UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Navegação e Monitoramento de Ambientes Internos Utilizando Robôs Móveis

Heitor Luis Polidoro

São Carlos / SP
Novembro de 2007

Navegação e Monitoramento de Ambientes Internos Utilizando Robôs Móveis

Heitor Luis Polidoro

Orientador: Prof. Dr. Denis Fernando Wolf

Monografia preliminar de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação (ICMC / USP) para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Robótica Móvel, Monitoramento, Navegação.

São Carlos / SP
Novembro de 2007

Agradecimento

Agradeço ao professores Denis Fernando Wolf e Eduardo Marques que me orientaram e incentivaram a fazer esse projeto. Agradeço ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP - por prover os recursos necessários para minha formação e conclusão desse projeto.

Dedicatória

Dedico o resultado deste projeto aos meus pais, Heitor e Sonia, pois sem o esforço e dedicação de ambos não seria possível eu estudar em uma universidade de renome. Exalto os esforços de ambos para que eu tivesse uma educação de qualidade e pelos exemplos de profissionalismo e educação, fazendo com que desta forma, me tornasse um bom cidadão e profissional capacitado. Por toda a sua dedicação, carinho e compreensão, lhes sou muito grato.

Resumo

A robótica móvel é uma área de pesquisa que está obtendo grande atenção da comunidade científica. O desenvolvimento de robôs móveis autônomos, que sejam capazes de interagir com o ambiente, aprender e de tomar decisões corretas para que suas tarefas sejam executadas com êxito é o maior desafio em robótica móvel. O desenvolvimento destes sistemas inteligentes e autônomos consiste em uma área de pesquisa multidisciplinar, considerada recente e extremamente promissora que envolve: inteligência artificial, aprendizado de máquina, estimação estatística e sistemas embarcados, por exemplo. Um dos problemas fundamentais na área de robótica é a navegação, que consiste em determinar o local para onde o robô deve se mover, o melhor caminho até esse local e como chegar até esse local de forma segura, evitando obstáculos. Esse projeto de pesquisa propõe a comparação de algoritmos de monitoramento de ambientes internos, no qual o robô deve percorrer o ambiente de maneira eficiente, levando em consideração a prioridade de cada local a ser visitado e o tempo decorrido desde a última visita a cada região do ambiente.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo

1

Introdução

1.1 Contextualização e Motivação

A robótica consiste em uma área multidisciplinar de pesquisa, envolvendo elementos de engenharia mecânica, elétrica, computação e áreas de humanas como psicologia e estudos do comportamento de animais.

Inicialmente, os robôs foram utilizados para automação industrial, mas com a evolução tecnológica os robôs passaram a ser utilizados em outras áreas como: medicina de precisão, ambientes perigosos, entretenimento, serviços domésticos, etc. Nesse contexto surgiram as pesquisas para o desenvolvimento de robôs móveis autônomos, que sejam capazes de atuar em ambientes reais e reagir a situações desconhecidas de forma inteligente.

Entre todas essas diversas aplicações da robótica móvel, pode-se citar o robô Sojourner (Figura 1.1a) da NASA, que explora, envia fotos e outras informações do planeta Marte para a Terra [Nasa & JPL, 2007], e o robô desenvolvido pela universidade Camegie Mellon, chamado Groundhog, que explora minas abandonadas, que além do risco de desabamento, em muitos casos também contém gases tóxicos [Thrun et al., 2004].

A comunidade científica aposta que sistemas robóticos estejam cada vez mais presentes em nossa vida cotidiana, o que torna esta área de pesquisa extremamente promissora e desafiadora. Segundo Bill Gates, estamos entrando numa nova era da computação, comparando os robôs industriais com os mainframes de antigamente, e prevê que existirá um robô em cada casa no futuro [Gates, 2007].

A robótica móvel, além de ser uma área de grande potencial científico, empresas de tecnologia estão cada vez mais investindo em produtos. Como por exemplo o desenvolvimento de robôs que realizam trabalhos domésticos autonomamente, entre os quais o aspirador de pó Roomba da IRobot [IRobot, 2007] (Figura 1.1b) e o robô cortador de grama Robomow (Figura 1.1c) da Friendly Robotics [FriendlyRobotics, 2007]. Ambos apresentam sucesso comercial.

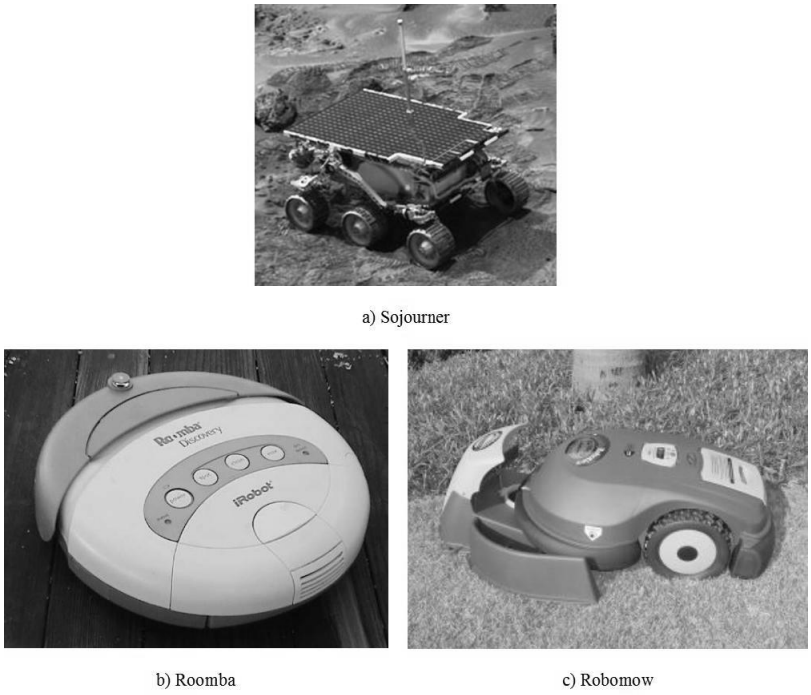

Figura 1.1: Exemplos de robôs.

Neste projeto de pesquisa, a robótica é vista de uma perspectiva da ciência da computação, mais especificamente das áreas de inteligência artificial e algoritmo e estrutura de dados. Nesse contexto, o maior desafio no desenvolvimento de robôs móveis é a capacidade de interagir com o ambiente e tomar decisões corretas para que suas tarefas sejam executadas com êxito. Robôs móveis autônomos devem ser capazes de atuar em ambientes desconhecidos e dinâmicos e de reagir diante de situações imprevistas. Devido à utilização de robôs móveis em ambientes dinâmicos, ou seja, ambientes em que o robô pode encontrar objetos não presentes na sua representação de mundo, ou não encontrar algum objeto que estava presente, os dados provenientes dos sensores, na grande maioria

dos casos, possuem ruídos e imprecisões, o que dificulta as tarefas de navegação e exploração [Bianchi, 2003]. A interação com o ambiente é realizada através de sensores (câmeras de vídeo, LASERs, sonares, bússolas e sensores de deslocamento) e atuadores (rodas e manipuladores).

O processamento das informações obtidas por sensores não é uma tarefa trivial. Por exemplo, apesar de ser fácil para seres humanos entenderem as imagens de uma câmera de vídeo, obter automaticamente informações úteis através do processamento dessas imagens por um computador é uma tarefa difícil que vem sendo estudada por pesquisadores da área de visão computacional. Mesmo depois de décadas de estudos, essa área ainda apresenta limitações e problemas a serem resolvidos.

A motivação deste projeto está em expandir os conhecimentos nas áreas de navegação e exploração de ambientes internos, com utilização de técnicas de inteligência artificial, o interesse particular por robótica, o contato anterior com o robô que será utilizado para os testes reais, a familiaridade com a ferramenta de desenvolvimento e com a ferramenta de simulação Player/Stage [Gerkey et al., 2001] e, também, a possibilidade de dar continuidade ao projeto iniciado na disciplina *Projeto Supervisionado ou de Graduação I*, colocando em prática conceitos aprendidos nas disciplinas cursadas na graduação. Para tanto, é proposto uma aplicação de robótica móvel para o monitoramento de ambientes internos, onde, normalmente, nesses ambientes, existem regiões criticas que possuem prioridade distintas para serem monitoradas, analisando e avaliando diferentes algoritmos de definição da seqüência de visita das áreas.

1.2 Objetivos do Trabalho

Esse projeto tem como objetivos ampliar os conhecimentos do aluno na área de robótica móvel, bem como possibilitar uma experiência prática na aplicação de conceitos aprendidos em disciplinas cursadas na graduação. Para isso, é proposto o desenvolvimento e avaliação de algoritmos de determinação da seqüência em que as áreas devem ser visitadas em um ambiente interno, utilizando algoritmos de navegação, incluindo o desvio de obstáculos, determinando qual a melhor trajetória entre as áreas visitadas, dando continuidade ao projeto de título *Desenvolvimento de técnicas de monitoramento de ambientes internos utilizando-se robôs móveis*, iniciado na disciplina *Projeto Supervisionado ou de Graduação I*.

Esta monografia apresenta o desenvolvimento e avaliação de algoritmos para de-

terminar a seqüência das áreas a serem visitadas em ambientes internos com o objetivo de monitoramento destes ambientes utilizando-se um robô móvel. O problema a ser resolvido consiste na divisão de um ambiente previamente conhecido em áreas de interesse. A cada uma dessas áreas é atribuído um valor (peso) referente a sua importância de monitoramento. A prioridade com que o robô deve visitar determinadas áreas é calculada com base na importância dessas áreas e no tempo decorrido desde a sua última visita. Áreas de maior importância devem ser visitadas mais freqüentemente.

As avaliações consistem na comparação dos algoritmos. Serão utilizados dois critérios de comparação, um critério de pontuação final, onde a pontuação é calculada pela somatória dos pontos de cada sala, e os pontos de cada sala são calculados multiplicando a freqüência absoluta de visitas pela prioridade. O segundo critério é um gráfico mostrando a progressão da somatória dos graus de urgência de todas as salas.

Capítulo

2

Revisão Bibliográfica

Esta seção descreve as seguintes técnicas utilizadas no desenvolvimento das atividades do projeto: o paradigma da robótica (Seção 2.1), o algoritmo de desvio de obstáculos (Seção 2.2), a busca de menor caminho (Seção 2.3) e o controle do robô (Seção 2.4).

2.1 Paradigmas da Robótica

Um paradigma é uma filosofia, um conjunto de regras, técnicas que definem uma abordagem de um tipo de problema. Os paradigmas robóticos são as definições de como a "inteligência" dos robôs pode ser estruturada. Existem três primitivas básicas: SENTIR, PLANEJAR e AGIR; e a forma como são organizadas é que diferenciam os paradigmas. A primitiva SENTIR é responsável por adquirir informações sobre o mundo. A primitiva PLANEJAR recebe as informações adquiridas do mundo e gera as ações que o robô deve seguir. E a primitiva AGIR comanda diretamente os atuadores do robô [Bianchi, 2003]. Utilizando-se dessas primitivas, é possível caracterizar até o momento três paradigmas robóticos: o Deliberativo (ou Hierárquico), o Reativo e o Híbrido Reativo / Deliberativo.

2.1.1 Paradigma Deliberativo

O paradigma deliberativo foi o primeiro paradigma robótico, e as primitivas estão organizadas como mostra a Figura 2.1.

O problema principal desse paradigma é a velocidade de resposta, pois os processos de criação dos modelos de mundo e de planejamentos exigem muito processamento, fazendo com que demore para obter uma resposta, e como as primitivas estão organizadas em seqüência, o robô precisa esperar que uma primitiva seja executada por completo para seguir para a outra [Bianchi, 2003]. Porém tem-se preferência por utilizar o paradigma deliberativo quando o mundo puder ser modelado com precisão, as incertezas são limitadas e o ambiente não muda durante a execução [Faria, 2006]. As arquiteturas mais conhecidas que utilizam o paradigma deliberativo são: SOAR (do inglês, *State, Operator and Result*), *Blackboard*, NCH (do inglês, *Nested Hierarchical Controller*) PRODIGY e NASREM (do inglês, *NASA/NBS Standard Reference Model*).

Figura 2.1: Paradigma Deliberativo.

2.1.2 Paradigma Reativo

O paradigma reativo foi inspirado no comportamento de animais. Não existe modelo de mundo, portanto não há a necessidade da primitiva PLANEJAR (Figura 2.2), fazendo com que a velocidade de resposta se torne maior que no paradigma deliberativo [Bianchi, 2003].

Esse paradigma é orientado a comportamentos, ou seja, o comportamento final do robô é uma combinação de vários comportamentos independentes. Isso implica que as primitivas SENTIR e AGIR se tornem locais para cada comportamento. Portanto, a ação final do robô deve ser uma integração de todas as ações geradas por cada comportamento [Bianchi, 2003]. *Subsumption* e Esquema Motor são exemplos de arquiteturas que utilizam o paradigma reativo.

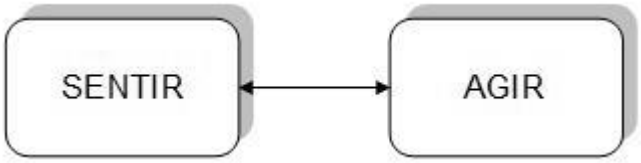

Figura 2.2: Paradigma Reativo.

2.1.3 Paradigma Híbrido Reativo / Deliberativo

Embora o paradigma reativo apresente um desempenho robusto e consiga lidar com facilidade em ambientes dinâmicos e complexos, tarefas como localização, ou navegação de forma autônoma por longo período de tempo ou por grandes distâncias, requerem que o robô tenha uma certa memória do mundo [Bianchi, 2003]. Por outro lado o paradigma deliberativo pode não conseguir responder com rapidez e eficiência a uma mudança no mundo 2.3, restringindo assim, o paradigma deliberativo a nichos específicos [Faria, 2006].

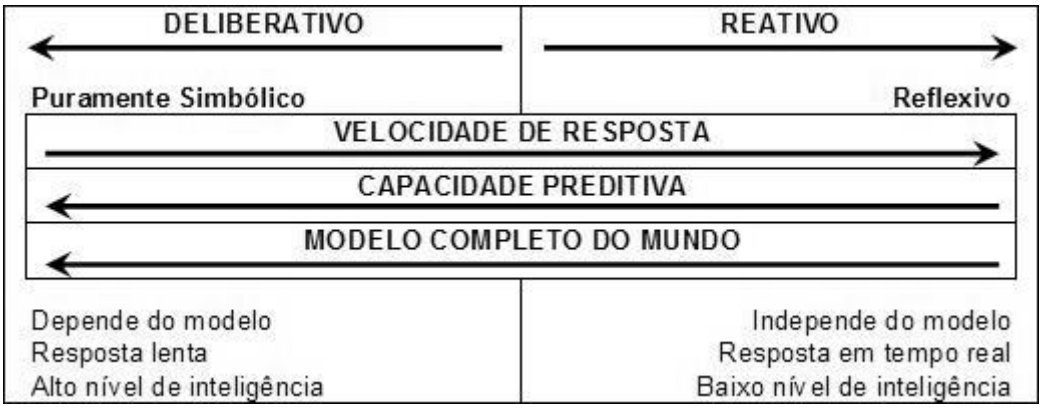

Figura 2.3: Comparação entre paradigmas: Deliberativo e Reativo.

Surgiu então, a necessidade de um paradigma que unisse o melhor dos paradigmas Deliberativo e Reativo. Assim foi criado o paradigma híbrido, unindo os paradigmas anteriores de forma que os robôs não são tão lentos como no paradigma deliberativo e podem planejar ações utilizando modelos do mundo. Nesse paradigma, a primitiva PLANEJAR não esta ligada em seqüência com as outras (Figura 2.4). Isso quer dizer que o planejamento é feito em um passo e os atos de SENTIR e AGIR são realizados juntos em outro passo, fazendo com que o planejamento não faça parte do ciclo principal (percepção-ação), interferindo somente quando há algo relevante [Bianchi, 2003]. São exemplos de arquiteturas híbridas: AuRA (do inglês, *Autonomous Robot Architecture*), *Atlantis*, DAMN (do inglês, *Distributed Architecture for Mobile Navigation*), TCA (do inglês, *Task Control Architecture*), *Saphira* e NOS.

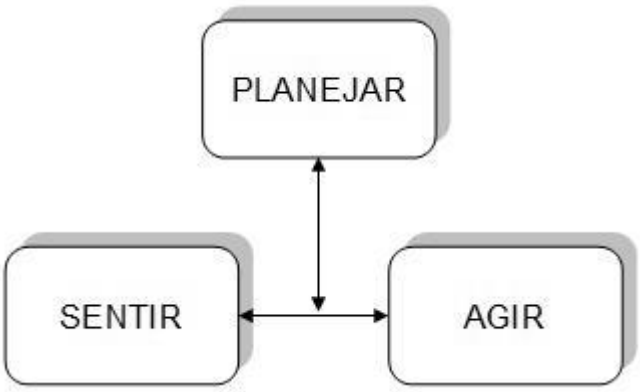

Figura 2.4: Paradigma Híbrido Reativo / Deliberativo.

2.2 Desvio de Obstáculos

Para a tarefa de desvio de obstáculos utilizou-se o algoritmo de campos potencias. Esse algoritmo consiste em calcular e representar a meta e os obstáculos do ambiente como forças que atraem e repelem o robô [Koren & Borenstein, 1991] [Goodrich, 2007]. Normalmente esse algoritmo pode ser implementado de maneira simples com sensores que provêem informações sobre a distância entre o robô e os obstáculos, como sonar ou LASER, por exemplo.

No algoritmo de campos potenciais a direção e a velocidade que o robô deve seguir são dadas pelo vetor resultante da combinação das forças de atração da meta, e repulsão dos obstáculos. Por exemplo, a Figura 2.5 mostra o vetor resultante em cada ponto, num plano contendo somente a meta.

Para gerar esse campo, basta considerar o vetor $v = [x, y]$ como vetor gradiente $\Delta = [\Delta x, \Delta y]$ e calcular e segundo os seguintes termos:

- Seja (x_m, y_m) a posição da meta onde o robô deve chegar;
- Seja (x, y) a posição do robô;
- Considerar a meta como um circunferência de raio r;
- Considerar um anel de extensão de raios interno r e externo $s + r$;
- Seja d a distância entre a meta e o robô: $d = \sqrt{(x_m - x)^2 + (y_m - y)^2}$;
- Seja θ o ângulo entre a meta e o robô: $\theta = tan^{-1}\left(\frac{y_m - y}{x_m - x}\right)$;

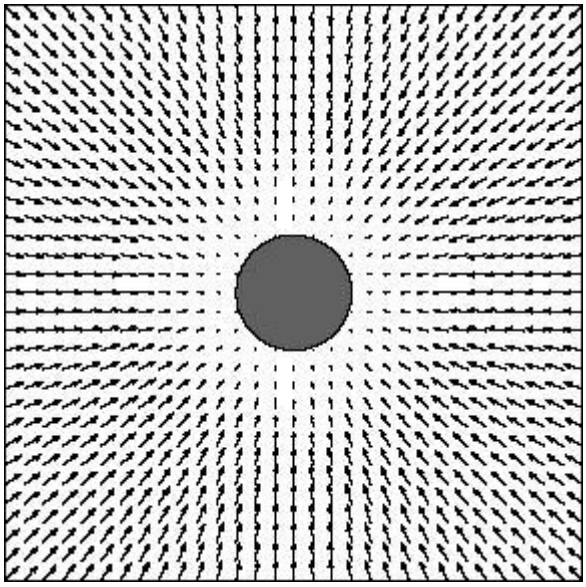

Figura 2.5: Campo Potencial de Atração.

- Definir Δx e Δy de acordo com o seguinte:

$$\begin{array}{cc} \text{Se } d < r & \Delta x = \Delta y = 0 \\ \text{Se } r \leq d \leq s + r & \Delta x = \alpha(d - r)cos(\theta) \text{ e } \Delta y = \alpha(d - r)sin(\theta) \\ \text{Se } d > s + r & \Delta x = \alpha cos(\theta) \text{ e } \Delta y = \alpha sin(\theta) \end{array}$$

Quando $d < r$ significa que o robô está dentro da circunferência que define a meta, portanto ele deve parar. Quando $r \leq d \leq s + r$ significa que o robô está no anel de extensão (Figura 2.6), então a velocidade deverá ser proporcional à distância do robô para a meta. E, finalmente, quando $d > s + r$ significa que o robô está muito longe da meta, portanto sua velocidade é fixada em α , sendo $\alpha > 0$

No caso de um campo potencial de repulsão, o cálculo é similar ao cálculo de um campo potencia de atração.

- Seja (x_o, y_o) a posição do obstáculo onde o robô deve chegar;
- Seja (x, y) a posição do robô;
- Considerar o obstáculo como um circunferência de raio r;
- Considerar um anel de extensão de raios interno r e externo $s + r$;
- Seja d a distância entre o obstáculo e o robô: $d = \sqrt{(x_o - x)^2 + (y_o - y)^2}$;
- Seja θ o ângulo entre o obstáculo e o robô: $\theta = tan^{-1}\left(\frac{y_o - y}{x_o - x}\right)$;

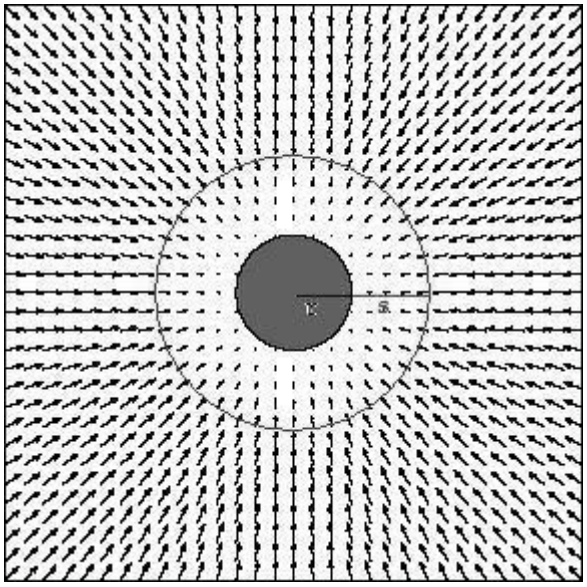

Figura 2.6: Anel de Extensão no Campo Potencial de Atração.

- Definir Δx e Δy de acordo com o seguinte:

$$\begin{array}{cc} \text{Se } d < r & \Delta x = -sign(cos(\theta)) \times \infty \text{ e } \Delta y = -sign(sin(\theta)) \times \infty \\ \text{Se } r \leq d \leq s + r & \Delta x = \beta(s + r - d)cos(\theta) \text{ e } \Delta y = \beta(s + r - d)sin(\theta) \\ \text{Se } d > s + r & \Delta x = \Delta y = 0 \end{array}$$

Dentro do raio do obstáculo ($d < r$) o campo potencial de repulsão é infinito e aponta para fora do centro do obstáculo. No anel de extensão ($r \leq d \leq s + r$) a velocidade é inversamente proporcional à distância do robô para o obstáculo. E fora do anel de extensão ($d > s + r$) o obstáculo não oferece perigo, portanto não interfere na velocidade nem na direção do robô. A Figura 2.7 ilustra os vetores resultantes.

A direção e a velocidade finais do robô são correspondentes à direção e ao módulo do vetor resultante da combinação do campo potencial da meta e de todos os campos potenciais dos possíveis obstáculos, ou seja, $v = v_a + \sum v_r i$ onde v_a é o vetor do campo de atração e $v_r i$ é o i-ésimo vetor do campo de repulsão (Figura 2.8).

Uma desvantagem é que as forças de atração e repulsão podem se anular em determinados pontos, gerando mínimos locais [Faria et al., 2006]. Na Figura 2.9 os mínimos locais estão representados por células em branco, onde o vetor resultante tem módulo zero, impossibilitando o algoritmo de decidir para onde o robô deve seguir.

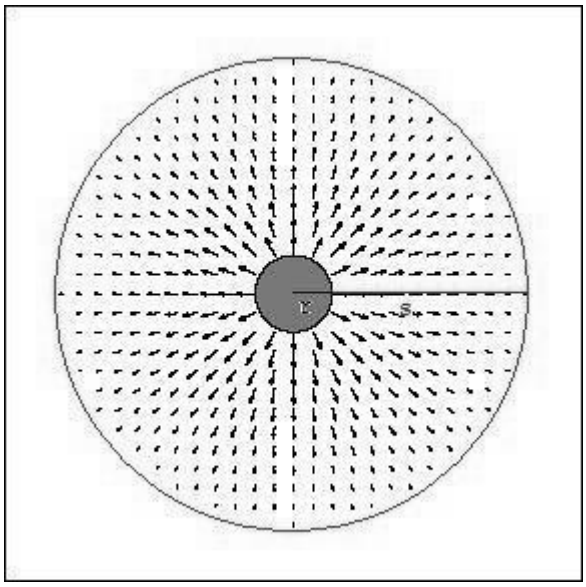

Figura 2.7: Campo Potencial de Repulsão com Anel de Extensão.

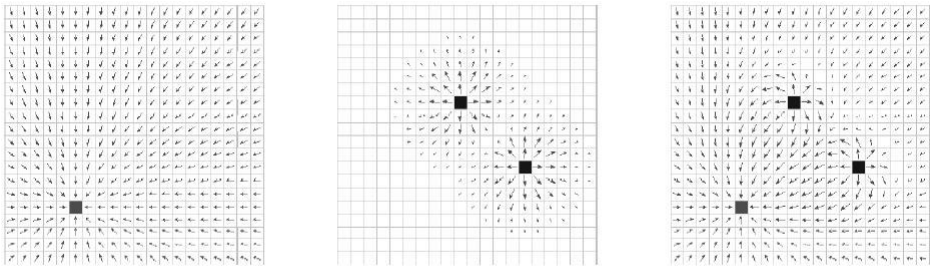

Figura 2.8: Campo de Atração (a), Campo de Repulsão (b), Combinação dos Campos (c).

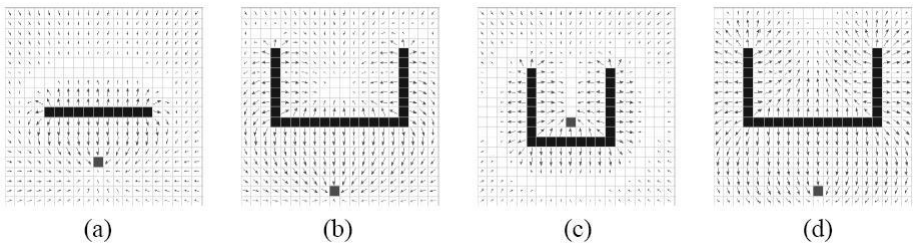

Figura 2.9: Mínimos Locais.

2.3 Busca do Menor Caminho

Para determinar a melhor trajetória entre as áreas do ambiente, pretende-se utilizar um algoritmo chamado *Ants*, desenvolvido pelo autor. Nesse algoritmo, o ambiente (mapa topológico, já conhecido) é representado como um grafo, onde os vértices (m) representam as áreas de interesse e as arestas(n) representam a distância, ou o custo, para o robô se deslocar de uma área para a outra (Figura 2.10).

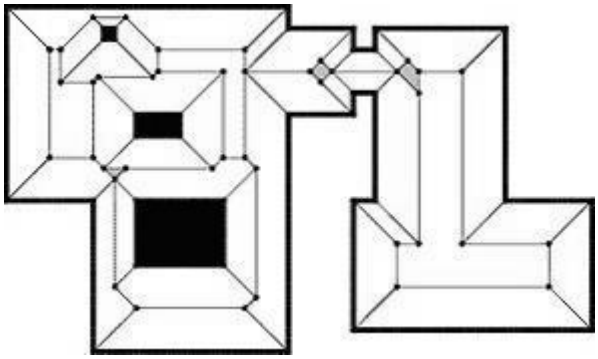

Figura 2.10: Mapa Topológico.

2.3.1 Ants

A idéia desse algoritmo é uma busca em profundidade, porém sem retroceder, visando paralelismo. Inicialmente é criado um agente no vértice inicial, e esse agente é inserido numa fila. A rotina do algoritmo consiste em:

- Se a fila de agentes não estiver vazia, retira um agente da fila, se não finaliza a rotina;
- Para cada vértice adjacente ao vértice onde se encontra o agente é criada uma cópia do agente que é colocada nesse vértice adjacente somando o peso da aresta e armazenando o caminho percorrido.
- Compara-se a soma do caminho desse novo agente com a variável `minimo_encontrado` (iniciada com ∞) se for menor, o novo agente é inserido no final da fila, se não ele é simplesmente descartado.
- Quando um agente chega ao vértice destino, o agente é guardado, e a variável `minimo_encontrado` é atualizada.

Essa rotina se repete até que não exista mais agentes na fila. Se o vértice destino for alcançável a partir do vértice inicial, o algoritmo *Ants* retornará o menor caminho do vértice inicial até o vértice destino.

2.4 Controle do Robô

O controle do robô será desenvolvido utilizando-se a biblioteca Player/Stage [Gerkey et al., 2001], a qual permite que sejam realizadas simulações antes que os algoritmos sejam testados com o robô real (Figura 2.11). Essas simulações são importantes para o aperfeiçoamento dos parâmetros, como distâncias, velocidades, etc, antes do teste no Pioneer.

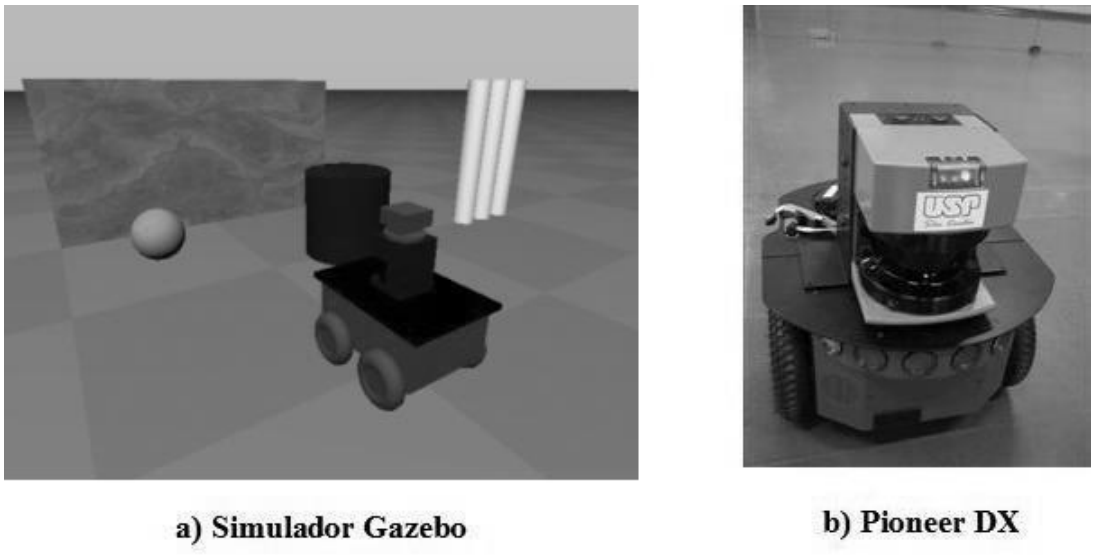

Figura 2.11: Robô Pioneer, Simulado e Real.

2.4.1 Player

O Player é um servidor de rede para controlar robôs. Executando embarcado no robô, o Player provê uma interface simples e clara dos sensores e atuadores do robô sobre uma rede IP. O programa cliente "conversa" com o Player utilizando sockets TPC, lendo dados dos sensores, escrevendo comandos nos atuadores e configurando dispositivos em tempo de execução. O servidor Player foi desenvolvido para ser independente de linguagem e de plataforma. O programa cliente pode executar em qualquer máquina que tenha conexão de rede com o robô, e pode ser escrito em qualquer linguagem que suporte sockets TCP. Atualmente existem clientes disponíveis em C++, Tcl, Java em

Phyton [Player, 2007]. Futuramente, o Player não fará suposições sobre como o programa de controle do robô é estruturado, ou seja, poderá-se escrever desde programas multi-threads altamente concorrentes até programas seqüências simples.

2.4.2 Stage

O Stage é usado normalmente como um plugin para o Player, provendo uma série de dispositivos virtuais para os clientes Player. Os usuários escrevem as rotinas e algoritmos normalmente, como clientes para um servidor Player. Não é possível para clientes distinguir a diferença entre os dispositivos reais do robô e os equivalentes simulados pelo Player/Stage. Com isso clientes Player desenvolvidos usando o Stage precisarão de pouca ou nenhuma modificação para trabalhar com o robô real, e vice-versa. Em muitos casos basta somente mudar no cliente o endereço IP de onde está o servidor. O Stage também pode simular uma população de robôs móveis, sensores e objetos num ambiente bi-dimensional (Figura 2.12) [Player, 2007]. Neste projeto será utilizado somente um robô e o sensor LASER.

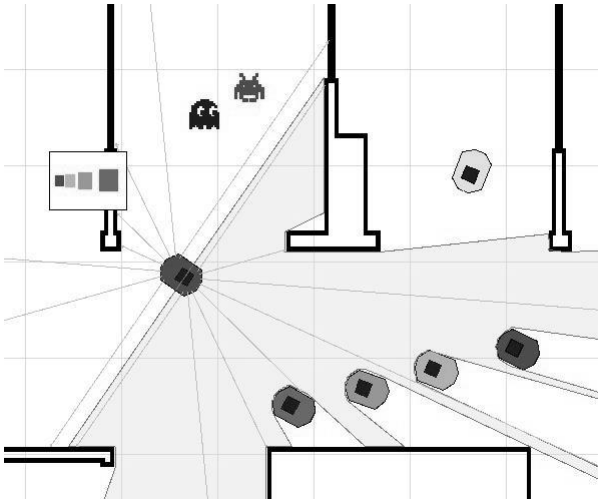

Figura 2.12: Simulação com 5 robôs, 2 objetos, LASER, sonar e blobfinder.

Capítulo

3

Estado Atual do Trabalho

3.1 Projeto

A proposta desse projeto é a elaboração, implementação, validação e comparação de algoritmos de definição de áreas a serem visitadas em ambientes internos utilizando algoritmos de busca de trajetórias e desvio de obstáculos, objetivando o monitoramento desse ambiente.

3.2 Descrição das Atividades Realizadas

Nesta seção está detalhado todas as atividades realizadas para a conclusão do projeto: Controle do Robô (3.2.1), Desvio de Obstáculos (3.2.2), Planejamento de trajetória (3.2.3), Determinação da seqüência de áreas (3.2.4) e Integração e validação dos algoritmos (3.2.5)

3.2.1 Estudo do Ambiente de Controle de Robôs Móveis Player/Stage

Apesar do contato anterior com o Player/Stage nas disciplinas *"Projeto e Implementação de Sistemas Embarcados I"* e *"II"*, foi necessário um novo estudo, pois o Player/Stage foi adaptado pelo Prof. Dr. Denis Fernando Wolf para facilitar a adaptação dos alunos com o Player/Stage nas disciplinas.

3.2.2 Estudo e Implementação do Algoritmo de Desvio de Obstáculos

Existem vários algoritmos para desvio de obstáculos: Desvio Tangencial, desvio utilizando-se Redes Neurais Artificiais, Campos Potenciais, etc. O algoritmo escolhido para o projeto foi o de Campos Potenciais, por ser de fácil implementação e ter um resultado satisfatório. O algoritmo de campos potenciais foi implementado da seguinte forma:

- É feita a leitura de 9 LASERs do robô: 0º, 30º, 60º, 75º, 90º, 105º, 120º, 150º e 180º (Figura 3.1);
- Foi estabelecido um valor (`MAX_DIST`) que se a distância do robô ao obstáculo foi superior que o mesmo, não há necessidade de desvio;
- Calcula-se a diferença entre a leitura do LASER (distância para o obstáculo) e `MAX_DIST` (somente se a leitura do LASER for menor que `MAX_DIST`);

$$dif_i = \texttt{MAX\_DIST} - [leitura\_do\_laser\_i]$$

- Para os ângulos maiores que 180º a diferença é multiplicada por -1;
- Somam-se todas as diferenças;

$$CP = \sum dif_i$$

Com isso, se obtém um valor que é proporcional à velocidade com que o robô deve virar ($|CP|$) e para qual lado ($sign(CP)$ positivo vira para a direita, negativo vira para a esquerda).

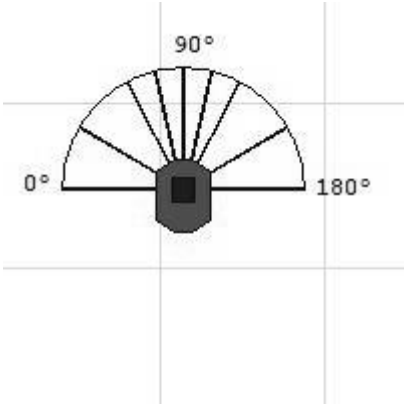

Figura 3.1: LASERs Utilizados para Desvio de Obstáculos

3.2.3 Estudo e Implementação dos Algoritmos de Planejamento de Trajetória

Foi utilizado um algoritmo de planejamento de trajetória chamado *Ants*. Veja a seguir o algoritmo 3.2.1 e o fluxograma (Figura 3.2) desse algoritmo.

Algoritmo 3.2.1 Algoritmo Ants

```
1: Cria um agente e coloca na fila
2: minimo_encontrado ⇐ ∞
3: enquanto Existir Agente na Fila faça
4:    Retira o Agente da Fila
5:    para Cada nó adjacente não visitado ao nó do Agente faça
6:       Cria uma cópia do Agente
7:       "Anda" com o Agente para o nó
8:       Armazena o caminho
9:       Soma o peso
10:      se soma < minimo_encontrado então
11:         se Novo nó = Destino então
12:            minimo_encontrado ⇐ soma
13:            Guarda/Substitui o Agente que encontrou o menor caminho
14:         senão
15:            Insere o Agente na fila
16:         fim-se
17:      fim-se
18:   fim-para
19: fim-enquanto
```

A intensão de usar esse algoritmo é que ele é facilmente paralelizável, basta cada processo retirar e analisar um agente da fila, aumentando a velocidade de resposta do algoritmo.

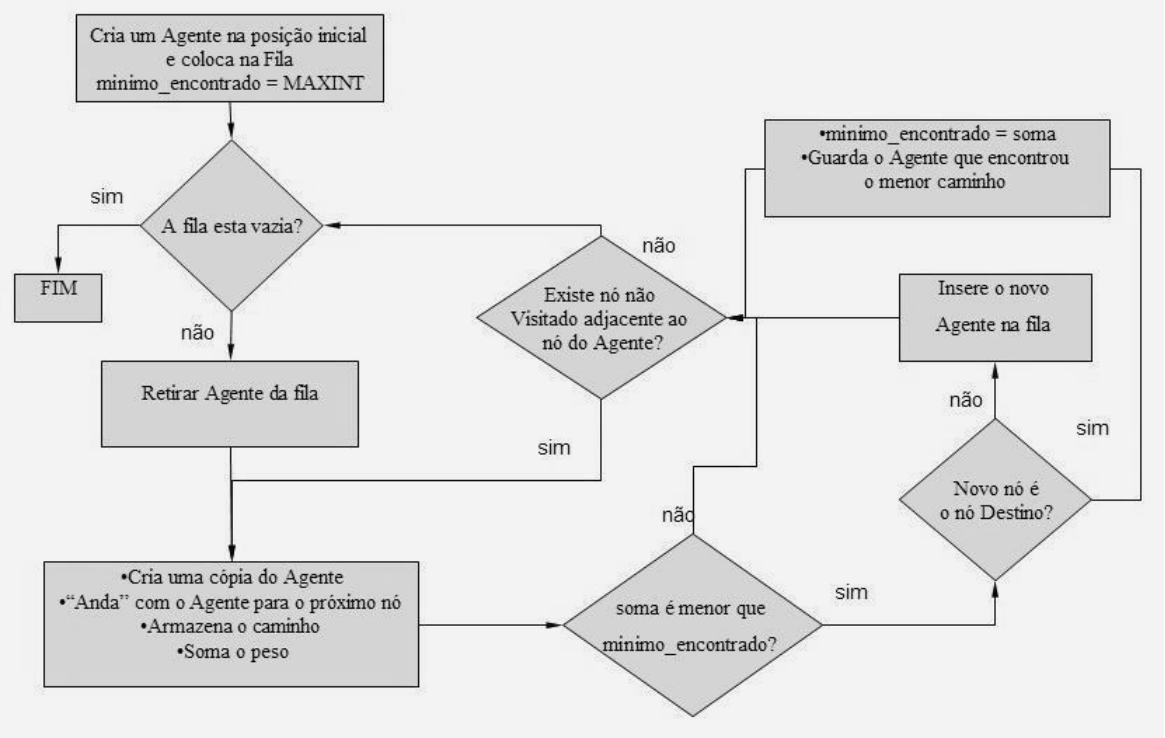

Figura 3.2: Fluxograma do Algoritmo Ants

3.2.4 Desenvolvimento e Implementação dos Algoritmos de Determinação das Áreas que Devem Ser Visitadas

Foram criados dois ambientes virtuais, um com baixa conectividade, mais perto da realidade (Mapa 1 - Figura 3.3) e outro com alta conectividade (Mapa 2 - Figura 3.4) para testar os algoritmos de planejamento de trajetória. Como o algoritmo de campos potencias fatalmente cairá no problema de mínimos locais, foram criados pontos de interseção para evitar esse problema, e os círculos numerados são as salas. A posição dos pontos e das salas foram criados pelo autor, da forma que achou conveniente, sem seguir um padrão ou regra específica. As prioridades das áreas foram escolhidas de forma arbitrária e estão descritas nas tabelas, Tabela 3.1, Tabela 3.2. As salas inicias escolhidas foram, Sala 5 e Sala 10, para os mapas 1 e 2, respectivamente.

Para auxiliar as decisões, foi criado um grau de urgência de visita U para cada sala, esse grau de urgência é calculado multiplicando-se a prioridade da sala P pelo tempo decorrido desde a última visita t. A casa visita o t é zerado.

$$U = P \times t$$

Foram desenvolvidos três algoritmos. O algoritmo 1 (3.2.2) verifica qual sala que possui

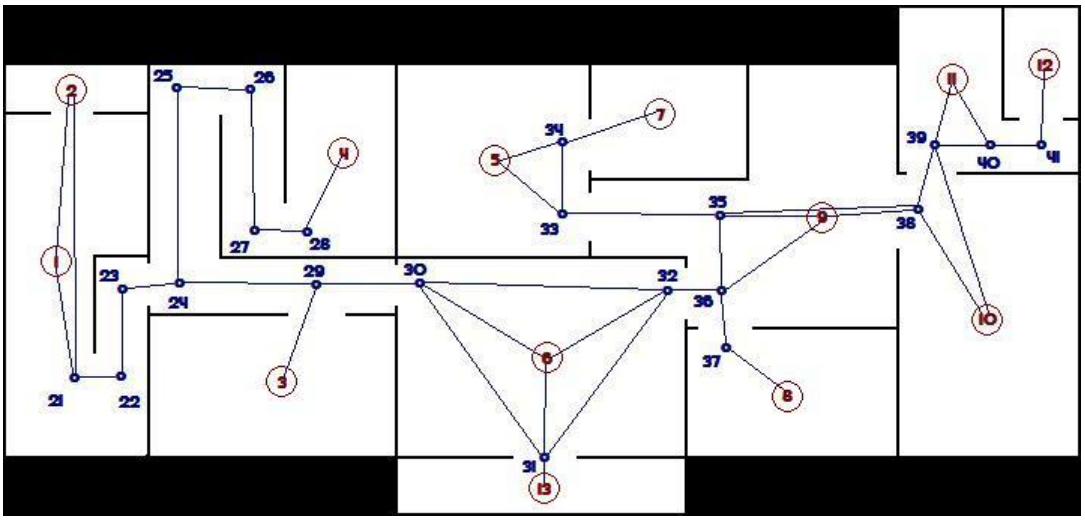

Figura 3.3: Mapa 1.

Sala	Prioridade
1	9
2	2
3	9
4	4
5	9
6	6
7	2
8	9
9	6
10	7
11	1
12	2
13	4

Tabela 3.1: Prioridades do Mapa 1

Sala	Prioridade
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	1
9	2
10	3

Tabela 3.2: Prioridades do Mapa 2

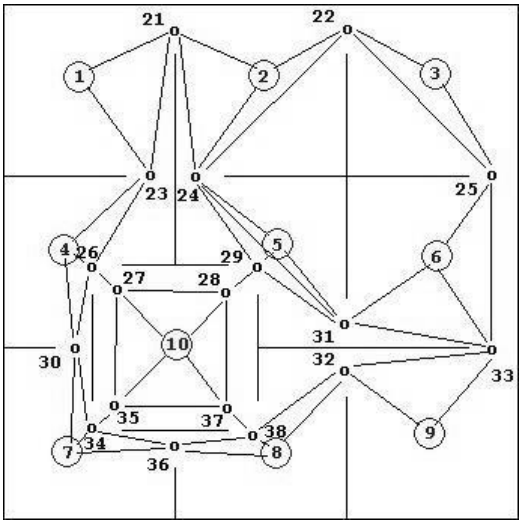

Figura 3.4: Mapa 2

o maior grau de urgência e manda o robô visitar aquela sala. O algoritmo 2 (3.2.3) manda o robô visitar a sala que possui o maior grau de urgência visitando as salas que são adjacentes à algum vértice da trajetória até a sala destino. Finalmente, o algoritmo 3 (3.2.4) calcula um fator para cada sala, e manda o robô visitar a sala com maior fator. Esse fator é calculado dividindo o grau de urgência da sala pelo custo para chegar até a sala.

$$Fator = [Grau\_de\_Urgencia]/[custo]$$

Algoritmo 3.2.2 Algoritmo 1

1: $sala \Leftarrow sala\_inicial$
2: **loop**
3: Visita sala atual //Zera o t da sala atual na tabela
4: $sala \Leftarrow$ sala_de_maior_grau_de_urgência()
5: Vai para *sala*
6: **fim-loop**

3.2.5 Integração e Validação dos Algoritmos Implementados no Simulador Stage

Todos os três algoritmos (Algoritmo 3.2.2, Algoritmo 3.2.3 e Algoritmo 3.2.4) foram testados nos dois mapas (Figura 3.3 e Figura 3.4). A freqüência de visitas e a trajetória dos algoritmos nos mapas são descritos e analisados na seção a seguir.

Algoritmo 3.2.3 Algoritmo 2

```
1: sala ⇐ SALA_INICIAL
2: loop
3:    Visita sala atual //Zera o t da sala atual na tabela
4:    sala ⇐ sala_de_maior_grau_de_urgência()
5:    Vai para sala
6:    para todo vértice no caminho faça
7:       para todo Sala adjacente ao vértice faça
8:          Vai para sala adjacente
9:          Visita a sala //Zera o t da sala adjacente na tabela
10:         Volta para o vértice
11:      fim-para
12:   fim-para
13: fim-loop
```

Algoritmo 3.2.4 Algoritmo 3

```
1: sala ⇐ SALA_INICIAL
2: melhor_fator ⇐ 0
3: loop
4:    Visita a sala atual //Zera o t da sala atual na tabela
5:    para todo Sala s da tabela ≠ sala faça
6:       custo ⇐ custo_sala(s) //Custo para chegar à sala s da sala atual
7:       fator ⇐ sala[s].U/custo //Divide o grau de urgência da sala pelo seu custo
8:       se fator > melhor_fator então
9:          melhor_fator ⇐ fator
10:         sala = s
11:      fim-se
12:   fim-para
13:   Vai para a sala com o melhor (maior) fator
14: fim-loop
```

3.3 Resultados Obtidos

Após vários testes e aperfeiçoamentos nas simulações como: O robô esperar 5 segundos para considerar que a sala foi visitada, simulando uma análise de dados qualquer; E a tolerância para o robô chegar em um vértice intermediário ser maior do que para chegar em um vértice-sala, o que faz com o que o robô não perca tempo chegando exatamente no ponto de um vértice intermediário, podendo ir para o vértice seguinte, otimizando a navegação. Foi realizada uma bateria de testes. Cada teste foi executado durante trinta minutos com atualização dos graus de urgência, para montar gráficos, de trinta segundos. A seguir, a trajetória do robô em cada teste, juntamente com uma tabela contendo freqüência absoluta das visitas e a quantidade de pontos e também um gráfico contendo a progressão do grau de urgência total do mapa, que é a soma dos graus de urgência das salas.

A Figura 3.5 mostra a trajetória do robô no Mapa 1 utilizando o Algoritmo 1. Na Tabela 3.3 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 1 no Mapa 1.

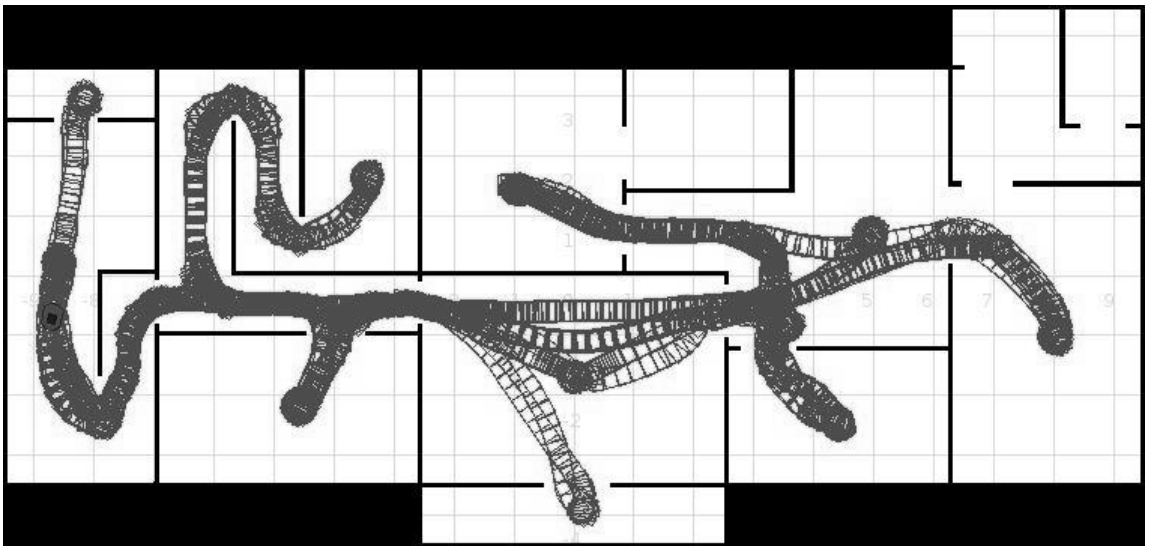

Figura 3.5: Mapa 1 - Algoritmo 1.

Sala	Prioridade	Visitas	Pontos
1	9	5	45
2	2	1	2
3	9	4	36
4	4	2	8
5	9	5	45
6	6	3	18
7	2	0	0
8	9	4	36
9	6	2	12
10	7	3	21
11	1	0	0
12	2	0	0
13	4	1	4
Total			**227**

Tabela 3.3: Freqüência absoluta e pontos do Algoritmo 1 no Mapa 1.

A Figura 3.6 mostra a trajetória do robô no Mapa 1 utilizando o Algoritmo 2. Na Tabela 3.4 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 2 no Mapa 1.

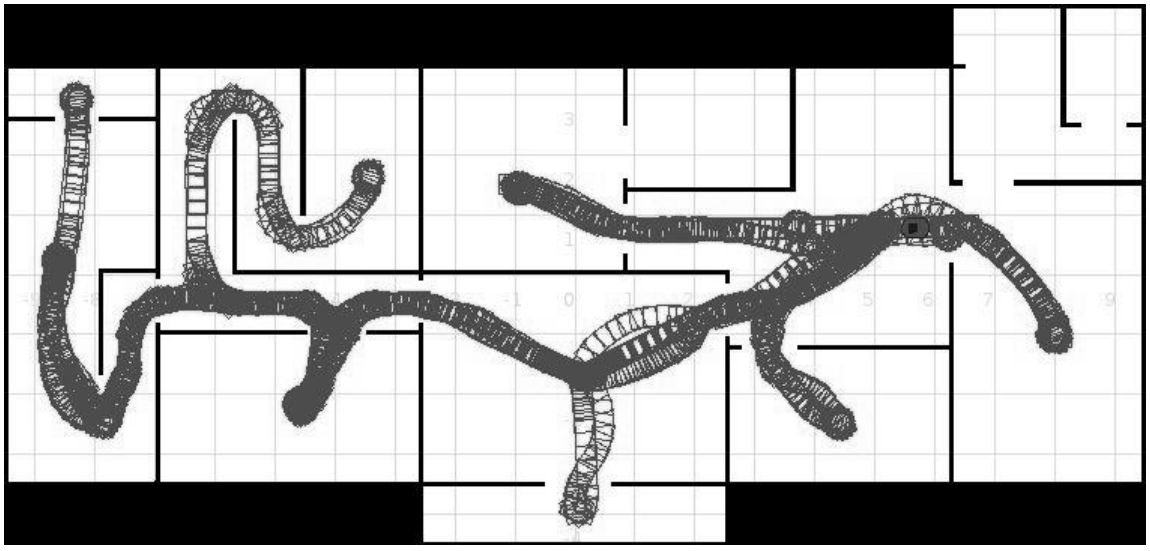

Figura 3.6: Mapa 1 - Algoritmo 2.

Sala	Prioridade	Visitas	Pontos
1	9	5	45
2	2	1	2
3	9	10	90
4	4	1	4
5	9	4	36
6	6	11	66
7	2	0	0
8	9	3	27
9	6	15	90
10	7	3	21
11	1	0	0
12	2	0	0
13	4	1	4
Total			**385**

Tabela 3.4: Freqüência absoluta e pontos do Algoritmo 2 no Mapa 1.

A Figura 3.7 mostra a trajetória do robô no Mapa 1 utilizando o Algoritmo 3. Na Tabela 3.5 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 3 no Mapa 1.

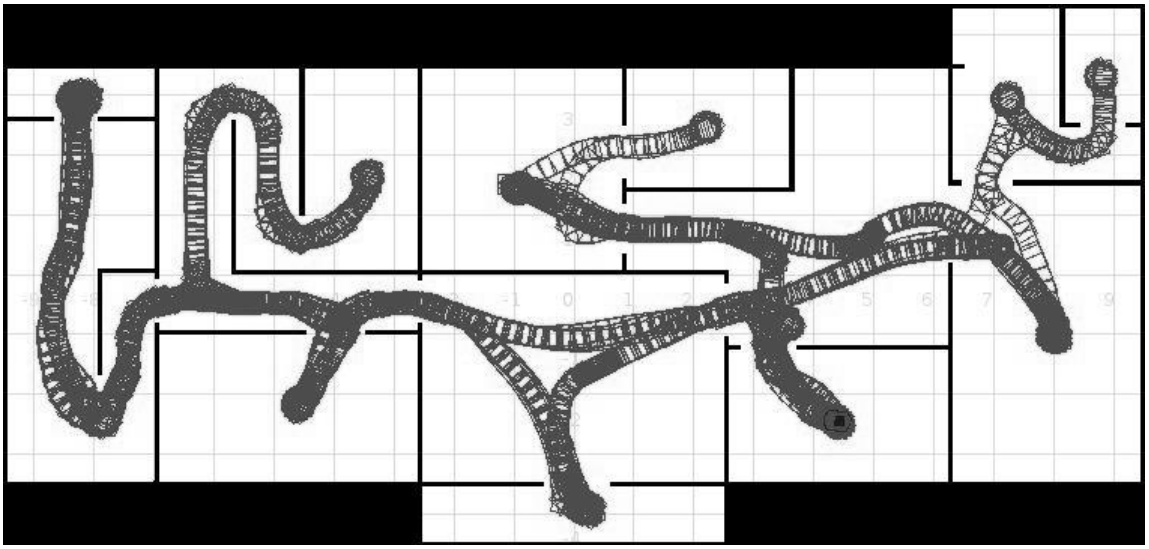

Figura 3.7: Mapa 1 - Algoritmo 3.

O Gráfico na Figura 3.8 demonstra a evolução do Grau de Urgência total no Mapa 1, comparando os três algoritmos.

Sala	Prioridade	Visitas	Pontos
1	9	4	36
2	2	4	8
3	9	4	36
4	4	2	8
5	9	5	45
6	6	4	24
7	2	1	2
8	8	5	45
9	6	5	30
10	7	5	35
11	1	1	1
12	2	1	2
13	4	4	16
Total			**288**

Tabela 3.5: Freqüência absoluta e pontos do Algoritmo 3 no Mapa 1.

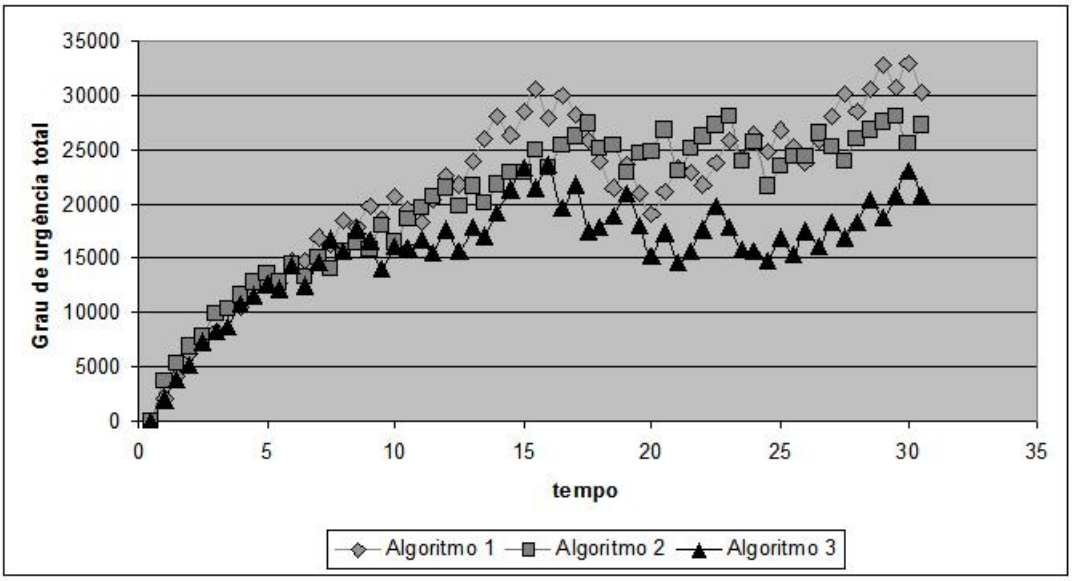

Figura 3.8: Progressão do Grau de Urgência Total no Mapa 1

Sala	Prioridade	Visitas	Pontos
1	2	6	12
2	1	3	3
3	2	6	12
4	1	3	3
5	1	3	3
6	1	3	3
7	2	6	12
8	1	3	3
9	2	6	12
10	3	9	27
Total			**90**

Tabela 3.6: Freqüência absoluta e pontos do Algoritmo 1 no Mapa 2.

A Figura 3.9 mostra a trajetória do robô no Mapa 2 utilizando o Algoritmo 1. Na Tabela 3.6 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 1 no Mapa 2.

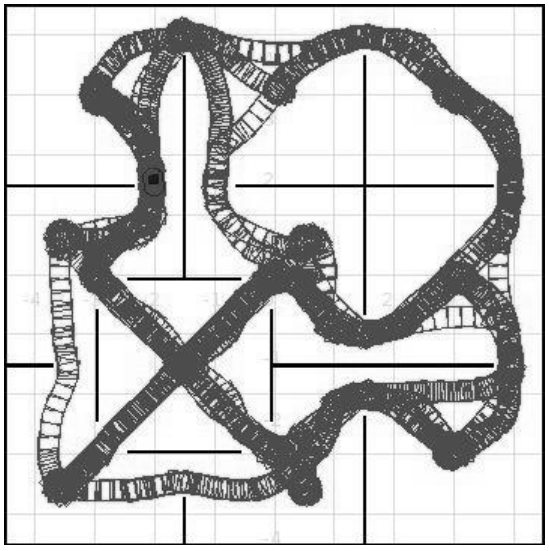

Figura 3.9: Mapa 2 - Algoritmo 1.

Sala	Prioridade	Visitas	Pontos
1	2	6	12
2	1	5	5
3	2	6	12
4	1	7	7
5	1	7	7
6	1	7	7
7	2	6	12
8	1	11	11
9	2	6	12
10	3	13	39
Total			**124**

Tabela 3.7: Freqüência absoluta e pontos do Algoritmo 2 no Mapa 2.

A Figura 3.10 mostra a trajetória do robô no Mapa 2 utilizando o Algoritmo 2. Na Tabela 3.7 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 2 no Mapa 2.

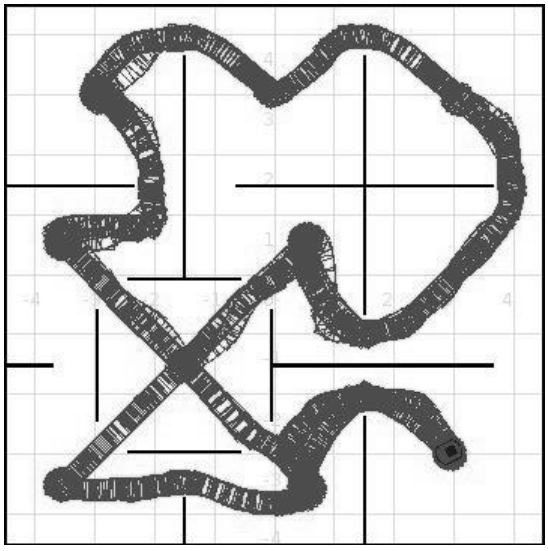

Figura 3.10: Mapa 2 - Algoritmo 2

Sala	Prioridade	Visitas	Pontos
1	2	7	14
2	1	5	5
3	2	8	16
4	1	5	5
5	1	4	4
6	1	4	4
7	2	8	16
8	1	5	5
9	2	7	14
10	3	13	36
Total			**122**

Tabela 3.8: Freqüência absoluta e pontos do Algoritmo 3 no Mapa 2.

A Figura 3.10 mostra a trajetória do robô no Mapa 2 utilizando o Algoritmo 3. Na Tabela 3.8 estão as visitas em cada sala e o total de pontos adquiridos pelo Algoritmo 3 no Mapa 2.

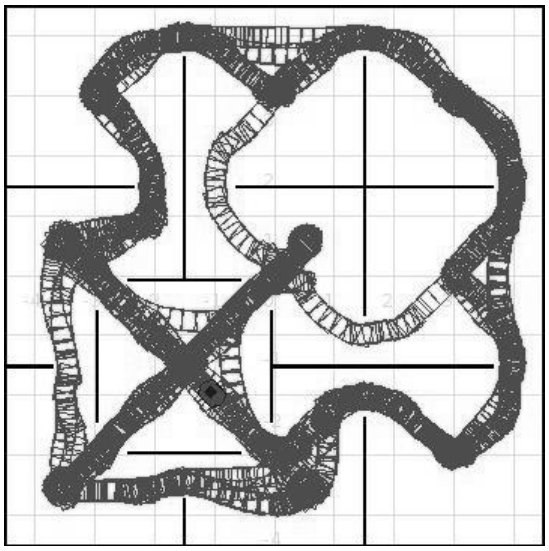

Figura 3.11: Mapa 2 - Algoritmo 3.

O Gráfico na Figura 3.12 demonstra a evolução do Grau de Urgência total no Mapa 2, comparando os três algoritmos.

3.4 Dificuldades e Limitações

Das dificuldades encontradas, vale destacar a dificuldade de dar uma ordem para o robô que é simples do ponto de vista humano, como por exemplo no algoritmo 2 (Algoritmo

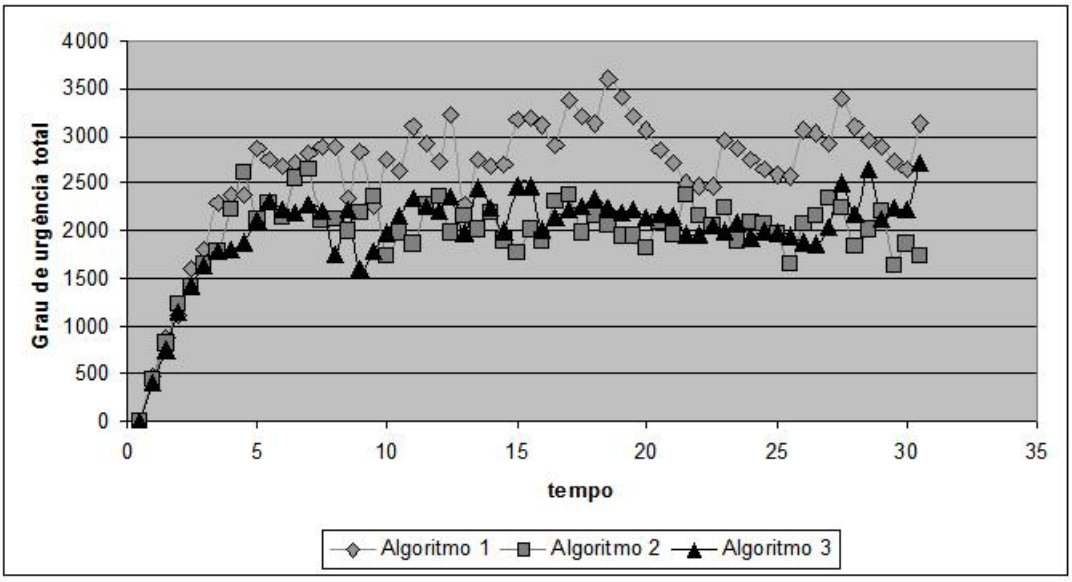

Figura 3.12: Progressão do Grau de Urgência Total no Mapa 2

3.2.3), no qual o robô visita salas adjacentes, no Mapa 1 (Figura 3.3), inicialmente a sala 1 era adjacente à sala 2 e compartilhavam um vértice em comum, isso gerou um problema, pois o robô antes de chegar à sala 1, visitava a sala 2, e assim que visitava a sala 1, ao querer sair da sala ele visitava novamente a sala 2. Para tentar resolver esse problema, foi colocada uma restrição de tempo, onde o robô só visitaria uma sala passado x segundos, mas isso gerou um outro problema, pois o robô, deixava de visitar as salas adjacentes no início do algoritmo, que no início considera que todas as salas acabaram de serem visitadas. A melhor solução encontrada foi colocar um vértice na porta da sala 2, evitando assim que as salas 1 e 2 fossem adjacentes.

Outra dificuldade, nesta mesma área, ocorreu durante o desenvolvimento de um algoritmo que simulava as visitas às salas em todos os caminhos possíveis da sala de origem à sala de destino, na simulação, quando o robô visitava uma sala, contava pontos. Esses pontos eram o Grau de Urgência da sala quando o robô chegava na mesma. Sendo assim, a trajetória com mais pontos seria a ideal, porém o robô gastava tempo desnecessariamente para chegar a uma sala, fazendo com que ganhasse mais pontos ao visita-la. Esse algoritmo foi descartado.

Capítulo

4

Conclusões e Trabalhos Futuros

Analisando os resultados obtidos descritos na seção 3.3, podemos concluir que o desempenho do algoritmo 1 foi inferior em todos os testes, obtendo sempre uma pontuação menor e um grau de urgência total superior aos outros algoritmos. Os algoritmos 2 e 3 alternaram seu desempenho, sendo o algoritmo 2 melhor na pontuação do mapa 1, mas praticamente empatado com o algoritmo 3 no mapa 2 como mostra a tabela 4.1.

Analisando melhor esses dados, podemos verificar que no Mapa 1 (Figura 3.3), o Algoritmo 2 visitou com maior freqüência as salas de maior prioridade (Salas 1 e 3), como mostra na Tabela 3.4. Porém, as salas 7, 11 e 12 não foram visitadas. E as salas 6 e 9 que estão no caminho foram visitadas com muita freqüência. No caso do Algoritmo 3 o robô visitou todas as salas pelo menos uma vez, entretanto visitou com freqüência semelhante salas com prioridades opostas, como as salas 1, 2 e 3 (Tabela 3.5.

Ao analisarmos os gráficos obtemos conclusões diferentes. Pois no gráfico da

	Algoritmo 1	Algoritmo 2	Algoritmo 3
Mapa 1	227	385	288
Mapa 2	90	124	122

Tabela 4.1: Comparação da Pontuação dos Algoritmos nos Mapas.

Figura 3.8 mostra claramente um desempenho melhor do Algoritmo 3, mantendo o Grau de Urgência Total sempre inferior aos outros algoritmos. No entanto, no gráfico da Figura 3.12 os algoritmos 2 e 3 tem um desempenho equivalente, e no final do gráfico existe uma mudança brusca.

Com isso, é possível afirmar que o desempenho dos algoritmos desenvolvidos variam de acordo com a necessidade da aplicação.

Como continuação desse projeto, pretende-se desenvolver um projeto semelhante utilizando uma equipe de robôs móveis, criando novos algoritmos de determinação de trajetória, alternando entre se haverá comunicação entre os robôs, se os robôs serão totalmente autônomos ou obedecerão a um mestre.

Capítulo

5

Comentários sobre o Curso de Graduação

O curso de Bacharelado em Ciências de Computação no ICMC/USP é excelente, contudo tenho algumas críticas. Senti falta de mais opções de disciplinas sobre inteligência artificial cursei somente as disciplinas *Inteligência Artificial* e *Introdução a Redes Neurais*. A disciplina *Inteligência Artificial* não correspondeu às minhas espectativas pois eu esperava aprender conceitos de inteligência artificial mas no curso foi ensinado mais a utilizar a linguagem de programação *ProLog* do que conceitos de inteligência artificial.

Uma outra crítica é com relação à postura do instituto quanto às ênfases. Não existe um programa de palestras com pessoas da área de cada ênfase, escolhi a ênfase de sistemas embarcados pela afinidade com a área e por não ter afinidade com as outras ênfases, mas não obtive nenhuma instrução do que seria a ênfase, e o que existe de pesquisa dentro dela, entrei para a área de robótica durante o curso, pois tivemos matérias (*Projeto e Implementação de Sistemas Embarcados I* e *II*) que utilizamos a biblioteca e o simulador Player/Stage, e gostei. Caso, antes da escolha da ênfase, tivesse existido alguma palestra explicando mais sobre a mesma, teria sido mais fácil a decisão.

Referências Bibliográficas

[Bianchi, 2003] Bianchi, R. E. (2003). Sistemas de navegação de robôs móveis autônomos para o transporte de documentos. Tese de Mestrado, ICMC/USP.

[Faria, 2006] Faria, G. (2006). Sistemas de navegação de robôs móveis autônomos para o transporte de documentos. Tese de Doutorado, ICMC/USP.

[Faria et al., 2006] Faria, G., da Silva Martins, P. & Pereira, M. C. (2006). XXVI Congresso da SBC 26, 443–448.

[FriendlyRobotics, 2007] FriendlyRobotics (2007). http://www.friendlyrobotics.com. Visitado em Novembro de 2007.

[Gates, 2007] Gates, B. (2007). Scientific American.com. Visitado em Novembro de 2007.

[Gerkey et al., 2001] Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. S. & Mataric, M. J. (2001). In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001) pp. 1226–1231,.

[Goodrich, 2007] Goodrich, M. A. (2007). Notas de aula, Georgia Institute of Technology.

[IRobot, 2007] IRobot (2007). http://www.irobot.com. Visitado em Novembro de 2007.

[Koren & Borenstein, 1991] Koren, Y. & Borenstein, J. (1991). Proceedings of the IEEE International Conference on Robotics and Automation 2, 1398–1404.

[Nasa & JPL, 2007] Nasa & JPL (2007). http://mpfwww.jpl.nasa.gov/default.html. Visitado em Novembro de 2007.

[Player, 2007] Player (2007). http://playerstage.sourceforge.net/index.php?src=index. Visitado em Novembro de 2007.

[Thrun et al., 2004] Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hähnel, D., Montemerlo, M., Morris, A., Omohundro, Z., Reverte, C. & Whittaker, W. (2004). IEEE Robotics and Automation Magazine 11, 79–91.

Apêndice

A

Atividades Desenvolvidas no Projeto de Graduação I

No primeiro semestre de 2007 o autor desta monografia cursou a disciplina *SCE 292:Projeto de Graduação I* e apresentou o trabalho intitulado *Desenvolvimento de técnicas de monitoramento de ambientes internos utilizando-se robôs móveis* sob orientação do Prof. Dr. Denis Wolf. Para uma melhor compreensão deste trabalho, que dá continuidade ao projeto iniciado no semestre anterior, as atividades já realizadas são brevemente descritas neste apêndice.

Estudo do ambiente de controle de robôs móveis Player/Stage

Onde foi estudado a biblioteca e o simulador Player/Stage.

Estudo e implementação do algoritmo de desvio de obstáculos

Onde foi estudado e implementado o algoritmo de Campos Potenciais.

Estudo e implementação dos algoritmos de planejamento de trajetória

Onde foi analisado os diferentes algoritmos de melhor caminho (Busca em Profundidade, Busca em Largura, Dijsktra e Ants), e foi implementado o Algoritmo *Ants* pela facilidade de paralelismo

Desenvolvimento e implementação dos algoritmos de determinação das áreas que devem ser visitadas

Onde foi desenvolvido e implementado dois tipos de algoritmos para uma primeira análise do uso da robótica para monitoramento de ambientes internos.

Integração e validação dos algoritmos implementados no simulador Stage

Onde foi testado os algoritmos desenvolvidos, verificando que ambos obedeceram à prioridade das salas, visitando com maior freqüência as salas de maior prioridade e visitando com menor freqüências as salas de menor prioridade.