas por quantidade de robôs. Com os resultados pode-se perceber que em ambientes acima de 500 metros quadrados, independente da quantidade de robôs, utilizar a comunicação entre os robôs do enxame robótico é mais eficiente do que os outros métodos de busca, e essa eficiência pode ser vital no cenário de busca e salvamento de pessoas.

ABSTRACT

The application of robotic swarm combined with wireless communication networks form an excellent tool for possible solutions to various problems, such as identification and search for lost people equipped with devices that have signal transmitter. In this way, this paper proposes the search and location of objects using RSSI and a robotic swarm. Three search methods are used: by trial and error, search without communication and with communication. In the first method the robots come out in search of the lost object signal through random tries; the second method uses a device to obtain the position of the object when detected; and the latter method also gets the position of the object when detected and uses the communication between the robots of the swarm, for cooperation. Beyond that, are considered environments with different areas and different numbers of robots that compose the swarm. Several tests were performed for the three methods and combinations of various areas in relation of the number of robots. The results show that in environments with over 500 square meters, regardless of the number of robots, the search with communication between the robots is more efficient than other methods, and this efficiency can be vital in search and rescue scenario.

CAPÍTULO 1

INTRODUÇÃO

Quando um grupo de pessoas se perde na mata ou em algum lugar isolado, são destinados profissionais treinados para busca e salvamento. Porém, estes profissionais possuem diversas limitações como a visão, necessidade de descanso e sono, obstáculos por intempéries, entre outros, o que pode acarretar em riscos de vida ou a não localização das vítimas [22].

Uma forma de dirimir estes problemas é usar robôs na realização desta busca e salvamento, uma vez que a maioria destas limitações não se aplicam aos robôs, principalmente no tempo de realização da busca. Com o uso de robôs é possível atuar em diferentes ambientes e sob diversas condições climáticas, porém deve-se considerar as limitações inerentes à robótica, como a limitação de energia, obstáculos e visão para navegação e a restrição do poder computacional.

Nesta busca é possível utilizar mais de um robô formando um grupo com o mesmo objetivo, pois espera-se que um grupo robótico possa alcançar o objetivo mais rápido. Uma forma de organizar o comportamento dos robôs desse grupo é com o uso de enxames, tornando-o um enxame robótico. Com o enxame robótico, as limitações de cada robô podem ser superadas pelo todo, pois as tarefas podem ser distribuídas de acordo com as possibilidades de cada um, além de trazer características como autonomia e cooperação.

O estudo de enxames robóticos aliado à comunicação sem fio têm apresentado resultados aplicados à busca e salvamento, como [14], [36], [37], [40], [35], [48], [60], [27], [41], [39] e [42]. Neste cenário, a robótica destaca-se como uma ferramenta para diferentes soluções, e quando aliada com as tecnologias de redes sem fio permite a comunicação e cooperação entre os robôs, e assim, é possível a localização de objetos perdidos em um menor tempo.

Mesmo com o uso dos robôs e conceitos do enxame, é necessária uma forma de localizar o objeto perdido. A indicação da intensidade do sinal recebido, ou *Received Signal Strength Indicator (RSSI)* é uma forma de identificar um objeto ou uma pessoa perdida no cenário

de busca e salvamento, pois permite ler a frequência de um sinal emitido por tecnologias que trabalham com rádio frequência, como *smartphones*. Assim, uma pessoa pode ser encontrada com essa forma de busca, desde que porte um objeto emissor de sinal, como o seu celular.

Nesta pesquisa, foi utilizada a intensidade de sinal na busca pelo objeto perdido e três métodos de busca: busca por tentativa e erro, busca com localização e busca com localização e comunicação, todos os métodos se baseiam em enxames e leitura de intensidade de sinal. Após a realização dos testes, notou-se que em ambientes pequenos as diferenças no desempenho entre os métodos não foram tão grandes, porém, em ambientes grandes o método com localização e comunicação obteve um desempenho melhor do que os demais métodos.

1.1 Objetivos

O objetivo principal desta dissertação é usar um enxame robótico para localizar pessoas perdidas, que portem objetos com emissão de sinal, utilizando o RSSI. Para alcançar este objetivo, foram determinados alguns objetivos específicos:

- estudar os conceitos e fundamentos que servirão de base para essa pesquisa;
- pesquisar as soluções de busca e salvamento que utilizem robôs, comunicação e intensidade de sinal com o objetivo de identificar suas características específicas, vantagens e desvantagens;
- propor métodos de busca utilizando um enxame robótico e a leitura da intensidade de sinal;
- implementar, testar, obter e analisar os resultados dos métodos propostos;
- apresentar as conclusões e apontar direcionamentos para novas pesquisas.

1.2 Contribuições

Este trabalho propõe um enxame de robôs com objetivo de encontrar pessoas perdidas através da leitura de intensidade de sinal. Dentre as contribuições deste trabalho, destacam-se:

- a evidência da busca pela intensidade de sinal e sua aplicação em cenários de salvamento;
- a aplicação dos conceitos de enxame no contexto de busca mediada pelos robôs;
- a implementação e testes dos métodos.

1.3 Organização do trabalho

Este trabalho está organizado em 6 capítulos:

- Capítulo 2: apresenta a teoria e os conceitos necessários que fundamentam a pesquisa;
- Capítulo 3: contém os trabalhos relacionados que servem como referência e ponto de partida para a pesquisa;
- Capítulo 4: apresenta as metodologias utilizadas para a identificação do objeto e da realização da busca;
- Capítulo 5: contém a implementação da busca, testes em diferentes cenários, obtenção e análise dos resultados;
- Capítulo 6: apresenta a conclusão e os trabalhos futuros.

CAPÍTULO 2

FUNDAMENTAÇÃO

Este capítulo apresenta os conceitos e fundamentos sobre a robótica aplicada, enxames de robôs, intensidade de sinal e os tipos de rádio. Estes conceitos são apresentados de forma objetiva, coerente com a necessidade desta pesquisa, e os conceitos mais aprofundados podem ser encontrados nas referências citadas.

2.1 Robótica Aplicada

Robôs são dispositivos que contêm uma estrutura mecânica com sensores, atuadores e um sistema de controle automático. Os primeiros robôs começaram a ser construídos na década de 60 por Engelberger, que foi considerado o pai da robótica, porém foi na década de 80 que a robótica cresceu, principalmente com o incentivo da indústria automotiva [33].

Atualmente, os robôs auxiliam nas aplicações industriais e são usados para diversas operações tecnológicas, podendo trabalhar em baixas ou altas temperaturas, muitas vezes, executando operações melhor do que as pessoas, com maior precisão e repetibilidade, sem precisar de luz, descanso, bonificações, entre outros. Os robôs podem ser classificados em 3 grupos [33]:

- Robôs Móveis: podem se locomover no ambiente. Existem três tipos de robôs: terrestres, aéreos e subaquáticos. Os robôs possuem aplicações para diferentes áreas, como por exemplo [59]:
    - Aplicações Domésticas e de Entretenimento: aspiradores de pó, cortadores de grama, jogos, entre outros;
    - Industriais: Transportes Automatizados, solda, montagem, entre outros;
    - Urbanas: Cadeiras de Rodas, vigilância, entre outros;

- Segurança e Defesa Civil e Militar: Resgate e Exploração em Ambientes Hostis, ataque e defesa, entre outros;

- Robôs Humanóides: Trabalham em ambientes compartilhados com os humanos. São ajudantes e devem ser capazes de aprender a realizar novas tarefas [11]. Realizam ações semelhantes a dos Humanos, como por exemplo: andar, pular, dançar, subir / descer escadas e transportar objetos [56];

- Robôs Industriais: Trabalham em ambientes isolados ou com poucas pessoas. Suas tarefas são repetitivas e limitadas, porém possuem precisão, alta carga, velocidade e rigidez.

Um robô é um dispositivo mecânico reprogramável, que pode: obter informações do ambiente que está inserido utilizando sensores, tomar decisões sobre o que fazer com base nas informações obtidas do ambiente e manipular objetos do ambiente utilizando atuadores.

Os robôs seguem princípios comuns e básicos, como por exemplo o modo que interagem entre si para executarem um objetivo. Entre estes princípios estão [51]:

- Sistemas de locomoção: possibilita que um robô móvel se mova no ambiente. Nos robôs móveis terrestres a locomoção está suportada nas rodas, patas ou esteiras. Nos robôs aéreos (drones) a locomoção é possível através das hélices.

- Motores e energia: são fundamentais para um robô móvel. Os motores mais usados são os elétricos ou de combustão. Os robôs devem possuir o fornecimento de energia a todos os sistemas que necessitam (motores, atuadores, sensores, computadores, sistemas de comunicação, entre outros). O fornecimento de energia se relaciona com a autonomia que se pretende com o robô e para isto, o mais comum é utilizar baterias.

- Percepção, navegação e atuação: podem ser atribuídas funções aos robôs e estes podem trabalhar em grupos com objetivos em comum. As funções de navegação possuem os ciclos:

- Percepção: para o robô se deslocar em um ambiente desviando de obstáculos, ele deve adquirir, analisar e interpretar as informações do ambiente, para isto é usada a percepção que dá conhecimento do ambiente que está inserido.

- Navegação: para o robô navegar em um ambiente, saber onde está e para onde deve ir, definindo o caminho que deve seguir, o robô usa a navegação.

- Atuação: para que o robô perceba o ambiente e consiga tomar decisões a partir do ambiente que está inserido, é necessário a atuação.

- Comunicação: é importante para que os robôs comuniquem entre si ou para a comunicação com as pessoas, através de envio de vídeos, imagens ou para a tomada de decisões.

- Interface com o utilizador: os robôs foram desenvolvidos para auxiliarem as pessoas na realização de tarefas pesadas, repetitivas ou em ambientes perigosos, e muitas vezes é necessário uma interface para que haja troca de informações entre o robô e as pessoas que estão auxiliando na operação. Usando a interface é possível disponibilizar os dados dos sensores e receber instruções para realizar uma determinada operação.

Com o avanço das pesquisas na área da robótica, houve o interesse pelo comportamento dos enxames de animais (formigas e abelhas), do movimento sincronizado e da definição de regras simples para cada indivíduo. Percebeu-se que o enxame torna-se mais inteligente do que um único indivíduo, pois há a cooperação para o alcance de um objetivo comum. Estas observações foram vitais para a ideia do enxame robótico [19] [5] [3] [1] [2] [6] [4].

2.2 Enxame de Robôs

Enxames de robôs são compostos por vários robôs móveis simples que, inseridos em um mesmo espaço, interagem para alcançar um objetivo [16]. No enxame cada robô realiza uma tarefa simples, que juntas resultam em uma tarefa complexa. Assim, a falha de um robô não compromete a tarefa do enxame [49].

O comportamento do enxame de robôs se baseia no comportamento de colônias de insetos, como por exemplo: abelhas e formigas, nas quais não há um único inseto que controle os demais, além do processamento ser individual e haver uma comunicação local [16].

Um problema comum em um enxame com muitos robôs é o congestionamento, principalmente quando os robôs se movimentam para o mesmo local simultaneamente. Um exemplo é o transporte de escombros onde os robôs percorrem o mesmo caminho. Como solução, pode ser utilizado uma Unidade Central de Processamento (CPU - Central Processing Unit) que trace o melhor caminho para cada robô percorrer, porém, o uso deste processamento tornaria o enxame centralizado [16].

Segundo [49], a ausência de uma unidade central pode ter pontos positivos, como:

- Robustez: Capacidade de realizar tarefas mesmo quando há um ou vários indivíduos com falha;
- Flexibilidade: Capacidade de adaptar o comportamento a diferentes tarefas atribuídas. Por exemplo, se a tarefa é empurrar um objeto e um robô não tem força suficiente para a realização desta tarefa, outros robôs podem colaborar para sua realização;
- Escalabilidade: Permite manter a execução das tarefas mesmo com as alterações que ocorrem com a quantidade de robôs.

Segundo [49], a coordenação de enxame de robôs é uma área interessante para a pesquisa, pois há muitas aplicações em diferentes cenários (exploração ambiental [15], monitoramento [25], operações de busca e resgate [54], entre outros).

Existem vários tipos de enxames, como por exemplo:

- Enxame de formigas – As formigas de um determinado formigueiro se guiam através de uma trilha química composta por feromônios. Quando as formigas caminham do formigueiro até a fonte de alimentação e vice-versa, elas liberam a substância no chão. Se tiver vários caminhos com a substância, a formiga tende a escolher o que

possui maior concentração de feromônio [53] [30]. O feromônio evapora depois de um determinado tempo, por este motivo, os caminhos mais longos tendem a possuir menos feromônio [26].

- Enxame de abelhas – Um grupo de abelhas saem separadas em procura de fontes de alimento, quando uma ou mais abelhas encontram, elas retornam para a colmeia para avisar as demais abelhas através da dança do balanço, esta dança informa as demais abelhas se há ou não bastante alimento e o quão longe está da colmeia. Com base nestas informações as abelhas decidem em qual fonte ir buscar o alimento [34] [53].

- Cardume de peixes – Os peixes possuem um comportamento coletivo que traz benefícios à vida social de cada peixe, como por exemplo: a evasão do predador com eficiência. Tanto a desova quanto a alimentação dependem do clima do ambiente. O nado entre um cardume é coordenado através do alinhamento, da proximidade e da velocidade [10].

- Colônia de bactérias – são capazes de migrar para lugares ricos em nutrientes utilizando a quimiotaxia, que determina o movimento de uma bactéria, sendo possível nadar em uma direção específica ou girar sem direção no meio do líquido [53]. Em um ambiente neutro, as bactérias possuem um comportamento padrão de deslocamentos contínuos e mudanças aleatórias de direção. Em um ambiente nutritivo, as bactérias reduzem as mudanças de direção. Em um ambiente nocivo, as bactérias elevam seu tempo de deslocamento e retornão ao seu comportamento padrão [18] [17] [38].

- Manada de Lobos: Uma manada possui entre 5 a 20 lobos, estes lobos são divididos da seguinte forma: Lobo líder: é o lobo mais forte, feroz e inteligente da manada. O líder comanda os demais lobos e toma as decisões; Lobos escoteiros: são os lobos mais fracos da manada. Estes lobos procuram as caças de forma independente em ambientes próximos do restante do bando, quando encontra a presa, o lobo uiva convocando os lobos ferozes; Lobos ferozes: assim que os lobos escoteiros uivam,

estes lobos se movem rapidamente para a direção do uivo para capturar as presas. Depois de capturada a presa, há duas formas de dividir a comida: do forte para os fracos ou de forma aleatoria [32] [57].

2.3 Robótica Colaborativa

Robôs Colaborativos (Cobots) trocam informações entre si e podem trabalhar em conjunto no alcance de um objetivo. Este comportamento se assemelha há alguns tipos de enxame, porém os robôs colaborativos podem ter sensores diferentes. Os Cobots foram criados pela Universidade Northwestern e General Motors Corporation, e possuem dois importantes conceitos: Ergonomia e Segurança [51] [9].

O comportamento dos Cobots se baseia em regras simples e por isso, grupos de pesquisadores têm estudado o comportamento coletivo dos robôs, como por exemplo, o comportamento do enxame de robôs militares, que possuem robôs simples e baratos e reagem apenas às situações locais [7]. Os Cobots possuem três características:

- possuem um conjunto de regras de comportamento;
- só respondem a situações locais e;
- o robô é homogêneo.

Os robôs colaborativos podem trabalhar ao lado de seres humanos, cooperar, adaptar e aprender com o comportamento humano. Comparados com os robôs mais antigos, que predominava os robôs grandes com força e velocidade, os robôs colaborativos são menores e mais lentos [8].

2.4 Rádios

A tecnologia de rádio é muito utilizada para comunicação. Ela consiste em ondas que podem percorrer longas distâncias e entrar facilmente dentro de prédios, além de serem fáceis de gerar. As ondas possuem algumas propriedades que variam de acordo com a frequência: nas frequências baixas, as ondas atravessam os obstáculos e a potência cai

à medida que a distância da origem aumenta; nas frequências altas, as ondas tendem a viajar em linha reta, ricocheteando nos obstáculos [50].

O caminho que estas ondas percorrem pode variar entre uma linha reta ou obstruída por obstáculos. Para os caminhos com obstáculos, devem ser considerados os mecanismos básicos de propagação, que são [50]:

- Reflexão: ocorre quando uma onda em propagação colide com um obstáculo e retorna ao meio de propagação, como por exemplo morros, montanhas, prédios e paredes.
- Difração: ocorre quando o caminho da onda em propagação é obstruído por obstáculos, e que, ao bater nestes obstáculos, as ondas geram ondas secundárias que mudam a direção de propagação da onda principal, e desta forma, as ondas são capazes de atravessar os obstáculos.
- Dispersão: ocorre quando o caminho da onda em propagação é obstruído por vários objetos pequenos produzindo várias ondas dispersas. Exemplo de obstáculos que induzem à dispersão: susperfícies ásperas, folhagens, pequenos objetos, postes, entre outros.

As ondas de rádio podem percorrer longas distâncias, e por isso, pode haver interferências entre os usuários, dessa forma, o uso das frequências é controlado pelo governo. No próximo capítulo serão abordados alguns protocolos que utilizam ondas de rádio para transmissão [50].

2.4.1 Comunicação

Existem vários protocolos que usam ondas de rádio para a comunicação. Neste capítulo serão abordados os protocolos Bluetooth, WiMax, ZigBee e Redes sem fio.

- Bluetooth (padrão 802.15.1) – Este protocolo utiliza uma conexão de rádio de curto alcance, usa a banda de 2,4 GHz e, em uma distância de 10 metros possui a capacidade de transmissão de 1 Mbps. Esta tecnologia se tornou o padrão sem fio de curto alcance para uma ampla variedade de dispositivos, como por exemplo, para

usar teclado e mouse no computador, ligar computadores à impressoras, usar o fone de ouvido com o celular, entre outros [50].

- WiMax (padrão 802.16) – Este protocolo consiste em uma estação base fixa em um ponto alto da cidade e estações clientes que recebem o sinal da estação base. Existem dois tipos de estações clientes: estações fixas - residências com antenas fixas no telhado; e estações móveis – dispositivos móveis com a tecnologia WiMax. As estações fixas possuem uma banda de 3,5 GHz e as estações móveis variam de 2,5 GHz, 3,5 GHz e 5,8 GHz, dentre as quais apenas a última banda não é licenciada no Brasil [28].

- ZigBee (padrão 802.15.4) – Este protocolo não requer licença para funcionamento, oferece imunidade contra interferências, possui a capacidade para vários dispositivos em uma mesma rede com taxas de transferências de dados variando entre 20 à 250 kbps e trabalha na frequência de 2,4 GHz. Uma rede ZigBee é formada por dois tipos de dispositivos: *Full Function Device* (FFD) e *Reduced Function Device* (RFD). Os dispositivos FFD possuem a função de coordenador/roteador da rede e os dispositivos RFD são conhecidos como escravos e só comunicam com o coordenador/roteador da rede que está associado [60], [12], [29].

- Redes sem fio *WiFi*(padrão 802.11) – Este protocolo é o principal padrão de LAN sem fio e pode ser usado de duas formas: na primeira, em lugares públicos, aeroportos, escritórios, entre outros para fornecer as pessoas conexão à Internet através de seus smartphones, tablets ou laptops; na segunda, dois ou mais computadores podem se comunicar sem usar a internet, este tipo de rede é conhecida como rede *ad hoc*. Esse padrão 802.11 surgiu em 1997, e atualmente, existem algumas variações conhecidas como: 802.11a – surgiu em 1999, usa uma banda de frequência de 5 GHz e uma velocidade de 54 Mbps; 802.11b – surgiu no início do padrão 802.11, utiliza uma banda de 2,4 GHz e velocidade de 1 a 2 Mbps, esta velocidade foi alterada para até 11 Mbps; 802.11g – surgiu em 2006, utiliza a banda de frequência de 2,4 GHz e oferece velocidade de 54 Mbps; 802.11n que utiliza simultaneamente várias antenas

no transmissor e receptor para aumentar a velocidade, que pode chegar até 300 Mbps; entre outras. Todas as técnicas do padrão 802.11 utilizam rádios de curto alcance para a transmissão de sinais nas bandas de 2,4 ou 5 GHz [50].

2.4.2 RSSI

A realização de buscas em uma determinada área é possível utilizando o RSSI, trata-se de um parâmetro que mede a energia recebida de um sinal de rádio, esse parâmetro é usado para determinar a qualidade do sinal de uma rede sem fio em um determinado ambiente [58].

Se existirem vários receptores calculando a intensidade de sinal de um mesmo transmissor, através dos valores da intensidade de sinal percebidos pelos receptores é possível calcular a posição do transmissor. Este cálculo é possível quando se conhece a força do transmissor, o modelo de perda de sinal pelo caminho bem como o coeficiente desta perda. Assim o receptor pode usar a força do sinal recebido e calcular a sua distância, como demonstrado em [21].

Em ambientes internos é mais difícil estimar a posição devido à refração, difração, reflexão ou absorção [31]. Devido às várias reflexões de vários objetos ou paredes, as ondas eletromagnéticas se deslocam para diferentes caminhos com comprimentos variados. Conforme a distância entre transmissor e receptor aumenta, a intensidade do sinal dessa onda diminui [58].

O RSSI pode ser avaliado como Bom, Aceitável e Fraco de acordo com a intensidade do sinal recebido, ou seja, se a intensidade do sinal recebido for maior que 40 dBm é considerado bom, se o valor for menor que 40 dBm e maior que 35 dBm é considerado aceitável, e se este valor for menor que 35 dBm é considerado fraco [58].

Para obter o RSSI é necessário o uso de um receptor *Wi-Fi* para ler o sinal e a sua intensidade. O RSSI pode ser medido tanto em ambientes interno quanto externo. Há alguns aspectos que devem ser considerados referente ao RSSI [21]:

- Os valores RSSI podem oscilar, mesmo não havendo movimento entre emissor e receptor.

- Dependendo da configuração dos dispositivos, um mesmo valor de intensidade do sinal pode resultar em diferentes valores de RSSI para diferentes dispositivos.

- Os obstáculos podem fazer com que o sinal percorra vários caminhos. A atenuação do sinal ao longo de vários caminhos é maior do que em um caminho direto, podendo obter uma distância de caminho maior do que a distância real.

- As pessoas também são obstáculos móveis para os sinais de rádio, podendo ter variação no RSSI.

Nesse capítulo foi apresentada uma visão geral de robótica, enxames e rádios com conceitos e fundamentos necessários para o entendimento dessa pesquisa. O próximo capítulo apresenta três trabalhos que utilizam os conceitos descritos nesse capítulo com objetivo de busca, sejam para localização, leitura de intensidade de rádio ou cooperação entre subequipes.

CAPÍTULO 3

TRABALHOS RELACIONADOS

Esta seção apresenta um estudo de trabalhos com propostas relacionadas ao objetivo desta pesquisa. Após uma revisão da literatura da área, três trabalhos foram selecionados por suas abordagens ao problema da localização e por utilizar metodologias e tecnologias que podem auxiliar no desenvolvimento desta pesquisa. Estes trabalhos são:

- *The Search for Survivors: Cooperative Human-Robot Interaction in Search and Rescue Environments using Semi-Autonomous Robots* (A busca por sobreviventes: Cooperativa de Interação Humano-Robô na Busca e Salvamento em Ambientes usando robôs semiautônomos) [27];
- *Multi-robot search and rescue team* (Multi-robôs para equipes de busca e resgate) [41];
- *Search Strategy of a Mobile Robot for Radiation Sources in an Unknown Environment* (Estratégia de Busca de um Robô Móvel para fontes de radiação em um ambiente desconhecido) [39].

Os próximos subcapítulos explicam de uma forma resumida estes artigos.

3.1 A busca por sobreviventes: Cooperativa de Interação Humano-Robô na Busca e Salvamento em Ambientes usando robôs semiautônomos

O artigo proposto por Doroodgar, Mobedi e Nejat da Universidade de Toronto - Canadá e por Ficocelli da Universidade de Nova York - USA ([27]) apresenta a interação entre pessoas (denominadas operadores) e robôs para a busca de sobreviventes em escombros. Para isto, os robôs são equipados com câmeras que transmitem em tempo real as imagens

do ambiente. Através destas imagens é possível que o operador consiga identificar se há possíveis sobreviventes. Para esta interação foi proposto um equilíbrio entre a autonomia do robô e a quantidade de controle do operador sobre o robô, sendo que o robô possui a capacidade de aprender e tomar decisões.

Para que o robô consiga tomar decisões, foi desenvolvido um sistema de mapeamento 3D em tempo real que fornece imagens 2D e 3D, além de identificar pontos de referência. Além disso, este trabalho apresenta uma arquitetura de controle exclusivo para navegação semiautônoma de uma plataforma robótica, onde o próprio robô mapeia o ambiente e consegue se situar no ambiente que está inserido.

Para a navegação semiautônoma, foi desenvolvido um algoritmo de controle que permite que o robô aprenda e tome decisões sobre quais tarefas devem ser realizadas em um determinado momento e quem deve executar estas tarefas para obter melhores resultados: o operador ou o robô. O operador possui uma interface que permite obter informações do ambiente e do robô, a fim de controlar o robô somente quando for necessário.

A arquitetura do trabalho consiste de dois principais componentes: o mapeamento 3D em tempo real do ambiente que o robô está inserido e o controle do robô sobre a sua própria autonomia durante as operações de busca e salvamento.

Para o mapeamento do ambiente e controle do robô, foi utilizado um sensor que fornece imagens em 2D e 3D e em tempo real do ambiente com escombros, tanto em ambientes com luz quanto em ambientes escuros, a partir destas imagens um mapa 3D virtual é gerado com alguns pontos de referência para que o robô consiga se localizar no ambiente. A partir do cenário, o robô toma as decisões e decide quem irá executá-las.

Foram feitos experimentos em um ambiente desconhecido e com vários objetos incluídos, como madeira, metal, plástico, tijolo, cerâmica, concreto, papel, papelão, gesso e rochas, além de oito bonecos que representavam as vítimas. Cinco operadores realizaram os testes sendo que cada operador possuía um controle (*joystick*) para situações que o controle é passado do robô para o operador. Durante os testes, o robô colidiu algumas vezes com pequenos obstáculos que estavam fora do seu campo de visão e, também colidiu com o boneco quando os operadores estavam guiando por falta de experiência do operador. O

robô e o operador conseguiram localizar sete vítimas das oito que estavam no ambiente.

Este trabalho se difere dessa pesquisa por possuir apenas um robô e por precisar de um operador em algumas situações em que o robô não é capaz de resolver sozinho.

3.2 Multi-robôs para equipes de busca e resgate

O artigo proposto por Luo, Espinosa, Pranantha e Gloria da Universidade de Genoa - Itália ([41]) apresenta uma abordagem para busca e resgate de sobreviventes dentro de um edifício. Por não conhecer o interior do edifício - pode estar desmoronando ou com possíveis reféns - deu-se a necessidade de montar uma equipe. Esta equipe consiste em 4 subequipes: um veículo que mapeia o ambiente, um veículo aéreo, um veículo terrestre e um de *backup* para o resgate ou segurança e a estação de controle.

- Veículos terrestres: para as subequipes de veículos terrestres (a subequipe que mapeia o ambiente e a subequipe de veículo terrestre e de *backup*) foram utilizados robôs Legos equipados com sonares e bússolas.

- Veículo aéreo: foi utilizado um *drone* equipado com uma câmera vertical, uma câmera horizontal, um telêmetro para medir as distâncias, e controle automático, porém a qualquer momento a estação base pode mudar o controle para manual para controlá-lo.

- Estação de controle: é equipada com um computador portátil para o processamento de imagem, detecção de objetos e para o controle dos robôs. A estação consegue controlar e receber os dados do veículo aéreo através da rede *WiFi* e dos veículos terrestres através da rede *Bluetooth*.

Pelo fato de um veículo terrestre mapear o ambiente e os demais veículos usarem este mapeamento para acharem o edifício alvo e se localizarem, foi escolhido robôs Lego pela facilidade de utilização, desempenho da odometria e preço razoável.

As quatro equipes trabalham juntas, para isso, o veículo terrestre inicialmente se move para frente a partir da estação base, este ponto se torna um ponto de partida conhecido,

depois percorre o caminho até o edifício alvo para encontrar uma possível entrada, este percurso é mapeado e enviado para a estação base, que enviará estas informações para as demais subequipes. Para o mapeamento, foi utilizado o método de mapeamento SLAM (*Simultaneous Localization and Mapping*), este método é usado para construir o mapa do ambiente ao mesmo tempo que navega e se localiza no ambiente. Para corrigir possíveis dados incorretos devido a ruídos, o sistema utiliza um método baseado no método probabilístico Bayes.

Posteriormente, o veículo aéreo vai para o local utilizando o mapa fornecido pela estação base e com a ajuda de um telêmetro na tentativa de localizar o alvo e ver as condições que este alvo se encontra: reféns, ataques terrorista, desabamento, entre outros, e repassa esta informação a estação base, que decide a estratégia adequada a ser usada.

Por último, o veículo terrestre vai até o alvo para realizar a sua tarefa, que pode ser de resgate ou de segurança. Se a estação base perceber a necessidade de apoio para o veículo terrestre, ela acionará o veículo de *backup* para fornecer assistência ao veículo terrestre.

Foram realizados 72 testes de resgates. Os testes foram utilizados com e sem o apoio dos veículos aéreos, e percebeu-se que com o uso do veículo aéreo reduziu o tempo de busca de forma significativa e que a missão foi cumprida em menor tempo do que usando apenas uma equipe de resgate.

Este trabalho se difere dessa pesquisa por não possuir o conceito de enxames, embora usa uma equipe dividida em quatro subequipes, cada subequipe é responsável por uma tarefa específica e duas subequipes possuem apenas um veículo. Se este veículo falhar, a equipe não terá sucesso na operação. Além disso, uma subequipe é a base de apoio formada por humanos que é responsável em repassar todas as informações obtidas de uma equipe para a outra equipe, sem esta base de apoio, as demais subequipes não possuem comunicação entre si.

3.3 Estratégia de Busca de um Robô Móvel para fontes de radiação em um ambiente desconhecido

O artigo proposto por Lin e Tzeng da Universidade de Tecnologia de Taipei - Taiwan ([39]) apresenta um robô móvel que navega de forma autônoma em busca de possíveis fontes de radiação em um ambiente desconhecido. Para ser possível a localização de fontes de radiação, o robô possui um contador Geiger (dispositivo que detecta e mede radiações).

Para o robô conseguir navegar no ambiente e encontrar a fonte de radiação, foi adotada uma APF (Análises de Pontos de Função), nesta APF devem ser descritos os vetores de força atrativa e repulsiva. A força atrativa do vetor é prevista por um filtro de partículas que usa o contador Geiger, tornando-se mais forte à medida que o robô está se aproximando da fonte de radiação. A força repulsiva é regulada pela distância a partir do robô para um obstáculo, tornando-se mais forte quando o robô está mais próximo do obstáculo, para isso, foi utilizado um telêmetro a laser que retorna a distância e orientação sobre o robô.

O vetor de força de repulsão é estimado por um filtro de partículas cujos parâmetros são a intensidade da fonte de radiação, a orientação relativa e a distância entre o robô e a fonte de radiação. As partículas com maiores pesos são utilizadas para gerar a força de atração do vetor.

A força repulsiva do vetor é impulsionada pelo obstáculo e é relacionada com a distância a partir do robô para o obstáculo. A força atrativa do vetor é derivada a partir do sinal de radiação recebida do sensor. Porém, a intensidade de radiação recebida pelo sensor é influenciada pelos obstáculos, dessa forma, deve ser usado um filtro de partículas para estimar a possível localização e orientação da fonte de radiação.

O robô, quando percebe algum nível de radiação, segue em busca deste nível, sempre para a direção que o índice de radiação for maior, desviando apenas dos obstáculos. Para isto, foram criados vetores de todas as partículas que apontam a fonte de radiação. Com base nestas informações dos vetores, o robô consegue se deslocar até a fonte de radiação.

Foram realizados testes sem vetores de força de impulso e percebeu-se que os robôs se moviam em direção a fonte de radiação evitando os obstáculos, porém a busca do robô está

em zigue-zague porquê o filtro de partículas não chega a uma convergência. Para melhorar esta busca, foi adicionado um vetor de força dinâmica, e assim, o robô não navegou mais de forma aleatória, e sim, navega de forma eficiente para a fonte de radiação. A razão desta melhora é que o vetor de força estimado pelo filtro de partículas é instável no início, de modo que o vetor de força de atração é contribuído principalmente no vetor de força de impulso. Foram realizados vários testes em um espaço de 100 metros quadrados, todos os resultados das simulações foram satisfatórias.

Este trabalho se difere dessa pesquisa por usar apenas um robô e por buscar a leitura da intensidade de força da radiação, além disso, o robô apenas detecta e monitora as informações da radiação e repassa estas informações para humanos através de uma rede sem fio não informada no trabalho.

3.4 Comparação entre os trabalhos relacionados

A tabela 3.1 resume as características dos trabalhos relacionados. Todas as soluções são voltadas para a busca, contudo nenhuma aborda todas as características elencadas na tabela, além de não utilizarem o conceito de enxame. As principais vantagens e desvantagens destes trabalhos são:

- A vantagem do trabalho [27] é a autonomia do robô e a tomada de decisão de quando necessitar da ajuda de um ser humano, além de utilizar uma rede de comunicação para enviar imagens do robô para o homem, embora esta rede não seja detalhada no trabalho. A desvantagem está no uso restrito a apenas um robô, pois o sucesso da busca está limitada ao sucesso deste robô.

- As vantagens do trabalho [41] são: o uso de equipes, embora não utilize o conceito de enxame, e a distribuição das tarefas entre quatro equipes aplicando, de certa forma, o conceito de cooperação; a utilização de uma base de apoio das equipes para estabelecer a comunicação e tomadas de decisões, o que se assemelha muito ao conceito de líder em um enxame do tipo manada de lobos; e o posicionamento a partir de um ponto inicial. A desvantagem é a necessidade da interação com humano

na base de apoio, pois os robôs não se comunicam entre si.

- As vantagens do trabalho [39] são: o uso da leitura da intensidade de força da radiação, embora não utilize sinais de radio frequência de fontes como uma rede celular, este guia-se pela intensidade da propagação da onda de radiação; é autônomo por não precisar em nenhum momento de interação humana; e é capaz de comunicar através de uma rede sem fio com o ser humano informando os níveis de radiação presentes no ambiente, embora esta rede não seja detalhada no trabalho. A desvantagem está no fato do robô não encontrar um objeto perdido e sim, detectar os níveis de radiação para que seres humanos saibam quando é seguro para que eles possam entrar no ambiente e realizar a busca.

Tabela 3.1: Comparativo entre os trabalhos Relacionados.

Solução	**Grupo de robôs**	**Autonomia**	**RSSI**	**Redes sem fio**
[27]	-	x	-	x
[41]	x	x	-	x
[39]	-	x	x	x

Estes trabalhos contribuíram com suas características para a construção do enxame robótico proposto nessa pesquisa. As contribuições foram:

- trabalho [27]: a aplicação da autonomia dos robôs, cuja característica foi incorporada nos métodos propostos nesse trabalho.

- trabalho [41]: a capacidade de posicionar-se a partir de um ponto inicial, mesmo desconhecendo o ambiente, tornando este caminho conhecido. Este posicionamento foi levado em consideração para o desenvolvimento do método de busca com localização e comunicação dessa pesquisa. Outra contribuição foi a presença de uma base para comunicação e tomada de decisão, o que serviu de ponto de partida para o conceito de um líder no enxame robótico baseado em manada de lobos, proposto também no método de busca com localização e comunicação. Além disso, foi incorporado nessa pesquisa a característica de comunicação baseada em tecnologias de redes sem fio entre os robôs e uma base para posicionamento e notificações.

- trabalho [39]: o uso de leitura de intensidade de sinal para detecção e busca da fonte de propagação deste sinal. Esta característica foi adaptada para todos os métodos de busca dessa pesquisa, trazendo uma relação entre a intensidade de sinal e a proximidade do objeto a ser encontrado.

CAPÍTULO 4

A BUSCA FEITA PELO ENXAME DE ROBÔS

Este trabalho partiu da hipótese que é possível aplicar a comunicação de redes sem fio, a leitura da intensidade do sinal e um enxame robótico para buscar objetos ou pessoas perdidas. A relevância desta busca não está somente na importância do objeto a ser localizado e sim em explorar esta possibilidade e as tecnologias utilizadas para esta finalidade. Esta busca é possível com a identificação da intensidade de sinal emitida por determinado dispositivo, como o sinal *WiFi*, e a capacidade autônoma de cada robô pertencente ao enxame. Essa hipótese tem como base os fundamentos das tecnologias estudadas e os trabalhos relacionados que, de certa forma, serviram como ponto de partida e correlação no uso da intensidade de sinal para busca e salvamento. Para que esta hipótese de pesquisa seja verificada, é preciso definir em detalhes qual a metodologia da busca e delimitar o que será buscado.

4.1 O objeto a ser buscado

O objetivo dessa busca é encontrar um objeto perdido, seja um simples objeto ou uma pessoa portando um *smartphone*. Este objeto a ser procurado deve possuir emissão de sinal, para que os robôs do enxame, utilizando a leitura da intensidade do sinal, possam realizar a localização. Este sinal pode ser emitido por diferentes tecnologias, tais como os sinais GSM (900 MHz), WCDMA/HSDPA (1900/2100 MHz) e LTE (700 MHz), as quais são utilizados por *smartphones* nessas frequência no Brasil, segundo a homologação da Anatel [13], ou tecnologias utilizadas para transmissão de dados em redes sem fio, como *WiFi*, *ZigBee*, *WiMax* e Bluetooth usadas para estabelecer a comunicação.

A comunicação entre os robôs do enxame é fundamental para a coordenação e cooperação no alcance de um objetivo comum. Essa comunicação será estabelecida com redes sem fio de tal forma que permita a diferenciação entre os robôs do enxame e o objeto alvo.

Esta diferença pode ser implementada com o uso de dispositivos com tecnologias distintas, como os módulos *ZigBee* para os enxame e a *WiFi* (802.11) para o objeto a ser procurado, como ilustrado na figura 4.1.

Figura 4.1: Tecnologias de redes sem fio para a comunicação entre os robôs do enxame.

Além de utilizar tecnologias diferentes é possível também o uso de mesma tecnologia diferenciando o objeto perdido através de informações como ESSID ou BSSID, para a rede *WiFi*. Além disso, para que não haja interferências, podem ser usadas frequências diferentes ou estas interferências serem tratadas pelo próprio dispositivo de enlace. Dessa forma, nas realizações dos testes nessa pesquisa, para a comunicação entre os robôs foi utilizado a *WiFi*, detalhes do uso da tecnologia bem como a diferenciação entre sinais do enxame e da pessoa perdida serão detalhadas no capítulo 5.2.

Este trabalho não considerou a implementação de um módulo *ZigBee*, uma vez que a delimitação dessa pesquisa está na busca do objeto perdido e como a comunicação impacta nessa busca, e não em uma tecnologia específica. Esta escolha deu-se unicamente pela delimitação do escopo da pesquisa, embora o *ZigBee* seja bem aceito para esta finalidade. Desta forma, a implementação e uso deste módulo é indicada para trabalhos futuros.

O objeto perdido pode ser um dispositivo que possua um GPS (Global Position System), como um *smartphone*, ou não. Dessa forma, é interessante considerar formas de busca com e sem a localização exata do objeto quando detectado. Além disso, para delimitação da pesquisa, foi considerada a busca de um único objeto por todo o enxame, com o objetivo de alcançar o menor tempo necessário para encontrá-lo em diferentes cenários, embora

a busca por mais de um objeto seja totalmente válida, pois em um caso de busca por vítimas de um acidente há grande chance de haver um grande número de pessoas a serem resgatadas. A ampliação do número de objetos a serem encontrados, bem como melhores métodos para encontrá-los, foge ao escopo dessa pesquisa e são apontados como trabalhos futuros.

4.2 Metodologia da busca

Foram utilizados 3 métodos de busca: por tentativa e erro; busca com localização; e busca com localização e comunicação. Estes métodos foram necessários para abranger diferentes cenários: o primeiro considera um enxame simples e não há formas de obter a posição e nem comunicação, o segundo método considera a obtenção da posição enquanto que o terceiro possui a posição e a comunicação, além de um líder responsável pelo enxame. Todos os métodos utilizam a leitura da intensidade do sinal para identificar o objeto perdido, perceber a proximidade e estabelecer o momento correto de parar a busca, a fim de considerá-lo encontrado.

4.2.1 Busca por Tentativa e Erro

O método de busca por tentativa e erro é baseado na colônia de bactérias, no qual cada robô que compõe o enxame inicia de forma autônoma, ou seja, com movimentos não coordenados e independentes entre si. Este tipo de enxame foi escolhido por ter o comportamento mais próximo do que se almeja para esse método. Embora o enxame tenha um objetivo comum, cada robô possui a autonomia na forma de como buscá-lo e por onde percorrer. Se algum robô do enxame perceber o sinal do objeto perdido, deve guiar-se pela intensidade do sinal encontrado. Como não há uma posição exata a seguir, o robô tentará encontrar essa posição por tentativa e erro auxiliada pelo sinal do objeto.

Uma vez identificado o sinal do objeto perdido, o robô segue em frente na tentativa de aumentar a força do sinal, ou seja, aproximar-se do objeto, pois, os robôs não possuem visão computacional e são guiados somente pela intensidade do sinal. O melhor caso é o

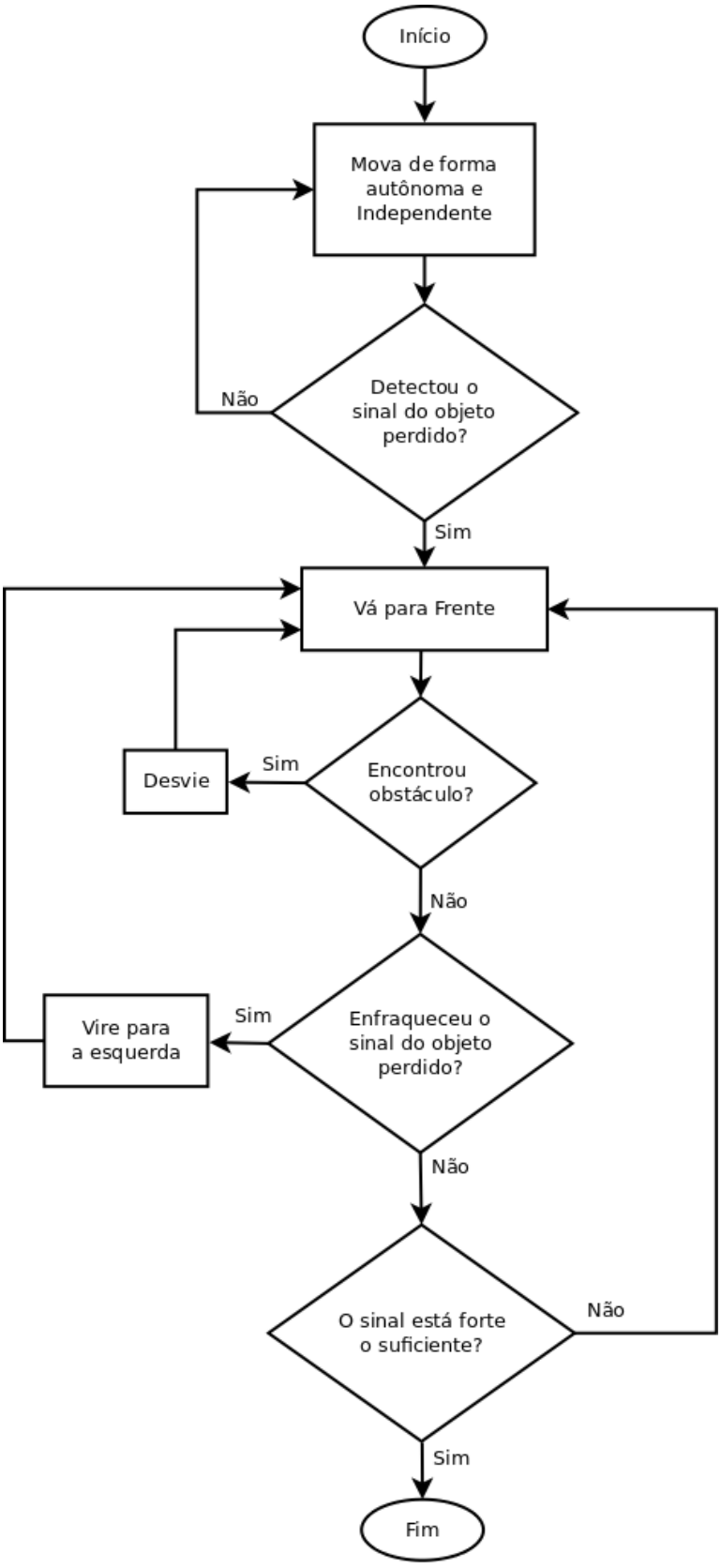

Figura 4.2: Fluxograma do método por Tentativa e Erro.

robô estar posicionado de frente ao objeto ao detectar este sinal, o que é pouco provável pois os robôs considerados nesse enxame possuem 4 lados. Se o robô estiver posicionado de lado ou de trás para o objeto, ao avançar ele se distanciará do alvo e a força do sinal atenuará e será detectado um erro. Quando isso ocorrer, o robô virará para o lado, escolhido arbitrariamente a esquerda, e avançará tantas vezes quanto necessárias para que a força do sinal aumente, colocando-o dessa forma de frente para o objeto procurado.

Os robôs são dotados de sensores de laser para evitar obstáculo e programados para desviar quando encontrados. Se após desviar de um obstáculo, ou em alguma outra ação, o robô perder o sinal do objeto, outrora detectado, ele reiniciará o processo de busca a partir do início, andando de forma autônoma e independente. Ao detectar novamente o sinal, o processo segue como explicado anteriormente. A figura 4.2 apresenta o fluxograma com o método de tentativa e erro. A perda do sinal não foi inserida no diagrama pois pode ocorrer em qualquer momento no processo de busca, não agindo de forma sequencial.

Este método foi criado por não necessitar de nenhuma comunicação entre os robôs do enxame e nem de uma localização exata, guiando-se somente pela força do sinal emitido do objeto. Pelo fato de não necessitar de uma cooperação ou acordos prévios, este método inicia-se rapidamente.

4.2.2 Busca com Localização

O método de busca com localização é análogo a busca por tentativa e erro, também baseado na colônia de bactérias, porém com uma pequena adaptação no momento de ir ao encontro do objeto procurado. Ao perceber o sinal do objeto procurado, o robô obtêm a localização exata do objeto e direciona-se a esta posição. Esta obtenção e direcionamento de um local específico podem ser realizados por diferentes formas, com uso de um GPS ou triangulação de sinal de rádio. A forma como é obtida esta localização não faz parte do escopo dessa pesquisa, apenas o uso dessa posição e como essa abordagem pode impactar no processo de busca e salvamento. Essa lacuna é apontada como trabalho futuro.

Com a localização exata do objeto perdido, inicia-se o processo de ir ao seu encontro guiado pela intensidade de sinal. Durante esse percurso, é possível que haja obstáculos,

e estes serão desviados para evitar as colisões, reposicionando-se guiado pela localização tão breve quanto possível. Quando a intensidade do sinal for forte o suficiente para que o robô esteja muito próximo ao objeto perdido, a busca terminará. A figura 4.3 apresenta o fluxograma com o método de busca com localização.

Esse método foi criado por supor que com uma localização específica e forma de navegação o robô encontrará o objeto perdido mais rapidamente do que pelo método de tentantiva e erro. Vale ressaltar que, assim como o método anterior, a busca com localização não possui a comunicação entre os robôs e consequentemente não estabelecem acordos prévios à busca, o que permite um rápido início do processo de busca.

4.2.3 Busca com Localização e Comunicação

Esse método considera a localização assim como no método de busca com localização, e a intensidade do sinal como nos métodos anteriores, porém diferencia-se dos outros métodos, pois é baseado numa manada de lobos. Este método de enxame foi escolhido pela necessidade de um líder comandar os demais e tomar as decisões guiando todo o enxame e direcionando-os para a busca do objeto perdido. Dessa forma, este método divide-se em dois comportamentos: do líder e dos demais robôs do enxame.

4.2.3.1 Robô Líder

Inicialmente, o líder deve ser escolhido. Numa manada de lobos, o líder é o mais forte do grupo, e nesse contexto o robô pode ser o que tem melhor processamento, mais recurso energético disponível ou maior força de sinal para comunicação com os demais. Estas características não foram contempladas no escopo da pesquisa, pois espera-se saber o quão eficiente será esse método em relação aos demais, e assim o líder foi escolhido arbitrariamente. Uma melhor seleção do líder, considerando o viés computacional, é apontado como trabalho futuro.

Assim como o lobo guia a sua manada comunicando por uivos, deve haver uma forma dos robôs do enxame se comunicarem. Para a busca e salvamento, o método mais viável para esta comunicação é o uso de redes sem fio, como demonstrado no trabalho relacionado

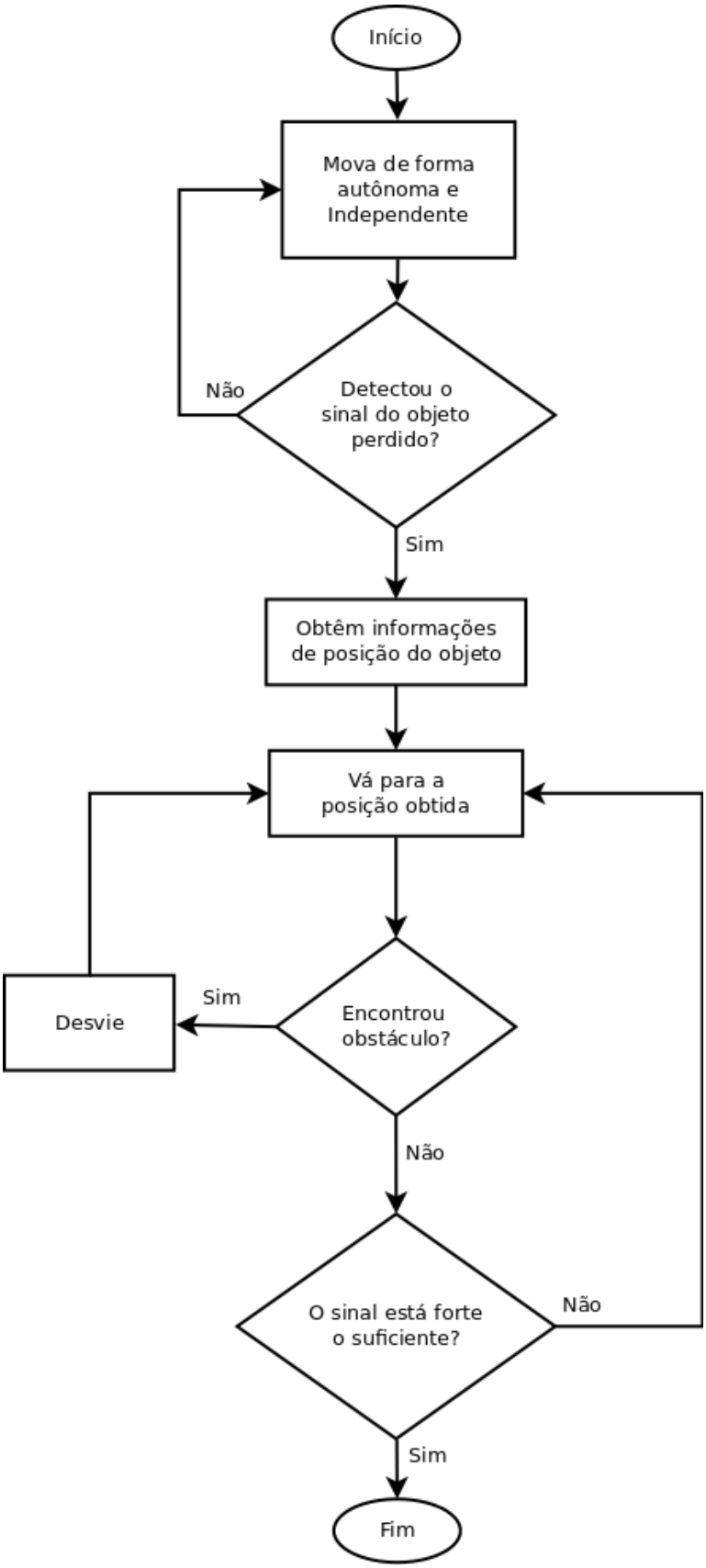

Figura 4.3: Fluxograma do método Busca com Localização.

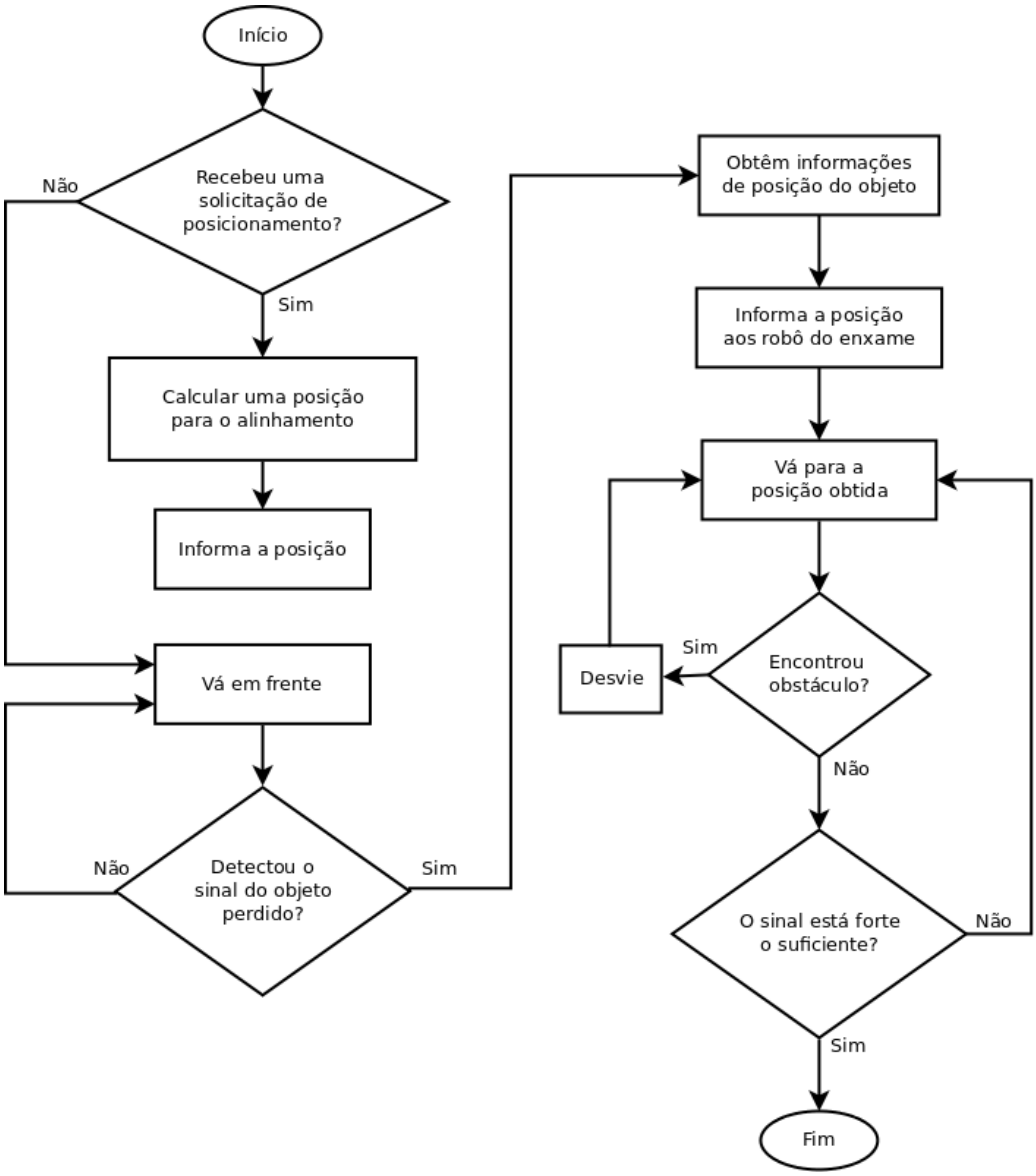

Figura 4.4: Interação entre os robôs na busca do objeto alvo.

[41]. O robô líder será responsável por agrupar todo o enxame e alinhá-los para abranger uma determinada área de busca e manter o grupo unido.

Por este motivo é necessária a comunicação entre os robôs e o líder, e ela deve ser estabelecida de tal forma que não interfira na leitura do sinal do objeto perdido. O robô líder calcula e informa a posição de todos os robôs do enxame baseado em sua própria posição, mas computando diferenças suficientes para que não hajam sobreposições e que o robô mais distante não se desconecte do líder por atenuação do sinal.

Por exemplo: se o raio de alcance dos robôs for de 100 metros, o líder busca a quantidade de robôs existente no enxame. Se a quantidade de robôs for 25, ele divide 100 por 25, que

resulta em 4 metros e dada a sua posição, informa para os demais robôs, um por um, o deslocamento de 4 metros, ou seja, o R0 é o líder e está na posição 0, ele vai mandar para o R1 a posição 4 (posição obtida através do cálculo 0 + 4), para o R2 a posição 8 (posição obtida através do cálculo 4 + 4), para o R3 a posição 12 (posição obtida através do cálculo 8 + 4), e assim por diante.

Quando o líder detecta o sinal do objeto perdido, assim como no método anterior, obtêm a localização exata do objeto e imediatamente informa esta posição aos demais robôs e direciona-se ao encontro do objeto. Esta informação é necessária para o cenário em que, mesmo o líder sendo o detector inicial do objeto perdido, um outro robô do enxame pode estar mais próximo do objeto mas não detectou o sinal por um simples obstáculo. Com a informação da localização, este robô pode direcionar-se ao objeto perdido, desviando do obstáculo que o impedia de detectar o alvo e rapidamente encontrá-lo, diminuindo o tempo da busca, e isto justifica a difusão da localização do objeto perdido, quando encontrado.

Em outro cenário, o líder está mais próximo do alvo ao detectá-lo e enfrentar obstáculos que toma demasiado tempo ao ponto de outro robô encontrar o objeto perdido. Este cenário confirma a necessidade do líder informar todos os demais robôs da localização do alvo detectado, com o objetivo de alcançar o menor tempo possível. De todas as formas, o líder irá ao encontro o objeto guiando-se pela intensidade do sinal, desviando de obstáculos e retomando a posição tão breve quanto possível. A intensidade do sinal será utilizada como nos outros métodos como critério de conclusão da busca. A figura 4.4 apresenta o diagrama com o método de busca com localização e comunicação, com foco no líder. O recebimento das solicitações de localização e a busca pelo objeto perdido acontecem concomitantemente, embora representado sequencialmente no fluxograma por limitação dessa representação.

4.2.3.2 Demais robôs do enxame

Os robôs do enxame devem conhecer o líder. Como este líder foi selecionado arbitrariamente, essa é uma informação que deve ser de conhecimento prévio entre todo o enxame, e foi feito de forma estática. Esse trabalho não contemplou uma forma dinâmica de identificar o líder

no grupo por almejar uma inicialização mais rápida, embora novas formas de identificação possam ser desenvolvidos em trabalhos futuros.

Uma vez que todos os robôs do enxame conhecem o líder, o primeiro passo para os demais robôs é, através de trocas de mensagens, solicitar ao líder um local e posicionar-se para constituir um grupo unido e iniciar o processo de busca. O líder saberá que a posição foi devidamente aceita e que o robô está ativo no grupo mediante uma mensagem de confirmação. Uma vez posicionados, todos os robôs direcionam-se para a frente buscando pelo sinal do objeto perdido. Quando um obstáculo é detectado o robô desvia dele e perde o alinhamento com os demais robôs, e novamente é solicitado o posicionamento do líder para que se reposicione assim que possível.

Ao detectar o sinal do objeto perdido, o robô obtêm a localização do objeto detectado e informa o líder imediatamente para que esta mensagem possa ser difundida em todo o enxame. Esta abordagem difere-se um pouco da manada de lobos, pois quando um lobo acha uma caça uiva para toda a manada. Para evitar um grande fluxo de mensagens na rede do enxame, optou-se por apenas o líder difundir a mensagem, pois vários robôs podem identificar o objeto alvo em tempos diferentes ou simultâneos.

Após notificar a posição, de forma similar ao robô líder, o robô direciona-se para o local do objeto perdido guiado pela intensidade do sinal até o sinal estar forte e consequentemente perto o suficiente para que seja considerado encontrado. Assim como nos outros métodos os obstáculos são desviados, e a posição é retomada tão breve quanto possível. A figura 4.5 apresenta o fluxograma de ações dos demais robôs do enxame.

Este método foi criado por supor que a comunicação e organização dos robôs do enxame guiadas por um líder traz maior precisão e aumenta a chance de encontrar o objeto perdido em ambientes maiores, do que a dispersão dos robôs que atuam autonomamente. A desvantagem esperada é a lentidão em relação aos outros métodos para a coordenação inicial, o que pode impactar significantemente em ambientes pequenos.

As expectativas desses dois últimos métodos baseiam-se na presença de um componente que obtêm a posição exata, quando possível, do objeto a ser encontrado. Embora não esteja na delimitação dessa pesquisa, não considerar tal componente nesses métodos impactaria no

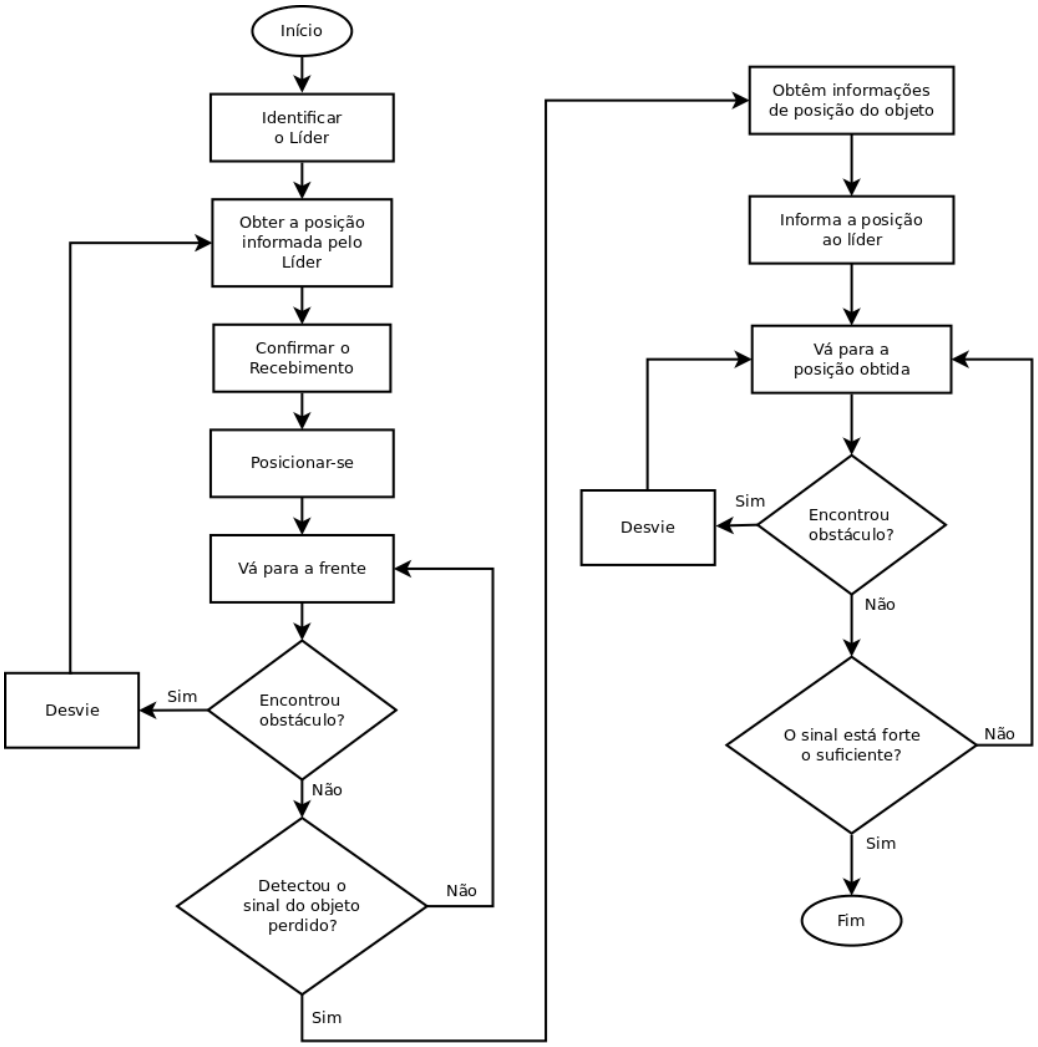

Figura 4.5: Interação entre os demais robôs do enxame na busca do objeto alvo.

processo de busca da seguinte forma: os robôs seguiriam em posições arbitrárias guiando-se pela leitura da intensidade do sinal e tomando como base métodos de tentativa e erro, por não possuírem visão computacional e apenas tentar movimentar-se mediante a leitura dos sinais encontrados. Assim, pressupõe-se que sem o dispositivo de localização o tempo de encontrar o objeto seja maior, independente do tamanho do cenário, da área de busca ou da quantidade de robôs.

CAPÍTULO 5

SIMULAÇÕES E RESULTADOS

Para a realização deste trabalho, foram pesquisados vários simuladores e interfaces voltados para a robótica, que possibilitem a inserção de robôs móveis e sensores (apêndice H).

5.1 Player/Stage

Com base nos simuladores pesquisados, foi escolhido a interface/simulador Player/Stage por permitir a simulação de sinal *WiFi* e possuir vários sensores que ajudam na construção de um robô [47], necessário para esse trabalho. Várias versões do Player/Stage foram testadas em diversas distribuições e versões do kernel Linux devido a incompatibilidade das bibliotecas necessárias para o funcionamento básico e suporte das adaptações realizadas. Após os testes, foram selecionadas as versões 3.0.2 do Player, 3.2.2 do Stage, GNU/Linux Ubuntu Nathy 11.04, kernel 2.6.38-8-generic e gcc version 4.5.2.

O simulador Player/Stage possui três principais arquivos, utilizados para a construção da simulação:

- .world: nestes arquivos são feitas as definições do tamanho da janela de simulação, da imagem que será utilizada como o ambiente, e da descrição do robô, dos seus sensores e, se possuir *WiFi*, do modelo da *WiFi* que será utilizado, assim como o IP, MAC e ESSID. Os apêndices A e B apresentam maiores detalhes para a construção do ambiente e o apêndice C apresenta o arquivo *world* usado na simulação desse trabalho.

- .inc: são arquivos de inclusão que descrevem os modelos a serem utilizados na construção do ambiente, como por exemplo a descrição de um robô ou a descrição de um mapa que pode ser utilizado em um ou mais arquivos .world. O robô utilizado nesta pesquisa é um modelo do Pioneer-2DX, contendo 2 rodas, 8 sonares frontais

com a visão de 0 até 5 metros e ângulo de 15°, além de possuírem um modulo *WiFi* para comunicação. O apêndice D apresenta o modelo utilizados nesse trabalho.

- .cfg: é o arquivo de configuração que contém os parâmetros dos robôs para o Player. Este arquivo informa ao Player todos os drivers usados pelos robôs no ambiente e define como será a interação dos robôs, na simulação ou em robôs reais, com os softwares externos. Este software externo é um programa que pode ser escrito nas linguagens de programação C, C++, Java, entre outros e que conecta com o player por meio de portas e sockets de comunicação criadas pelo próprio simulador. Além disso, quando um driver é inserido em um modelo no world, este deve ser especificado no Player para que haja uma identificação e dessa forma possa enviar e receber dados através do socket. O apêndice E mostra os arquivos de configuração utilizados.

Os drivers são códigos específicos que interagem diretamente com o hardware, para cada tipo de hardware ou modelo de hardware é necessário um driver diferente. No Player/Stage os drivers já estão prontos, bastando ao usuário somente adicionar ao robô. Os drivers podem produzir informações que podem ser visualizadas através de uma interface, esta interface pode enviar e receber informações a partir do Player. Um dispositivo é um driver vinculado a uma interface, dessa forma, o Player pode conversar diretamente com o dispositivo.

5.1.1 Adaptações no Player/Stage

Com as versões utilizadas do Player/Stage, o módulo *WiFi* teve que ser adaptado para que funcionasse corretamente com o método raytrace usado na simulação, bem como o uso do ESSID, MAC e do IP. Para esta adaptação, foram consultados diferentes fóruns e listas de emails, além de próprias correções oferecidas pelos desenvolvedores no site oficial do simulador.

O módulo *WiFi*, na versão do simulador utilizada, não está corretamente disponibilizado e em muitos métodos haviam apenas os blocos de códigos a serem feitos "TODO". Assim como a correção anterior, foi utilizado um *patch* indicado por um dos autores do

Player/Stage em listas de discussão, além da substituição de dois arquivos libstage/communication.cc e examples/ctrl/wander_wifi.cc. As principais alterações foram feitas no arquivo model_wifi, libstage/world.cc, libstage/CMakeLists.txt, libstage/stage.hh, libstage/typetable.cc, config.h.in, worlds/wifi.world, CMakeLists.txt e examples/ctrl/CMakeLists.txt.

O arquivo libstage/model_wifi.cc foi praticamente todo escrito para criar o modelo *WiFi*, as principais implementações são:

- Criação dos cinco modelos de propagação de rádio:

  ◦ Simple *WiFi* Model: este é o modelo mais simples e deve ser especificado somente o raio de propagação de rádio, do sinal *WiFi*.

  ◦ Friis Outdoor Model: esse modelo simula a perda de percurso para cada link, por isso, é usado em ambientes de espaço livre. Possui dois parâmetros obrigatórios: power (dbm) e sensibilidade (dbm).

  ◦ ITU Indoor Propagation Model: esse modelo é usado em ambientes fechados, pois estima a perda de percurso, para isto, é feita uma análise das definições do ambiente e o cálculo do coeficiente da distância e da perda de potência.

  ◦ Log Distance Path Loss Model: esse modelo calcula a perda total do caminho dentro de um edifício, para isso, é usado o expoente da distância e da perda de percurso (PLE), que varia entre 1,8 e 10 para acomodar diferentes configurações ambientais. Uma variável aleatória gaussiana com média zero e desvio padrão sigma é usada para refletir sombra desvanecimento.

  ◦ Simple Raytracing Model: este modelo é usado para refletir obstáculos de bloqueio de wi-fi dentro do raio de alcance da *WiFi*. Neste modelo é necessário especificar o $wall\_factor$, que simula a dificuldade do sinal *WiFi* em penetrar obstáculos.

- Implementação dos atributos:

  ◦ IP (*Internet Protocol* – Protocolo da Internet): endereço IP correspondente ao robô. Cada robô deve possuir um endereço IP.

- MAC (*Media Access Control* – controle de acesso à mídia): endereço MAC correspondente ao robô. O endereço MAC é o endereço físico do robô, por isso, cada robô deve possuir um endereço MAC diferente.

- ESSID (*Extended Service Set Identification* – Conjunto de Serviço de Identificação de Extensão): é um tipo de SSID (*Service Set Identification* – Conjunto de serviço de identificação). Existem dois tipos de SSID: o BSSID (*Basic Service Set Identification* – Conjunto de serviço de identificação básico) usado em redes sem fio *ad hoc* (redes que não possuem ponto de acesso) e ESSID usado em redes sem fio que incluem um ponto de acesso. O SSID é responsável por identificar o nome da rede que o robôs estão inseridos.

- freq: define a frequência da operação que, por padrão, é 2450 MHz.

- power: potência de saída do transmissor que, por padrão, é 45 dbm.

- sensitivy: sensibilidade do receptor, por padrão, é definida em -75dbm.

- range: raio de propagação em metros (usado no modelo Simple).

- plc: coeficiente de perda de energia (usado no modelo Indoor ITU).

- ple: coeficiente de perda de energia (usado no modelo Indoor ITU).

- sigma: desvio padrão (usado no modelo Log Distance Path Loss Model).

- range_db: visualização de propagação de ondas de rádio (usado no modelo Simple Raytracing Model).

- wall_factor: reflete a força dos obstáculos (usado no modelo Simple Raytracing Model).

- Retorno de informações da *WiFi* para os vizinhos, quando detectado;

- Representação visual das conexões com uso de linhas de ligação entre os robôs vizinhos, respeitando as configurações dos atributos;

- Cálculo da distância máxima para atenuação e perda da conexão.

Para permitir o uso da criação do modelo *WiFi*, os desenvolvedores do Player/Stage adicionaram o uso da biblioteca Boost e foram necessárias adaptações em diferentes arquivos:

- Arquivo libstage/world.cc: foi adicionado e implementado o método Raytrace de propagação de rádio no ambiente de simulação;
- Arquivo libstage/CMakeLists.txt: foram adicionados arquivos de comunicação e modelo *WiFi* para serem compilados junto ao simulador, além de adicionar o uso da bilioteca *boostRandomLib*;
- Arquivo libstage/stage.hh: foi adicionado o uso da biblioteca BOOST para aleatoridade e implementado os modelos *WiFi*, detecção e as trocas de mensagens entre os vizinhos;
- Arquivo libstage/typetable.cc: foi adicionado uma linha para registrar a criação de modelos *WiFi*;
- Arquivo config.h.in: foi definido valores para uso da biblioteca BOOST;
- Arquivo wifi.world Cria exemplos de ambiente com uso dos tipos de propagação de rádio *WiFi*;
- Arquivo CMakeLists.txt: prepara a compilação para o uso da biblioteca BOOST;
- examples/ctrl/CMakeLists.txt: adiciona exemplos de uso da *WiFi*.

Além das indicações do patch, foram realizadas outras adaptações, elas foram:

- Arquivo libstageplugin/p_driver.cc: Adição de um caso na estrutura de escolha para permitir o uso da *WiFi* pelo Player;
- Arquivo libstageplugin/p_wifi.cc: Foram mapeados do formato Stage para o formato do Player, para que pudessem ser recebidos no software de controle, os seguintes dados:

- Número de vizinhos encontrados
- Informações sobre os vizinhos, contendo: MAC, RSSI, frequência, ESSID, IP e posição.

Houve poucas adaptações para o Player, e estas foram necessárias somente para o funcionamento da Wireless no simulador e adaptação de versão da biblioteca BOOST:

No arquivo server/drivers/wifi/CMakeLists.txt:

- Antes: PLAYERDRIVER_REQUIRE_HEADER (linuxwifi build_linuxwifi linux/wireless.h)
- Depois: PLAYERDRIVER_REQUIRE_HEADER (linuxwifi build_linuxwifi linux/wireless.h sys/socket.h)

No arquivo client_libs/libplayerc++/playerclient.cc:

- Antes: boost::xtime_get(&xt, boost::TIME_UTC);
- Depois: boost::xtime_get(&xt, boost::TIME_UTC_);

E nos seguintes arquivos:

- arquivo libplayerinterface/interfaces: foram criados tipos diferentes de dados para as variáveis já existentes, com objetivo de facilitar o recebimento no Player e no software de controle dos valores em cadeia de caracteres (*String*)
- libplayertcp/playertcp: foram substituídas linhas de retorno no método *WriteClient* do Player que comunica por TCP com o software controle, presentes no apêndice G;

Essas adaptações possibilitaram a criação de rede entre os robôs do enxame e a identificação de um objeto perdido. Vale ressaltar que estas modificações são apenas adaptações para solução dessa pesquisa, e que embora possa contribuir com a comunidade, deve ainda ser melhor estruturada para que atendam demandas genéricas, e além disso, esta programação foge ao escopo da pesquisa, embora seja fundamental para obtenção dos resultados levando em conta os parâmetros necessários.

5.2 Software de Controle

Para controlar e realizar as tomadas de decisões de cada robô, foram construídos 3 softwares de controle, um para cada método de busca: por tentativa e erro, com localização, e com localização e comunicação. Estes softwares foram escritos na linguagem C++ com uso da biblioteca playerc++ e playerclient disponíveis junto com o simulador, para implementar os métodos exatamente como descritos nos fluxogramas 4.2, 4.3, 4.5 e 4.4 para os três respectivos métodos. O software de controle para o método de busca com localização e comunicação foi dividido em duas partes: um para o líder e o outro para os demais robôs do enxame, como mostrado nos fluxogramas.

A identificação do objeto perdido foi implementada através da detecção de um sinal *WiFi*, utilizando o método *Raytrace*, e a leitura do seu ESSID. Todos os robôs do enxame estão em uma mesma rede ESSID, e dessa forma é possível identificar um elemento não conhecido na rede e classificá-lo como o objeto perdido.

A identificação dos obstáculos e seu desvio é implementada automaticamente pelo próprio Player/Stage ao usar sensores como Laser ou Sonar. A distância entre um robô e o obstáculo pode ser obtida pelo métodos disponíveis nas bibliotecas do simulador. Estas bibliotecas permitem controlar o robô através do método *SetSpeed*, informando o grau a virar e a velocidade desejada, úteis para decidir como desviar dos obstáculos.

O RSSI foi implementado através do uso da *WiFi* com o método *Raytrace*. Em todos os métodos de busca os robôs possuem um dispositivo *WiFi* capaz de ler qualquer sinal *WiFi* de outros robôs ou dispositivos na simulação, independente se há comunicação ou não entre eles. Quando um robô entra no raio de alcance dos seus vizinhos ele conhece o nível de intensidade de sinal, dado em decibéis, bem como os vizinhos dele através do método *GetLinkLevel*, presente também na biblioteca do simulador. Quando o sinal encontrado é do objeto perdido, este valor passa a ser monitorado para que o robô aproxime-se do objeto perdido, conforme explicado no capítulo 4.

A obtenção da posição dos robôs e do objeto perdido na simulação foi implementada pelos métodos *GetXPos* e *GetYPos* da biblioteca do simulador, a qual retorna valores inteiros que identificam precisamente o objeto na área da simulação. De forma similar,

o método *GoTo*, também da bilioteca do simulador, traça um plano para que os robôs dirijam-se diretamente para a posição informada, através de coordenadas X e Y.

A identificação do líder foi implementada estaticamente, ou seja, no *software* de controle dos demais robôs existe uma constante que o identifica, e no *software* de controle do líder existe um vetor com informações dos demais robôs do enxame. A comunicação entre os robôs do enxame e o líder foi implementada com o uso de *socket*, em portas diferentes. Para que o líder e os robôs estejam sempre prontos para enviar e receber mensagens, foram utilizadas *Threads*, as quais tanto o líder quanto os demais robôs estão sempre "ouvindo" e "respondendo" por solicitações ao mesmo tempo que buscam pelo objeto perdido.

No método de busca com localização e comunicação, é realizado um posicionamento de todos os robôs em relação ao líder. Este alinhamento é realizado da seguinte forma:

- Através de trocas de mensagens os robôs solicitam o posicionamento para o líder, o líder sabe qual robô está ativo no enxame e obtém o seu identificador;
- O líder realiza uma proporção entre o número de robôs do enxame e a área em que o enxame atuará na busca (estes são parâmetros informados na execução do *software* de controle);
- O líder informa o mesmo eixo Y em que está para todos os robôs, alterando apenas o valor para o eixo X;
- Cada robô recebe uma posição diferente para o eixo X e se alinha a partir da posição do líder, respeitando o limite máximo para que não haja desconexões.
- O líder controla através do valor da razão, diminuindo-o quando necessário, para que o robô mais distante não exceda 100 metros, a fim de não perder a conexão entre eles.

Quando um robô do enxame encontra o objeto perdido ele comunica o líder, e ao receber, o líder realiza uma difusão de mensagens para todos os demais robôs do enxame. Se o próprio líder detecta o perdido, também realiza o mesmo procedimento. Nessa mensagem contém informações de posição do objeto perdido para que todo o enxame

possa ir ao encontro desse objeto guiado pela intensidade do sinal, além das informações de posição, com objetivo de encontrar o mais rápido possível.

5.3 Parâmetros Usados

Foram construídas áreas com tamanhos diferentes, raio de alcance e quantidade de robôs variados. As áreas construídas são: 10 x 10 que corresponde a 100 metros quadrados; 50 x 50 que corresponde a 2.500 metros quadrados; 80 x 80 que corresponde a 6.400 metros quadrados; 100 x 100 que corresponde a 10.000 metros quadrados; 500 x 500 que corresponde a 250.000 metros quadrados; 1000 x 1000 que corresponde a 1.000.000 metros quadrados e; 1500 x 1500 que corresponde a 2.250.000 metros quadrados.

Na primeira área, 10 x 10, os robôs possuem um raio de alcance de 10 metros para sinal *WiFi* do enxame, nas demais áreas, o raio de alcance dos robôs é de 100 metros para comunicação *WiFi*, esses parâmetros foram escolhidos para oportunizar a perda de conexão durante os testes. Para a área de 10 x 10 metros, foram realizados testes com um enxame de 5 robôs. Para as áreas de 50 x 50 e 80 x 80 metros, foram realizados testes com enxames de 5 e 25 robôs. Para as demais áreas, foram realizados testes com enxames de 5, 25 e 50 robôs.

O parâmetro de posicionamento inicial dos robôs, para todos os cenários, foi: os robôs foram distribuídos proporcionalmente do canto inferior direito para o centro, na área delimitada para a busca. Além disso, a velocidade dos robôs permaneceu a velocidade padrão do player, 0,2 metros por segundo. Para todas as áreas foi construído um modelo de ambiente com vários obstáculos simulando um ambiente real. As imagens 5.1, 5.2 e 5.3 mostram os ambientes e as áreas com tamanhos variados.

Para passar alguns parâmetros para os robôs (como por exemplo determinar o posicionamento, sinais recebidos e a sua intensidade, informações do robô e de seus vizinhos) foi desenvolvido um programa de controle que se comunica com o simulador interagindo diretamente com cada robô, recebendo informações de sensores e enviando para os atuadores, através do uso de sockets.

O programa de controle fica continuamente lendo os dispositivos de cada robô: o

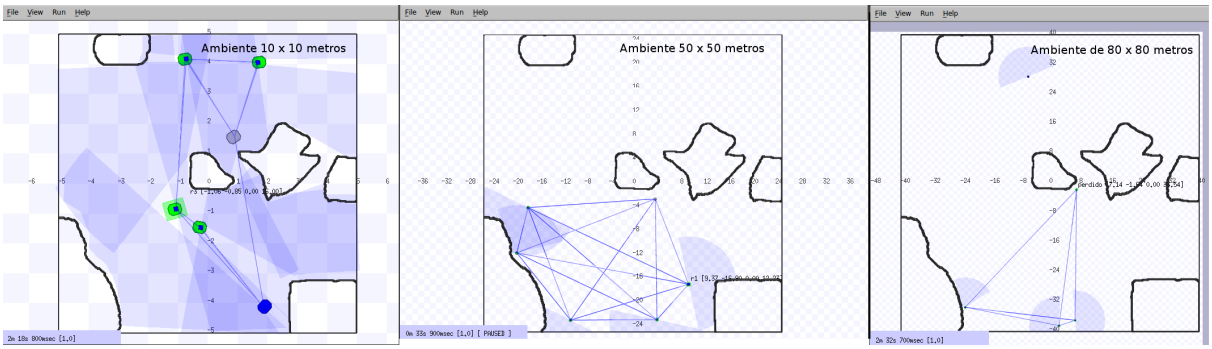

Figura 5.1: Imagem dos ambientes de 10, 50 e 80 metros quadrados.

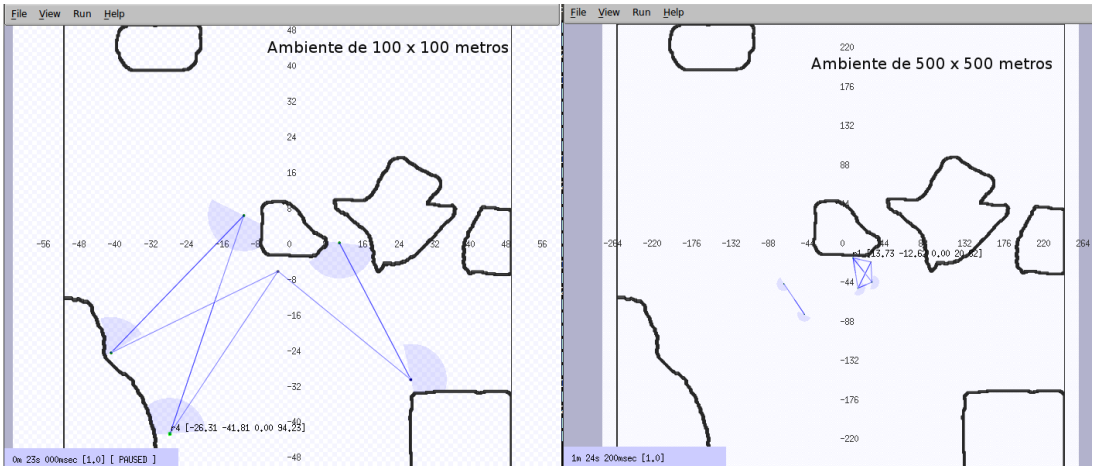

Figura 5.2: Imagem dos ambientes de 100 e 500 metros quadrados.

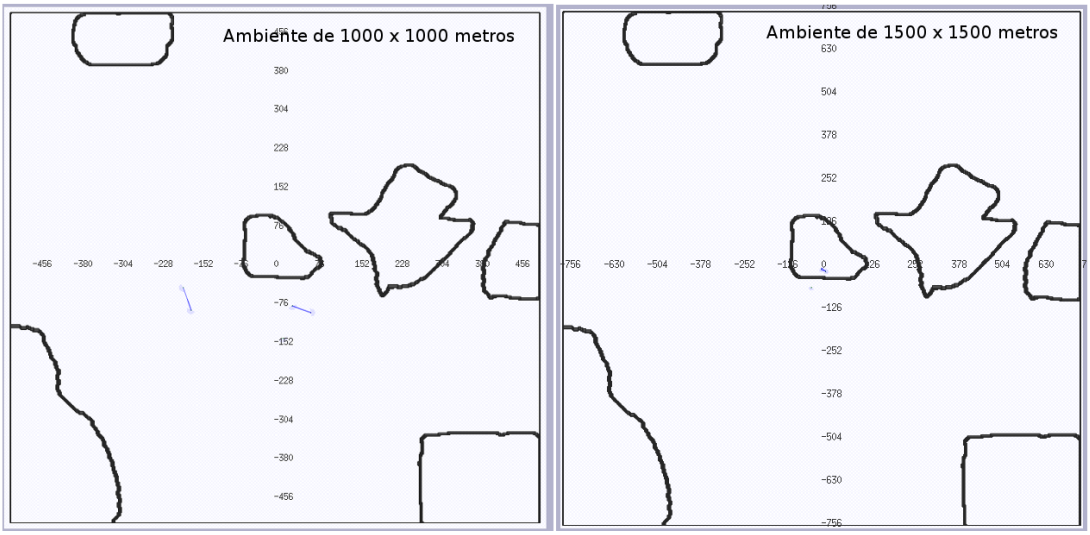

Figura 5.3: Imagem dos ambientes de 1000 e 1500 metros quadrados.

sensor de distância, o *blob finder* e o leitor *WiFi*; cada um desses dispositivos monta um proxy com uma porta, e na implementação do software controle, cada dispositivo desse pode interagir com o simulador através dessas portas de tal forma que, para cada robô

do simulador, é possível obter informações separadas de cada sensor bem como enviar informações para os atuadores.

Assim, o software de controle fica constantemente lendo a intensidade do sinal. Nesses sinais pode-se obter algumas informações, para a execução desse trabalho foi lido a intensidade de sinal da *WiFi*. Os robôs estão na mesma *WiFi* e assim conhecem o ESSID dessa rede, ao perceber um sinal de rede *WiFi* que não seja o seu (ESSID diferente), esse sinal não conhecido é interpretado como o alvo a ser buscado. O ESSID foi escolhido por questões de implementação, mas poderia ser facilmente substituído pelo MAC, no entanto, o objetivo é identificar uma rede diferente da rede a qual pertencem os robôs do enxame.

5.4 Resultados

Com o modelo de aplicação construído foram realizados vários testes em ambientes com tamanhos diferentes e com vários obstáculos simulando um ambiente real. Cada configuração de teste foi executada entre 15 e 20 vezes, foram excluídos o maior e o menor valor, a fim de alcançar o intervalo de confiança de 90%.

Para a obtenção dos resultados, foi feita uma comparação entre os três métodos testados em cada ambiente e para cada quantidade de robôs. O desvio padrão foi calculado com base em todos os resultados dos diversos testes para cada ambiente e quantidade de robôs.

5.4.1 Ambiente de 10 x 10 metros

Neste ambiente, foi utilizado apenas 05 robôs (figura 5.4) e o método com comunicação obteve um desempenho melhor do que os demais métodos: utilizou 56,36% do tempo gasto pelo método sem comunicação, ou seja, uma melhora de aproximadamente 77%; e apenas 18,02% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora aproximadamente de 450%.

Essa melhora foi possível pela comunicação dos robôs pois o robô que identifica o objeto perdido, por vezes, pode não ser o mais próximo e com a troca de mensagens, o robô mais próximo encontra-o mais rapidamente, mesmo sem o ter percebido inicialmente.

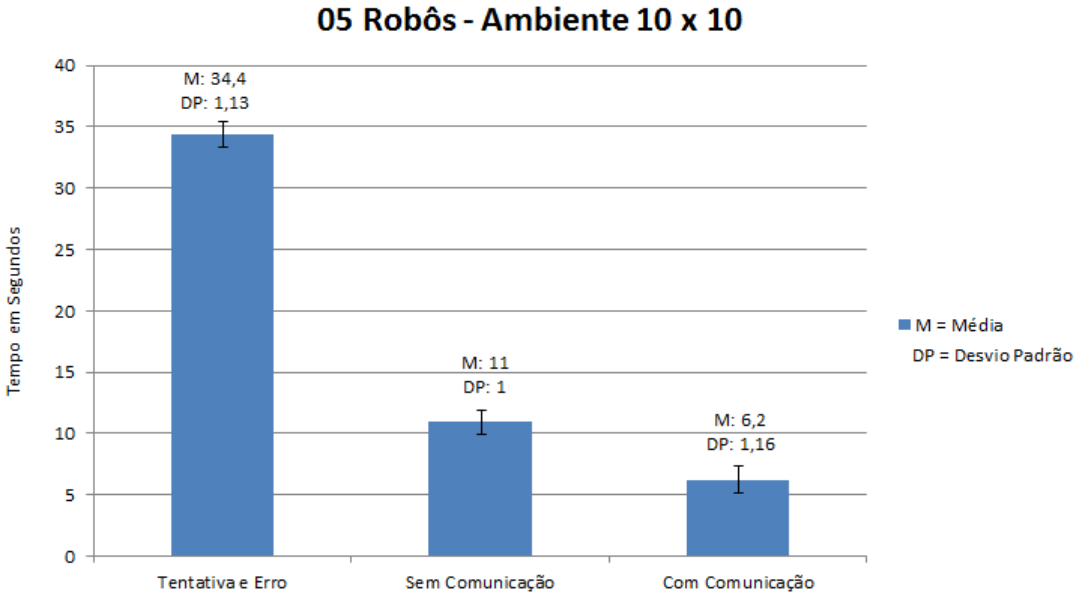

Figura 5.4: Comparação entre os métodos no ambiente de 10 metros quadrados com 05 robôs.

5.4.2 Ambiente de 50 x 50 metros

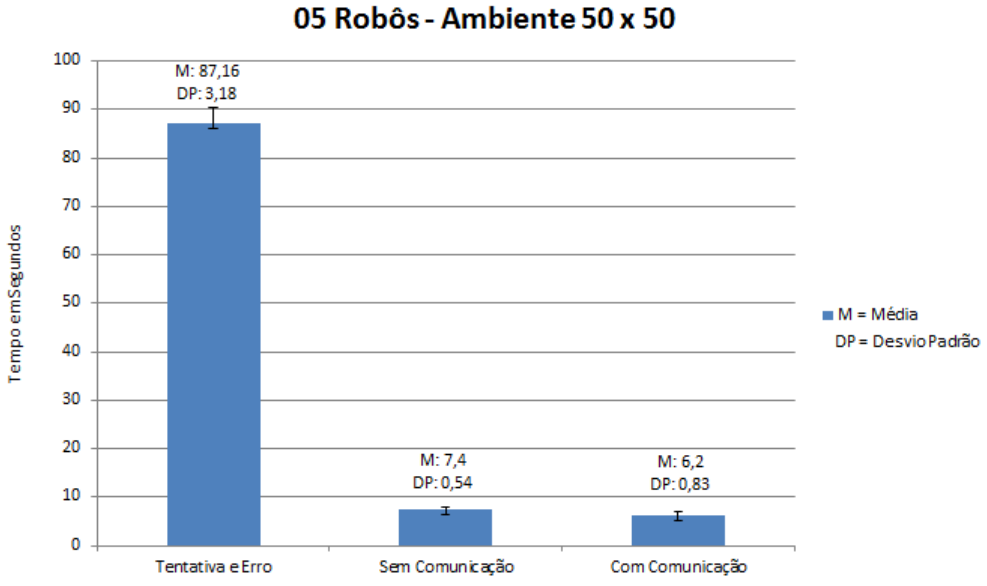

Figura 5.5: Comparação entre os métodos no ambiente de 50 metros quadrados com 05 robôs.

Neste ambiente com 05 robôs (figura 5.5), nota-se que o método com comunicação utilizou 83,78% do tempo gasto pelo método sem comunicação obtendo uma melhora de aproximadamente 19,35%. Em relação ao método de tentativa e erro, o método com comu-

nicação utilizou apenas 7,11% do tempo gasto obtendo uma melhora de aproximadamente 1305%. Assim como no primeiro ambiente, essa melhora foi possível pela comunicação dos robôs.

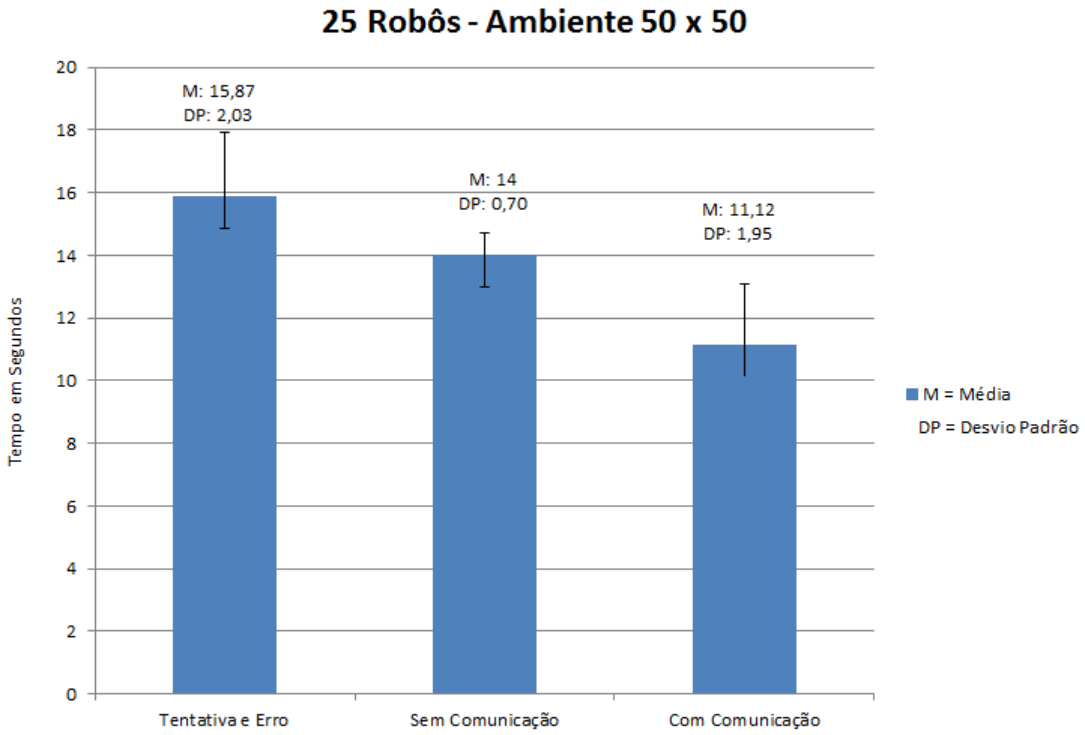

Figura 5.6: Comparação entre os métodos no ambiente de 50 metros quadrados com 25 robôs.

Neste ambiente com 25 robôs (figura 5.6), o método com comunicação obteve um desempenho melhor do que os demais métodos, com as seguintes proporções: utilizou 79,42% do tempo gasto pelo método sem comunicação, ou seja, uma melhora de aproximadamente 25%; e utilizou 70% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora de aproximadamente 42%.

5.4.3 Ambiente de 80 x 80 metros

Neste ambiente com 05 robôs (figura 5.7), o método com comunicação utilizou 63,57% do tempo gasto pelo método sem comunicação obtendo uma melhora de aproximadamente 57%. Em relação ao método de tentativa e erro, o método com comunicação utilizou apenas 4,80% do tempo gasto, obtendo uma melhora de aproximadamente 1982%. Assim como nos ambientes anteriores, essa melhora foi possível pela comunicação dos robôs.

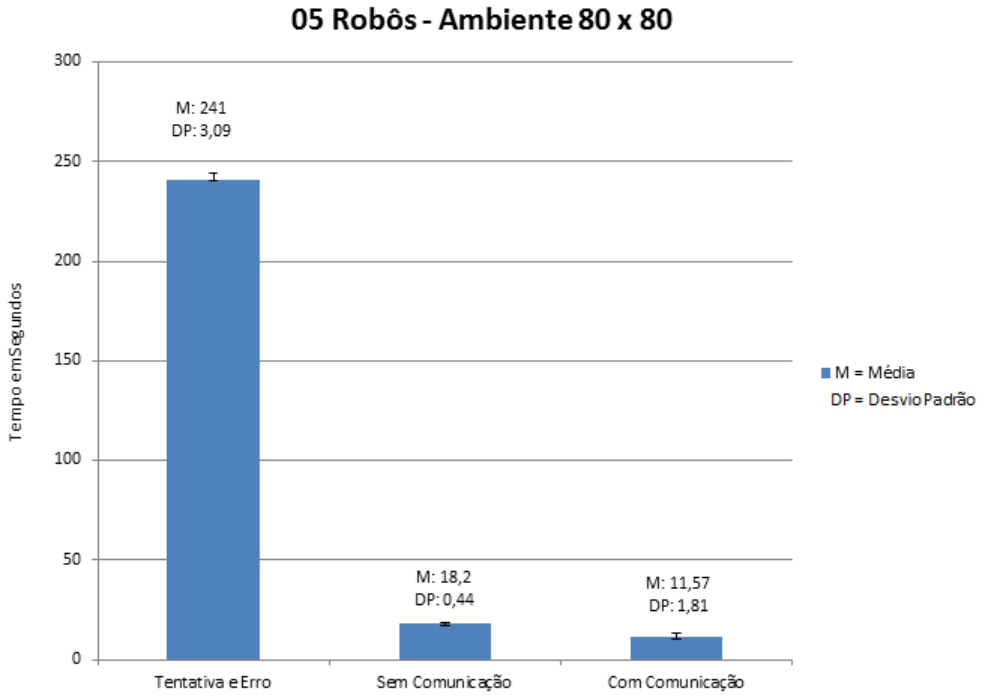

Figura 5.7: Comparação entre os métodos no ambiente de 80 metros quadrados com 05 robôs.

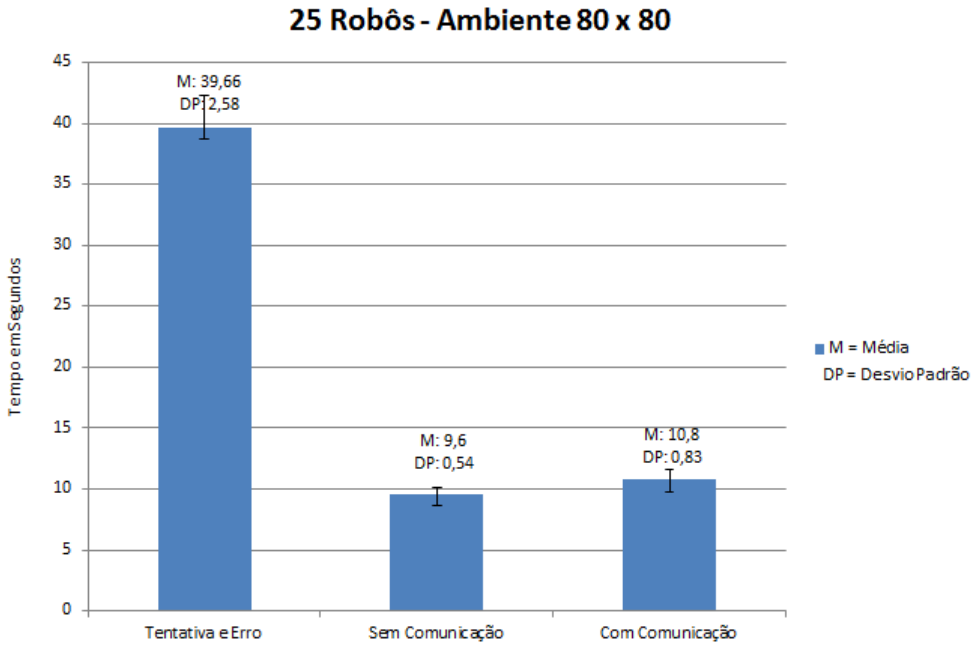

Figura 5.8: Comparação entre os métodos no ambiente de 80 metros quadrados com 25 robôs.

Neste ambiente com 25 robôs (figura 5.8), o método sem comunicação obteve um desempenho melhor do que os demais métodos: utilizou 88,88% do tempo gasto pelo método com comunicação, obtendo uma melhora de aproximadamente 12%; e 24,2% do

tempo gasto pelo método de tentativa e erro, obtendo uma melhora de aproximadamente 313%.

Neste ambiente, o desempenho do método com comunicação foi aproximadamente de 1 segundo pior do que o método sem comunicação. Essa diferença de desempenho é justificada pelo fato de que o método com comunicação inicialmente realiza um acordo entre os robôs para o alinhamento do enxame. Enquanto este método realiza o acordo e alinhamento, os demais métodos já estão realizando a busca, e pelo fato do ambiente ser pequeno, o método sem comunicação obteve um desempenho melhor na busca pelo objeto perdido.

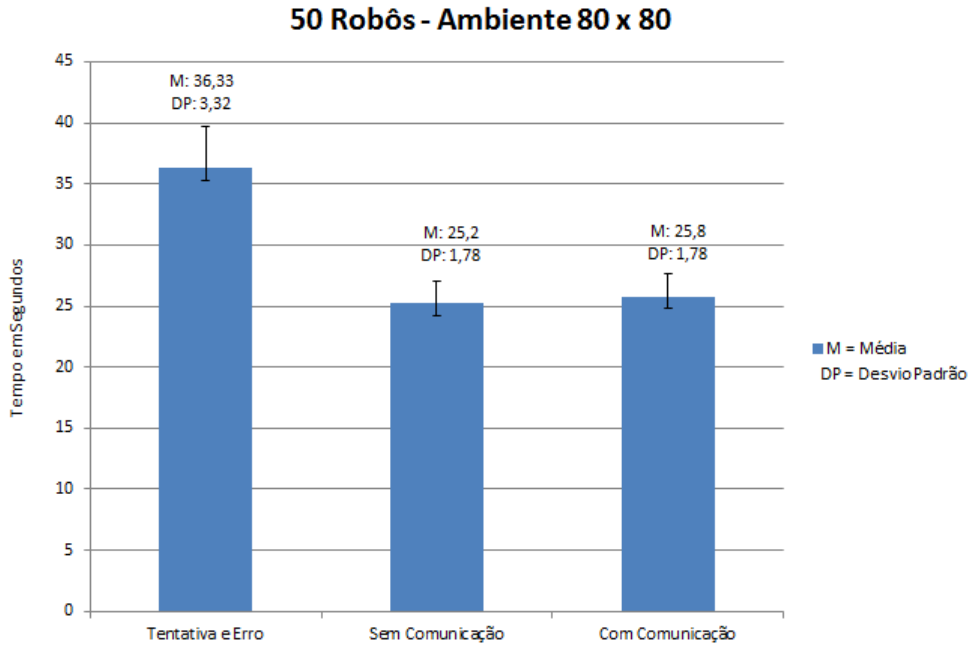

Figura 5.9: Comparação entre os métodos no ambiente de 80 metros quadrados com 50 robôs.

Neste ambiente com 50 robôs (figura 5.9), o método sem comunicação obteve um desempenho melhor do que os demais métodos, nas seguintes proporções: utilizou 97,67% do tempo gasto pelo método com comunicação, obtendo uma melhora de aproximadamente 2%; e utilizou 69,36% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora de aproximadamente 44,16%. Assim como no ambiente de 80 x 80 metros e com 25 robôs, esta diferença do método com comunicação de quase 1 segundo pior do que o método sem comunicação é justificada pelo acordo e alinhamento entre os robôs do

enxame.

5.4.4 Ambiente de 100 x 100 metros

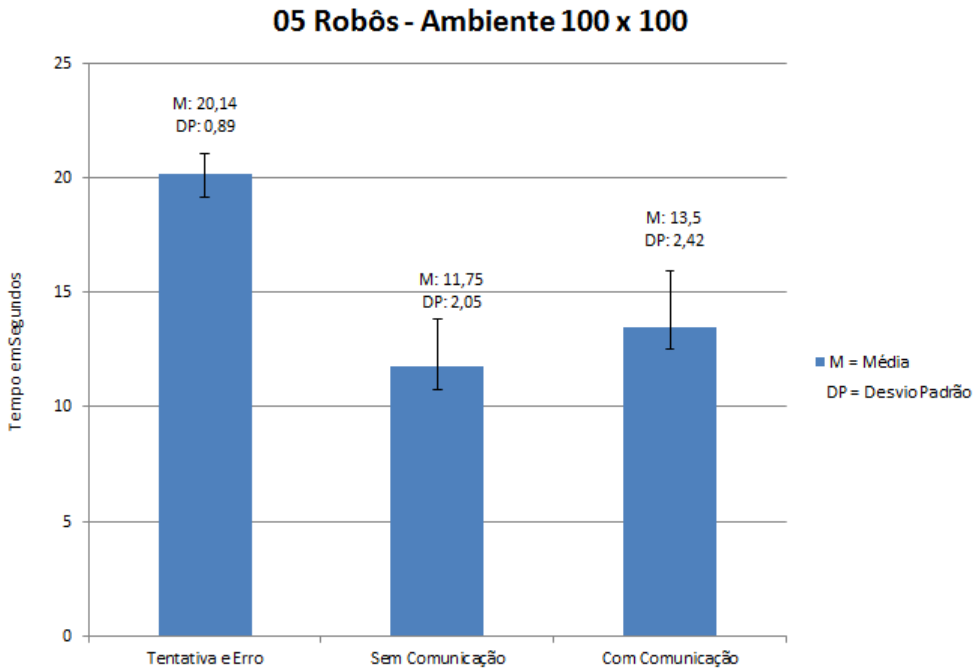

Figura 5.10: Comparação entre os métodos no ambiente de 100 metros quadrados com 05 robôs.

Neste ambiente com 05 robôs (figura 5.10), o método sem comunicação obteve um desempenho melhor do que os demais métodos: utilizou 87% do tempo gasto pelo método com comunicação, obtendo uma melhora de aproximadamente 14%; e utilizou 58,34% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora de aproximadamente 71%. Assim como no ambiente de 80 x 80 metros com 25 e 50 robôs, esta diferença de desempenho do método com comunicação para o método sem comunicação é justificada pelo acordo e alinhamento entre os robôs do enxame.

Neste ambiente com 25 robôs (figura 5.11), os métodos de tentativa e erro e sem comunicação obtiveram o mesmo desempenho, utilizando 81,73% do tempo gasto pelo método com comunicação e obtendo uma melhora de aproximadamente 22%. Assim como nos ambientes anteriores, esta diferença de desempenho é justificada pelo acordo e alinhamento entre os robôs do enxame.

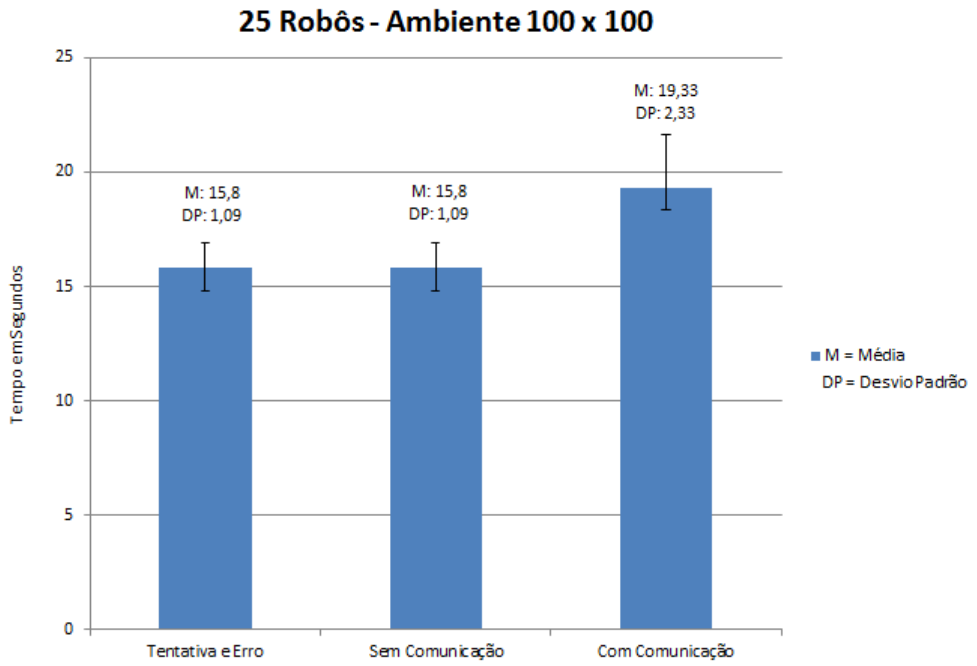

Figura 5.11: Comparação entre os métodos no ambiente de 100 metros quadrados com 25 robôs.

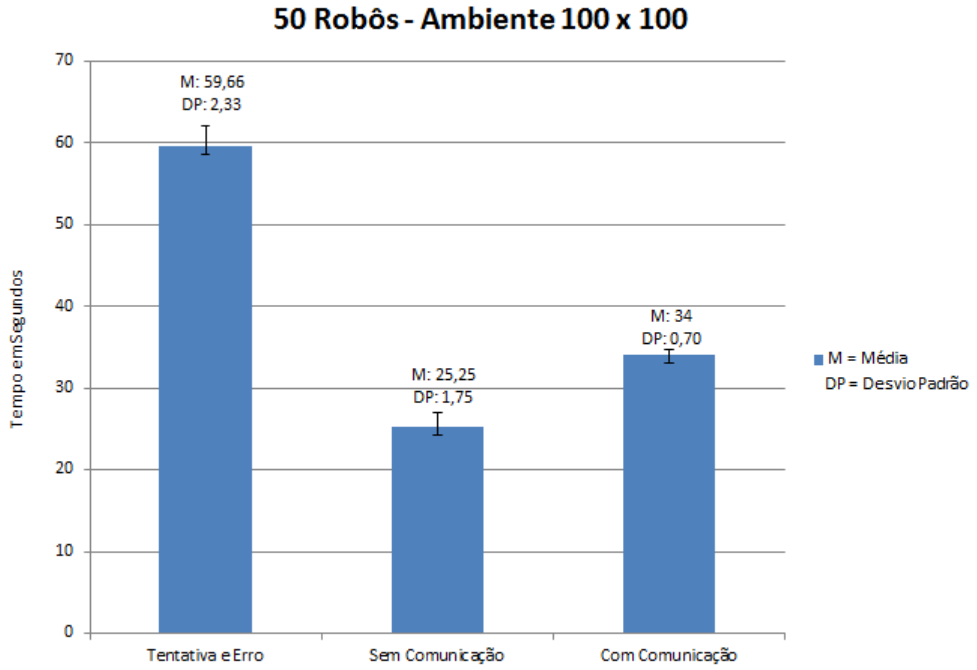

Figura 5.12: Comparação entre os métodos no ambiente de 100 metros quadrados com 50 robôs.

Neste ambiente com 50 robôs (figura 5.12), o método sem comunicação teve um desempenho melhor do que os demais métodos: utilizou 74,26% do tempo gasto pelo método com comunicação, obtendo uma melhora de aproximadamente 34%; e 42,32% do

tempo gasto pelo método de tentativa e erro, obtendo uma melhora de aproximadamente 136,27%. Esta diferença de desempenho do método com comunicação para o método sem comunicação é justificada pelo acordo e alinhamento entre os robôs do enxame.

5.4.5 Ambiente de 500 x 500 metros

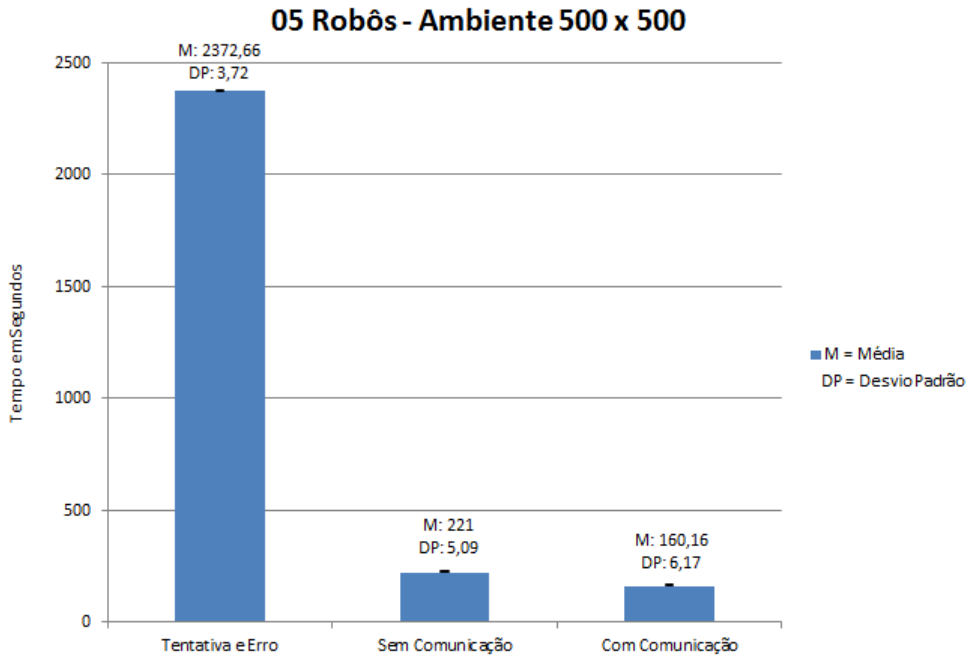

Figura 5.13: Comparação entre os métodos no ambiente de 500 metros quadrados com 05 robôs.

Neste ambiente com 05 robôs (figura 5.13), o método com comunicação utilizou 72,47% do tempo gasto pelo método sem comunicação, obtendo uma melhora de aproximadamente 38%. Em relação ao método de tentativa e erro, o método com comunicação utilizou apenas 6,75% do tempo gasto, obtendo uma melhora de aproximadamente 1.381%.

No ambiente de 500 metros quadrados, por ser maior do que os ambientes anteriores, o acordo inicial entre os robôs do enxame e o alinhamento não prejudicam o desempenho da busca realizada. Ao contrário dos últimos cinco resultados (figuras 5.8, 5.9, 5.10, 5.11 e 5.12),o uso da comunicação entre os robôs torna a busca mais rápida do que os demais métodos.

Neste ambiente com 25 robôs (figura 5.14), o método com comunicação teve um desempenho melhor do que os demais métodos: utilizou 91,88% do tempo gasto pelo

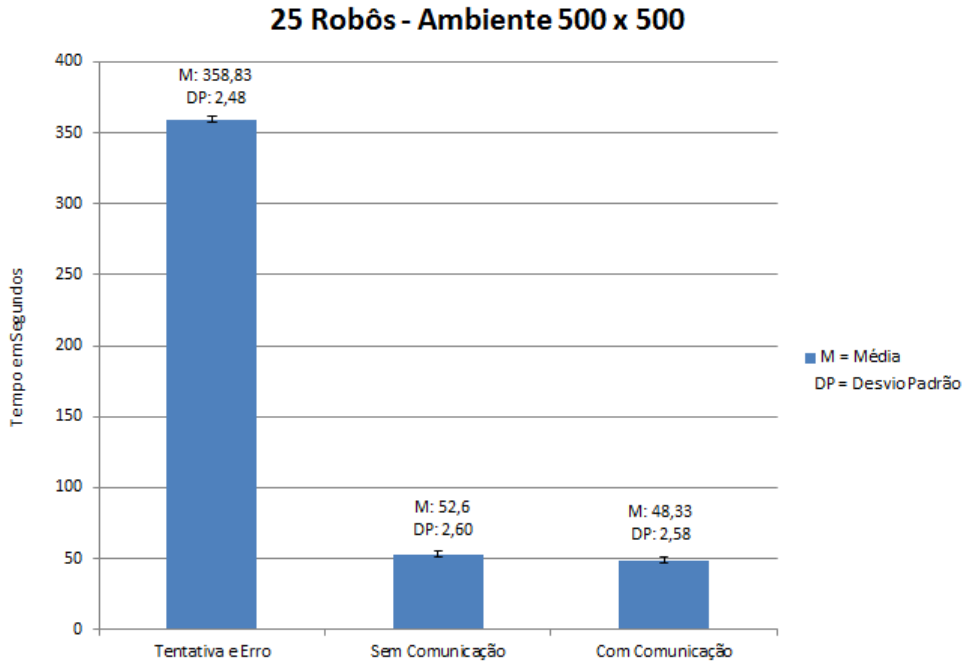

Figura 5.14: Comparação entre os métodos no ambiente de 500 metros quadrados com 25 robôs.

método sem comunicação, ou seja, uma melhora de aproximadamente 8%; e apenas 13,46% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora aproximadamente de 642%.

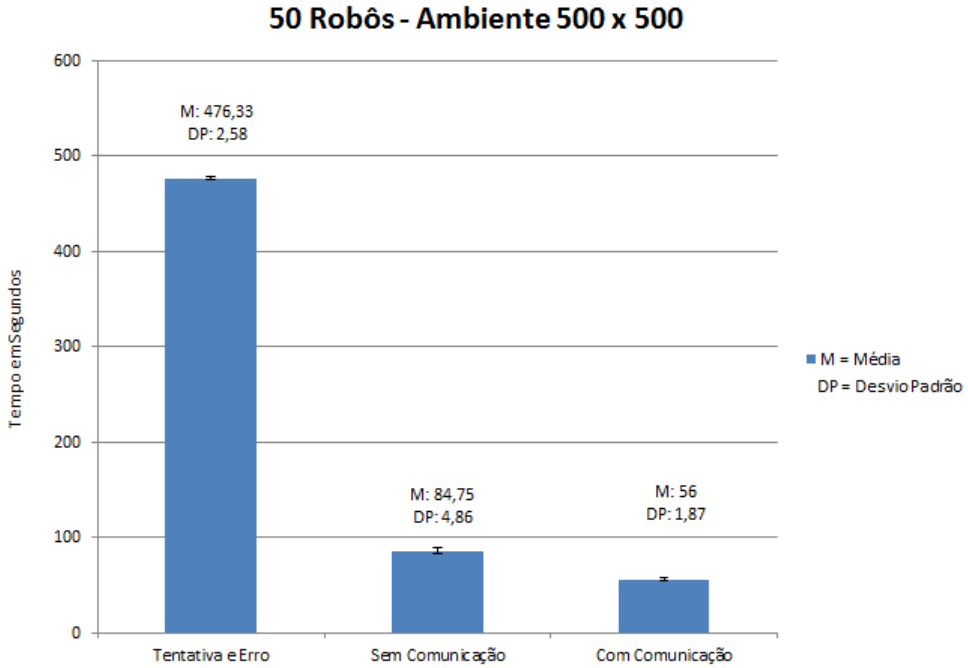

Figura 5.15: Comparação entre os métodos no ambiente de 500 metros quadrados com 50 robôs.

Utilizando 50 robôs (figura 5.15), o método com comunicação utilizou 66% do tempo gasto pelo método sem comunicação obtendo uma melhora de aproximadamente 51%. Em relação ao método de tentativa e erro, o método com comunicação utilizou apenas 11,75% do tempo gasto, obtendo uma melhora de aproximadamente 750%.

5.4.6 Ambiente de 1000 x 1000 metros

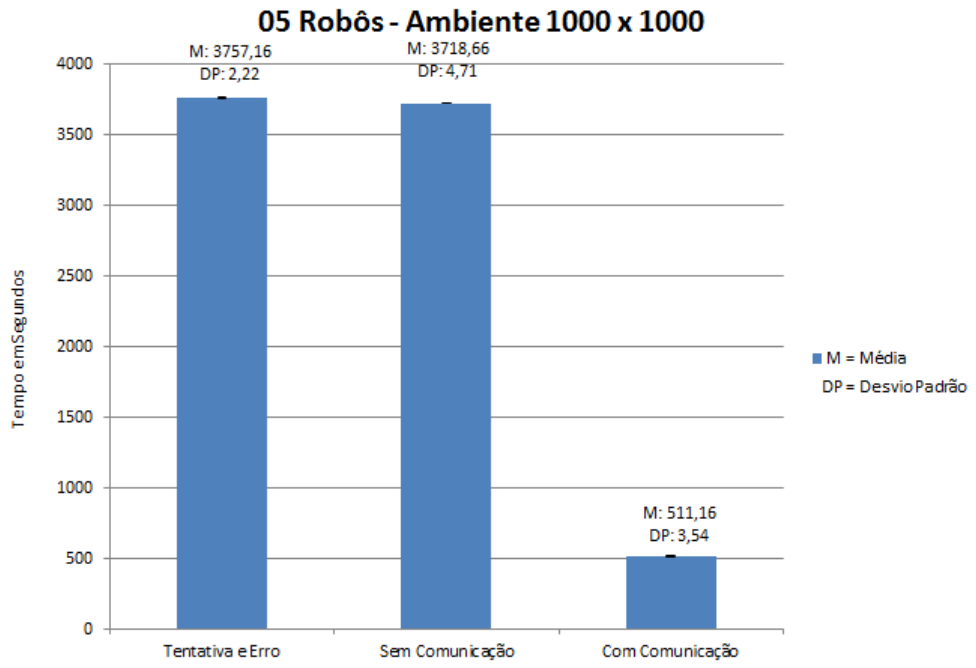

Figura 5.16: Comparação entre os métodos no ambiente de 1000 metros quadrados com 05 robôs.

Neste ambiente com 05 robôs (figura 5.16), o método com comunicação obteve um desempenho melhor do que os demais métodos, com as seguintes proporções: utilizou 13,74% do tempo gasto pelo método sem comunicação, ou seja, uma melhora de aproximadamente 627%; e apenas 13,60% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora aproximadamente de 635%.

Neste teste, houve um salto grande comparando com o ambiente anterior de 500 x 500, isto se deu devido ao tamanho total da área. Pelo fato do objeto perdido estar sempre na mesma posição (atrás de pedras), os robôs do enxame demoraram mais para encontrá-lo.

Neste ambiente com 25 robôs (figura 5.17), o método com comunicação utilizou 38,78% do tempo gasto pelo método sem comunicação, obtendo uma melhora de aproximadamente

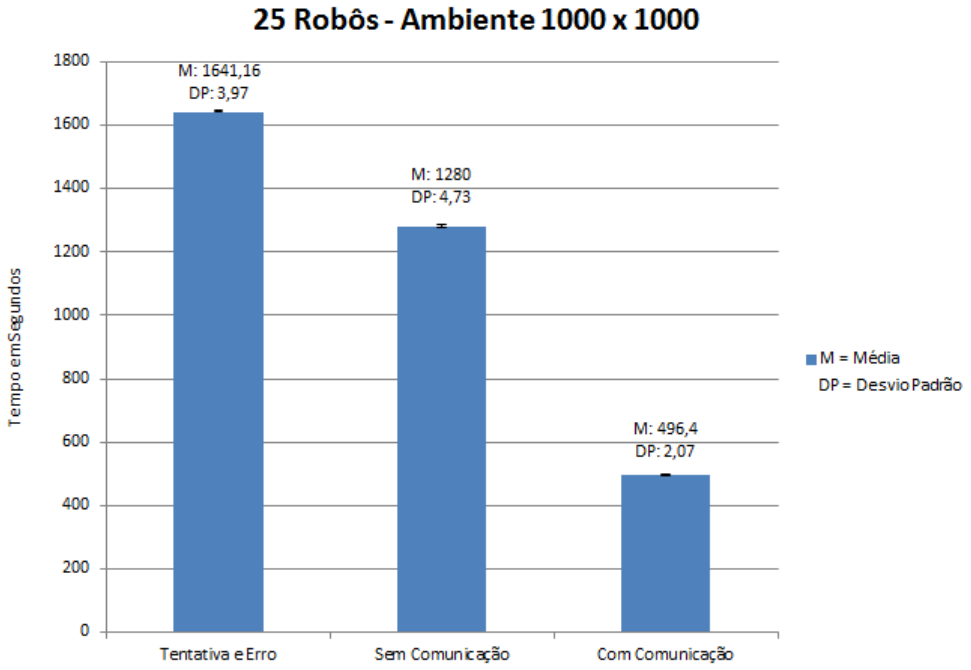

Figura 5.17: Comparação entre os métodos no ambiente de 1000 metros quadrados com 25 robôs.

157%. Em relação ao método de tentativa e erro, o método com comunicação utilizou 30,24% do tempo gasto, obtendo uma melhora de aproximadamente 230%.

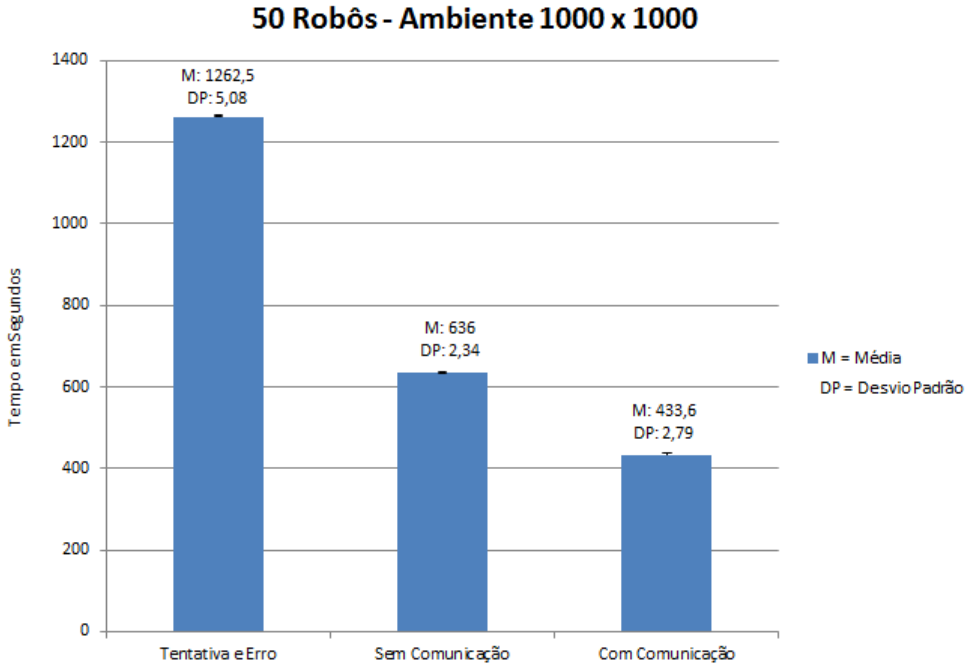

Figura 5.18: Comparação entre os métodos no ambiente de 1000 metros quadrados com 50 robôs.

Neste ambiente com 50 robôs (figura 5.18), o método com comunicação teve um

desempenho melhor do que os demais métodos: utilizou 68,17% do tempo gasto pelo método sem comunicação, ou seja, uma melhora de aproximadamente 46%; e apenas 34,34% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora aproximadamente de 191%.

5.4.7 Ambiente de 1500 x 1500 metros

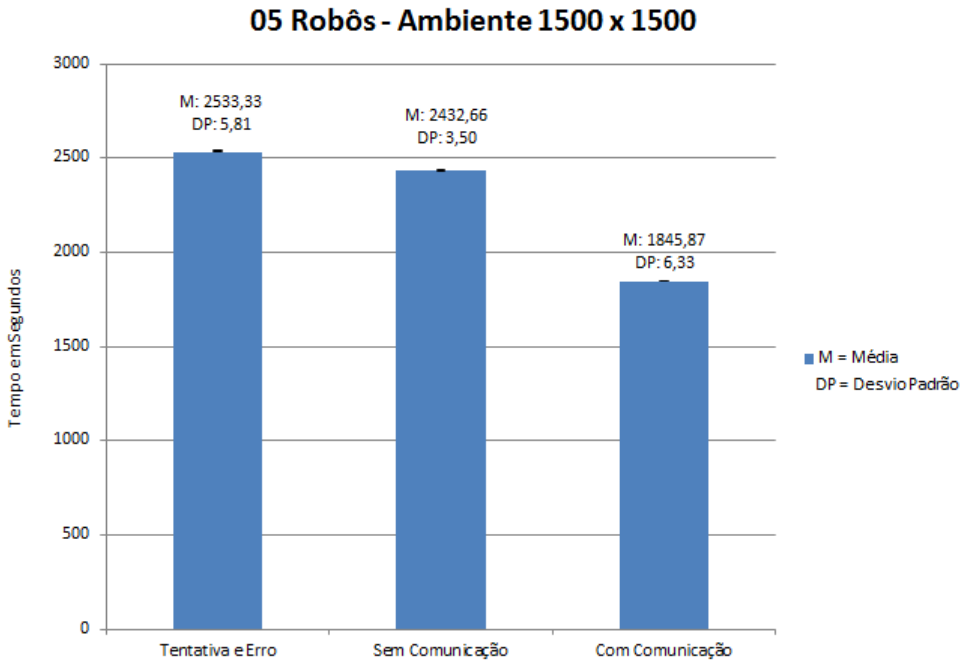

Figura 5.19: Comparação entre os métodos no ambiente de 1500 metros quadrados com 05 robôs.

Neste ambiente com 05 robôs (figura 5.19), o método com comunicação utilizou 75,87% do tempo gasto pelo método sem comunicação, obtendo uma melhora de aproximadamente 31%. Em relação ao método de tentativa e erro, o método com comunicação utilizou 72,86% do tempo gasto, obtendo uma melhora de aproximadamente 37%.

Neste ambiente com 25 robôs (figura 5.20), o método com comunicação teve um desempenho melhor do que os demais métodos, com as seguintes proporções: utilizou 55,18% do tempo gasto pelo método sem comunicação, ou seja, uma melhora de aproximadamente 81%; e apenas 27,25% do tempo gasto pelo método de tentativa e erro, obtendo uma melhora aproximadamente de 266%.

Neste ambiente com 50 robôs (figura 5.21), o método com comunicação utilizou 51,26%

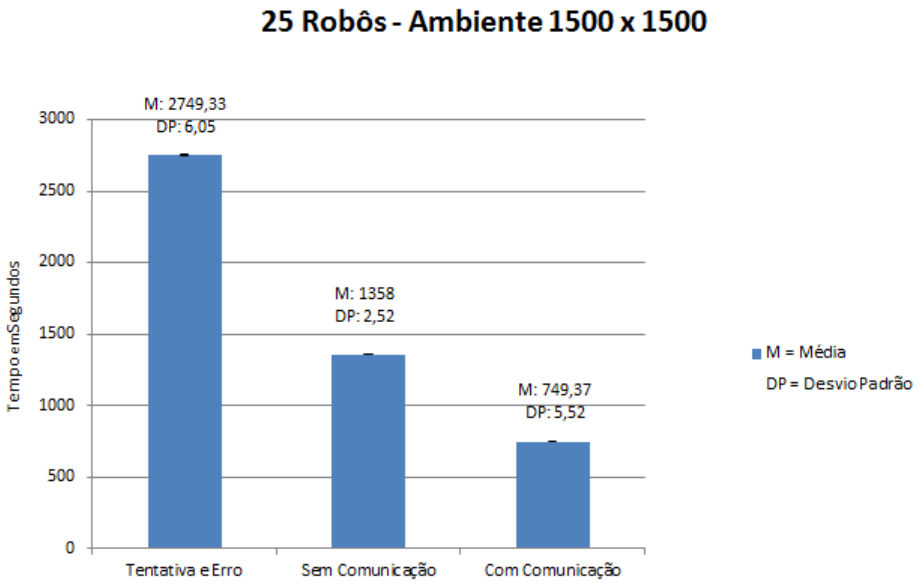

Figura 5.20: Comparação entre os métodos no ambiente de 1500 metros quadrados com 25 robôs.

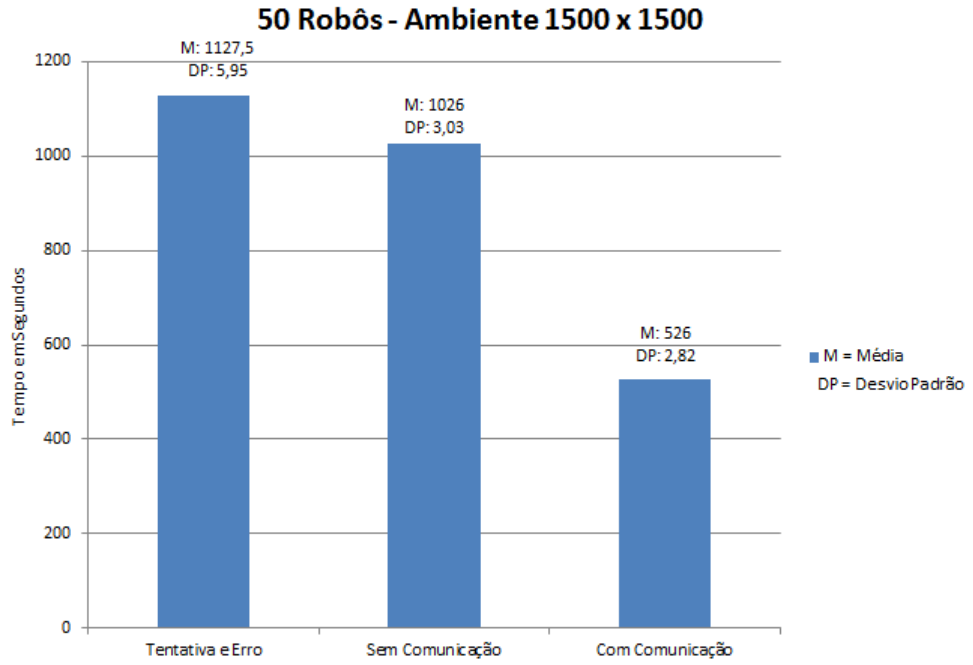

Figura 5.21: Comparação entre os métodos no ambiente de 1500 metros quadrados com 50 robôs.

do tempo gasto pelo método sem comunicação, obtendo uma melhora de aproximadamente 95%. Em relação ao método de tentativa e erro, o método com comunicação utilizou 46,65% do tempo gasto, obtendo uma melhora de aproximadamente 114%.

5.4.8 Comparação dos resultados dos métodos utilizados

Nota-se que nas figuras 5.8, 5.9, 5.10, 5.11 e 5.12, o método sem comunicação apresentou melhor desempenho do que o método com comunicação. Esta diferença de desempenho aconteceu pela grande quantidade de robôs em relação da área de busca, pois vale ressaltar que o método com comunicação inicialmente realiza um acordo entre todos os robôs do enxame para que se alinhem em relação ao robô líder para posterior varredura de sinal do perdido.

Este tempo necessário para a realização do alinhamento impactou em ambientes menores com uma quantidade razoável de robôs pois enquanto realiza este processo inicial, os outros métodos já estariam quase encontrando ou já teria encontrado o sinal do perdido.

Percebe-se que esta diferença é facilmente superada quando o ambiente de busca aumenta e consequentemente a quantidade de área buscada por cada robô do enxame. A busca nos ambientes de 500, 1000 e 1500 metros, independentemente da quantidade de robôs, o método com comunicação mostrou-se extremamente mais eficiente do que os demais métodos.

Esta eficiência deve-se à comunicação entre os robôs do enxame e o líder, de tal forma que quando qualquer robô do enxame detecta o perdido, informa-o. Ao receber a mensagem, o líder informa todo o enxame para que o robô mais próximo do sinal encontre-o rapidamente, independente da detecção inicial do objeto perdido.

As figuras 5.22, 5.23 e 5.24 mostram todos os resultados realizados com os métodos tentativa e erro, sem comunicação e com comunicação variando o ambiente e a quantidade de robôs, considerando a média do tempo gasto para cada teste e a taxa de erros dos testes.

Com os resultados dos três métodos podemos concluir que para uma busca eficiente é fundamental se ater à relação tamanho da área x quantidade de robôs, pois uma área pequena com um grande número de robôs ocorre em incessantes desvios de um robô com outro para que não se colidam. Além disso, a comunicação torna-se ineficiente, pois por simples tentativa e erro há maior chance de encontrar o objeto perdido em menor tempo.

Analisando as figuras 5.22, 5.23 e 5.24, os testes realizados com as quantidades de

robôs 05, 25 e 50, percebe-se que na maioria dos resultados os melhores desempenhos foram obtidos por: 05 robôs nas áreas de 10 e 50 metros quadrados; 25 robôs nas áreas de 80, 100 e 500 metros quadrados; e 50 robôs nas áreas de 1000 e 1500 metros quadrados.

A figura 5.23 mostra que no ambiente de 1000 X 1000 com 05 robôs o desempenho do método sem comunicação foi pior do que o desempenho no ambiente de 1500 X 1500 com 05 robôs. Fazendo uma comparação entre os dois ambientes, o ambiente maior obteve um desempenho melhor de aproximadamente 52,86%. Isso se deu pelo fato de que todos os ambientes usaram o mesmo mapa que contêm os obstáculos, e devido a isso, conforme o aumento das áreas os obstáculos distanciam-se uns dos outros, ampliando os espaços para a busca e diminuindo as chances de colisão ou desvio de obstáculos.

A imagem 5.25 mostra os três métodos em tipos de linhas diferentes para melhor visualização: o método de tentativa e erro utiliza uma linha azul; o método sem comunicação utiliza uma linha vermelha; e o método com comunicação utiliza uma linha verde. Os pontos usados foram obtidos através da média dos resultados dos testes (comentados anteriormente).

Neste comparativo entre os três métodos variando o tamanho da área e a quantidade de robôs, percebe-se que: à partir da área de 100 metros quadrados com a quantidade de 50 robôs, o desempenho do método de tentativa e erro foi pior do que os demais métodos; e à partir do área de 500 metros quadrados com a quantidade de 50 robôs, o desempenho do método sem comunicação foi pior do que o método com comunicação.

Em todos os métodos percebeu-se que durante o processo de busca houveram colisões ao tentar desviar de obstáculos, sejam com os próprios obstáculos ou com outros robôs. Em 90% dos casos de colisões os robôs demoraram cerca de 1 segundo para retomar a busca, e nos outros 10% não conseguiram sair da colisão e permaneceram colididos. Embora houveram as colisões, o objetivo do enxame não foi comprometido pois as colisões aconteceram apenas com uma pequena parte do enxame em diferentes momentos durante o processo de busca.

No terceiro método, o alinhamento inicial dos robôs do enxame demorou em média aproximadamente 10 segundos. Este alinhamento ocorre somente na presença do líder e

é estabelecida através da comunicação entre o líder e os demais robôs do enxame, como explicado no capítulo 4.2.3. Embora este tempo seja gasto para realizar o alinhamento, os robôs procuravam o objeto perdido em paralelo para otimização da busca.

Nos ambientes pequenos com os parâmetros 10 x 10 com 05 robôs, 50 x 50 com 05 e 25 robôs e 80 x 80 com 05 robôs, o método com comunicação teve melhor desempenho em relação ao método com localização. Isto se deve por causa do tamanho do ambiente, pois os robôs localizaram o alvo antes de concluir o alinhamento. Pelo fato da área ser pequena, o método com comunicação se comportou similar ao método com localização.

Esta comparação mostra que em ambientes grandes, o método com comunicação faz uma grande diferença, tornando bem mais rápido a procura pelo objeto perdido e diminuindo o tempo de busca, uma vez que este tempo pode ser vital para a busca e salvamento. Além disso, no método com comunicação, todos os robôs são avisados quando um robô acha o alvo perdido, diferentemente dos demais métodos que, quando um robô acha, os demais robôs não saberão dessa informação e continuarão procurando.

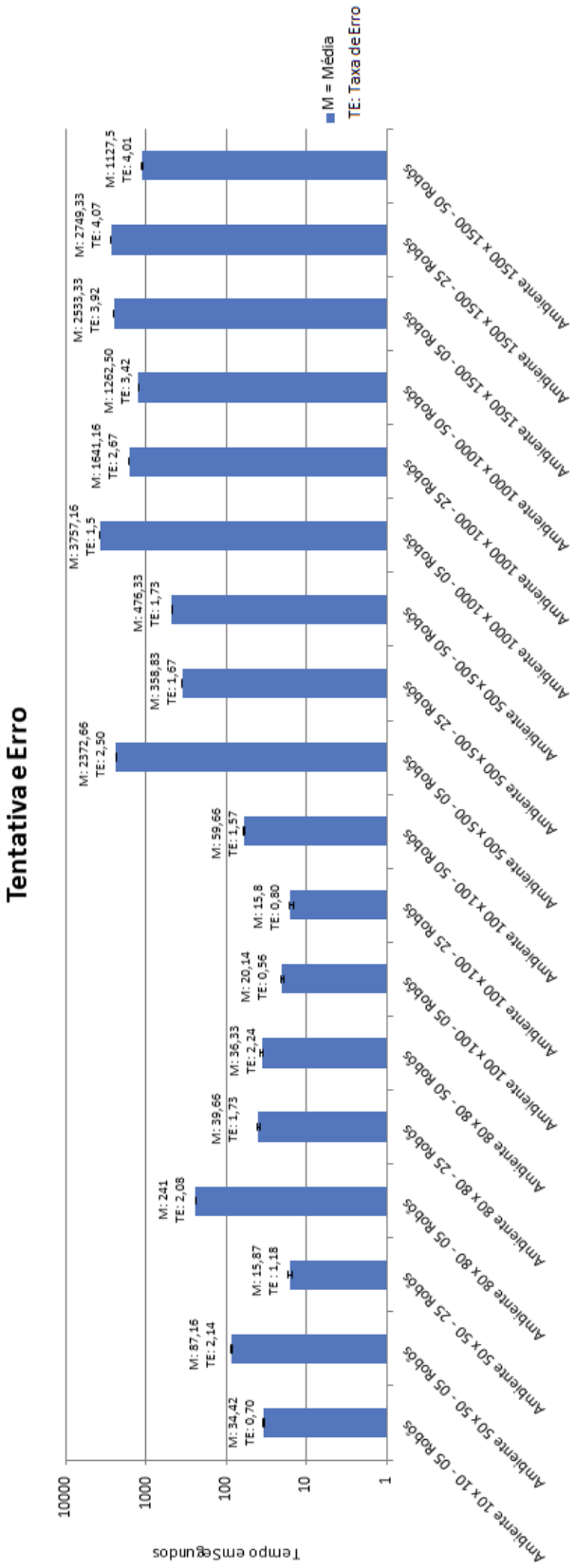

Figura 5.22: Tentativa e erro: comparação entre os tamanhos de área e quantidades de robôs. O gráfico está representado em escala logarítmica de base 10.

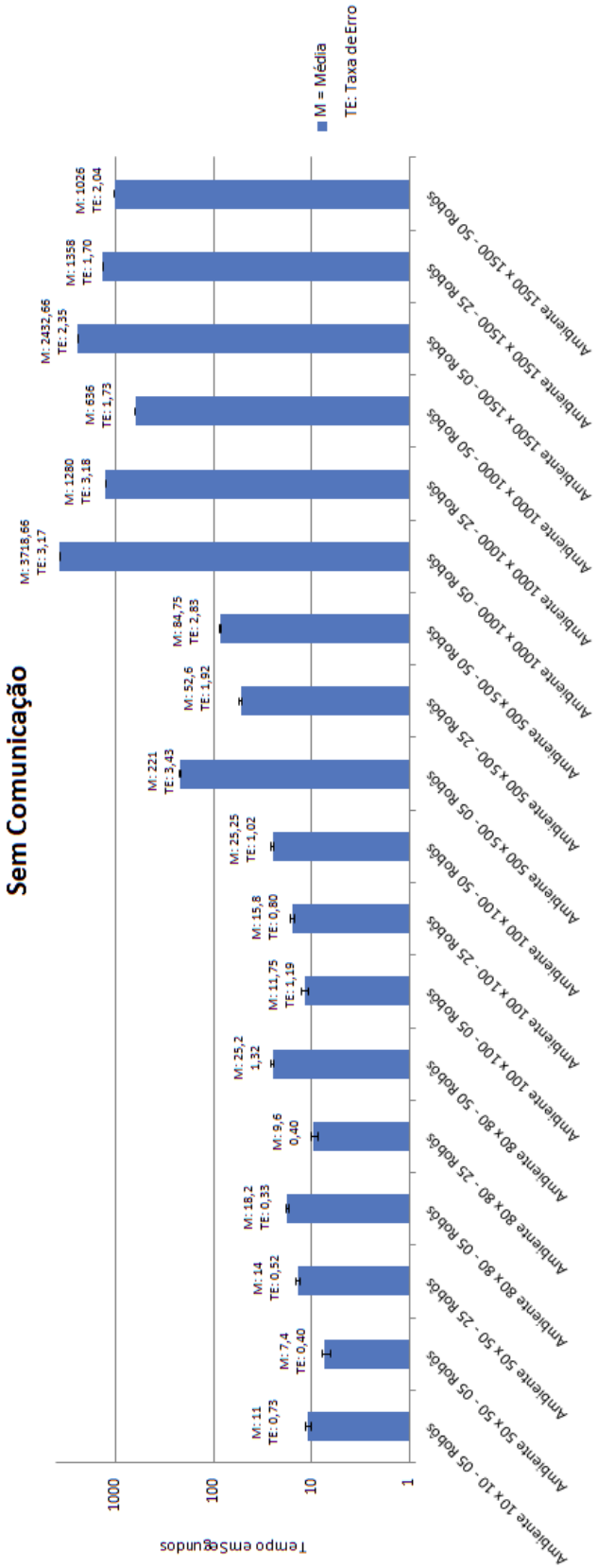

Figura 5.23: Sem comunicação: comparação entre os tamanhos de área e quantidades de robôs. O gráfico está representado em escala logarítmica de base 10.

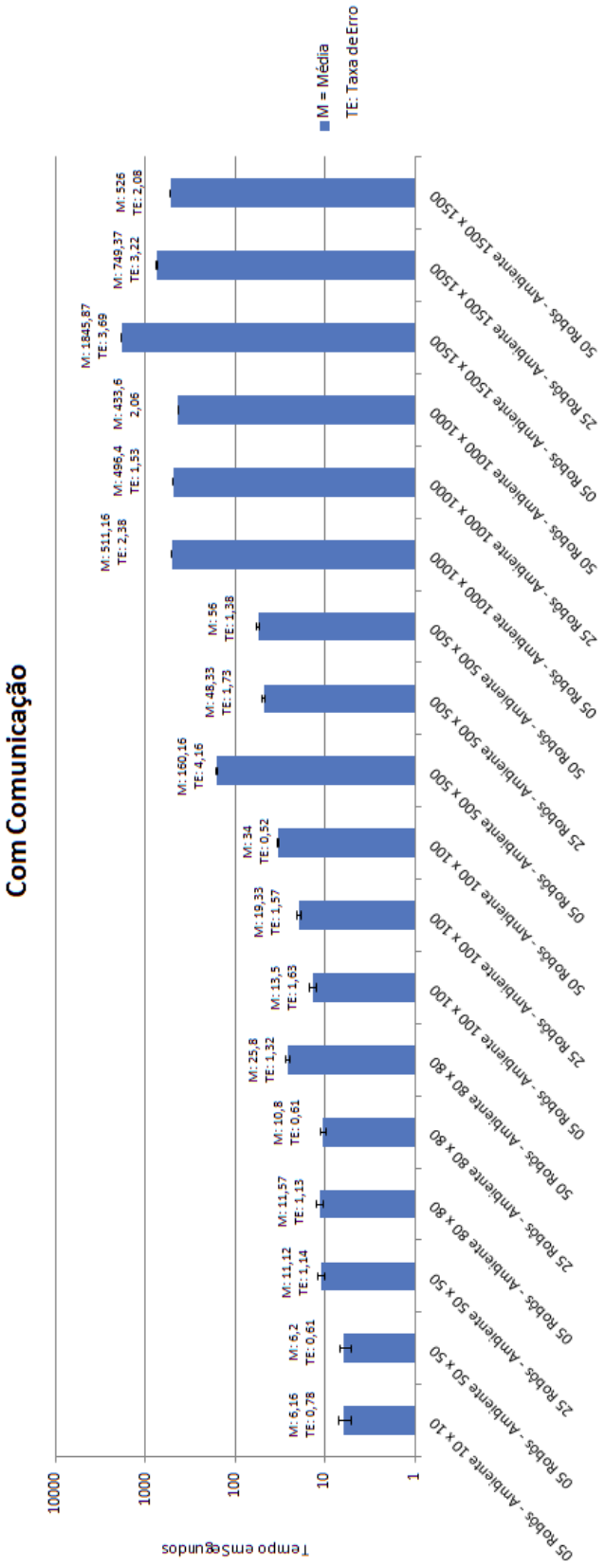

Figura 5.24: Com comunicação: comparação entre os tamanhos de área e quantidades de robôs. O gráfico está representado em escala logarítmica de base 10.

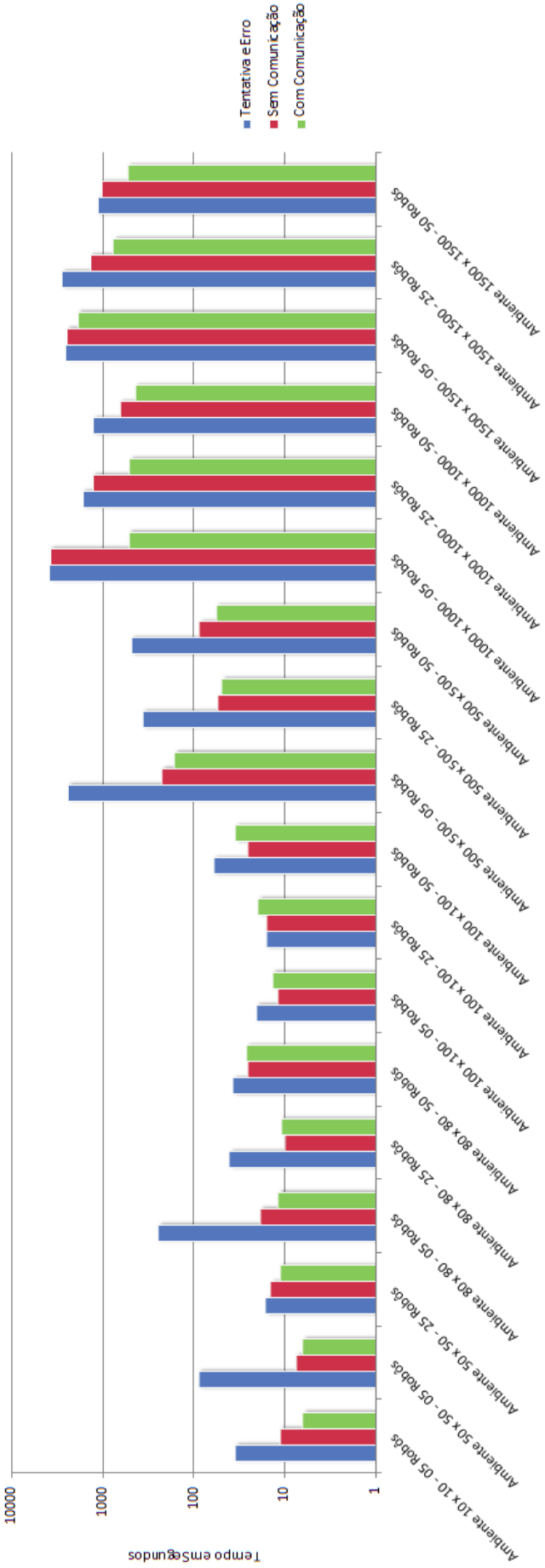

Figura 5.25: Comparação entre os métodos com tamanhos de área e quantidades de robôs diferentes. O gráfico está representado em escala logarítmica de base 10.

CAPÍTULO 6

CONCLUSÃO E TRABALHOS FUTUROS

Como exposto nesse trabalho, o enxame robótico é uma tecnologia em potencial na busca de objetos ou pessoas perdidas. No entanto, esta tecnologia traz alguns desafios, como melhores formas de localização e comunicação para cooperação entre os robôs do enxame.

Este trabalho apresentou uma revisão conceitual necessária para o entendimento da pesquisa, a qual abordou a robótica aplicada, o enxame de robôs e os tipos de rádios em potencial uso para a comunicação e busca. Após, foram apresentados trabalhos que relacionam-se com esta pesquisa e de certa forma, contribuiram, seja como ponto de partida ou em suas abordagens na busca e salvamento. Além disso, foram estudados vários simuladores que atendessem aos requisitos necessários para a simulação da proposta deste trabalho, dentre os estudados foi escolhido o simulador Player/Stage por ter o código aberto e assim permitir as adaptações necessárias, além de dispor de dispositivos e drivers fundamentais para a simulação pretendida, como a conexão *WiFi*.

Com a base conceitual definida e o simulador selecionado e adaptado para as necessidades, foi realizado vários testes com o objetivo de encontrar um objeto ou uma pessoa perdida através da intensidade de sinal. Foram considerados diferentes parâmetros variando entre o tamanho da área da busca e a quantidade de robôs. Para efetivar a busca com o RSSI, foram utilizados três métodos: busca por tentativa e erro, busca com localização e busca com localização e comunicação. Todos estes métodos possuem a leitura de intensidade do sinal e foram testados em todos os diferentes tamanhos de ambiente e número de robôs.

Com os testes realizados, percebeu-se que há uma grande relação do método pela razão tamanho de área/número de robôs. Em ambiente pequenos o método de busca com localização e comunicação não apresentou grandes vantagens em relação aos outros dois métodos, e esses resultados foram similares até o ambiente de 500 metros quadrados

independente da quantidade de robôs. Uma ressalva importante é que em ambientes pequenos percebeu-se que uma grande quantidade de robôs apresentou-se ineficiente, pois a proximidade entre os robôs fez com que perdessem muito tempo desviando entre si para evitar colisões, aumentando assim o tempo para encontrar o objeto perdido.

Em ambientes grandes o método de busca com localização e comunicação teve um desempenho superior em relação aos demais métodos, considerando a mesma quantidade de robôs. Assim pode-se concluir que a comunicação e a leitura de intensidade de sinal são grandes aliadas para a busca e salvamento em áreas superiores a 500 metros quadrados, e esta eficiência pode ser vital quando aplicadas a cenários como salvamento de pessoas.

Contudo, algumas questões não puderam ser atendidas por delimitação do foco e por considerar o tempo disponível, além de outras que surgiram ao longo do desenvolvimento da pesquisa, e assim têm-se como trabalhos futuros:

- Definição de áreas de busca: com as trocas de mensagens de localização e posicionamento, cada robô conhece a posição e o caminho percorrido por cada robô do enxame e podem dinamicamente definir e otimizar áreas de busca para evitar sobreposição.
- Estudar e propor formas de notificação da localização do objeto perdido quando encontrado.
- Identificar se a posição recebida de qualquer um dos robôs do enxame é verdadeira ou falsa.
- Estudar a escolha do líder através de seus recursos computacionais e como informar aos demais robôs do enxame quem é o líder.
- Permitir a busca de vários objetos perdidos considerando que esta quantidade não é conhecida e que os robôs do enxame possuem um tempo limitado de atuação.
- Estudar métodos de buscas presentes na literatura da área para comparar com os métodos apresentados nesta pesquisa com o objetivo de encontrar melhores abordagens para melhoria do processo de busca.

- Acordo do término da busca, como não há predefinições da quantidade de objetos a serem localizados, deve ser estudado um critério de parada de busca do enxame.

- Implementar a leitura de frequências diferentes das usadas pelo enxame.

BIBLIOGRAFIA

[1] The dynamics of collective sorting: robot-like ants and ant-like robots. *From Animals to Animats: Proc. 1st Int. Conf. on Simulation of Adaptive Behaviour*, páginas 356–365, 1991.

[2] Ant system: optimization by a colony of co-operating agents. *IEEE Transaction on Systems, Man and Cybernetics*, páginas 29–41, 1996.

[3] Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, páginas 256–279, 2004.

[4] Ant colony optimization: Introduction and recent trends. *Physics of life reviews*, páginas 353–373, 2005.

[5] Ant colony optimization for multi-objective optimization problems. *IEEE International Conference on Tools with Artificial Intelligence*, páginas 450–457, 2007.

[6] New task allocation methods for robotic swarms. *IEEE/RAS Conference on Autonomous Robot Systems and Competitions*, 2009.

[7] Cobots: Collaborative robots servicing multi-floor buildings. *International Conference on Intelligent Robots and Systems*, 2012.

[8] Control of collaborative mobile robots subject to nonholonomic constraints. *Department of Mechanical and Aerospace Engineering - University of Florida*, 2012.

[9] Localization and navigation of the cobots over long-term deployments. *International Journal of Robotics Research*, (14):1679–1695, 2013.

[10] Nicole Abaid e Maurizio Porfiri. Collective behavior of fish shoals in one-dimensional annular domains. *American Control Conference (ACC), 2010*, páginas 63–68, 2010.

[11] Albert Albers, Sven Brudniok, Jens Ottnad, Christian Sauter, e Korkiat Sedchaicharn. Design of modules and components for humanoid robots. *Advanced Robotic Systems International*, 2007.

[12] ZigBee Alliance e ZigBee Specifications. `http://www.zigbee.org`. Acessado em janeiro de 2008.

[13] Anatel. `www.anatel.gov.br`. Acessado em junho 2014.

[14] Joydeep Biswas e Manuela Veloso. Wifi localization and navigation for autonomous indoor mobile robots. *International Conference on Robotics and Automation*, 2010.

[15] P. Brass, F. Cabrera-Mora, A. Gasparri, e Jizhong Xiao. Multirobot tree and graph exploration. *Robotics, IEEE Transactions on*, 27(4):707–717, 2011.

[16] Luiz Chaimowicz e Yuri Tavares Passos. Controle de congestionamento para enxames de robôs no acesso a alvos em comum. *X SBAI - Simpósio Brasileiro de Automação Inteligente*, 2011.

[17] Hanning Chen, Yunlong Zhu, e Kunyuan Hu. Cooperative bacterial foraging algorithm for global optimization. *Control and Decision Conference, 2009. CCDC '09. Chinese*, páginas 3896–3901, 2009.

[18] Leandro S Coelho e Camila C Silveira. Improved bacterial foraging strategy for controller optimization applied to robotic manipulator system. *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, páginas 1276–1281, 2006.

[19] David Corne, Alan Reynolds, e Eric Bonabeau. Swarm intelligence. *Springer - Verlog Berlim Heidelberg*, 2012.

[20] Cyberbotics. `www.cyberbotics.com`. Acessado em janeiro de 2015.

[21] V. Daiya, J. Ebenezer, S.A.V.S. Murty, e Baldev Raj. Experimental analysis of rssi for distance and position estimation. *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, páginas 1093–1098, 2011.

[22] Corpo de bombeiros militar de Santa Catarina. `http://goo.gl/AOoKXe`. Acessado em julho de 2015.

[23] Universidade de Stanford. Website do ross. `http://wiki.ros.org`. Acessado em fevereiro de 2015.

[24] Universidade Politécnica de Valência. `http://robotica.isa.upv.es/virtualrobot`. Acessado em janeiro de 2015.

[25] Amit Dhariwal, Gaurav S. Sukhatme, e Aristides A. G. Requicha. Bacterium-inspired robots for environmental monitoring. *In Proceedings of IEEE International Conference on Robotics and Automation*, 2:1436–1443, 2004.

[26] Marco Dorigo, Mauro Birattari, e Thomas Stutzle. Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[27] Barzin Doroodgar, Maurizio Ficocelli, Babak Mobedi, e Goldie Nejat. The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, páginas 2858–2863, 2010.

[28] C. Eklund, Roger B. Marks, K.L. Stanwood, e S. Wang. Ieee standard 802.16: a technical overview of the wirelessman/sup tm/ air interface for broadband wireless access. *Communications Magazine, IEEE*, 40(6):98–107, 2002.

[29] Ata Elahi e Adam Gschwender. Zigbee wireless sensor and control network. *Prentice Hall*, páginas 1–288, 2009.

[30] T. Ghanbarzadeh, S. Goleijani, e M.P. Moghaddam. Reliability constrained unit commitment with electric vehicle to grid using hybrid particle swarm optimization

and ant colony optimization. *Power and Energy Society General Meeting, 2011 IEEE*, páginas 1–7, 2011.

[31] D. Gualda, J. Urena, J.C. Garcia, E. Garcia, e D. Ruiz. Rssi distance estimation based on genetic programming. *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, páginas 1–8, 2013.

[32] Wu Husheng e Zhang Fengming. A uncultivated wolf pack algorithm for high-dimensional functions and its application in parameters optimization of pid controller. *Evolutionary Computation (CEC), 2014 IEEE Congress on*, páginas 1477–1482, 2014.

[33] Reza N. Jazar. Theory of applied robotics - rinematics, dynamics, and control. *Springer*, Second Edition, 2010.

[34] Aleksandar Jevtic, Alvaro Gutierrez, e Mo Andina, Diego e Jamshidi. Distributed bees algorithm for task allocation in swarm of robots. *Systems Journal, IEEE*, 6(2):296–304, 2012.

[35] Takaaki Kadota, Toshiyuki Yasuda, Yoshiyuki Matsumura, e Kazuhiro Ohkura. An incremental approach to an evolutionary robotic swarm. *IEEE/SICE International Symposium on System Integration (SII)*, páginas 458–463, 2012.

[36] Yara Khaluf, Emi Mathews, e Franz Josef Rammig. Self-organized cooperation in swarm robotics. *14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, páginas 217–226, 2011.

[37] Albert Ko e Henry YK Lau. Robot assisted emergency search and rescue system with a wireless sensor network. *International Journal of Advanced Science and Technology*, 3:69–78, 2009.

[38] Fei Li, Yuting Zhang, Jiulong Wu, e Haibo Li. Quantum bacterial foraging optimization algorithm. *Evolutionary Computation (CEC), 2014 IEEE Congress on*, páginas 1265–1272, 2014.

[39] Hsien-I Lin e Hua Jr Tzeng. Search strategy of a mobile robot for radiation sources in an unknown environment. *Advanced Robotics and Intelligent Systems (ARIS), 2014 International Conference on*, páginas 56–60, 2014.

[40] Zhongli Liu, Yinjie Chen, Benyuan Liu, Chengyu Cao, e Xinwen Fu. Hawk: An unmanned mini-helicopter-based aerial wireless kit for localization. *Mobile Computing, IEEE Transactions on*, 13(2):287–298, 2014.

[41] Cai Luo, Andre Possani Espinosa, Danu Pranantha, e Alessandro Gloria. Multirobot search and rescue team. *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, páginas 296–301, 2011.

[42] Talles Henrique Medeiros, Luís Fabrício Wanderley Góes, Milene Barbosa Carvalho, e Ilg Menezes. Computação bioinspirada aplicada à robótica. *SBC. (Org.). Escola Regional de Informática de Minas Gerais (ERI-MG).*, 2005.

[43] Martin Mellado, Carlos Correcher, Juan Vicente Catret, e David Puig. Virtualrobot: An open general-purpose simulation tool for robotics. *The European Simulation and Modelling Conference (ESM2003), EUROSIS, Naples, Italy*, páginas 271–350, 2003.

[44] Microsoft. `https://msdn.microsoft.com/pt-br/library/cc580631.aspx`. Acessado em janeiro de 2015.

[45] Robô Mind. `www.robomind.net`. Acessado em janeiro de 2015.

[46] MobileRobots. `http://www.activrobots.com/Software/MobileSim.aspx`. Acessado em janeiro de 2015.

[47] University of Southem Califórnia. Website do player/stage/gazebo. `http://playerstage.sourceforge.net/`. Acessado em janeiro 2015.

[48] Carlo Pinciroli, Vito Trianni, R. O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, F. Ducatelle, Timothy

Stirling, Alvaro Gutierrez, L.M. Gambardella, e M. Dorigo. Argos: A modular, multi-engine simulator for heterogeneous swarm robotics. *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, páginas 5027–5034, 2011.

[49] Attilio Priolo. *Swarm Aggregation Algorithms for Multi-Robot Systems*. Tese de doutorado, University of Roma Tre, 2013.

[50] Theodore S. Rappaport. Comunicações sem fio: Princípios e práticas. *Pearson Prentice Hall*, 2, 2009.

[51] Maria Isabel Ribeiro. Uma viagem ao mundo dos robots. *Instituto de Sistemas e Robótica - Instituto Superior Técnico*, páginas 1–21, 2005.

[52] Uno Robótica. `http://www.unorobotica.com.br/docs`. Acessado em janeiro de 2015.

[53] Adriane B S Serapião. Fundamentos de otimização por inteligência de enxames: uma visão geral. *Controle & Automação Sociedade Brasileira de Automática*, 20, 2009.

[54] Angie Shia, Farokh Bastani, e Yen I-Ling. A highly resilient framework for autonomous robotic swarm systems operating in unknown, hostile environments. *In 10th International Symposium on Autonomous Decentralized Systems*, páginas 147–153, 2011.

[55] Universidade Carnegie Mellon University. Website do carmen. `http://carmen.sourceforge.net/using_carmen.html`. Acessado em janeiro de 2015.

[56] Jingguo Wang e Yangmin Li. Static force analysis for a mobile humanoid robot moving on a slope. *International Conference on Robotics and Biomimetics*, páginas 371–376, 2009.

[57] Alfredo Weitzenfeld, Alberto Vallesa, e Horacio Flores. A biologically-inspired wolf pack multiple robot hunting model. *Robotics Symposium, 2006. LARS '06. IEEE 3rd Latin American*, páginas 120–127, 2006.

[58] Rong-Hou Wu, Yang-Han Lee, Hsien-Wei Tseng, Yih-Guang Jan, e Ming-Hsueh Chuang. Study of characteristics of rssi signal. *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, páginas 1–3, 2008.

[59] Stefano R. Zeplim e Rafael R. S. Borba. Projeto e validação experimental de topologias de robôs móveis aplicadas à robótica educacional. *COBENGE - XL Congresso Brasileiro de Educação em Engenharia*, 2012.

[60] Baoding Zhang e Shulan Gao. The study of zigbee technology's application in swarm robotics system. *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*, páginas 1763–1766, 2011.

APÊNDICE A

TIPOS DE INTERFACE E DEFINIÇÕES DE PARÂMETROS

Tipos de Interface disponíveis no Player:

- player: representa o próprio servidor e na configuração é usado o comportamento do servidor.
- null: não produz dados e não aceita comandos ou pedidos de configuração, simplesmente é a analogia do player para /dev/null.
- aio: fornece acesso a um dispositivo analógico de I/O (Input/Output – Entrada/Saída).
- audio e audiodsp: são usados para controlar o hardware de som, se o robô possuir este hardware. A diferença entre os dois dispositivos é que o audiodsp possui mais comandos disponíveis do que o audio.
- audiomix: é utilizada para controlar os níveis de ruído.
- blobfinder: fornece acesso a dispositivos que detectam cores.
- bumper: simula um vetor de sensores de toque.
- comms: permite a comunicação entre os clientes através do servidor Player.
- camera: é usada para obter as imagens da camera no sistema, seja ela física ou simulada.
- dio: fornece acesso a um dispositivo digital I/O.
- fiducial: permite o acesso a dispositivos que detectam marcadores colocados no ambiente.
- gps: fornece acesso a posição de algum dispositivo.

- gripper: fornece acesso a pinça robótica.
- ir: fornece acesso a sensores IR (infravermelho).
- laser: fornece acesso a um sensor de varredura.
- localize: fornece informações para estimar a posição do robô, através da comparação de leituras leituras do sensor e de um pré mapa do ambiente.
- mcom: serve para troca de informações entre clientes, onde um cliente envia uma mensagem e outros clientes podem requisitar as mensagens.
- position: é usado para controlar a base de um robô móvel planar.
- position3d: é usado para controlar uma base de robô móvel em 3D.
- power: provê acesso ao subsistema de energia do robô.
- ptz: é utilizado para controlar um pan-tilt-zoom (câmera com a capacidade de direcionar remotamente e que possui controle do zoom) da unidade.
- sonar: fornece acesso a um conjunto de sensores de alcance fixo.
- Sound: permite a reprodução de um som pré-registrado.
- speech: fornece acesso a um sistema de síntese de voz.
- truth: fornece acesso ao estado absoluto de entidades.
- wavefront: recebe amostras digitais arbitrárias, como por exemplo, de um dispositivo de áudio digital.
- wifi: provê acesso ao estado de uma interface de redes sem fio.

Definições de parâmetros que devem ser definidos no arquivo .world:

- Tamanho: é o tamanho da janela da simulação em pixels, para isto, é necessário definir a largura e a altura da janela da seguinte forma: tamanho [largura altura].

- Escala: define a quantidade de metros que cada pixel mostra.
- *size*: representa em metros, o tamanho do ambiente da simulação, podendo ter um ou mais mapas dentro deste ambiente de simulação executando ao mesmo tempo.
- *interval_sim*: é a quantidade de milissigundo simulados entre cada atualização da janela de simulação, por padrão vem configurado em 100 milissigundos e
- *interval_real* – define a quantidade de milissigundos entre cada atualização da janela de simulação, por padrão vem configurado em 100 milissigundos.
    - Para controlar a velocidade da simulação, deve ser equilibrado os parâmetros $interval_sim$ e $interval\_real$.

APÊNDICE B

TIPOS DE MODELOS

Existem vários modelos que ajudam na construção de um robô e são utilizados para definir os sensores e atuadores que o robô possui, atuando como uma interface entre a simulação e o player, os modelos são:

- câmera: adiciona uma câmera para o robô e permite que o programa de controle do usuário interaja com a câmera simulada.
- blobfinder: simula a detecção de cores através da imagem da câmera do robô.
- fiducialfinder: localiza pontos fixos em uma imagem, sendo capaz de localizar objetos na simulação.
- ranger: simula qualquer tipo de dispositivo de detecção de obstáculos (sonar, infra-vermelho, entre outros) e, normalmente, aceita vários rangers.

  laser: é um sensor de ranger que só permite um ranger, mas possui um grande campo de visão.
- Gripper: serve para colocar uma garra no robô, que serve para o robô ser capaz de pegar objetos e ser capaz de movê-los dentro do ambiente de simulação.
- posição: simula a posição através da odometria do robô, ou seja, o robô mantém o controle de sua localização através da gravação de quantas vezes as rodas giram o ângulo e ele se transforma. Este modelo utiliza a interface de posição position2d, que especifica aonde o robô está no ambiente. Este modelo possui os parâmetros: unidade – informa como o robô é conduzido, ou seja, como as rodas são controladas, se as duas deslocam na mesma velocidade se altera a velocidade das rodas esquerda ou direita ou se o próprio robô controla a forma como ele se move; localização – diz ao modelo a forma que deve ser gravada a odometria, se o robô calcula a partir do

seu próprio movimento ou com o uso de GPS; odom_error – quantidade de erros que o robô fará nas gravações da odometria e; massa – como o robô é pesado.

APÊNDICE C

ARQUIVO WORLD

```
include "map.inc"
include "pioneer.inc"
include "sick.inc"
interval_sim 100 # simulation timestep in milliseconds
interval_real 20 # real-time interval between simulation updates in
   milliseconds
quit_time 1800
paused 0
resolution 0.02 # Resoluo  de cada pixel
# configure the GUI window
size [10 10]  #mudar o tamanho da rea em metros
window
(
  size [ 700.000 700.000 ] # in pixels
  scale 52.592          #Dividir o tamanho da janela pelo tamanho da rea
  # pixels per meter
  center [ -0.047 -0.002 ]
  rotate [ 0 0 ]
  show_data 1            # 1=on 0=off
)
# load an environment bitmap
floorplan
(
  name "cave"
  size [10.000 10.000 1.000] #mudar o tamanho da rea em metros
  pose [0 0 0 0]
  bitmap "bitmaps/cave.png"
)
pioneer2dx
(
  name "perdido"
  color "gray"
  pose [0.900 1.579 0 33.541]
  wifi(
    mac "99:99:99:99:99:99"
    essid "vivo"
    model "raytrace"
    wall_factor 5
  )
)
pioneer2dx
(
  name "r1"
  color "green"
  pose [-4.085 1.619 0 88.612]
  sicklaser()
  ctrl "wander_wifi"
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.1"
    netmask "255.255.255.0"
    mac "01:01:01:01:01:01"
    essid "mestrado"
    model "raytrace"
    wall_factor 5
```

```
    power 45
    sensitivity -53
    range_db -45
        )
)
pioneer2dx
(
  name "r2"
  color "green"
  pose [-1.554 -1.822 0 69.440]
  sicklaser()
  ctrl "wander_wifi"
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.2"
    netmask "255.255.255.0"
    mac "02:02:02:02:02:02"
    essid "mestrado"
    model "raytrace"
    wall_factor 5
    power 45
    sensitivity -53
    range_db -45
  )
)
pioneer2dx
(
  name "r3"
  color "green"
  pose [-3.055 -0.232 0 -29.838]
  sicklaser()
  ctrl "wander_wifi"
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.3"
    netmask "255.255.255.0"
    mac "03:03:03:03:03:03"
    essid "mestrado"
    model "raytrace"
    wall_factor 5
    power 45
    sensitivity -53
    range_db -45
  )
)
pioneer2dx
(
  name "r4"
  color "green"
  pose [-4.225 -0.376 0 157.259]
  sicklaser()
  ctrl "wander_wifi"
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.4"
    netmask "255.255.255.0"
    mac "04:04:04:04:04:04"
    essid "mestrado"
    model "raytrace"
    wall_factor 5
```

```
    power 45
    sensitivity -53
    range_db -45
  )
)
pioneer2dx
(
  name "r5"
  color "green"
  pose [-2.188 0.708 0 -151.123]
  sicklaser()
  ctrl "wander_wifi"
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  wifi(
    ip "192.168.0.5"
    netmask "255.255.255.0"
    mac "05:05:05:05:05:05"
    essid "mestrado"
    model "raytrace"
    wall_factor 5
    power 45
    sensitivity -53
    range_db -45
  )
)
```

APÊNDICE D

ARQUIVO INC

Arquivo map.inc

```
define floorplan model
(
  # sombre, sensible, artistic
  color "gray30"
  # most maps will need a bounding box
  boundary 1
  gui_nose 0
  gui_grid 0
  gui_move 0
  gui_outline 0
  gripper_return 0
  fiducial_return 0
  laser_return 1
)
define zone model
(
  color "orange"
  size [ 2 2 0.02 ]
  gui_nose 0
  gui_grid 0
  gui_move 0
  gui_outline 0
  # insensible to collision and range sensors
  obstacle_return 0
  laser_return 0
  ranger_return 0
)
```

Arquivo pioneer.inc

```
# The Pioneer2DX sonar array
define p2dx_sonar ranger
(
  scount 16 # the number of transducers
  # define the pose of each transducer [xpos ypos heading]
  spose[0] [ 0.075 0.130 90 ]
  spose[1] [ 0.115 0.115 50 ]
  spose[2] [ 0.150 0.080 30 ]
  spose[3] [ 0.170 0.025 10 ]
  spose[4] [ 0.170 -0.025 -10 ]
  spose[5] [ 0.150 -0.080 -30 ]
  spose[6] [ 0.115 -0.115 -50 ]
  spose[7] [ 0.075 -0.130 -90 ]
  spose[8] [ -0.155 -0.130 -90 ]
  spose[9] [ -0.195 -0.115 -130 ]
  spose[10] [ -0.230 -0.080 -150 ]
  spose[11] [ -0.250 -0.025 -170 ]
  spose[12] [ -0.250 0.025 170 ]
  spose[13] [ -0.230 0.080 150 ]
  spose[14] [ -0.195 0.115 130 ]
  spose[15] [ -0.155 0.130 90 ]
  # define the field of view of each transducer [range_min range_max view_angle
      ]
  sview [0 5.0 15]
  # define the size of each transducer [xsize ysize] in meters
  ssize [0.01 0.05]
```

```
)
define p2dx_sonar_front ranger
(
  scount 8 # the number of transducers
  # define the pose of each transducer [xpos ypos heading]
  spose[0] [ 0.075 0.130 90 ]
  spose[1] [ 0.115 0.115 50 ]
  spose[2] [ 0.150 0.080 30 ]
  spose[3] [ 0.170 0.025 10 ]
  spose[4] [ 0.170 -0.025 -10 ]
  spose[5] [ 0.150 -0.080 -30 ]
  spose[6] [ 0.115 -0.115 -50 ]
  spose[7] [ 0.075 -0.130 -90 ]
  # define the field of view of each transducer [range_min range_max view_angle
     ]
  sview [0 5.0 15]
  # define the size of each transducer [xsize ysize] in meters
  ssize [0.01 0.05]
)
# The Pioneer3DX sonar array
define p3dx_sonar ranger
(
  scount 16
  # define the pose of each transducer [xpos ypos heading]
  spose[0] [ 0.069 0.136 90 ]
  spose[1] [ 0.114 0.119 50 ]
  spose[2] [ 0.148 0.078 30 ]
  spose[3] [ 0.166 0.027 10 ]
  spose[4] [ 0.166 -0.027 -10 ]
  spose[5] [ 0.148 -0.078 -30 ]
  spose[6] [ 0.114 -0.119 -50 ]
  spose[7] [ 0.069 -0.136 -90 ]
  spose[8] [ -0.157 -0.136 -90 ]
  spose[9] [ -0.203 -0.119 -130 ]
  spose[10] [ -0.237 -0.078 -150 ]
  spose[11] [ -0.255 -0.027 -170 ]
  spose[12] [ -0.255 0.027 170 ]
  spose[13] [ -0.237 0.078 150 ]
  spose[14] [ -0.103 0.119 130 ]
  spose[15] [ -0.157 0.136 90 ]
  # define the field of view of each transducer [range_min range_max view_angle
     ]
  sview [0.1 5.0 30] # min (m), max (m), field of view (deg)
  # define the size of each transducer [xsize ysize] in meters
  ssize [0.01 0.04]
)
# The Pioneer3AT sonar array
define p3at_sonar ranger
(
  scount 16
  # define the pose of each transducer [xpos ypos heading]
  spose[0] [0.147 0.136 90]
  spose[1] [0.193 0.119 50]
  spose[2] [0.227 0.079 30]
  spose[3] [0.245 0.027 10]
  spose[4] [0.245 -0.027 -10]
  spose[5] [0.227 -0.079 -30]
  spose[6] [0.193 -0.119 -50]
  spose[7] [0.147 -0.136 -90]
  spose[8] [-0.144 -0.136 -90]
  spose[9] [-0.189 -0.119 -130]
  spose[10] [-0.223 -0.079 -150]
  spose[11] [-0.241 -0.027 -170]
```

```
  spose[12] [-0.241 0.027 170]
  spose[13] [-0.223 0.079 150]
  spose[14] [-0.189 0.119 130]
  spose[15] [-0.144 0.136 90]
  # define the field of view of each transducer [range_min range_max view_angle
     ]
  sview [0.1 5.0 30] # min (m), max (m), field of view (deg)
  # define the size of each transducer [xsize ysize] in meters
  ssize [0.01 0.04]
)
define pioneer_base position
(
  color "red"                # Default color.
  drive "diff"               # Differential steering model.
  gui_nose 1                 # Draw a nose on the robot so we can see which way
      it points
  obstacle_return 1          # Can hit things.
  laser_return 1              # reflects laser beams
  ranger_return 1            # reflects sonar beams
  blob_return 1              # Seen by blobfinders
  fiducial_return 1          # Seen as "1" fiducial finders
  localization "gps"
  localization_origin [0 0 0 0]     # Start odometry at (0, 0, 0).
  # alternative odometric localization with simple error model
  # localization "odom"                    # Change to "gps" to have impossibly
      perfect, global odometry
  #  odom_error [ 0.05 0.05 0.1 ]    # Odometry error or slip in X, Y and Theta
                                     # (Uniform random distribution)
)
define pioneer2dx_base_no_sonar pioneer_base
(
  # actual size
  size [0.44 0.38 0.22] # sizes from MobileRobots' web site
  # the pioneer's center of rotation is offset from its center of area
  origin [-0.04 0 0 0]
  # draw a nose on the robot so we can see which way it points
  gui_nose 1
  # estimated mass in KG
  mass 23.0
  # differential steering model
  drive "diff"
 )
define pioneer2dx_base pioneer2dx_base_no_sonar
(
  # use the sonar array defined above with a small vertical offset to
  # drop the sensors into the robot body
  p2dx_sonar( pose [0 0 -0.03 0] )
)
define pioneer2dx_base_front_sonar pioneer2dx_base_no_sonar
(
  # use the sonar array defined above with a small vertical offset to
  # drop the sensors into the robot body
  p2dx_sonar_front( pose [0 0 -0.03 0] )
)
define pioneer2dx pioneer2dx_base
(
  # simplified Body shape:
  block(
    points 8
    point[0] [-0.2 0.12]
    point[1] [-0.2 -0.12]
    point[2] [-0.12 -0.2555]
    point[3] [0.12 -0.2555]
```

```
    point[4] [0.2 -0.12]
    point[5] [0.2 0.12]
    point[6] [0.12 0.2555]
    point[7] [-0.12 0.2555]
    z [0 0.22]
  )
)
# as above, but with front sonar only
define pioneer2dx_front_sonar pioneer2dx_base_front_sonar
(
  # simplified Body shape:
  block(
    points 8
    point[0] [-0.2 0.12]
    point[1] [-0.2 -0.12]
    point[2] [-0.12 -0.2555]
    point[3] [0.12 -0.2555]
    point[4] [0.2 -0.12]
    point[5] [0.2 0.12]
    point[6] [0.12 0.2555]
    point[7] [-0.12 0.2555]
    z [0 0.22]
  )
)
# a Pioneer 2 or 3 in standard configuration
define fancypioneer2dx pioneer2dx_base
(
  # this set of blocks approximates the shape of a real Pioneer
  #  The geometry is from the Webots v5.3.0 manual. Thanks to Webots
  #  and Olivier Michel. If Stage or Gazebo do not do what you want,
  #  take a look at Webots. It's a very nice commercial simulator.
  # main body
  block
  (
    points 8
    point[0] [ -0.185 -0.135 ]
    point[1] [ 0.095 -0.135 ]
    point[2] [ 0.11 -0.08 ]
    point[3] [ 0.11  0.08 ]
    point[4] [ 0.095 0.135 ]
    point[5] [ -0.185 0.135 ]
    point[6] [ -0.215 0.1 ]
    point[7] [ -0.215 -0.1 ]
    z [ 0.059 0.234 ]
  )
  # sonar case
  block
  (
    points 9
    point[0] [ -0.135 0.136 ]
    point[1] [ -0.185 0.136 ]
    point[2] [ -0.223 0.101 ]
    point[3] [ -0.248 0.054 ]
    point[4] [ -0.258 0    ]
    point[5] [ -0.248 -0.054 ]
    point[6] [ -0.223 -0.101 ]
    point[7] [ -0.185 -0.136 ]
    point[8] [ -0.135 -0.136 ]
    z [ 0.184 0.234 ]
  )
  # sonar case
  block
  (
```

```
  points 9
  point[0] [ 0.046 -0.136 ]
  point[1] [ 0.096 -0.136 ]
  point[2] [ 0.134 -0.101 ]
  point[3] [ 0.159 -0.054 ]
  point[4] [ 0.168 0     ]
  point[5] [ 0.159 0.054 ]
  point[6] [ 0.134 0.101 ]
  point[7] [ 0.096 0.136 ]
  point[8] [ 0.046 0.136 ]
  z [ 0.184 0.234 ]
)
# left wheel
block
(
  points 4
  point[0] [ 0.083 0.177 ]
  point[1] [ -0.083 0.177 ]
  point[2] [ -0.083 0.140 ]
  point[3] [ 0.083 0.140 ]
  z [0 0.165 ]
  color "gray15"
)
# right wheel
block
(
  points 4
  point[0] [ 0.083 -0.14 ]
  point[1] [ -0.083 -0.14 ]
  point[2] [ -0.083 -0.177 ]
  point[3] [ 0.083 -0.177 ]
  z [ 0 0.165 ]
  color "gray15"
)
# castor
block
(
  points 4
  point[3] [ -0.2475 0.012 ]
  point[2] [ -0.1825 0.012 ]
  point[1] [ -0.1825 -0.012 ]
  point[0] [ -0.2475 -0.012 ]
  z [ 0 0.065 ]
  color "gray15"
)
# lid
block
(
  points 22
  point[21] [ 0.174 0 ]
  point[20] [ 0.166 -0.056 ]
  point[19] [ 0.145 -0.107 ]
  point[18] [ 0.112 -0.155 ]
  point[17] [ 0.064 -0.190 ]
  point[16] [ -0.074 -0.190 ]
  point[15] [ -0.096 -0.160 ]
  point[14] [ -0.151 -0.160 ]
  point[13] [ -0.2  -0.155 ]
  point[12] [ -0.236 -0.107 ]
  point[11] [ -0.256 -0.056 ]
  point[10] [ -0.264 0     ]
  point[9] [ -0.256 0.056 ]
  point[8] [ -0.236 0.107 ]
```

```
    point[7] [ -0.2  0.155 ]
    point[6] [ -0.151 0.160 ]
    point[5] [ -0.096 0.160 ]
    point[4] [ -0.074 0.190 ]
    point[3] [ 0.064 0.190 ]
    point[2] [ 0.112 0.155 ]
    point[1] [ 0.145 0.107 ]
    point[0] [ 0.166 0.056 ]
    z [ 0.234 0.24 ]
    # a dark top looks more realistic, but isn't very useful
    # for a top-down view
    #color "gray10"
  )
)
# define 10 straight bumpers around the edge of the robot
#
# (these angles are correct for p2dx but the offsets are approximate - RTV)
# format: bumper[x] [x y th length radius] (zero radius gives a straight line)
# WARNING: bumpers are not currently supported by Stage>=1.5
# define pioneer2dxbumper bumper
# (
#   bumpers10
#   bumper[0] [ 0.17 -0.22 -52 0.105 0.0 ]
#   bumper[1] [ 0.24 -0.12 -19 0.105 0.0 ]
#   bumper[2] [ 0.26 0.00   0  0.105 0.0 ]
#   bumper[3] [ 0.24 0.12  19  0.105 0.0 ]
#   bumper[4] [ 0.17 0.22  52  0.105 0.0 ]
#   bumper[5] [ -0.25 0.22 128 0.105 0.0 ]
#   bumper[6] [ -0.32 0.12 161 0.105 0.0 ]
#   bumper[7] [ -0.34 0.00 180 0.105 0.0 ]
#   bumper[8] [ -0.32 -0.12 199 0.105 0.0 ]
#   bumper[9] [ -0.25 -0.22 232 0.105 0.0 ]
# )
# The Pioneer3DX standard configuration
define pioneer3dx pioneer_base
(
  # Actual size
  size [0.511 0.4 0.22 ]
  # The pioneer's center of rotation is offset from its center of area
  origin [-0.04465 0.0 0.0]
  # Estimated mass in KG
  mass 23.0
  # Body shape:
  block(
    points 8
    point[0] [-0.2 0.12]
    point[1] [-0.2 -0.12]
    point[2] [-0.12 -0.2555]
    point[3] [0.12 -0.2555]
    point[4] [0.2 -0.12]
    point[5] [0.2 0.12]
    point[6] [0.12 0.2555]
    point[7] [-0.12 0.2555]
    z [0 0.22]
  )
  # Use the sonar array defined above
  p3dx_sonar( pose [ 0 0 -0.03 0 ] )
)
# The Pioneer3AT standard configuration
define pioneer3at pioneer_base
(
  # Actual size
  size [0.626 0.505]
```

```
  # The pioneer's center of rotation is offset from its center of area
  origin [-0.04465 0.0 0.0]
  # Estimated mass in KG
  mass 40.0
  # Body shape:
  block(
    points 8
    point[0] [-0.18 0.313]
    point[1] [0.18 0.313]
    point[2] [0.2525 0.18]
    point[3] [0.2525 -0.18]
    point[4] [0.18 -0.313]
    point[5] [-0.18 -0.313]
    point[6] [-0.2525 -0.18]
    point[7] [-0.2525 0.18]
  )
  # Use the sonar array defined above
  p3at_sonar( pose [ 0 0 -0.03 0 ] )
)

### AMIGOBOT ####
# The AmigoBot sonar array
define amigo_sonar ranger
(
  scount 8
  spose[0] [ 0.073 0.105 90 ]
  spose[1] [ 0.130 0.078 41 ]
  spose[2] [ 0.154 0.030 15 ]
  spose[3] [ 0.154 -0.030 -15 ]
  spose[4] [ 0.130 -0.078 -41 ]
  spose[5] [ 0.073 -0.105 -90 ]
  spose[6] [ -0.146 -0.060 -145 ]
  spose[7] [ -0.146 0.060 145 ]
)
define amigobot position
(
  size [0.330 0.280 0.25]
  origin [0 0 0 0] # what should this value be? send email to vaughan@sfu.ca.
  amigo_sonar( pose [0 0 -0.02 0 ] )
)
```

Arquivo sick.inc

```
 define sicklaser laser
(
  # laser-specific properties
  # factory settings for LMS200
  range_max 8.0
  fov 180.0
  samples 361
  #samples 90 # still useful but much faster to compute
  # generic model properties
  color "blue"
  size [ 0.156 0.155 0.19 ] # dimensions from LMS200 data sheet
)
# extends sicklaser to add nice-looking but relatively expensive geometry
define fancysicklaser sicklaser
(
  # bottom
  block(
    points 4
    point[0] [ -0.02 -0.077 ]
    point[1] [ 0.078 -0.077 ]
    point[2] [ 0.078 0.077 ]
    point[3] [ -0.02 0.077 ]
```

```
    z [0 0.02 ]
  )
  # back
  block(
    points 4
    point[0] [ -0.078 -0.077 ]
    point[1] [ -0.02 -0.077 ]
    point[2] [ -0.02 0.077 ]
    point[3] [ -0.078 0.077 ]
    z [0 0.21 ]
  )
  # top
  block( points 4
    point[0] [ -0.02 -0.077 ]
    point[1] [ 0.078 -0.077 ]
    point[2] [ 0.078 0.077 ]
    point[3] [ -0.02 0.077 ]
    z [0.12 0.21 ]
  )
  # laser bit
  block( points 4
    point[0] [ -0.02 -0.05 ]
         point[1] [ 0.06 -0.05 ]
         point[2] [ 0.06 0.05 ]
         point[3] [ -0.02 0.05 ]
         z [0.02 0.12 ]
   color "gray10"
  )
)
```

APÊNDICE E

ARQUIVO CFG

```
 driver
(
  name "stage"
  provides ["simulation:0" ]
  plugin "stageplugin"
  worldfile "wifi_50x50_05robos.world"
)
driver
(
  name "stage"
  provides ["6660:wifi:0" "6660:position2d:0" ]
  model "perdido"
  alwayson 1
)
driver
(
  name "stage"
  provides ["6661:position2d:1" "6661:wifi:1" ]
  model "r1"
  alwayson 1
)
driver
(
  name "stage"
  provides ["6662:position2d:2" "6662:wifi:2" ]
  model "r2"
  alwayson 1
)
driver
(
  name "stage"
  provides ["6663:position2d:3" "6663:wifi:3" ]
  model "r3"
  alwayson 1
)
driver
(
  name "stage"
  provides ["6664:position2d:4" "6664:wifi:4" ]
  model "r4"
  alwayson 1
)
driver
(
  name "stage"
  provides ["6665:position2d:5" "6665:wifi:5" ]
  model "r5"
  alwayson 1
)
```

APÊNDICE F

RESULTADOS DOS TESTES REALIZADOS

Tabela da média dos testes realizados, com o desvio padrão e a taxa de erros de 10 %.

Legenda
TE = Tentativa e Erro
SC = Sem Comunicação
CC = Com comunicação
MR = Média dos Resultados
DP = Desvio Padrão
ME90%C = Margem de erro para 90% de confiança

Testes	MR	DP	ME90%C
TE–10x10–05Rb	34,42	1,13	0,7
SC–10x10–05Rb	11	1	0,73
CC–10x10–05Rb	6,16	1,16	0,78
TE–50x50–05Rb	87,16	3,18	2,14
SC–50x50–05Rb	7,4	0,54	0,4
CC–50x50–05Rb	6,2	0,83	0,61
TE–50x50–25Rb	15,87	2,03	1,18
SC–50x50–25Rb	14	0,7	0,52
CC–50x50–25Rb	11,12	1,95	1,14
TE–80x80–05Rb	241	3,09	2,08
SC–80x80–05Rb	18,2	0,44	0,33
CC–80x80–05Rb	11,57	1,81	1,13
TE–80x80–25Rb	39,66	2,58	1,73
SC–80x80–25Rb	9,6	0,54	0,4
CC–80x80–25Rb	10,8	0,83	0,61
TE–80x80–50Rb	36,33	3,32	2,24
SC–80x80–50Rb	25,2	1,78	1,32
CC–80x80–50Rb	25,8	1,78	1,32
TE–100x100–05Rb	20,14	0,89	0,56
SC–100x100–05Rb	11,75	2,05	1,19
CC–100x100–05Rb	13,5	2,42	1,63
TE–100x100–25Rb	15,8	1,09	0,8
SC–100x100–25Rb	15,8	1,09	0,8
CC–100x100–25Rb	19,33	2,33	1,57
TE–100x100–50Rb	59,66	2,33	1,57
SC–100x100–50Rb	25,25	1,75	1,02
CC–100x100–50Rb	34	0,7	0,52
TE–500x500–05Rb	2372,66	3,72	2,5
SC–500x500–05Rb	221	5,09	3,43
CC–500x500–05Rb	160,16	6,17	4,16
TE–500x500–25Rb	358,83	2,48	1,67
SC–500x500–25Rb	52,6	2,6	1,92
CC–500x500–25Rb	48,33	2,58	1,73
TE–500x500–50Rb	476,33	2,58	1,73
SC–500x500–50Rb	84,75	4,86	2,83
CC–500x500–50Rb	56	1,87	1,38
TE–1000x1000–05Rb	3757,16	2,22	1,5
SC–1000x1000–05Rb	3718,66	4,71	3,17
CC–1000x1000–05Rb	511,16	3,54	2,38
TE–1000x1000–25Rb	1641,16	3,97	2,67
SC–1000x1000–25Rb	1280	4,73	3,18
CC–1000x1000–25Rb	496,4	2,07	1,53
TE–1000x1000–50Rb	1262,5	5,08	3,42
SC–1000x1000–50Rb	636	2,34	1,73
CC–1000x1000–50Rb	433,6	2,79	2,06
TE–1500x1500–05Rb	2533,33	5,81	3,92
SC–1500x1500–05Rb	2432,66	3,5	2,35
CC–1500x1500–05Rb	1845,87	6,33	3,69
TE–1500x1500–25Rb	2749,33	6,05	4,07
SC–1500x1500–25Rb	1358	2,52	1,7
CC–1500x1500–25Rb	749,37	5,52	3,22
TE–1500x1500–50Rb	1127,5	5,95	4,01
SC–1500x1500–50Rb	1026	3,03	2,04
CC–1500x1500–50Rb	526	2,82	2,08

APÊNDICE G

TRECHO DO ARQUIVO PLAYERTCP.CC DO PLAYER

```
int PlayerTCP::WriteClient(int cli){
  int numwritten;
  playertcp_conn_t* client;
  Message* msg;
  player_pack_fn_t packfunc;
  player_msghdr_t hdr;
  void* payload;
  int encode_msglen;
#if HAVE_Z
  player_map_data_t* zipped_data=NULL;
#endif
  client = this->clients + cli;
  for(;;){
    // try to send any bytes leftover from last time.
    if(client->writebufferlen){
      numwritten = send(client->fd, client->writebuffer, MIN(client->
          writebufferlen, PLAYERTCP_WRITEBUFFER_SIZE), 0);
      if(numwritten < 0){
        if(ErrNo == ERRNO_EAGAIN){
          // buffers are full
          return(0);
        }else{
#if defined (WIN32)
          LPVOID buffer = NULL;
          FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
              FORMAT_MESSAGE_FROM_SYSTEM, NULL,
                      ErrNo, 0, reinterpret_cast<LPTSTR> (&buffer), 0, NULL);
          PLAYER_MSG1(2, "send() failed: %s", reinterpret_cast<LPTSTR> (buffer))
              ;
          LocalFree(buffer);
#else
          PLAYER_MSG1(2,"send() failed: %s", strerror(ErrNo));
#endif
          return(-1);
        }
      }
      else if(numwritten == 0)
      {
        PLAYER_MSG0(2,"wrote zero bytes");
        return(-1);
      }

      memmove(client->writebuffer, client->writebuffer + numwritten,
              client->writebufferlen - numwritten);
      client->writebufferlen -= numwritten;
    }
    // try to pop a pending message
    else if((msg = client->queue->Pop()))
    {
      // Note that we make a COPY of the header. This is so that we can
      // edit the size field before sending it out, without affecting other
      // instances of the message on other queues.
      hdr = *msg->GetHeader();
      payload = msg->GetPayload();

      // Make sure there's room in the buffer for the encoded messsage.
```

```
      // 4 times the message (including dynamic data) is a safe upper bound
      size_t maxsize = PLAYERXDR_MSGHDR_SIZE + (4 * msg->GetDataSize());
      if(maxsize > (size_t)(client->writebuffersize))
      {
        // Get at least twice as much space
        client->writebuffersize = MAX((size_t)(client->writebuffersize * 2),
                                      maxsize);
        // Did we hit the limit (or overflow and become negative)?
        if((client->writebuffersize >= PLAYERXDR_MAX_MESSAGE_SIZE) ||
           (client->writebuffersize < 0))
        {
          PLAYER_WARN1("allocating maximum %d bytes to outgoing message buffer",
                       PLAYERXDR_MAX_MESSAGE_SIZE);
          client->writebuffersize = PLAYERXDR_MAX_MESSAGE_SIZE;
        }
        client->writebuffer = (char*)realloc(client->writebuffer,
                                             client->writebuffersize);
        assert(client->writebuffer);
        memset(client->writebuffer, 0, client->writebuffersize);
      }

      // HACK: special handling for map data to compress it before sending
      // them out over the network.
      if((hdr.addr.interf == PLAYER_MAP_CODE) &&
         (hdr.type == PLAYER_MSGTYPE_RESP_ACK) &&
         (hdr.subtype == PLAYER_MAP_REQ_GET_DATA))
      {
#if HAVE_Z
        player_map_data_t* raw_data = (player_map_data_t*)payload;
        zipped_data = (player_map_data_t*)calloc(1,sizeof(player_map_data_t));
        assert(zipped_data);

        // copy the metadata
        *zipped_data = *raw_data;
        uLongf count = compressBound(raw_data->data_count);
        zipped_data->data = (int8_t*)malloc(count);

        // compress the tile
        int ret;
        ret = compress((Bytef*)zipped_data->data,&count,
                       (const Bytef*)raw_data->data, raw_data->data_count);
        if((ret != Z_OK) && (ret != Z_STREAM_END))
        {
          PLAYER_ERROR("failed to compress map data");
          free(zipped_data);
          client->writebufferlen = 0;
          delete msg;
          return(0);
        }

        zipped_data->data_count = count;

        // swap the payload pointer to point at the zipped version
        payload = (void*)zipped_data;
#else
        PLAYER_WARN("not compressing map data, because zlib was not found at
            compile time");
#endif
      }

      if (payload)
      {
        // Locate the appropriate packing function
```

```
        if(!(packfunc = playerxdr_get_packfunc(hdr.addr.interf,
                                               hdr.type, hdr.subtype)))
        {
          // TODO: Allow the user to register a callback to handle unsupported
             messages
          PLAYER_WARN4("skipping message from %s:%u with unsupported type %s:%u
             ",
                       interf_to_str(hdr.addr.interf), hdr.addr.index,
                          msgtype_to_str(hdr.type), hdr.subtype);
        }
        else
        {
          // Encode the body first
          if((encode_msglen =
              (*packfunc)(client->writebuffer + PLAYERXDR_MSGHDR_SIZE,
                       maxsize - PLAYERXDR_MSGHDR_SIZE,
                       payload, PLAYERXDR_ENCODE)) < 0)
          {

            //  ALTERAO
            /*
            PLAYER_WARN4("encoding failed on message from %s:%u with type %s:%u
               ERRO EM PLAYERTCP.CC LINHA 777",
                     interf_to_str(hdr.addr.interf), hdr.addr.index,
                        msgtype_to_str(hdr.type), hdr.subtype);
            #if HAVE_Z
            if(zipped_data)
            {
              free(zipped_data->data);
              free(zipped_data);
              zipped_data=NULL;
            }
            #endif
            client->writebufferlen = 0;
            delete msg;
            return(0);
            */
            return(1);
          }
        }
      }
      else
      {
        encode_msglen = 0;
      }
      // Rewrite the size in the header with the length of the encoded
      // body, then encode the header.
      hdr.size = encode_msglen;
      if((encode_msglen = player_msghdr_pack(client->writebuffer,
                  PLAYERXDR_MSGHDR_SIZE, &hdr,
                  PLAYERXDR_ENCODE)) < 0)
      {
        PLAYER_ERROR("failed to encode msg header");
#if HAVE_Z
        if(zipped_data)
        {
          free(zipped_data->data);
          free(zipped_data);
          zipped_data=NULL;
        }
#endif
        client->writebufferlen = 0;
        delete msg;
```

```
        return(0);
      }
      client->writebufferlen = PLAYERXDR_MSGHDR_SIZE + hdr.size;
      delete msg;
#if HAVE_Z
      if(zipped_data)
      {
        free(zipped_data->data);
        free(zipped_data);
        zipped_data=NULL;
      }
#endif
    }
    else
      return(0);
  }
}
```

APÊNDICE H

SIMULADORES E INTERFACES PARA ROBÔS

- Player: É uma interface para robôs e teve o seu desenvolvimento iniciado no ano de 2000 por pesquisadores da *University of Southem Califórnia – EUA*. Foi desenvolvido para o sistema operacional Linux, possui o código aberto e, pesquisadores de várias instituições colaboram com o seu desenvolvimento. O Player possui uma interface de acesso ao hardware dos sensores e robôs móveis tornando mais simples a utilização, sua estrutura é baseada no modelo cliente/servidor: o servidor é responsável pela comunicação entre os sensores e o robô e; o cliente é responsável por controlar o robô. Um cliente pode controlar diversos servidores e vários clientes podem controlar diferentes sensores de um mesmo robô. A comunicação entre cliente e servidor é feita através de TCP/IP [47].

- ROS (*Robot Operating System*): É um ambiente de desenvolvimento que integra diversos pacotes orientados para a robótica, teve o seu desenvolvimento inicial no ano de 2000 pela Universidade de Stanford. Foi desenvolvido para plataformas baseadas em Unix (principalmente para os sistemas Ubuntu e Mac OS X), possui o código aberto e vários pesquisadores colaboram com o seu desenvolvimento. Este simulador serve de interface para o robô. O ROS fornece abstração de hardware, controles de dispositivos de baixo nível, implementação de funcionalidades de uso comum, passagem de mensagens entre processos, gerenciamento de pacotes e ferramentas e bibliotecas para construção e execução do código em vários computadores. O ROS é similar ao Player, CARMEN e Microsoft Robots Studio, porém diferenciam em seus objetivos: não é objetivo do ROS possuir muitos recursos, mas sim apoiar a reutilização de códigos em pesquisas e desenvolvimentos [23].

  ◦ Player x ROS: Possuem conceitos semelhantes. O Player foi projetado para plataformas móveis simples e não articuladas, fornecendo acesso fácil aos sensores e motores equipados com laser oferecendo mais opções de drivers. Pelo fato do ROS ser projetado para plataformas móveis mais complexas, ele é mais complexo que o Player. O código do Player, Stage e Gazebo podem ser aproveitados no ROS [23].

- Stage: é um simulador de robôs compatível com o Player e com o ROS (figura H.1). Os robôs simulados são baseados no Pioneer fabricado pela empresa *MobileRobots*. Neste simulador, vários robôs e sensores podem ser simulados ao mesmo tempo, controlados por um ou vários clientes. O Stage pode ser usado tanto em robôs virtuais quanto em robôs reais, o cliente consegue distinguir esses dois tipos possíveis de robô através da rede TCP/IP [47].

- Gazebo: é um simulador de robôs em 3D compatível com o Player e com o ROS (figura H.2), possibilitando uma simulação mais real, porém, requer mais recursos computacionais, limitando a quantidade de robôs simulados ao mesmo tempo. Normalmente é utilizado quando for necessário a precisão real do robô no ambiente, o Stage, por fazer uma simulação bidimensional não traz essa precisão, como por exemplo, quando o robô precisa mapear ambientes externos ou quando o solo for irregular comprometendo o funcionamento dos sensores e dos robôs [47].

- Carmen: é um simulador de controle de plataformas robóticas e sensores para sistemas operacionais Linux, desenvolvido pela Universidade *Carnegie Mellon University*, possui o código aberto e disponibiliza alguns sensores, além de aplicações como navegação, mapeamento e localização (figura H.3). Permite simular robôs móveis e sensores lasers [55].

- ARIA (*Advanced Interface for Applications*): desenvolvido pela empresa MobileRobots, é um sistema de controle para robôs, dentre os robôs destaca-se o pioneer, muito utilizado para pesquisas (figura H.4). Este simulador funciona nos sistemas

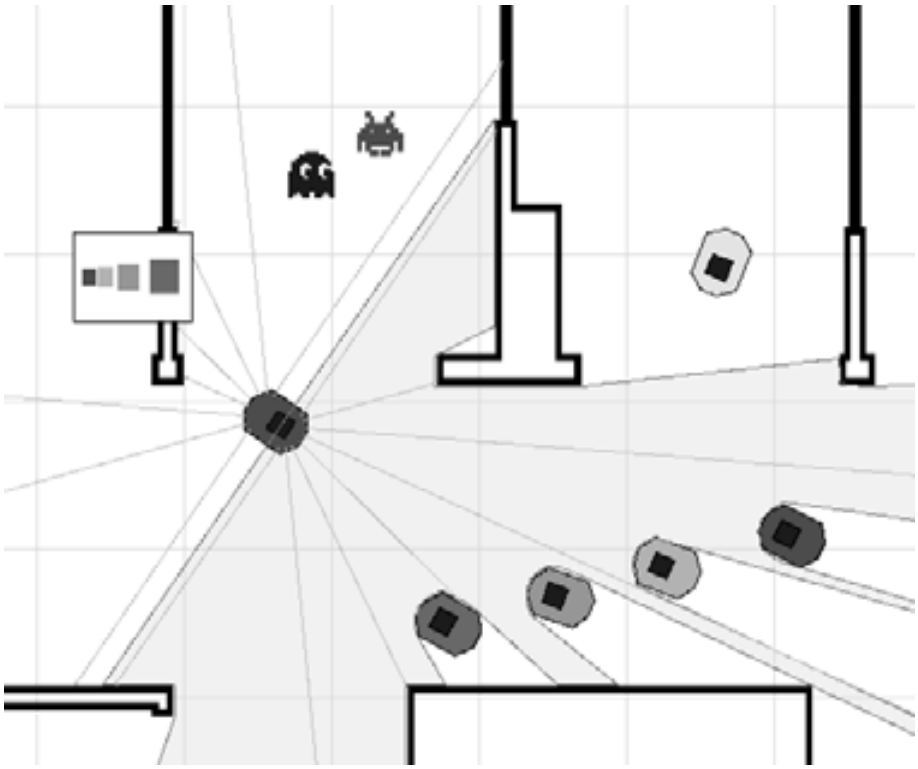

Figura H.1: Stage: Simulando robôs com sensores [47].

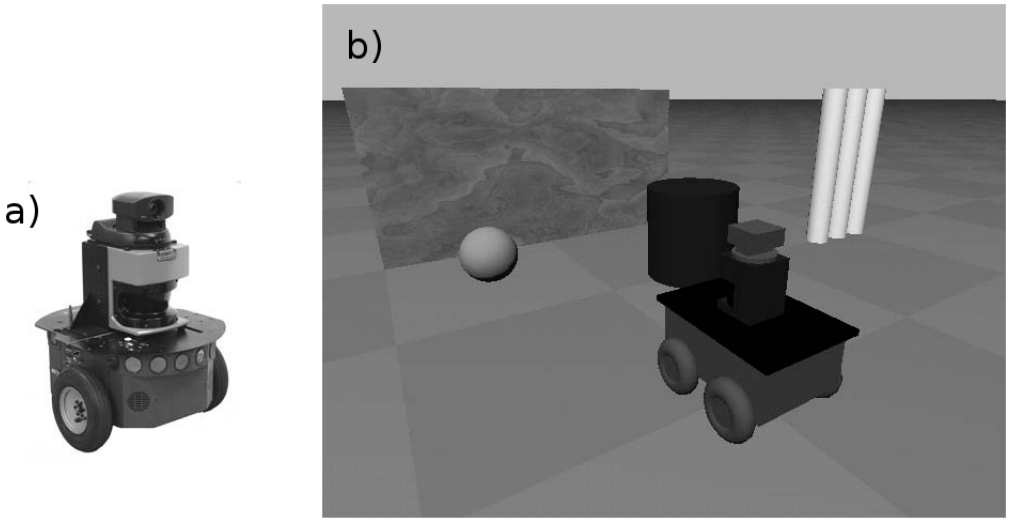

Figura H.2: Gazebo: a) Robô real e b) Robô simulado [47].

operacionais Linux e Windows. O simulador ARIA utiliza o MobileSim que é um simulador de ambientes baseado no Stage [46].

- Microsoft Robotics Studio (MRS): desenvolvido pela Microsoft, é compatível somente com o sistema operacional Windows e possui 3 versões: profissional, acadêmica e *express* (única versão de uso gratuito com restrições de uso). Este simulador tem como opção a simulação em ambientes 3D e a simulação física (figura H.5). Suporta as simulações de robôs humanóides, luta de robôs, braços mecânicos, futebol de robôs, entre outros [44].
- Webots: é um simulador desenvolvido pela empresa *Cyberbotics*, compatível com

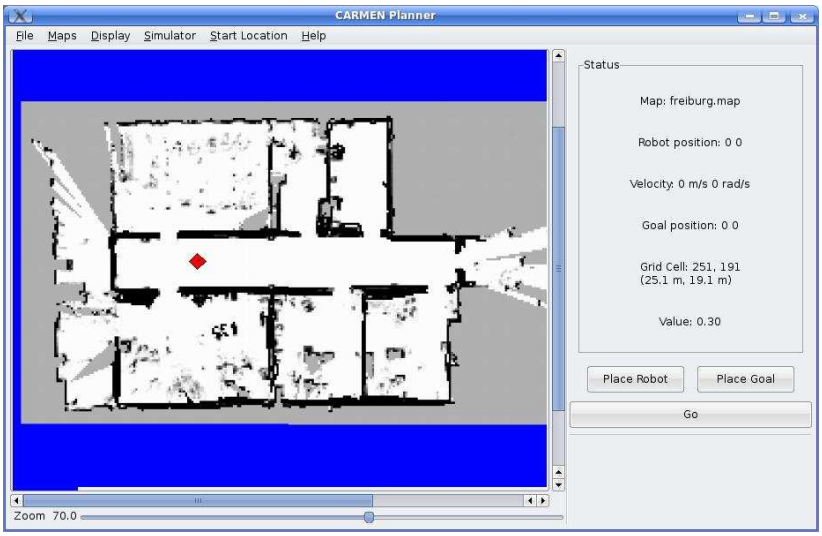

Figura H.3: Carmen [55].

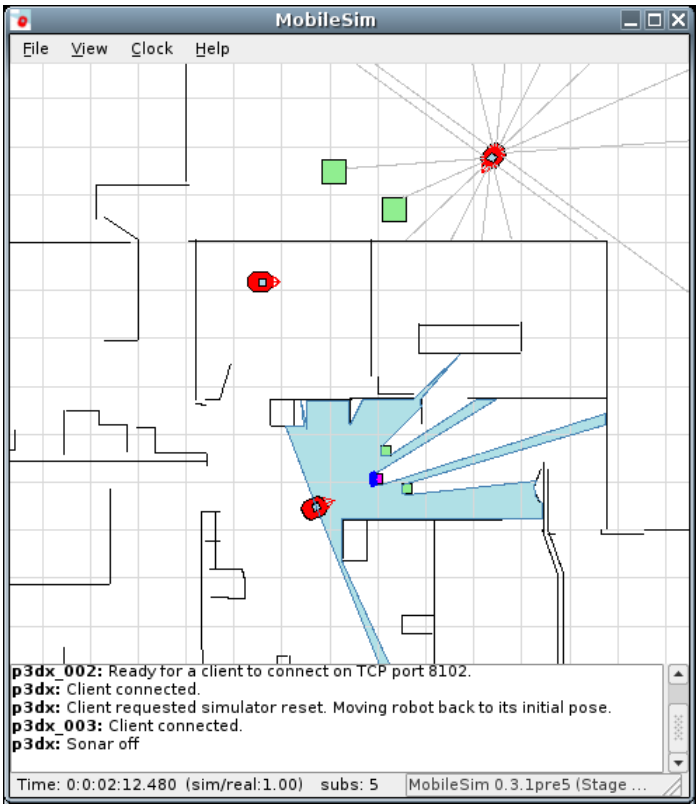

Figura H.4: ARIA [46].

plataformas Linux, Windows e Macintosh. Possui duas versões: a versão completa que necessita o pagamento de licença e a versão para teste que é gratuita. Este simulador permite modelar os robôs e o ambiente em 3D (figura H.6), e serve para modelar sistemas como veículos robóticos, robôs humanóides, futebol de robôs, cachorros robóticos, entre outros [20].

- RoboMind: é um simulador voltado para a educação, podendo ser adaptado o nível de dificuldade ao público (figura H.7). O ambiente é constituído por blocos, o robô é capaz de dirigir, olhar ao redor, mover itens e pintar. A linguagem de programação

Figura H.5: Microsoft Robotics Studio [44].

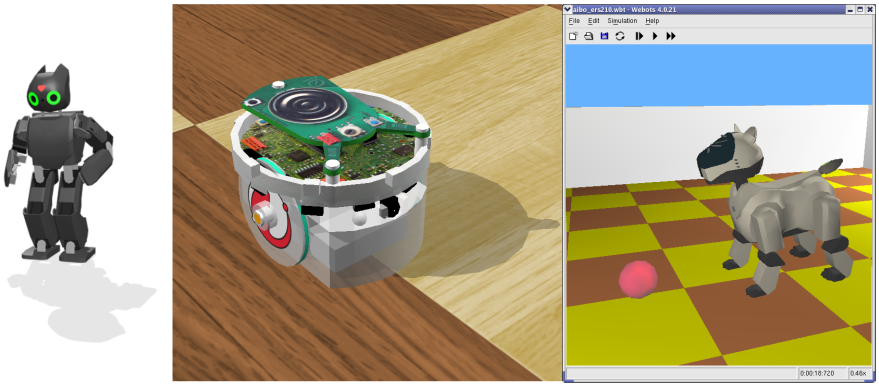

Figura H.6: Webots [20].

é um conjunto de instruções básicas que controlam o robô [45].

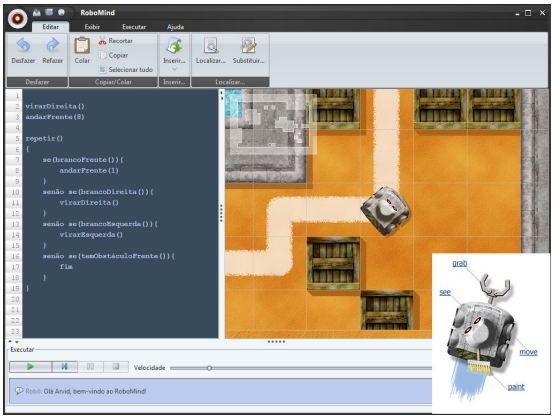

Figura H.7: Robô Mind [45].

- Studio Uno: desenvolvido pela UNO Robótica, é um ambiente fácil e divertido para a programação do robô UNO, baseia-se em programação por blocos, onde os comandos são arrastados para a área do programa e encaixando diferentes blocos entre si. É um programa gratuito para crianças a partir dos oito anos de idade (figura H.8). Para novos usuários, é possível programar o robô sem precisar digitar linhas de código e para usuários mais avançados podem importar bibliotecas de código ou digitar comandos na linguagem C. Após realizar a compilação, o programa é transferido pela conexão USB para a memória do robô [52].

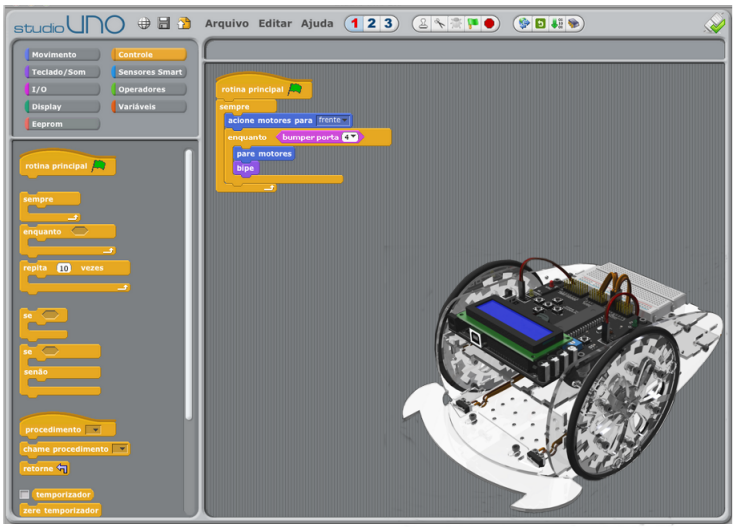

Figura H.8: Studio Uno .

[52]

- Virtual Robot Simulator (VRS): foi desenvolvido em 1988 pela *European Commission Joint Research Centre*, é gratuito e roda no sistema operacional Microsoft Windows (figura H.9). Este simulador é composto por: um modelador geométrico básico *VirtualRobot Modeller* (VRM); um tradutor de dados geométricos, *VirtualRobot Translator* (VRT); e a principal plataforma de simulação, *VirtualRobot Simulator* (VRS). O VRS é utilizado tanto para fins industriais quanto para fins educacionais e possui acompanhamento online para monitorar o estado do processo e robôs [24] [43].

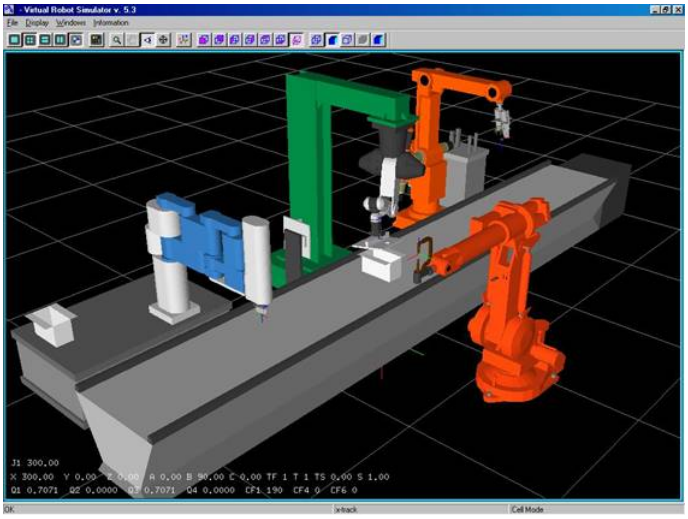

Figura H.9: Virtual Robot Simulator [24].