BRUNO LUIS SENON DE CARVALHO
TUPÃ NEGREIROS

SIMULAÇÃO DE ALGORITMOS DE DISPERSÃO E MAPEAMENTO PARA ENXAME DE ROBÔS

SÃO PAULO
2007

BRUNO LUIS SENON DE CARVALHO
TUPÃ NEGREIROS

SIMULAÇÃO DE ALGORITMOS DE DISPERSÃO E MAPEAMENTO PARA ENXAME DE ROBÔS

TRABALHO DE FORMATURA
APRESENTADO À ESCOLA
POLITÉCNICA PARA OBTENÇÃO DO
TÍTULO DE ENGENHEIRO

ÁREA DE CONCENTRAÇÃO:
ENGENHARIA MECATRÔNICA

ORIENTADOR: PROF. DR.
JUN OKAMOTO JR.

SÃO PAULO
2007

FICHA CATALOGRAFICA

Carvalho, Bruno Luis Senon de
Simulação de algoritmos de dispersão e mapeamento para enxame de robôs / B.L.S. de Carvalho, T. Negreiros. -- São Paulo, 2007.
51 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Enxames de robôs 2.Algoritmos de dispersão 3.Simulação de robôs I.Negreiros, Tupã II.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos III.t.

AGRADECIMENTOS

Agradecemos a todos o apoio para realizar não só esse trabalho como o apoio durante todo o curso. Agradecemos, em especial, a Vitor Guizilini por nos ter ensinado a usar o simulador, Ronaldo Viere por ter feito o vídeo da simulação e ao Professor Jun Okamoto Jr. pelo apoio na realização desse projeto.

RESUMO

O trabalho é baseado na simulação de algoritmos de dispersão e mapeamento para enxame de robôs. A simulação consiste em colocar um enxame de 30 robôs em um ponto inicial de um ambiente, uma casa com paredes, na qual eles terão a tarefa de, primeiramente, se distribuir pelo ambiente de uma forma otimizada e depois, através de informações obtidas por sensores de distância, criar o mapa do ambiente. Foram desenvolvidos e implementados dois algoritmos diferentes de dispersão, simulados em paralelo para se obter uma comparação entre eles. A ferramenta para a simulação utilizada foi o Player/Stage, utilizando programação em C++. Os robôs utilizados são simulações dos robôs do laboratório de Microprocessadores.

Palavras chave: Enxames de robôs, Algoritmos de dispersão, Simulação de robôs.

ABSTRACT

The paper is based on the simulation of dispersion and mapping algorithms for swarm robots. The simulation consists in placing a swarm of 30 robots in an initial point of an environment, a house with walls, where they will have the task of, at first, distributing itself in a optimized form and later, through information gotten from sensors, create the map of the environment. It will be developed and implemented two different algorithms of dispersion; they will be simulated in parallel to get a comparison of them. The tool for the simulation is software Player/Stage, programmed in C++. The robots used in the simulation are the ones of the laboratory of Microprocessors.

Keywords: Swarm robots, algorithms of dispersion, simulation of robots.

LISTA DE ILUSTRAÇÕES

Índice

1 INTRODUÇÃO

Em robótica autônoma, a chamada robótica coletiva ou robótica de enxame é baseada em metáforas e inspiração tomada de sistemas biológicos, em particular de insetos sociais, para o projeto de sistemas de controle distribuído ou estratégias de coordenação para grupos de robôs. Comportamentos coletivos de insetos sociais fornecem fortes evidências de que sistemas compostos por agentes simples são capazes de realizar tarefas complexas específicas. Sabe-se, entretanto, que as capacidades cognitivas destes insetos são muito restritas. Sendo assim, os comportamentos complexos que surgem devem ser propriedades emergentes resultantes das interações dos agentes e deles com seu ambiente, onde cada agente geralmente segue regras comportamentais simples.

Existe um crescente interesse pela robótica de enxame nos últimos anos. Isso acontece porque algumas tarefas são inerentemente muito complexas de serem resolvidas por um único robô e melhorias de desempenho podem ser conseguidas utilizando-se múltiplos robôs. Além disso, a construção e utilização de robôs simples é geralmente muito mais barata, flexível e tolerante a falhas do que projetar um único robô com alta capacidade de processamento de informação e sensores complicados.

A simulação do comportamento de robôs também vem surgindo como uma alternativa interessante para o desenvolvimento de software. Com este método não é necessário o protótipo do robô, o que diminui o custo do processo de desenvolvimento de software. Muitas vezes o protótipo não é viável tecnicamente de ser produzido e testado, especialmente no caso de enxame de robôs. A simulação permite, portanto, o desenvolvimento do software onde o robô simulado não é viável de ser produzido com a tecnologia atual, mas que poderá ser num futuro próximo.

O algoritmo de mapeamento serve como base para inúmeras atividades que podem ser realizadas por robôs. Simples robôs que se movimentam em uma área definida necessitam de prévio mapeamento do local. Sistemas mais complexos com enxames de robôs usados para localizar pessoas em incêndios ou desmoronamentos necessitam de algoritmos de mapeamento como os abordados neste Projeto.

2 PESQUISA

Podem existir dois tipos básicos de controle usados em enxame de robôs, eles são: o controle centralizado e o controle descentralizado. No sistema centralizado os robôs comunicam-se com um sistema de controle centralizado e executam-se de acordo com ordens desse sistema. No sistema descentralizado cada robô toma a decisão através das informações de sua vizinhança, cada robô segue um algoritmo de decisão.

O sistema de controle centralizado não é o mais eficiente para o controle de enxames, porque a comunicação é limitada pelo peso e pelo poder elétrico dos robôs, e é muito importante reduzir custos de uma comunicação. Além disso, como o tráfego de comunicação aumenta conforme o número de robôs aumenta, fica difícil para que os robôs executem as tarefas que requerem a cooperação em tempo real. Também é, geralmente, difícil definir uma função global apropriada que atribua comportamento otimizado e racional a todos os robôs.

Outra vantagem do controle descentralizado é a independência. No caso de uma falha com o sistema de controle centralizado, todo o enxame pode interromper sua execução. No controle descentralizado, não há um sistema central passível de falhas e caso haja uma falha num dos robôs do enxame, esta falha pode ser facilmente detectada pelo resto do enxame e o enxame pode continuar a execução sem problemas.

No sistema de controle descentralizado os robôs seguem uma série de regras predefinidas, ele armazena na memória informações sobre o estado do ambiente em que se encontra, o robô não possui informações sobre o ambiente todo, somente sobre a parte onde está. Os robôs auto-avaliam seus comportamentos através de suas funções locais de avaliação e assim tomam a decisão de como cooperar com o resto do grupo.

Apesar de a comunicação ser menor no controle descentralizado, os robôs do enxame podem se comunicar entre si, usando robôs intermediários para retransmitir as informações. Por exemplo, no caso de mapeamento, os robôs podem informar uns aos outros os locais que já foram mapeados, mesmo não tendo um controle central; ou, em outro caso, os robôs podem informar suas posições entre si a fim de evitar colisões. Com isso também é possível reduzir bastante o peso e a potência elétrica dos transmissores, pois a distância fica reduzida.

Através das vantagens citadas, foi decidido adotar o controle descentralizado. Apesar de o controle ser descentralizado, um sistema central se fará necessário para receber a informação do mapeamento completo e unificá-lo.

Através do estudo do projeto foi possível distinguir o movimentação dos robôs em duas partes: dispersão e mapeamento. Ou seja, o movimento dos robôs deverá ser inicialmente de ocupar todo o ambiente com um espaçamento uniforme (dispersão), para finalmente fazer uma varredura do ambiente através dos sensores e unificar toda informação recebida por cada um dos indivíduos do enxame em um servidor (mapeamento). Durante o movimento também é necessário que cada robô identifique obstáculos e outros robôs e faça desvios conforme necessário. O desvio de obstáculos do ambiente faz parte do algoritmo de dispersão.

3 ALGORITMO DE DISPERSÃO 1: HEXÁGONO

O primeiro algoritmo utilizado é o do Hexágono. Nele o comportamento global desejado deve emergir da interação local entre agentes. Para conseguir esse comportamento são usadas as leis naturais da física. Por isso é adotado a chamada "física artificial". O movimento dos robôs é causado por forças virtuais. A vantagem dessa abordagem pode ser vista no mundo físico real onde é possível obter comportamentos complexos através de simples interações entre as entidades [2].

O algoritmo de dispersão adotado baseia-se na utilização de forças virtuais para a formação de um hexágono. A construção de hexágonos pode ser feita através dos círculos de raio R como visto na Figura 3.1. Cada partícula repele outras partículas que estão a uma distância menor do que R e atraem as partículas que estão a uma distância maior do que R. Assim cada partícula tem seu campo de potencial ao seu redor, e partículas vizinhas vão ser separadas por uma distância R. A interseção desses círculos formam uma interferência construtiva onde estarão os nós de baixa energia potencial. Cada partícula sofrerá a influencia dos seus vizinhos mais próximos, num hexágono esses vizinhos estão a uma distância de $\sqrt{3}R$. A magnitude da força será calculada através da fórmula $F = G/R^2$, onde G é a constante gravitacional e adotamos que $F < F_{\max}$. Esse parâmetro F_{max} é adotado para poder restringir a aceleração dos robôs.

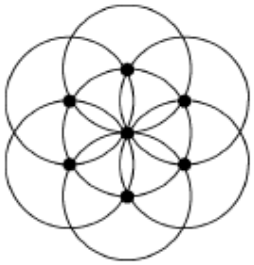

Figura 3.1 – Formação de um Hexágono.

Aplicando esse algoritmo não se obtêm um perímetro hexagonal perfeito, pois as forças atuantes são locais e não existe nenhuma condição global que permita a obtenção de um hexágono perfeito. Além disso, temos o problema de aglomeração, isso é um nó apresenta mais de uma partícula. Esses problemas podem ser vistos na figura 3.2.

Figura 3.2 – Dispersão em de partículas em forma de hexágono.

Forças grandes resultam em campos potenciais altos, permitindo que as partículas adotem uma forma estável, mas com múltiplas partículas aglomerando-se em nós de baixo potencial. Nesta situação a formação é parecida com a de um sólido. A transição de fase ocorre quando os campos de potencial diminuem. Neste momento, as forças que promovem a fragmentação da aglomeração são mais fortes do que as forças que promovem aglomeração. Nesta situação a formação é parecida com a de um líquido. Precisamos agora fazer uma analise quantitativa para sabermos o valor de G que será adotado.

Sabemos que a energia potencial é calculada por $P = -\int_s \vec{F} \cdot d\vec{s}$. Quando usamos um G pequeno, o campo de energia potencial fica menor e assim uma partícula fica cercada por pontos com baixo potencial. Supondo duas partículas no mesmo nó, uma partícula A deverá sair dessa posição adotando uma rota que não entre em conflito com o perímetro das partículas ao seu redor. Como a partícula A está muito próxima de outra partícula, agirá sobre ela uma força de repulsão $F_{\max}$. Ela é mantida no centro por seis partículas que estão ao seu redor. Se A se mover ligeiramente para a direita (ou esquerda), duas partículas a puxarão para o centro (atração), e duas partículas a empurrarão de volta para o centro (repulsão). Para cada partícula, o valor desta força aplicada em A é de G/R^2. A projeção desta força o eixo horizontal é $\sqrt{3}G/2R^2$ (porque o ângulo em relação a esse eixo é de 30º). Quando a força da coesão é maior que a força da fragmentação, há uma aglomeração no centro. Quando a força da fragmentação é maior, o conjunto central se separa. A transição da fase ocorre quando as duas forças estão no contrapeso:

$F_{\max} = \frac{2\sqrt{3}G}{R^2}$. Assim, temos $G_v = \frac{F_{\max} R^2}{2\sqrt{3}}$.

Ao iniciar a simulação, os robôs estão aglomerados em um canto da sala. As forças de repulsão irão começar a agir e fazer com que eles se distribuam na sala. O deslocamento dos robôs será parecido com o deslocamento das moléculas de um liquido. E eles vão se distribuir até chegar a uma condição de equilíbrio, onde eles vão ter preenchido toda a sala.

Para aperfeiçoarmos o mapeamento do ambiente os robôs deverão ser atraídos para os obstáculos e ficarem a uma distância de metade do alcance máximo do sensor de infravermelho. Para isso, serão usadas forças virtuais que fazem com que os robôs que estejam a uma distância maior que metade do alcance máximo do sensor sejam atraídos para o obstáculo, enquanto que robôs que estão a uma distância menor que essa sejam repelidos.

Primeiramente, o robô deverá determinar a sua posição para em seguida fazer uma varredura para determinar quais robôs estão a uma distância de 1,5 R, onde R é o raio da circunferência que formará os hexágonos. Após determinar os robôs que estão na vizinhança o robô fará a leitura dos sensores para determinar a distâncias dos obstáculos. Para cada robô da vizinhança será calculada uma força que atua somente no robô que está calculando o algoritmo. Para robôs que estão a uma distância menor que R a força será de repulsão enquanto que robôs que estão a uma distância maior que R a força será de atração. Os obstáculos determinados pelos sensores também irão criar forças virtuais, de repulsão se o obstáculo estiver próximo do robô e de atração se o obstáculo estiver afastado do robô. Depois de determinadas as forças, será determinada a somatória dessas forças e, através da resultante, é determinado o ângulo que o robô deve girar e andar. Depois de andar, o robô executará todos esses cálculos novamente. O robô só sairá desse ciclo depois de atingir a condição de parada.

Na implementação desse algoritmo os parâmetros utilizados foram:

- Robôs a uma distância maior que 1,7 m de um outro robô são atraídos e com um raio menor que 1,7 m são repelidos.
- Somente robôs que estão a uma distância menor que 2,55 m interagem e criam forças virtuais entre si.

- Para o cálculo da força virtual que um robô aplica no outro foi utilizado o valor de 370 para $G_{robô}$, depois de alguns ensaios foi percebido que a força de atração entre os robôs impedia que eles ocupassem toda a sala, então, depois de alguns ensaios, foi determinada que para força de atração o valor de $G_{robô}$ seria dividido por 1,3 obtendo 285.
- Obstáculos que estão a uma distância menor que 1,0m provocam uma força de repulsão nos robôs e obstáculos que estão a uma distância maior que 1.4m provocam uma força de atração, obstáculos que estão a uma distância entre 1,0m e 1,4m não provocam nenhuma força no robô.
- Para o cálculo das forças virtuais causadas por obstáculos foi usado 250 para G_{parede}, esse valor foi menor que $G_{robô}$ porque o algoritmo deve dar prioridade à dispersão, o valor de $G_{robô}$ maior que G_{parede} permite que os robôs sejam repelidos entre eles e entrem em quartos fechados sem que as paredes que constituem as portas os repilam da entrada.
- Os robôs, depois de determinada a direção do movimento, andam uma distância fixa. Se o sensor da frente do robô detectar obstáculos a uma distância menor que 0.2m ele não irá andar, se esse mesmo sensor obter uma leitura maior que 0,8 m e os sensores laterais obterem leituras maiores que 0,5m, o robô irá andar 0,23m. Se o sensor da frente obtiver leituras maiores que 0,2m, os sensores laterais também captarem leituras maiores de 0,2m e o robô ainda não tiver andado 0,23m nessa iteração ele irá andar apenas 0,1m.
- São utilizados dois parâmetros de parada: o software deverá satisfazer ambos para que a dispersão seja concluída. A cada iteração é calculada a menor distância entre dois robôs, e esse valor deve ser maior que 1,2m para o robô atingir uma das condições de parada. Foi determinado que os robôs se repelissem quando estiverem a uma distância menor que 1,7m, por isso, foi adotado um valor menor que esse para a condição de parada, dado que nem sempre, devido a condições do ambiente, será possível que a distância entre eles seja a máxima. Foi observado que quando um robô chega a um ponto de equilíbrio, depois de três iterações quaisquer ele volta a uma posição próxima a que ele estava no inicio da primeira iteração, então, essa seria a segunda

condição de parada, foi utilizado que essa distância entre três iterações deveria ser menor que 0,32m.

4 ALGORITMO DE DISPERSÃO 2: DHM

Outro algoritmo de dispersão adotado foi o Modelo de Hormônio Digital (DHM). Nele existe uma distribuição fictícia no ambiente, com alusão a um hormônio virtual. Próximo a outros robôs a concentração desse hormônio é alta e em áreas vazias a concentração é baixa. Desse modo, basta programar o robô para que ele migre de áreas com alta concentração de hormônio para áreas com baixa concentração [3,4,5].

O DHM foi inspirado na biologia. Nesse modelo os robôs são vistos como células biológicas que se comunicam e colaboram via hormônios. Nesse modelo os robôs não têm um líder fixo, mas sim controlam suas ações a partir de um controle totalmente descentralizado. Eles podem se reorganizar automaticamente caso algum robô falhe.

Esse algoritmo possui várias aplicações, inclusive permitindo que os robôs se comuniquem através de mensagens em forma de concentração de hormônio. Ele pode ser usado para que os robôs procurem e se movam a determinado ponto alvo. Também pode ser usado para reorganização em ambiente para o caso de certa quantidade de robôs apresente falha. Nesse projeto o essencial é fazer com que os robôs se dispersem em determinada região, evitando os obstáculos.

Os robôs devem se dispersar em um ambiente desconhecido. Quando uma grande quantidade de robôs é colocada no ambiente, eles são orientados a entrar no ambiente vazio, a baixa concentração de hormônios faz com que eles sejam puxados para dentro. Cada robô tem grande probabilidade de se mover para longe de um lugar onde há grande concentração de hormônio, ou seja, onde há uma grande quantidade de robôs. Os robôs também não se afastarão demais devido a atração da baixa concentração de hormônio que fazem atrair uns aos outros. O equilíbrio entre forças repulsivas e atrativas asseguram a formação desejada.

Ao iniciar a simulação os robôs estarão todos concentrados no canto inferior esquerdo. Logo depois eles começam a ser atraídos pela baixa concentração de hormônio na borda da distribuição total. Em seguida, os robôs internos começam a ser atraídos pela baixa concentração deixada pelos robôs que já iniciaram seu movimento. Com o passar do tempo todos os robôs estarão se movendo em direção a área não mapeada. Os robôs serão então dispersos no ambiente.

Por uma outra ótica, podemos interpretar que as regiões com alta concentração de hormônio repelem os robôs, enquanto que as regiões com baixa concentração atraem os robôs.

Veja a Figura 4.1. Cada robô gera um campo de distribuição ao seu redor. O ponto central indica a posição do robô. Próximo a ele, em amarelo, a concentração do hormônio é alta, indicando aos outros robôs que esta região deve ser evitada. Na borda dessa distribuição aparece uma baixa concentração de hormônio. Essa região com baixa concentração é interpretada pelos outros robôs como um possível destino. Na implementação do algoritmo foi adotado a concentração de 10 no círculo central e 5 na coroa externa.

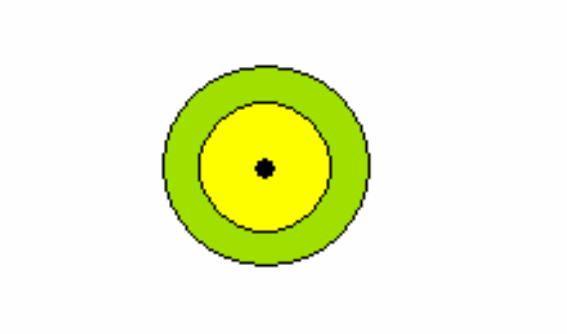

Figura 4.1 - Distribuição de Hormônio gerada por um único robô.

Ao iniciar o algoritmo, uma matriz deverá ser criada no controle central com a concentração de hormônio em cada ponto do ambiente. A concentração de hormônio será nula, exceto próximo a posição inicial dos robôs. Em cada robô deve-se criar um círculo com alta concentração de hormônio e uma coroa externa com baixa concentração.

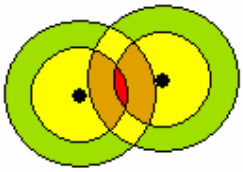

Figura 4.2 - Distribuição de Hormônio gerada pela interação de dois robôs.

Já na Figura 4.2 temos a distribuição de hormônio da interação de dois robôs. As concentrações se somam ponto a ponto. No exemplo da figura temos concentração de 20 na parte vermelha, 15 na parte laranja, 10 na parte amarela e 5 na parte

verde. Um destes robôs ao fazer uma varredura (que será explicada em detalhes adiante) deverá detectar a região em verde ao seu redor como melhor destino.
Os mesmos dois robôs ao se distanciarem atingem a configuração da Figura 4.3. Nota-se que não ocorre interação da influência dos hormônios. Esse então deve ser o critério de parada da dispersão. Um robô ao atingir um ponto onde ao seu redor só ocorra uma distribuição de hormônio gerada por ele mesmo deverá parar de se mover. Caso um outro robô se aproxime dele o ciclo de movimento irá se reiniciar.

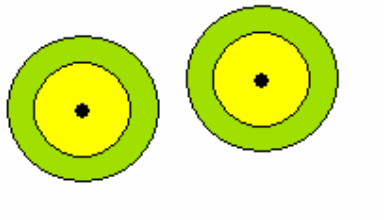

Figura 4.3 – Distribuição de Hormônio gerada por dois robôs afastados.

Cada Movimentação é executada da seguinte maneira:

- Registrar sua posição futura em uma variável visível a todos os robôs.
- Movimentar-se.
- Na posição inicial, retirar do mapa as concentrações de hormônio causadas por si próprio, o círculo de alta concentração e a coroa de baixa concentração.
- Na posição final, incluir novamente as concentrações de hormônio, agora no novo ponto.
- Registrar sua nova posição atual, também em uma variável visível a todos os robôs.

Cada robô executa os seguintes passos a cada Ciclo de Movimento:

- Verificar se o estado do robô ainda é de Mapeamento.

- Determinar sua posição.

- Varrer uma circunferência de raio 1,0m ao redor da sua posição, procurando por um ponto na região com menor concentração de hormônio.

- Girar em direção a esse ponto.

- Fazer uma leitura do sensor frontal. Caso exista uma leitura menor do que 0,5m, retornar ao início sem se movimentar.

- Movimentar 0,1m em frente.

- Gravar a distância percorrida nesta iteração.

- Retornar ao início.

O Ciclo de Varredura descrito acima é executado da forma abaixo. A Figura 4.4 complementa a explicação.

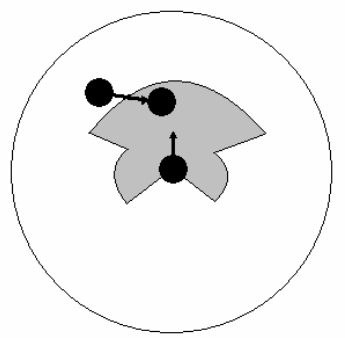

Figura 4.4 – Exemplo de Ciclo de Varredura.

- Varrer uma região circular de raio 1,0m ao redor da posição atual do robô. Determinar uma região de pontos onde a concentração de hormônios é menor.

- Escolher aleatoriamente um destes pontos

- Tendo esse ponto como destino, verificar se a sua frente, a uma distância de 0,5m e um ângulo de -60° a +60°, algum robô tem sua posição atual ou futura. Caso positivo, retornar ao início.

- Verificar também uma região em cada um dos lados, a uma distância de 0,2m e ângulo entre 60° e 120°. Caso positivo, retornar ao início.

Dada esta implementação, alguns pontos importantes devem ser ressalvados.
O algoritmo certamente não é determinístico, pois ocorre escolha aleatória de um ponto dentro de uma região.
A verificação de um obstáculo à frente, seja um robô detectado durante o Ciclo de Varredura, ou uma parede detectada pelo sensor durante o Ciclo de Movimento, faz o retorno a varredura das regiões de baixa concentração e posterior escolha aleatória de pontos. Ou seja, o robô ficará escolhendo pontos até que seja encontrado um ponto onde não ocorram obstáculos próximos, ou que o robô a sua frente se retire.

5 ALGORITMO DE MAPEAMENTO

O algoritmo de mapeamento será iniciado ao término do algoritmo de dispersão. Não ocorrerá mapeamento durante a movimentação dos robôs. Os robôs podem estar, portanto, em dois estados distintos: Dispersão e Mapeamento.

Todos os robôs deverão mudar de estado ao mesmo tempo. A mudança ocorre quando a maior das últimas distâncias movimentadas por cada robô for menor que determinado intervalo. Isso significa que todos os robôs estão parados (ou praticamente parados, dentro de um intervalo). Mais detalhes destes critérios de parada foram dados na descrição dos algoritmos.

Essa mudança de estado é verificada por uma função no controle central. Ao verificar que os robôs estão todos parados essa função informa os robôs que devem mudar de estado. Finalizado o estado de Dispersão os robôs entram em estado de Mapeamento.

O algoritmo de mapeamento em si constitui de:

- Fazer a leitura dos quatro sensores e armazenar os dados;
- Girar de 1º em 1º armazenando as informações dos sensores, até chegar a +90º.

As leituras dos sensores são registradas em uma matriz booleana. Inicialmente a matriz não registra nenhum obstáculo. A cada leitura do sensor indicando um obstáculo, esse obstáculo é registrado na posição indicada como um ponto na matriz booleana.

Os robôs detectam uns aos outros como obstáculos no mapa. Após a varredura temos de excluir do mapa os próprios robôs. Cada posição atual do robô deve ser marcada na matriz booleana como espaço vazio.

Ao final do mapeamento temos uma matriz booleana representando o ambiente sem os robôs que executaram o mapeamento. Uma figura no formato .PGM é exportada pelo programa, permitindo a visualização pelo usuário.

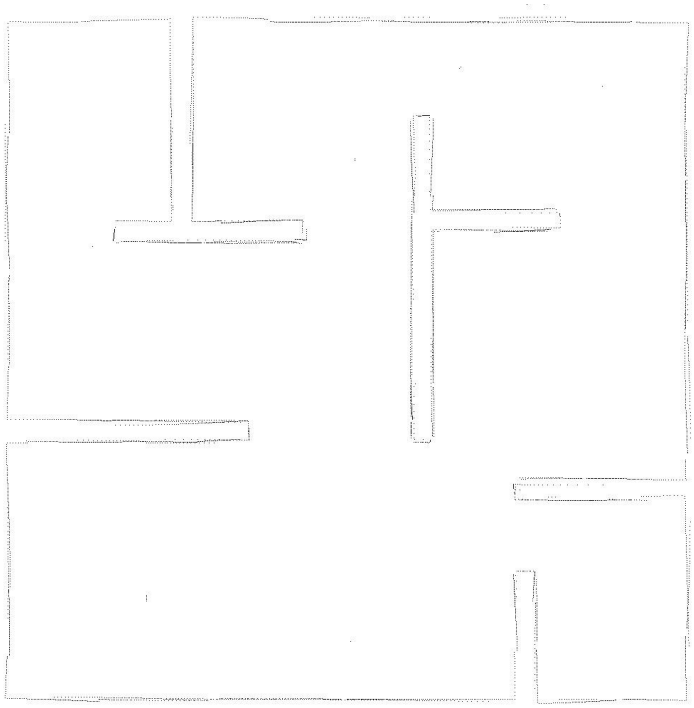

Figura 5.1 – Um exemplo de mapeamento obtido.

6 O SIMULADOR

O software utilizado nessa simulação foi o Player/Stage [1]. Pode ser utilizado nas plataformas Linux, Solaris, BSD e MacOS, no caso foi utilizado Linux. Permite o controle e simulação de múltiplos robôs, o que é essencial na escolha de um simulador para este projeto. É um software gratuito distribuído sobre licença GNU.

Player é um pacote de bibliotecas de interface de controle para uma grande variedade de robôs e sensores. O modelo do Player permite que programas de controle desenvolvidos nele se conectem a robôs reais através de conexão em rede ou mesmo em robôs simulados. A programação do Player pode ser feita em C++, Tcl, Java e Python, no caso foi utilizado C++.

Stage é um simulador de robôs em ambiente bidimensional. Vários modelos de sensores podem ser utilizados na simulação operando da mesma forma que um robô real em um ambiente real. Dessa forma a programação feita no Player pode ser utilizada sem modificações para controle de robôs reais.

Houve uma tentativa no início do projeto de utilizar o Robotics Studio da Microsoft. O Robotics Studio lançou sua primeira versão oficial em Novembro de 2006, com algumas versões de teste (Beta) lançadas ao longo de 2006. Porém, muito pouco material de suporte ao Robotics Studio foi lançado. Nenhum manual oficial foi produzido e nem se tem nenhuma previsão para tanto.

A qualidade gráfica do Stage é inferior comparada com outros simuladores. É baseado em gráficos 2D, enquanto que o Gazebo, que é do mesmo projeto de desenvolvimento, é baseado em gráficos 3D. Já o Robotics Studio utiliza a interface DirectX para gerar gráficos 3D, possuindo uma qualidade superior até ao Gazebo. Essa qualidade foi a motivação que levou a tentativa de utilização do Robotics Studio.

7 O ROBÔ

O robô base utilizado nesse projeto é o robô desenvolvido pelo LPA utilizado no Laboratório de Microprocessadores.

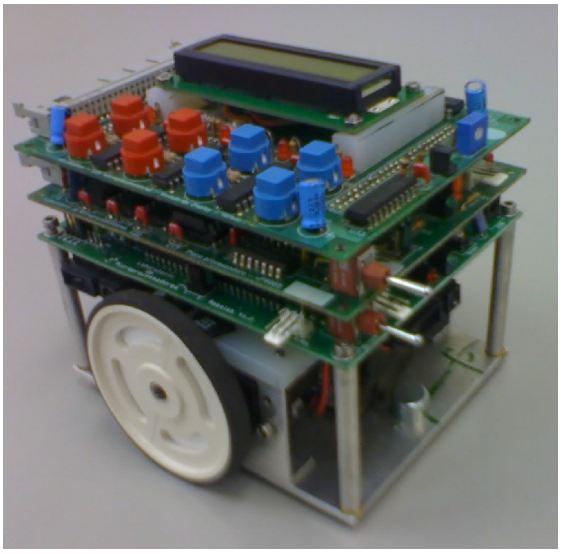

Figura 7.1 – Foto do Robô do Laboratório.

Para utilizá-lo no simulador do simulador Stage foi feita uma aproximação simplificada. A parte azul representa o chassi do robô e as linhas cinza que saem do robô são os sensores de distância.

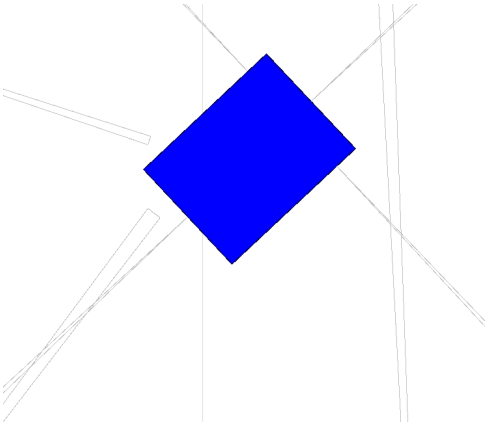

Figura 7.2 - Concepção do Robô no Stage.

As métricas do robô são: 130 mm de comprimento, 100 mm de largura e 90 mm de altura. A roda tem medidas de 68 mm de diâmetro e 3 mm de largura.

O Robô em questão possui duas rodas, com um motor em cada roda, no conceito conhecido como Motor Diferencial. Isso permite que o robô faça rotações sem se

mover, bem como se mover nas duas direções em linha reta ou em curva. Junto aos motores estão encoders de posição que permitem obter a rotação dos motores e posterior posição do robô no ambiente.

O Robô também possui quatro sensores de distância infravermelho, um situado em cada face do robô. Essa disposição de sensores permite uma varredura do ambiente ao seu redor a partir da rotação do robô. Os sensores têm um alcance máximo de 40cm e um ângulo de abertura de 15°. A partir do aprofundamento do tema foi visto que esse tipo de sensor seria insuficiente para mapear um ambiente muito grande. Devido ao pequeno alcance cada robô consegue varrer uma área muito pequena, dessa forma seriam necessários muito mais robôs. Cálculos preliminares indicaram que para mapear uma área de 10m x 10m seriam necessários mais de 1000 robôs.

Como os robôs são simulados, existe a liberdade de alterar suas características sem nenhum custo ou dificuldade. Foi decidido alterar o sensor por um sensor de distância com alcance máximo de 2m e ângulo de abertura de 1º. Desse modo, 30 robôs seriam suficientes para mapear a área de 10m x 10m.

8 CONFIGURAÇÃO DO SIMULADOR

Primeiro foi configurado o ambiente. Foram desenhados mapas na forma de imagens .PNG preto e branco, onde as partes pretas da imagem representam paredes e as partes brancas representam ambiente vazio. Foi descrito em um arquivo .WORLD os parâmetros de configuração do ambiente como tamanho, zoom e janela; bem como foram colocados cada um dos robôs com sua posição inicial.

O ambiente configurado foi baseado na planta de uma casa comum, de tamanho 10m por 10m. Veja a Figura mostrando a planta do ambiente visto de cima.

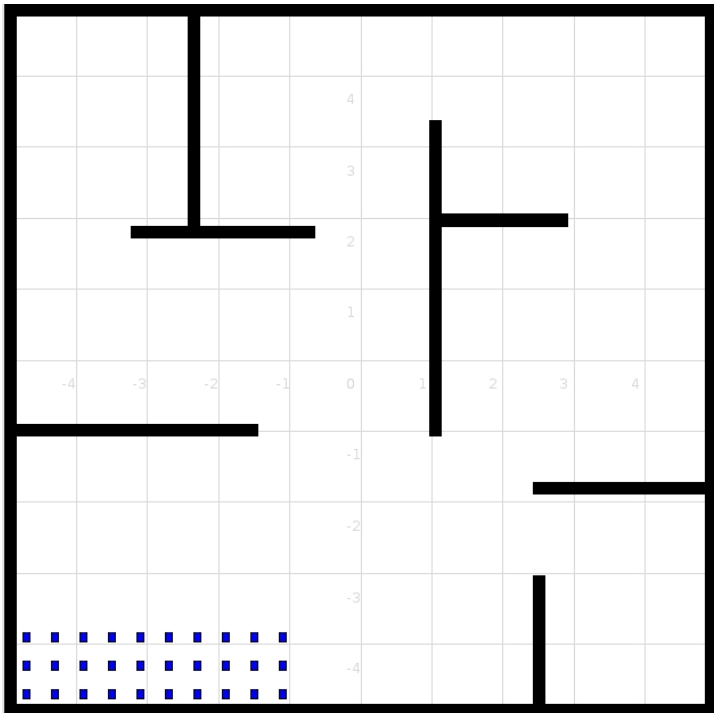

Figura 8.1 – O ambiente.

Foi configurado um arquivo .INC com os parâmetros do robô. O robô foi aproximado por um polígono retangular de comprimento 130mm e largura 100mm. Foram colocados os sensores de distância como sonares nas posições 0°, 90°, 180° e -90°. Os parâmetros dos sensores são distância mínima 0m, distância máxima 2m e ângulo de abertura de 1°. Foi definida a massa do robô como zero, dessa forma foi evitado erros de posicionamento devido a acelerações nos movimentos do robô.

Por fim foi configurado um arquivo .CFG unindo as configurações do ambiente com as do robô, declarando cada um dos robôs com seus componentes: posicionamento

e sonar. Com isso pode-se abrir o arquivo .CFG usando o software de simulação Stage.

Após a descrição dos robôs e do ambiente, foi escrito um programa de controle no software Player. O algoritmo de cada um dos robôs é executado em paralelo no processador do computador, na forma de threads. Como foi dito anteriormente, o controle dos robôs é descentralizado, ou seja, o controle de cada robô é executado pelo próprio robô. Algumas variáveis e/ou algumas rotinas podem ser centralizadas; como por exemplo as posições de cada robô é uma variável global que outro robô pode ler, funções de verificação de mudança de estado são executadas em modo centralizado.

Também como foi explicitado anteriormente, a movimentação dos robôs pode ser divida em dois estados distintos: Dispersão e Mapeamento. Não ocorre mapeamento enquanto os robôs estão se dispersando. Primeiro os robôs buscam se distribuir da maneira mais uniforme e o mais rápido possível, para em seguida fazer uma varredura do ambiente marcando as posições onde há leitura positiva dos sensores como obstáculos.

Na estrutura do programa do programa principal pode-se notar as seguintes divisões:

- Inicialização das variáveis
- Inicialização das threads
- Verificação de finalização do estado de Dispersão
- Verificação de finalização do estado de Mapeamento

Já na estrutura do programa das threads, os programas individuais de cada robô, pode-se notar as divisões:

- Algoritmo de Dispersão
- Algoritmo de Mapeamento

Foi tentado introduzir uma maior quantidade de robôs no enxame estudado. A princípio a idéia do projeto era utilizar 100 robôs. Por algum motivo, possivelmente por limitações de memória ou processamento, a quantidade máxima de robôs possíveis foi de 35 robôs. Por julgar que 30 robôs seriam necessários para mapear o ambiente descrito, foram utilizados 30 robôs.

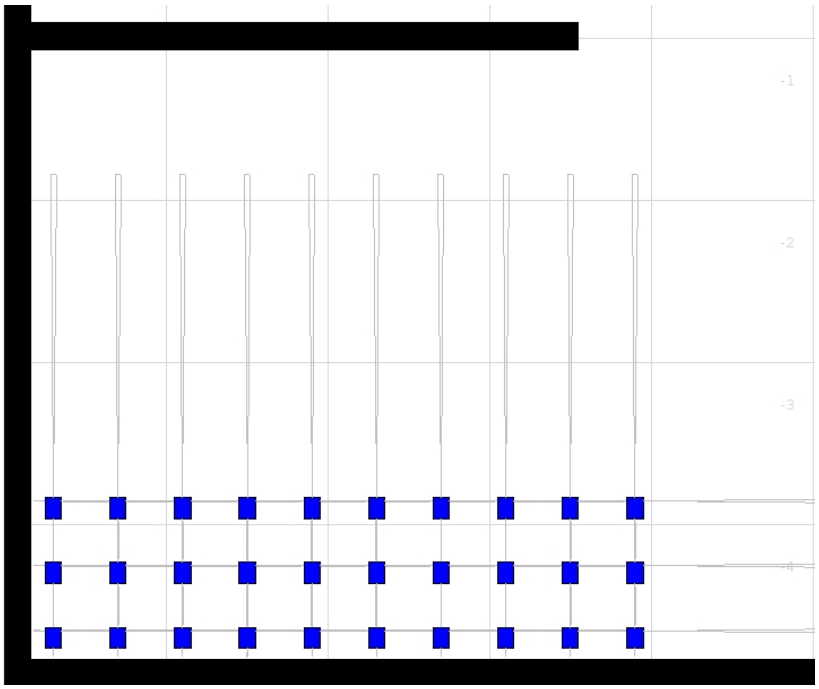

Figura 8.2 - Múltiplos robôs no ambiente de simulação.

9 ENSAIOS

Foram realizados ensaios para refinamento e estudo dos algoritmos. Três ambientes foram utilizados para teste, um ambiente sem paredes para verificação do algoritmo e os outros dois algoritmos simulavam a planta de uma casa normal. As plantas dos ambientes estão abaixo.

O ambiente 1 foi o ambiente utilizado para implementação dos algoritmos e para fazer a comparação entre os algoritmos. Desse ambiente foram gravadas imagens da simulação de uns dos algoritmos para montagem de um vídeo para a apresentação, as imagens foram gravadas de 10 em 10 segundos. O ambiente 2 foi utilizado para verificar a eficácia de operação dos algoritmos em outro ambiente.

Nos resultados adiante pode-se notar em ambos os algoritmos que no canto superior direito, do lado oposto onde os robôs iniciam o movimento, existe maior dificuldade na dispersão dos robôs.

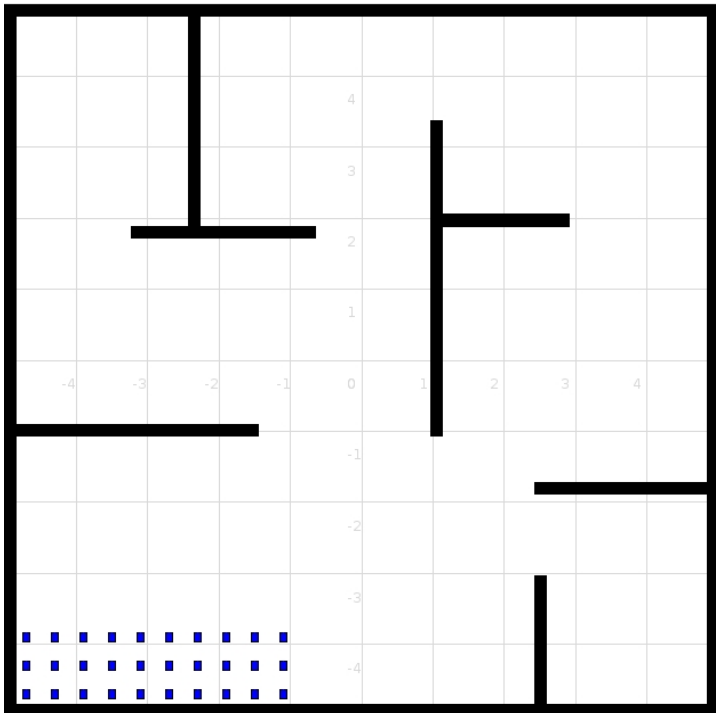

Figura 9.1 - Ambiente 1.

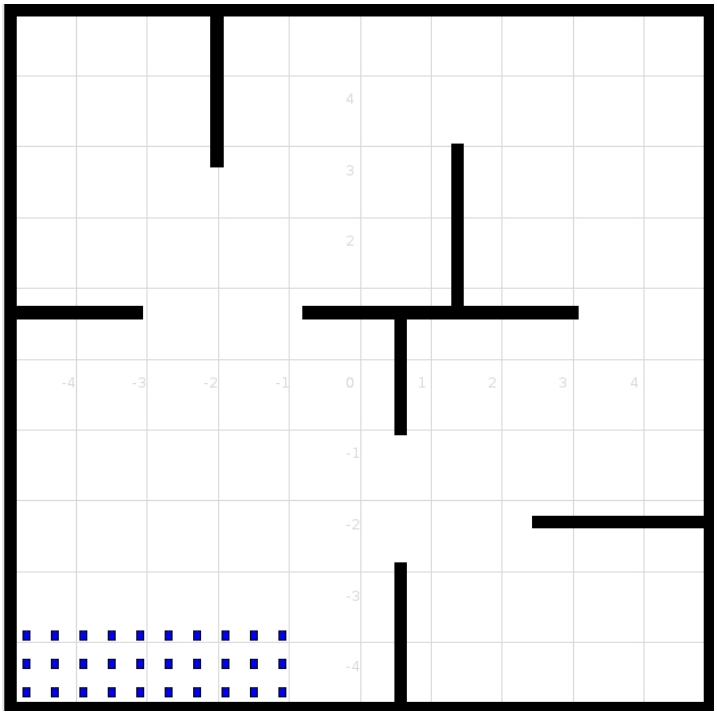

Figura 9.2 - Ambiente 2.

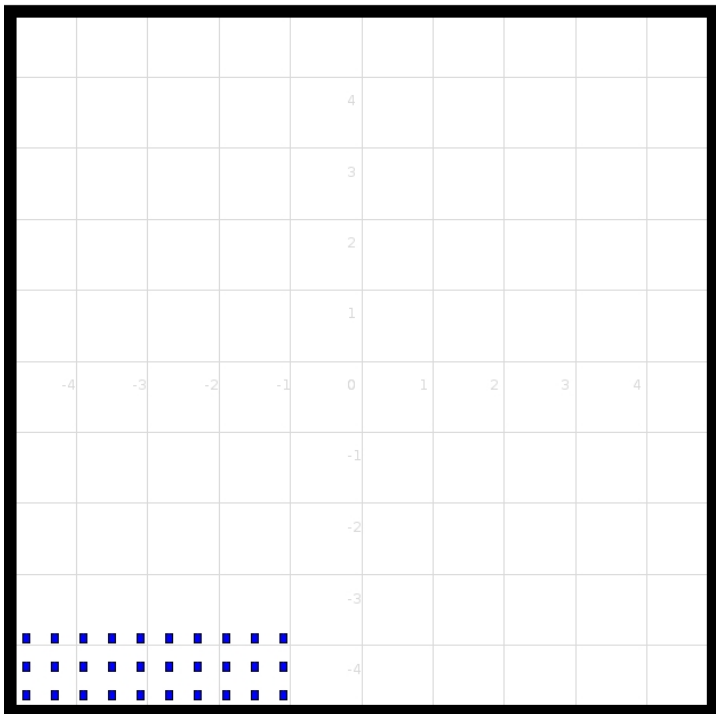

Figura 9.3 - Ambiente 3.

10 RESULTADOS DO ALGORITMO 1: HEXÁGONO

A simulação do algoritmo Hexágono no Ambiente 1 demorou 3 horas e 50 minutos para a condição de parada ser atingida. Os parâmetros de paradas foram: menor raio: 1,2045 m e distância entre iterações de 0,305413.

Entretanto, podemos observar, na figura 10.1, que depois de passados 1 hora e 30 minutos de simulação, os robôs já conseguiram sair da sala inicial e já preencheram quase todo o ambiente.

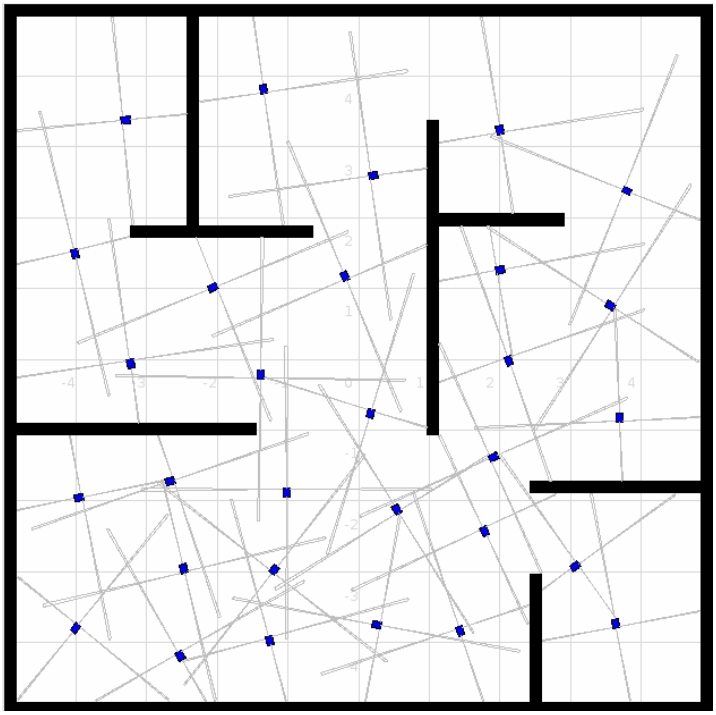

Figura 10.1 - Simulação depois de 1 hora e 30 minutos.

No início da simulação o enxame se dispersa rapidamente, devido à proximidade dos robôs na posição inicial. Depois de se espalharem, eles passam a se dispersar mais lentamente, isso acontece porque começam a agir as forças de atrações entre os robôs, isso impede que eles se desloquem a locais muito afastados rapidamente, eles são repelidos pelo robô mais próximo e, depois de se afastarem, são atraídos de volta por outros robôs mais afastados, isso acaba diminuindo muito a velocidade de dispersão.

Depois de 2 horas de simulação, os robôs apresentam uma distribuição semelhante à de 1 hora e 30 minutos, observa-se ainda, que nenhum dos sensores dos robôs

consegue cobrir o canto superior direito do ambiente. Podemos ver isso na Figura 10.2.

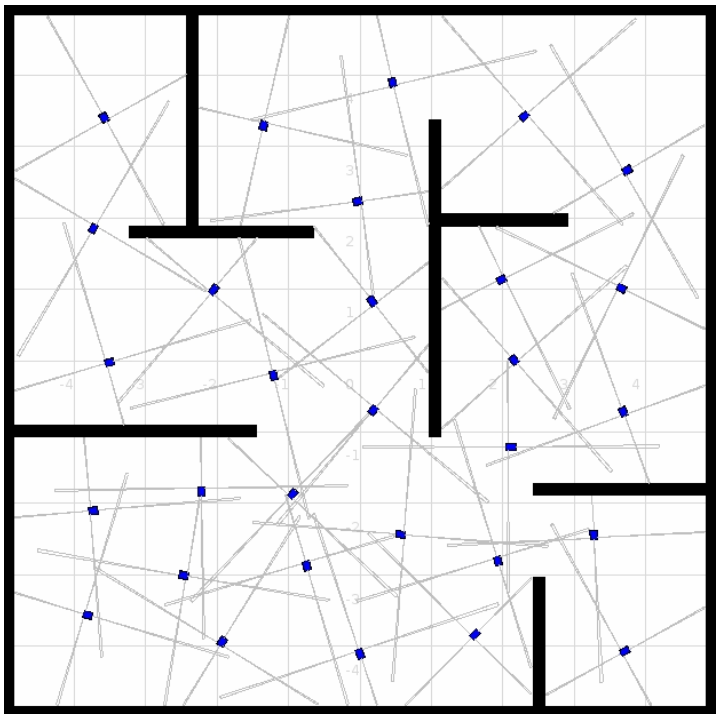

Figura 10.2 - Simulação depois de 2 horas.

O enxame tem dificuldades em se dispersar pela sala toda devido à força de atração entre eles, por isso, foi utilizado um fator que diminuísse essa força de atração. Sem, entretanto, diminuir demais essa força de atração, pois senão o enxame perderia a sua formação. Os robôs que estão nas primeiras fileiras iriam sair do grupo e chegar a lugares muito longe do resto do enxame, esse comportamento não é desejado, pois queremos cobrir todo o ambiente, e é preciso que os robôs andem juntos para explorar o local e não se percam do resto.

Depois de 3 horas e 50 minutos de simulação observa-se que os sensores dos robôs conseguem mapear o ambiente todo. Observa-se, também, que os robôs estão mais espalhados nas outras salas, e pelas condições de parada eles já não estão mudando de posição.

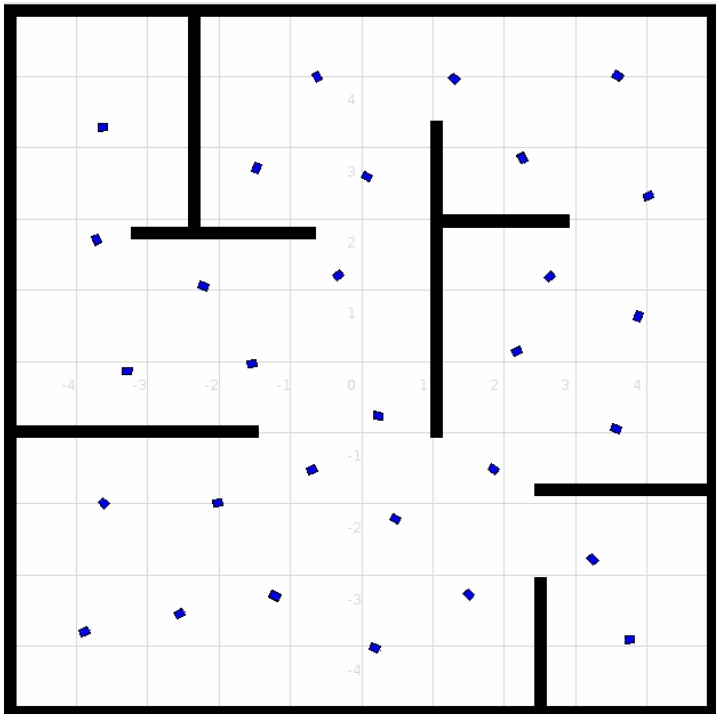

Figura 10.3 – Fim da Simulação.

Ao fim da dispersão, através do algoritmo de mapeamento, foi gerado um mapa do ambiente como pode ser visto na figura 10.4. Os robôs conseguiram fazer o mapeamento com precisão de todo o ambiente.

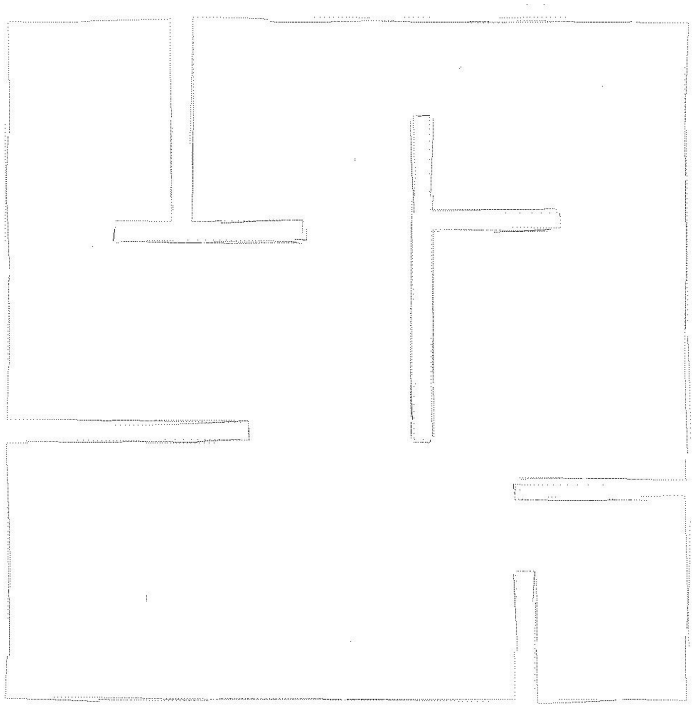

Figura 10.4 – Mapa gerado pelos robôs.

Os parâmetros $G_{robô}$ e G_{parede} foram escolhidos de para otimizar a dispersão e impedir que um robô colida com outro. Se fosse G_{parede} maior que $G_{robô}$, no início da dispersão os robôs iam acabar colidindo uns nos outros, já que eles estão em um ambiente cercados de paredes. Os parâmetros de movimentação também foram determinados para que os robôs não se chocassem nesse início, que é o momento mais critico porque os robôs estão mais próximos uns dos outros.

O robô tem uma percepção limitada do ambiente, por ter apenas quatro sensores, existem muitos pontos cegos em que o robô não detecta um obstáculo próximo dele. Como pode ser visto na figura 10.5, mesmo estando perto de uma parede, nenhum dos sensores detecta o obstáculo. Para evitar que o robô colida nas quinas das paredes foi determinado parâmetros para movimento em que o robô se desloque apenas distâncias pequenas mesmo com os sensores não obtendo nenhuma leitura de obstáculo. Assim, após andar uma distancia pequena novas forças agirão no robô e ele irar girar, possibilitando que e ele identifique obstáculos.

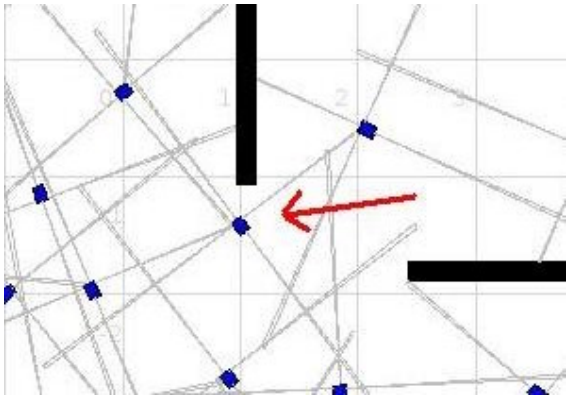

Figura 10.5 – Ponto cego dos sensores.

Logo no início da simulação, muitos robôs se chocavam, isso acontecia porque eles chegavam muito perto uns dos outros e quando giravam em direções opostas acabavam se chocando, pois não tinha espaço suficiente para eles girarem. Esse problema foi contornado aumentando a força de repulsão entre robôs e determinando parâmetros para movimentação, distâncias que eles deveriam andar e leituras de sensores que permitia a eles fazerem essa movimentação.

A simulação do algoritmo Hexágono no Ambiente 1 gravando imagens de 10 em 10 segundos demorou 2 horas e 29 minutos para a condição de parada ser atingida. Os parâmetros de paradas foram: menor raio 1,2435 m e distância entre iterações de 0,302443. Dessa simulação foi feito um vídeo para apresentação do algoritmo.

Podemos observar, na figura 10.6, que depois de passados 15 minutos de simulação, os robôs já conseguiram sair da sala inicial e já estão indo em direção às outras salas.

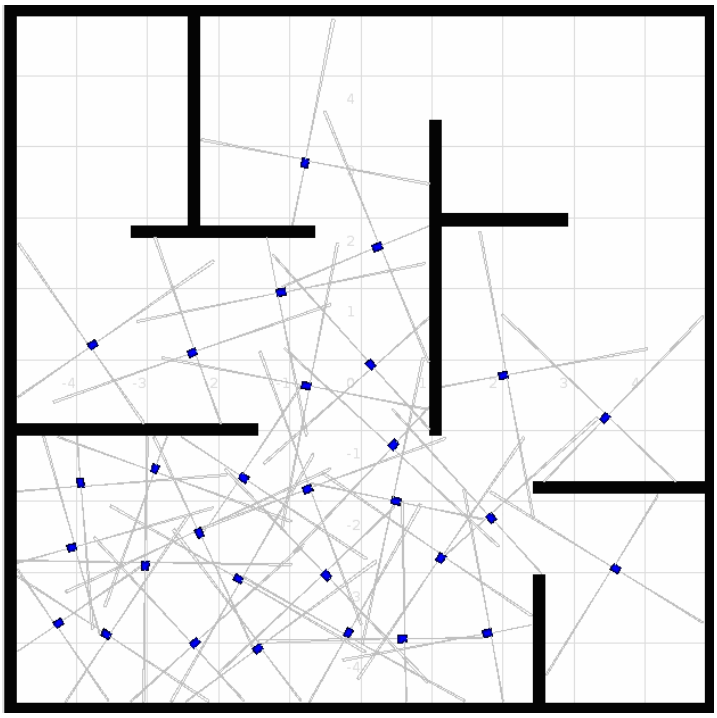

Figura 10.6 - Vídeo da simulação depois de 15 minutos de simulação.

Depois de uma hora de simulação, os robôs já conseguiram entrar em quase todas as salas, e só agora um robô está entrando na sala do canto superior direito. Podemos ver isso na figura 10.7.

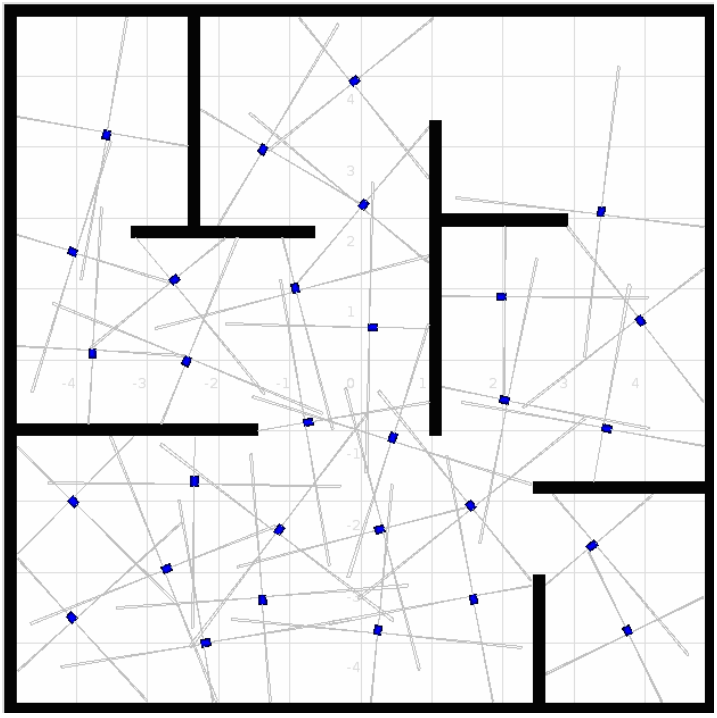

Figura 10.7 – Vídeo da simulação depois de 1 hora de simulação.

Depois observa-se que somente um robô conseguiu entrar na sala do canto superior direito, e ele sozinho não consegue fazer o mapeamento da sala toda onde ele se encontra. Observa-se, também, que os robôs estão mais espalhados nas outras salas, e pelas condições de parada eles já não estão mudando de posição.

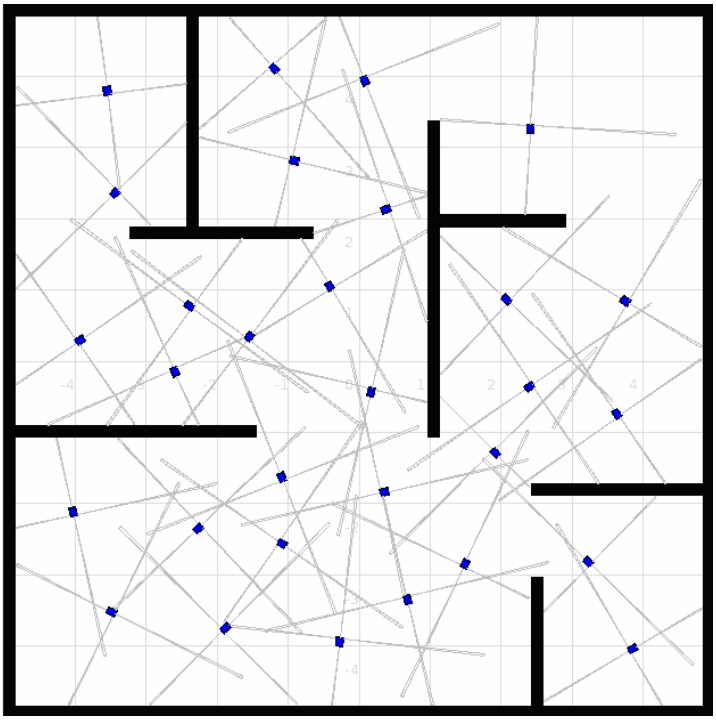

Figura 10.8 – Final da dispersão no vídeo da simulação.

O algoritmo apresenta resultados diferentes mesmo se simulado nas mesmas condições e ambientes, pois várias threads estão sendo executadas em paralelo. Erros de trucamento e a performance do processador do computador afetam também o resultado da simulação, por estar executando 30 threads paralelamente, o algoritmo fica sensível a alterações no processamento. Por exemplo, se executar a simulação sem salvar imagens será obtido um comportamento diferente se for salvo imagens de 10 em 10 segundos.

Foi feita uma simulação no Ambiente 2 para verificação do comportamento do algoritmo em um ambiente diferente. A condição de parada foi atingida depois de 2 horas e 38 minutos, com menor raio 1,2046 m e distância entre iterações de 0,317100.

O comportamento do enxame foi similar ao do ambiente anterior, em 1 hora de simulação somente o quarto do canto inferior direito não havia sido preenchido.

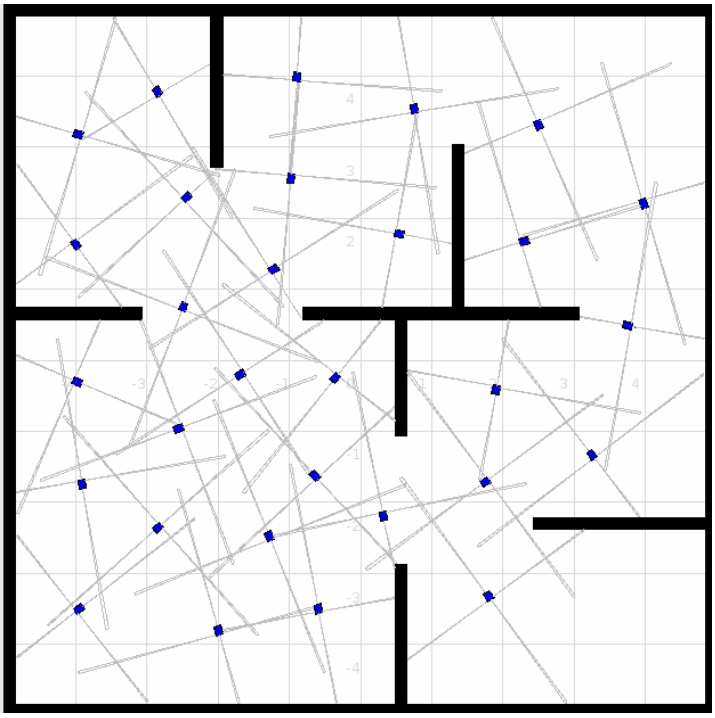

Figura 10.9 – Simulação no Ambiente 2 depois de 1 hora.

Ao final da dispersão, os robôs conseguiram se distanciar nos diversos quartos do ambiente e obter melhores leituras de sensores mas não chegaram a preencher o quarto que faltava. No mapa gerado da Figura 10.11, observa-se que ficou faltando mapear o canto inferior esquerdo do mapa.

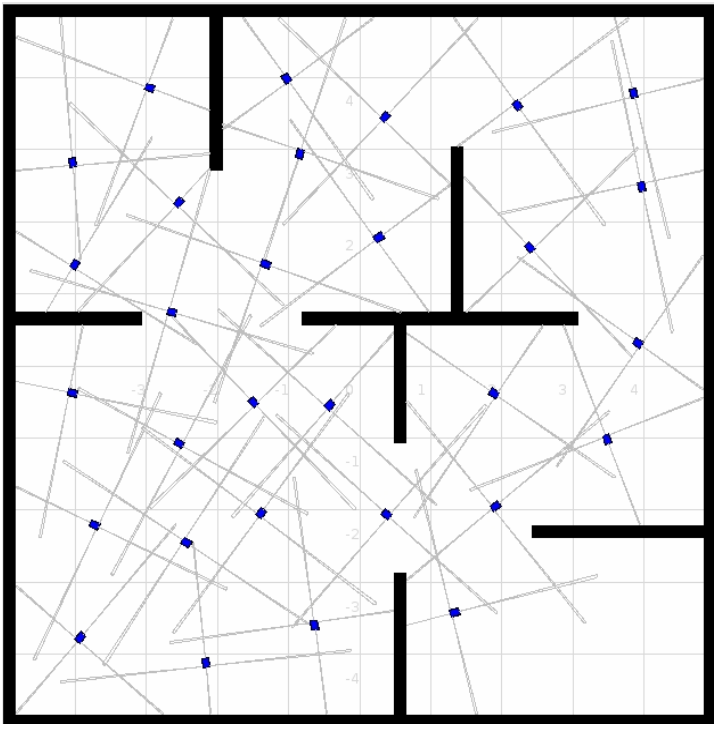

Figura 10.10 – Simulação no Ambiente 2 ao final da dispersão.

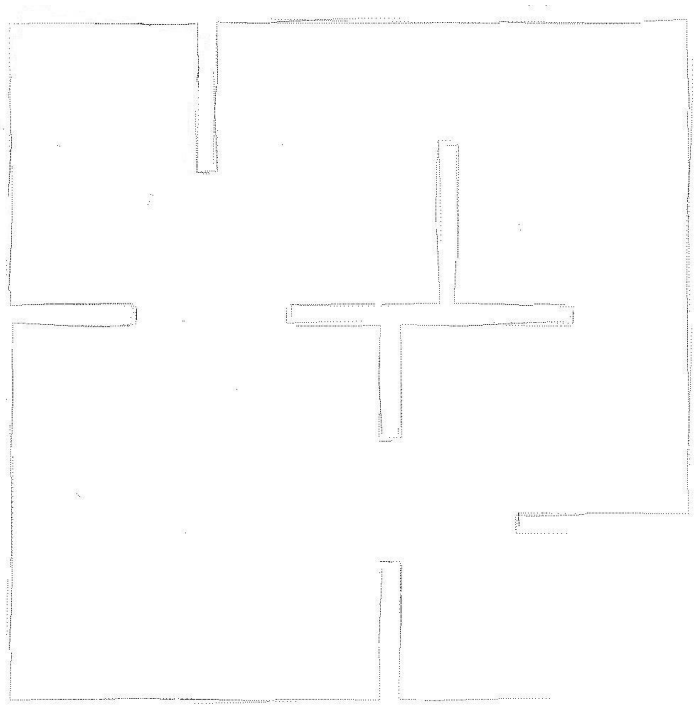

Figura 10.11 – Mapa do Ambiente 2 gerado pelos robôs.

Para fazer uma análise dos resultados obtidos, foi calculada a área de cobertura dos sensores do robô durante a simulação, assim, é possível saber com que velocidade os robôs conseguem mapear uma porcentagem da área do ambiente. Esses valores estão na Figura 10.12.

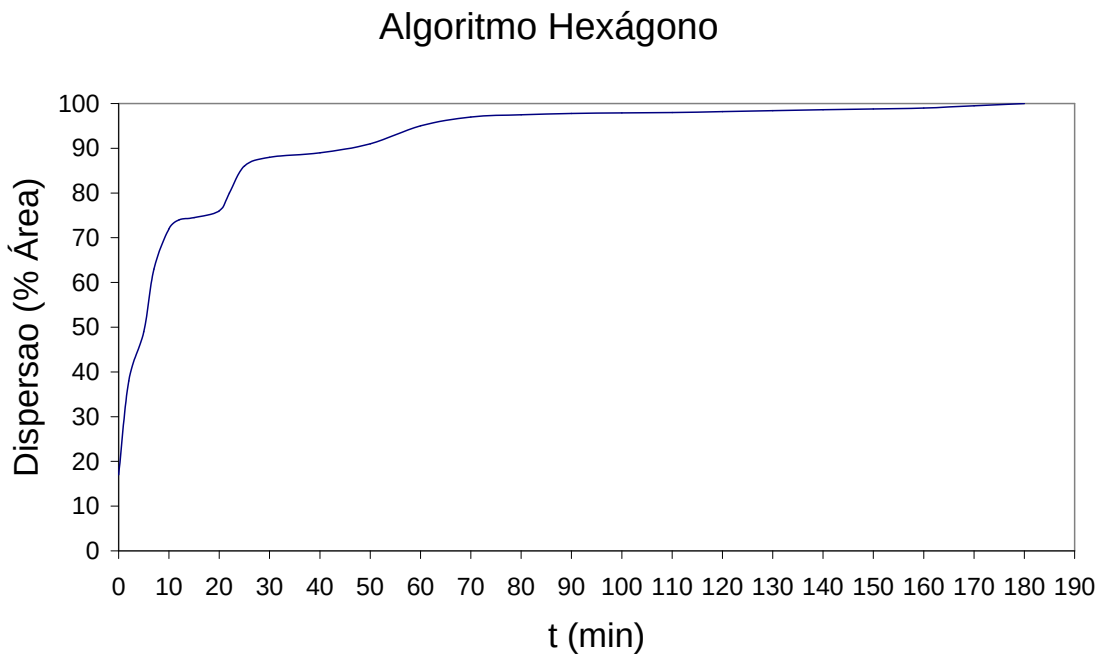

Figura 10.12 – Gráfico da área mapeada pelo tempo para o Algoritmo Hexágono.

Observa-se que em pouco tempo os robôs conseguem cobrir uma área grande do mapa, em 10 minutos é possível fazer o mapeamento de 72% do mapa. Como já foi dito, o enxame começa a se dispersa rapidamente e depois de atingir uma situação mais estável, ele passa a se dispersa mais lentamente.

Em 30 minutos de simulação, o enxame já consegue cobrir 88% do mapa, e a partir desse ponto que ele passa a se dispersar muito lentamente.

Pode-se concluir que, para ambientes menores, seria possível fazer o mapeamento de uma forma mais rápida, assim, seria possível fazer a dispersão e o mapeamento em até 30 minutos.

11 RESULTADOS DO ALGORITMO 2: DHM

Observa-se na Figura 11.1 a movimentação das concentrações de hormônio no ambiente após algumas iterações de movimentação. Estas imagens geradas a partir dos valores da própria simulação; através de uma escala de cinza onde mais escuro significa maior concentração e mais claro indica menor concentração.

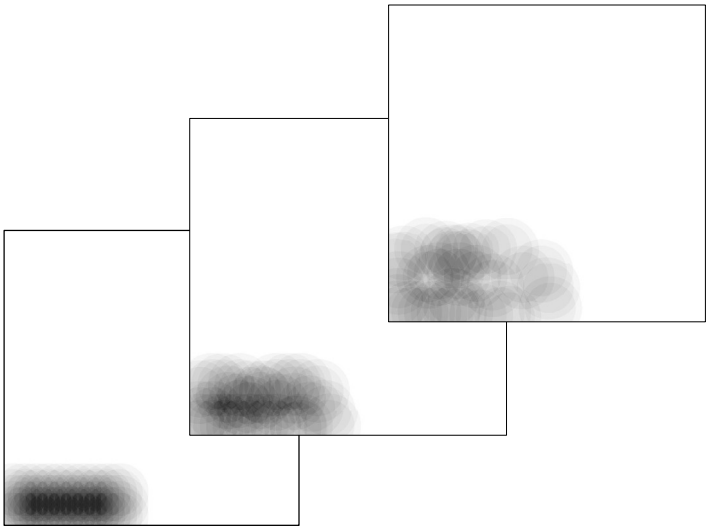

Figura 11.1 – Dispersão de Hormônio no ambiente.

Nota-se que na concentração inicial (a primeira figura, na esquerda) toda região dos robôs é quase preta, indicando altíssima concentração. Com exceção a periferia da região dos robôs, onde aparece uma região mais acinzentada. Essa região apresenta menor concentração e é a preferível para destino dos robôs.

Na segunda figura nota-se que já ocorreram algumas movimentações dos robôs e a distribuição de hormônio começou a se espalhar pelo ambiente. A região central se tornou um pouco mais clara.

Já na terceira figura, temos uma maior dispersão do hormônio. A figura apresenta uma distribuição cinza média. Notam-se alguns pontos internos mais claros com menor distribuição; esses pontos surgem devido a robôs que se afastam demais do grupo. Na próxima iteração ou esses robôs voltam preenchendo esses espaços ou os outros robôs são atraídos para estas posições.

O Algoritmo 2 não apresentou desempenho satisfatório devido a colisões que ocorrem durante a dispersão. Neste projeto tentou-se somente evitar colisões, não

se planejou um método para tratá-las depois de ocorridas. Talvez com um tratamento desse tipo o Algoritmo 2 poderia completar sua tarefa.

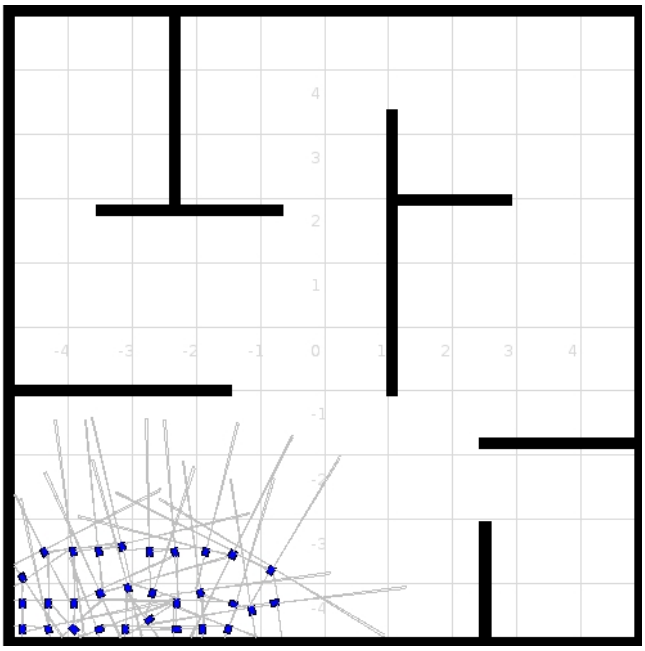

Figura 11.2 - Início da Dispersão.

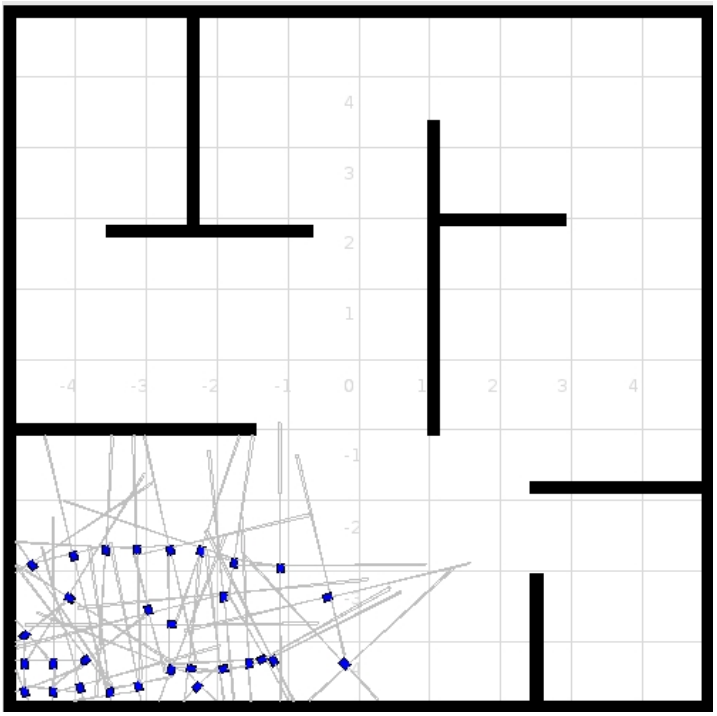

Figura 11.3 – Dispersão após algumas iterações.

Na Figura 11.3 nota-se que já ocorrem algumas colisões. Na ocorrência de uma colisão os robôs perdem a referência de sua posição atual, pois a medição é feita através de encoders. Como a roda do robô gira em falso os encoders detectam somente o movimento e fazem a leitura como se o robô tivesse se movimentado normalmente. Devido a essa perda de referência outros robôs começam a colidir em outros robôs que já colidiram; veja na Figura 11.3 esse caso onde três robôs estão agrupados.

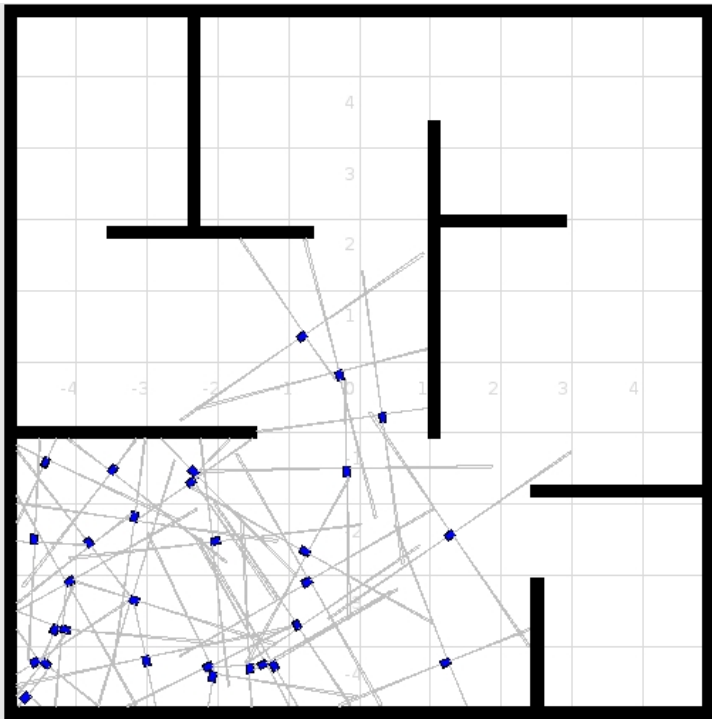

Figura 11.4 - Melhor resultado da dispersão.

Já na Figura 11.5 temos o melhor resultado da dispersão. Devido às colisões, muitos robôs são perdidos pelo movimento e acabam por retrair a dispersão do grupo. Pode se verificar que os robôs começaram a entrar nos outros quartos do ambiente, desviando das paredes.

Testes foram feitos usando uma menor quantidade de robôs de modo a evitar as colisões e a dispersão ocorreu com sucesso. Foram feitos testes dos critérios de parada do algoritmo e também foi obtido sucesso. Isto indicaria que tratando o problema das colisões, evitando-as ou corrigindo-as depois de ocorridas, o algoritmo deve apresentar um desempenho satisfatório.

Foi calculada a área de cobertura dos sensores do robô durante a simulação, assim, mesmo sem completar a dispersão, é possível saber com que velocidade os robôs conseguem mapear uma porcentagem da área do ambiente. Esses valores estão na Figura 11.5.

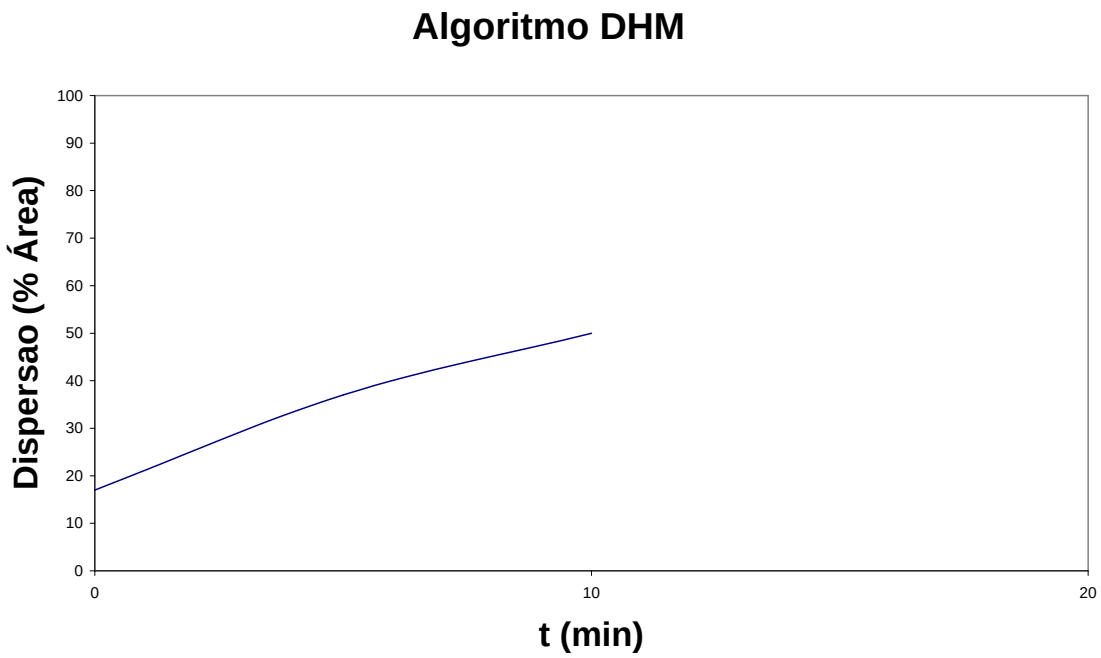

Figura 11.5 – Gráfico da área mapeada pelo tempo para o Algoritmo DHM.

Observa-se que, mesmo sem terminar a dispersão, o enxame conseguiu cobrir 50% da área do ambiente em pouco tempo.

12 CONCLUSÕES

Este projeto tinha como objetivo estudar o comportamento de um enxame de robôs, para isso, foram implementados dois algoritmos de dispersão. Para cada algoritmo, foram feitas diversas simulações para observar o funcionamento do algoritmo, estudar e refinar sua implementação, além de determinar os parâmetros que otimizariam seu funcionamento. Os algoritmos também foram simulados em outros dois ambientes diferentes para ser estudado o comportamento do mesmo em situações diferentes.

Como visto nos resultados das simulações, mesmo utilizando robôs com capacidade de percepção restrita, foi possível realizar uma tarefa complexa de mapear um ambiente.

O Algoritmo do Hexágono apresentou um resultado melhor que o DHM, pois pôde contornar o problema das colisões. Nele é possível ordenar diretamente que os robôs muito próximos se afastem; o que não ocorre no Algoritmo DHM. Através dos resultados obtidos, vemos que o algoritmo DHM começa a se dispersar rapidamente, mas é impedido de continuar a se dispersar devido às várias colisões.

Por isso, o Algoritmo do Hexágono foi o único a finalizar a tarefa de dispersão e mapeamento. No início da simulação, os robôs se dispersão muito rapidamente, chegando a cobrir a maior porcentagem do mapa em pouco tempo, a partir desse ponto a velocidade da dispersão é reduzida significativamente. Inicialmente, o projeto previa a utilização de um enxame de 100 robôs, mas devido às restrições de processamento e memória dos computadores utilizados na simulação foi possível utilizar somente 30 robôs. Os resultados obtidos mostram que, para um número maior de robôs executando esse algoritmo, o enxame teria maior facilidade em ocupar todo o ambiente e realizar a tarefa com maior eficiência.

Este projeto visava movimentar os robôs evitando colisões e não tratá-las depois de ocorridas. Se fosse implementado um modo de tratar essas colisões, seria possível obter resultados melhores do algoritmo DHM.

13 REFERÊNCIAS BIBLIOGRAFIA

[1] The Player Project, disponível em http://playerstage.sourceforge.net/ , acesso em 10 de fevereiro de 2007.

[2] SPEARS, W. M.; SPEARS D. F.; HAMANN, J.C; HEIL, R.. **Distributed, Physics-Based Control of Swarms of Vehicles***, Autonomous Robots 2004.*

[3] FREDSLUND, J.; MATARIÉ M. J. **A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication***, IEEE Transactions on Robotics and Automation 2001.*

[4] SHEN, W.; CHUONG, C. M.; WILL, P. **Digital Hormone Models for Self-Organization***, International Conference on Simulation and Synthesis of Living Systems. Sydney Australia 2004.*

[5] SHEN, W.; CHUONG, C. M.; WILL, P. **Simulating Self-Organization for Multi-Robot Systems***, International Conference on Intelligent and Robotic Systems. Switzerland, 2002.*

[6] SHEN, W.; CHUONG, C. M.; WILL, P.; GALSTVAN, A. **Hormone-Inspired Self-Organization and Distributed Control of Robotic Swarms,** *Autonomous Robots 2004.*

[7] THRUN, S.; BURGARD, W.; FOX, D. **A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping***, IEEE International Conference on Robotics and Automation, San Francisco, April 2000.*

[8] MUNIZ, A. L. N. **Resumo sobre a técnica de mapeamento 3D**

[9] VALDASTRIA, P.; CORRADIA, P.; MENCIASSIA, A.; SCHMICKLB, T.; CRAILSHEIMB, K.; SEYFRIEDC, J.; P. DARIOA P. **Micromanipulation, communication and swarm intelligence issues in a swarm microrobotic platform***, Robotics and Autonomous Systems, 2006.*