Sistema remoto para controle de robôs móveis via web[1]

Andrew Wasley Barbosa

Orientador: ***Profª. Drª. Roseli Aparecida Francelin Romero***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências da Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	15/12/2005
Visto do Orientador:	Roseli Romero

USP – São Carlos
Fevereiro/2006

[1]Trabalho realizado com o apoio financeiro do CNPq

Este documento foi preparado utilizando-se o formatador de textos LaTeX. Sua bibliografia é gerada automaticamente pelo BibTeX, utilizando o estilo Chicago.

Resumo

Nesta dissertação foi desenvolvido um sistema para controle remoto de robôs móveis via Internet. Este sistema é composto por um módulo de controle do robô e uma interface Web. O módulo de controle do robô é responsável pela navegação do robô que envolve as seguintes tarefas: mapeamento do ambiente, localização automática do robô e planejamento de trajetórias. A interface Web foi deseuvolvida em Java e é responsável por estabelecer a comunicação entre o usuário e o robô. Através desta interface, o usuário pode controlar o robô e visualizar as imagens capturadas durante a execução de uma determinada tarefa.

Este sistema foi testado tanto em ambientes simulados quanto em um ambiente real. Para realização dos testes, foram utilizados vários usuários enviando requisições ao sistema simultaneamente. Nestes testes foram utilizados vários ambientes diferentes. Em todos os experimentos, o sistema proposto neste trabalho apresentou resultados satisfatórios tanto em relação à comunicação entre os vários módulos que compõem o sistema quanto na realização das tarefas por parte do robô.

Este sistema integra recursos de hardware e software e disponibiliza esses recursos a qualquer usuário que esteja conectado na Internet. Usuários de qualquer parte do mundo poderão, graças ao sistema desenvolvido, controlar um robô remotamente.

Abstract

In this work we develop a system for robot remote control through the Internet. This system is composed by a robot control module and a Web interface. The robot control module is in charge of robot navigation and executes the following tasks: environment mapping, automatic localization and planning. The Web interface was developed in Java and provides the user interface, as well as the communication between the user and the robot. Through the interface the user can control the robot and look at the images taken by the robot on the execution of a given task.

The system was tested in either simulated and real-world environments. For testing, several users were allowed to send requisitions simultaneously to the robot. We also used different environments. In all tests, our system produced satisfactory results regarding both the communication between its modules and in the task execution by the robot.

Our system integrates hardware and software resources and enables any user located all over the world to control the robot using the Internet.

Aos meus pais, sem o apoio deles nas horas difíceis seria impossível a conclusão desta dissertação. Por sempre confiarem no meu potencial, até mesmo quando eu não acreditava. Pelo esforço de me manter sem bolsa durante um ano, mesmo quando o dinheiro estava "curto". Vocês são tudo pra mim e eu os amo muito. Muito obrigado pela vida. Muito obrigado por tudo.

Agradecimentos

Em primeiro lugar gostaria de agradecer as pessoas que colaboraram diretamente para o desenvolvimento dessa dissertação. Agradeço a minha orientadora, Profa. Dra. Roseli Aparecida Francelin Romero, pela paciência e pela ótima orientação durante todo o desenvolvimento deste trabalho. Agradeço aos amigos que me ajudaram durante todo o mestrado, Rodrigo Bianchi, Gédson Faria, Marcos Quiles e Rodrigo Calvo, muito obrigado por sempre trocarem idéias comigo, por me ajudarem na solução de problemas e me ajudarem nos testes com o robô. Agradeço a Huei por sempre me incentivar, desde a época da graduação, sem o seu incentivo provavelmente eu nunca tentaria fazer o mestrado. Agradeço ao Ronaldo pela consultoria em LaTeXe pela revisão do meu abstract. A todos vocês o meu Muito Obrigado.

Agradeço aos meus pais, por todo o apoio e pelas palavras de carinho sempre que alguma coisa dava errado. Agradeço pela força que vocês me deram sempre, tanto nos momentos difíceis de adaptação em uma nova cidade, nas horas que tudo parecia conspirar contra mim, quanto nos momentos de alegria. Muito Obrigado.

Agradeço a minha namorada e melhor amiga Paula pelo seu companheirismo, pelo seu amor, carinho e dedicação. Por me fazer muito feliz e por sempre tentar me alegrar em dias em que tudo dava errado. Você foi um anjinho da guarda que Deus colocou no meu caminho. Te amo muito e Muito Obrigado por você existir.

Agradeço aos meus familiares por sempre me apoiarem durante a minha caminhada no mestrado. Vó Nenem, Vô Augusto, Tia Rosa, Tia Vivi, minha irmã Aliandra e principalmente ao meu tio Kaw, que além de ser o meu melhor amigo e me dar a honra de ser padrinho da sua filha, ainda me bancou financeiramente durante o tempo que eu fiquei sem bolsa. Muito Obrigado.

Agradeço aos meus amigos pelo incentivo, pelas festas, pelos churrascos, pelas cervejadas e por sempre me distrairem quando eu estava muito preocupado com o mestrado. Vocês tiveram um papel importantíssimo nesse trabalho. Muito Obrigado.

Ao pessoal do LABIC, alguns se tornaram meus amigos, outros apenas são meus colegas, mas eu gostaria de agradecê-los pelo companheirismo, pelo bom convívio no laboratório e por sempre se mostrarem dispostos a ajudar. Muito Obrigado a todos vocês.

Finalmente, agradeço ao CNPq pelo apoio financeiro dado a este trabalho. Muito Obrigado.

Sumário

1 Introdução **1**

2 Robôs Teleoperados **5**

2.1 Teleoperação 5

2.2 Robôs Manipuladores 7

2.2.1 *TeleGarden* 7

2.2.2 O projeto *Telelabs* 7

2.3 Robôs Móveis Autônomos 8

2.3.1 O Robô Xavier 10

2.3.2 O Robô Minerva 15

2.4 Pesquisas no Brasil 18

2.5 Considerações Finais 19

3 O Sistema de Navegação Autônomo **21**

3.1 Localização 21

3.1.1 Localização de Markov 24

3.2 Mapeamento 26

3.3 Planejamento de Trajetórias 29

3.3.1 *Vector Field Histogram* 31

3.3.2 Planejador de Trajetórias de Alto Nível 33

3.4 Considerações Finais 33

4 A Interface Web **35**

4.1 Descrição da Interface 35

4.1.1 Linguagem Java 37

4.1.2 *Applets* 40

4.2 Funcionamento da Interface 43

4.3 Considerações Finais 45

5 Descrição do Sistema **47**
5.1 Arquitetura do Sistema de Teleoperação 47
5.2 Funcionamento do Sistema 48
5.3 Hardware Robótico 50
5.4 Considerações Finais 53

6 Experimentos Realizados **55**
6.1 Testes do Sistema de Navegação 55
6.2 Testes Simulados 56
6.3 Testes com o Robô 58
6.4 Limitações do Sistema 60
6.5 Considerações Finais 64

7 Conclusão e Trabalhos Futuros **67**
7.1 Trabalhos Futuros 68

Referências Bibliográficas **73**

Lista de Figuras

2.1 O braço robótico responsável pelo jardim remoto *TeleGarden* 8
2.2 Robô(TeleRobot) utilizado no projeto Telelabs 9
2.3 Interface de controle do robô (TeleRobot) 10
2.4 Xavier 11
2.5 Módulos que compõem o sistema de navegação do robô Xavier 12
2.6 Interface Web do Robô Xavier 13
2.7 Sistemas de controle On-Board e Off-Board e Arquitetura de Controle do robô Xavier com o módulo de interface Web 14
2.8 Minerva 15
2.9 Interface Web Minerva 18

3.1 Grafo de uma representação relacional para um planejamento topológico 30
3.2 Obtenção do mapa métrico utilizando o sistema de navegação do robô Pioneer 1 e visualização do mapa topológico construído a partir desse mapa métrico 34

4.1 Ciclo de vida das applets 41
4.2 Visualização da interface de controle do robô com o grafo correspondente ao mapa topológico do sistema de controle do robô 44

5.1 Arquitetura cliente-servidor do sistema de teleoperação 48
5.2 Arquitetura do Sistema de Controle do Robô 49
5.3 Robô Pionner 1 equipado com o sensor laser SICK PLS 52

6.1 Exemplo de uma mapa em forma de "H" com quatro requisições de usuários diferentes 57
6.2 Exemplo de um mapa fictício de um escritório com requisições de quatro usuários diferentes 58
6.3 Caminho realizado pelo robô para executar a requisição do usuário 1 nos dois casos de teste 59

6.4 Caminho realizado pelo robô para executar a requisição do usuário 2 nos dois casos de teste 60
6.5 Caminho realizado pelo robô para executar a requisição do usuário 3 nos dois casos de teste 61
6.6 Caminho realizado pelo robô para executar a requisição do usuário 4 nos dois casos de teste 62
6.7 Planta do prédio de laboratórios do ICMC 63
6.8 Mapa métrico obtido do mapeamento do corredor do prédio de laboratórios do ICMC 63
6.9 Mapa topológico criado manualmente sobre o mapa métrico 63
6.10 Interface web com o mapa correspondente ao corredor do prédio de laboratórios do ICMC 64

Capítulo

1

Introdução

A idéia de se controlar um robô é, para muitos, fascinante. Porém, o controle de robôs móveis é uma tarefa difícil de ser executada, dado que o robô tem que interagir com o ambiente, desviando de obstáculos e pessoas. Todo esse controle tem que ser realizado utilizando apenas as informações obtidas dos seus sensores.

A principal tarefa para o controle de um robô móvel consiste em realizar a navegação do robô de forma autônoma. A navegação é uma das tarefas mais desafiadoras a ser realizada, pois envolve todas as áreas da Robótica na Inteligência Artificial: sentir, agir, planejar, hardware, eficiência computacional e resolução de problemas (Murphy 2000).

Nos últimos anos, o uso da Internet cresceu de forma vertiginosa, fazendo parte do dia a dia das pessoas, seja para fazer compras, para ler notícias ou até mesmo para entretenimento. Por essa razão, muitos pesquisadores começaram a usar a Internet para sistemas teleoperados (Grange, Fong, and Baur 2000).

Existem alguns exemplos interessantes de robôs que puderam ser controlados online via Web. Um deles é o Xavier (Simmons et al.) do Learning Lab da Carnegie Melon University - CMU - USA; o Rhino[1] (Buhmann et al. 1995), existente na Universidade Bonn - Alemanha, que pode ser controlado durante uma apresentação no Museu de Deutsches, em Bonn; e um dos mais recentes, a Minerva[2] (Thrun et al. 1999), também do Learning Lab. da CMU. Dos robôs citados, o Xavier foi o que passou mais tempo podendo ser controlado por meio de um interface na Web. Acessando o site[3], as pessoas que navegavam pela Internet podiam enviar tarefas para o robô executar. Estima-se que mais de 30.000 requisições foram enviadas para

[1] `http://www.informatik.uni-bonn.de/~rhino/`
[2] `http://www.cs.cmu.edu/~Minerva`
[3] `http://www.cs.cmu.edu/~Xavier`

o Xavier e que o robô teve que se locomover por mais de 210km para poder realizar as requisições enviadas. No Brasil também existem alguns trabalhos nesta área, tais como, o laboratório virtual desenvolvido pelo CenPRA[4] em Campinas (Guimarães; et al. 2003) e o GRACO[5] (Grupo de Automação e Controle) da Universidade de Brasília.

O sistema proposto no presente trabalho faz a teleoperação de um robô móvel via Internet. Este sistema foi testado no robô Pioneer 1, pertencente ao LABIC[6]. Por meio deste sistema, usuários de qualquer parte do mundo podem controlar um robô de forma simples, utilizando uma interface Web. Este sistema é composto de um módulo responsável pela navegação do robô e um módulo responsável pela interface Web.

O módulo de navegação utilizado neste projeto foi desenvolvido em (Bianchi 2003). Este módulo de navegação é constituído de vários outros módulos, tais como, mapeamento, localização e planejamento de trajetórias. O módulo de mapeamento tem a finalidade de modelar o ambiente, de forma que o robô possa se locomover e desviar dos obstáculos. A localização é responsável por estimar o posicionamento do robô no ambiente. O módulo de planejamento de trajetórias consiste em fazer o deslocamento do robô de um ponto de partida a um ponto de chegada da melhor maneira possível.

Para se adequar a este projeto o módulo de navegação apresentado em (Bianchi 2003) precisou ser aperfeiçoado. Este aperfeiçoamento foi realizado pois o usuários necessitavam informar alguns pontos por onde o robô deveria passar, a fim de navegar de forma correta pelo ambiente. Um módulo de planejamento de trajetórias de alto nível foi implementado para sanar esse problema e é explicado em detalhes na Seção 3.3.2.

O módulo de interface é responsável por fazer a comunicação entre o sistema de navegação e o usuário. Por meio da interface o usuário envia as requisições de tarefas para o robô. Essas requisições são repassadas, via Internet, para o sistema de controle do robô e em seguida são enviadas para o robô via rádio modem.

Pretende-se, que com este trabalho, as pessoas que não são da área de robótica possam observar o robô executando suas tarefas e controlar um robô de forma simples, enviando requisições de tarefas para o robô realizar.

Os experimentos realizados mostram que o sistema funcionou de forma satisfatória na tarefa de controle remoto do robô. Limitações foram encontradas, discu-

[4]Centro de Pesquisas Renato Archer

[5]`http://www.graco.unb.br/`

[6]Laboratório de Inteligência Computacional do Instituto de Ciências Matemáticas e de Computação

tidas e são apontadas como trabalhos futuros.

Esta dissertação está organizada da seguinte maneira.

No Capítulo 2, são apresentadas algumas definições sobre teleoperação e os principais robôs controlados pela Internet são apresentados.

No Capítulo 3, é apresentada uma descrição detalhada dos módulos responsáveis pela navegação autônoma do robô. Esses módulos são: localização, mapeamento e planejamento de trajetórias (Bianchi 2003).

No Capítulo 4, a interface Web proposta é apresentada. O funcionamento da interface e os módulos que compõem essa interface são descritos.

Uma descrição detalhada do funcionamento dos módulos que compõem o sistema é apresentada no Capítulo 5.

No Capítulo 6, todos os experimentos realizados neste trabalho são apresentados e discutidos.

Finalmente, no Capítulo 7 algumas conclusões e trabalhos futuros são apresentados.

Capítulo

2

Robôs Teleoperados

Em tarefas de risco, tais como, detecção de minas em campo de batalha, situações de resgate de pessoas, em meio aquático, no espaço, entre outros, existe a necessidade de se ter um sistema de controle de robôs a distância. Esse tipo de controle é conhecido como teleoperação. A teleoperação consiste em um operador humano controlar um robô a distância. Este controle pode ser utilizado em robôs móveis autônomos e em robôs manipuladores.

No controle remoto de um robô móvel autônomo, o usuário necessita apenas enviar comandos para o robô. Após receber os comandos enviados pelo usuário, o robô realiza todas as tarefas necessárias para executar esse comando. O usuário apenas acompanha a execução da tarefa por meio de imagens.

No caso de robôs manipuladores, o usuário "guia" o robô durante toda a execução da tarefa. Esse controle é realizado utilizando-se as imagens coletadas por câmeras acopladas ao robô. No controle remoto de um robô autônomo, as imagens apenas mostram o robô realizando a tarefa. No caso de robôs manipuladores, as imagens são necessárias para auxiliar na realização da tarefa. O usuário necessita das imagens para poder controlar o robô e evitar possíveis obstáculos que estiverem pelo caminho.

Neste capítulo são apresentados os principais robôs que puderam, ou ainda podem, ser controlados remotamente pela Web. O capítulo é estruturado como segue: Na Seção 2.1 são explicados conceitos importantes sobre teleoperação. Por fim, alguns robôs teleoperados via Internet são mostrados nas Seções 2.2 e 2.3.

2.1 Teleoperação

Teleoperação consiste em um processo no qual um operador humano controla um robô (ou máquina) a distância (*tele* significa "remoto"). Inicialmente desenvolvida para manipulação de robôs em áreas que apresentam riscos aos humanos, a teleoperação permite que um operador exerça força e realize movimentos sobre uma

máquina remota e ainda receba retroalimentação sensorial, geralmente por meio de dados visuais, sonoros ou táteis.

Com a introdução da tecnologia de teleoperação, foi possível o desenvolvimento de interfaces capazes de prover uma interação satisfatória entre homem e máquina, permitindo que serviços de grande destreza fossem realizados remotamente. A interface responsável pelo controle do robô pode ser um joystick, um equipamento de realidade virtual ou uma página na Internet.

O operador humano, ou teleoperador, é referenciado como *local* (por estar em um computador local) e o robô é conhecido como *remoto* (por estar localizado distante do operador).

O operador local precisa ter algum tipo de display ou mecanismo de controle, enquanto o robô remoto precisa ter sensores, atuadores e energia. O teleoperador não pode observar o que o robô está fazendo diretamente, pois o robô é fisicamente remoto ou o local pode oferecer algum risco a vida humana (por exemplo, em uma usina nuclear, em outros planetas ou até mesmo em um campo minado).

Assim, os sensores que adquirem informações sobre o ambiente remoto, o display que permite que o operador veja os dados dos sensores e o canal de comunicação que conecta a parte local com a parte remota são componentes críticos de um sistema teleoperado (Murphy 2000).

Segundo Zhai and Milgram (1991), os sistemas teleoperados, de acordo com o grau de autonomia, podem ser classificados como:

1. Controle manual sem auxílio de computador;
2. Controle manual com um significativo auxílio ou transformação computacional;
3. Controle supervisionado com a maior parte do controle realizada pelo controlador humano;
4. Controle supervisionado com a maior parte do controle realizada pelo computador;
5. Controle completamente automático, onde os operadores humanos apenas observam o robô realizando suas tarefas.

O sistema proposto nesta dissertação, permite que um robô móvel, funcione de forma completamente autônoma via Web e seu funcionamento será descrito no Capítulo 5.

A seguir, são apresentados alguns robôs manipuladores teleoperados via Internet e na Seção 2.3 são apresentados alguns dos principais robôs móveis autônomos que puderam ser controlados remotamente, de forma online, via Internet.

2.2 Robôs Manipuladores

Segundo o Instituto Americano de Robótica, um robô manipulador é um mecanismo reprogramável e multi-funcional, que é desenvolvido para mover materiais, ferramentas ou outros dispositivos especializados por meio de vários movimentos programados para realizar tarefas variadas (Murphy 2000). Os robôs manipuladores são comumente projetados para serem capazes de realizar uma tarefa repetidamente e com um alto grau de precisão e velocidade.

Os primeiros robôs manipuladores, foram utilizados para manipular material radioativo e auxiliarem na construção de bombas atômicas. Esses robôs tinham que ser teleoperados para que os cientistas não tivessem contato com o material radioativo.

Nos dias atuais, os robôs manipuladores, controlados de forma remota, são utilizados em diversas aplicações, tais como, limpeza de oleodutos, braços robóticos utilizados para soldar tubulações, laboratórios de ensino remoto, entretenimento e etc. Dois exemplos de robôs manipuladores controlados pela Web são apresentados a seguir.

2.2.1 *TeleGarden*[1]

O *TeleGarden* (Figura 2.1) é uma instalação artística que permitiu aos usuários da Web visualizarem e interagirem com um jardim remoto, repleto de plantas vivas. Os usuários podiam plantar, regar e monitorar o crescimento das mudas por meio de movimentos sutis de um braço robótico industrial.

Esse projeto foi desenvolvido pela Universidade da Califórnia do Sul e pode ser acessado, de forma online pela Internet, em Junho de 1995. Desde então, o *TeleGarden* foi acessado continuamente durante sete anos.

No seu primeiro ano, mais de 9.000 membros ajudaram no cultivo das mudas. Em Setembro de 1996, o TeleGarden foi movido para o lobby do Centro de Eletrônica Ars na Áustria, onde permaneceu online até Agosto de 2004. Atualmente o *TeleGarden* está localizado no Museu de Eletrônica Ars na Austria. As pessoas podem visitar o jardim como convidados do museu.

2.2.2 *O projeto Telelabs*[2]

O projeto Telelabs, desenvolvido pela Escola de Engenharia Mecânica da Universidade da Austrália Ocidental, tem como principal objetivo desenvolver e testar novas tecnologias para aprendizado pela Internet. Os estudantes aprendem utilizando

[1] `http://queue.ieor.berkeley.edu/~goldberg/garden/Ars/`
[2] `http://telerobot.mech.uwa.edu.au/index2.html`

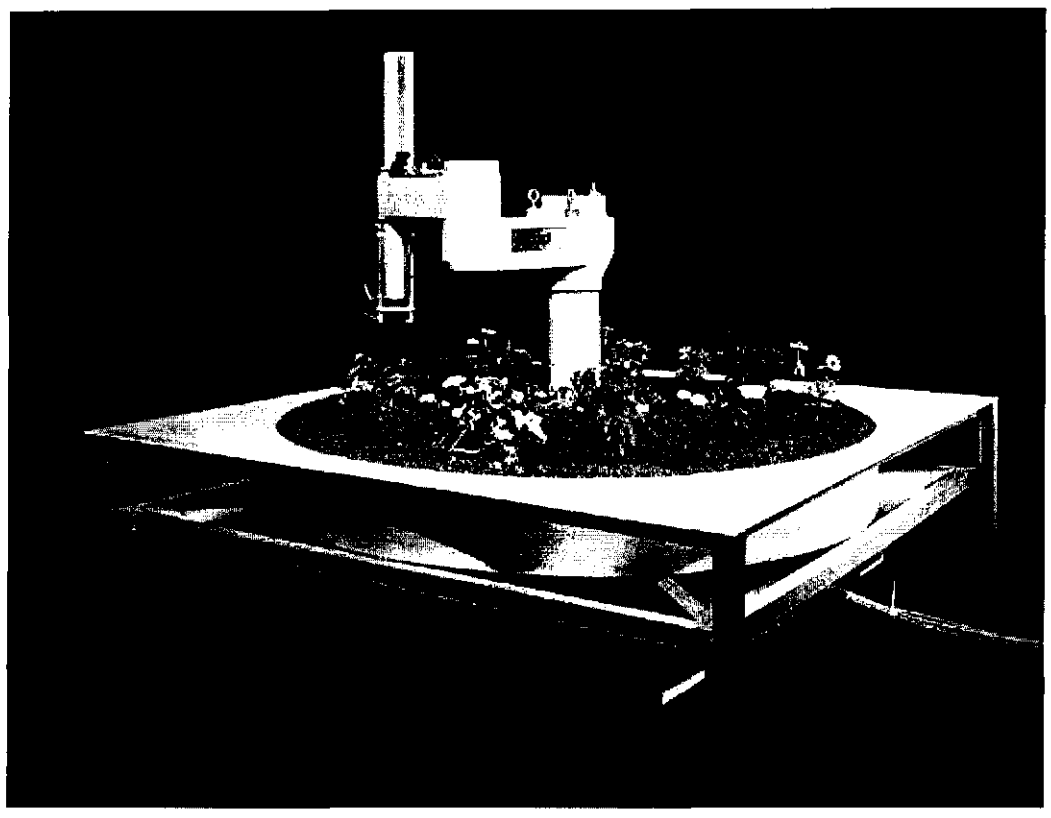


Figura 2.1: O braço robótico responsável pelo jardim remoto *TeleGarden* (figura retirada do site `http://queue.ieor.berkeley.edu/~goldberg/garden/Ars/`)

a teoria, simulação e equipamentos de laboratório.

Esse projeto conta com um robô industrial e algumas câmeras para mostrar ao usuário o que o robô está fazendo. Qualquer pessoa pode acessar o site[3], instalar o programa de acesso ao robô e se cadastrar. Após isso o usuário pode controlar o robô remotamente.

A Figura 2.2 mostra o robô utilizado neste laboratório remoto e a Figura 2.3 mostra a interface utilizada para o controle do robô do projeto Telelabs.

Este foi o primeiro projeto no mundo a possibilitar que pessoas em qualquer parte do mundo, por meio de um navegador (browser), pudessem operar um robô industrial. Estima-se que desde 1994, 500.000 usuários de mais de 100 países diferentes já participaram desse projeto operando esses laboratórios robóticos.

2.3 Robôs Móveis Autônomos

Robôs móveis inteligentes são agentes artificiais ativos, com capacidade de locomoção, imersos no mundo físico real. Agentes por atuarem de forma racional; artificiais por serem máquinas e não criaturas da natureza; ativos por atuarem no

[3] `http://telerobot.mech.uwa.edu.au/Telerobot/index.html`

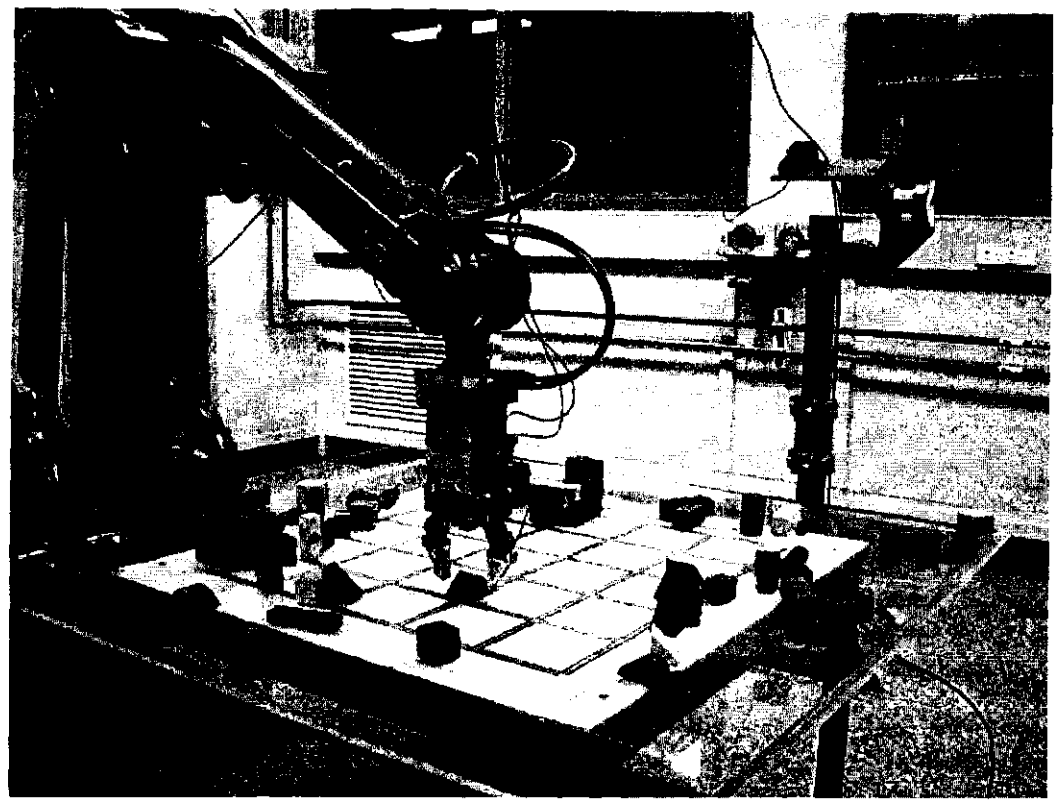


Figura 2.2: Robô(TeleRobot) utilizado no projeto Telelabs (figura retirada do site `http://telerobot.mech.uwa.edu.au/index2.html`).

ambiente de forma propositada, não passiva; com capacidade de locomoção por poderem se mover no ambiente real (Ribeiro et al. 2001).

A locomoção do robô é conseguida por meio de técnicas específicas. Essas técnicas provêem ao robô a capacidade de desviar de obstáculos, fazer o mapeamento do ambiente, saber sua localização no ambiente e fazer um planejamento de trajetória, a fim de conseguir realizar de forma segura as tarefas que forem passadas ao robô.

O controle autônomo de robôs móveis para a realização de tarefas simples, tais como, entrega de documentos (Bianchi 2003), coleta de objetos, é uma área de pesquisa bastante explorada na área de robótica. A teleoperação de robôs móveis também tem sido alvo de atenção de vários pesquisadores para se definir uma arquitetura padrão para realizar o controle de um robô móvel via Internet.

Ainda não se tem uma arquitetura de controle padrão a seguir. Porém, alguns experimentos realizados com robôs controlados via Web foram muito bem sucedidos. É o caso do robô Xavier (Simmons et al.) e do robô Minerva (Thrun et al. 1999).

Por esses robôs servirem de modelo para este projeto, uma descrição detalhada do funcionamento do robô Xavier é encontrada na próxima Seção e a descrição do funcionamento do robô Minerva é apresentada na Seção 2.3.2.

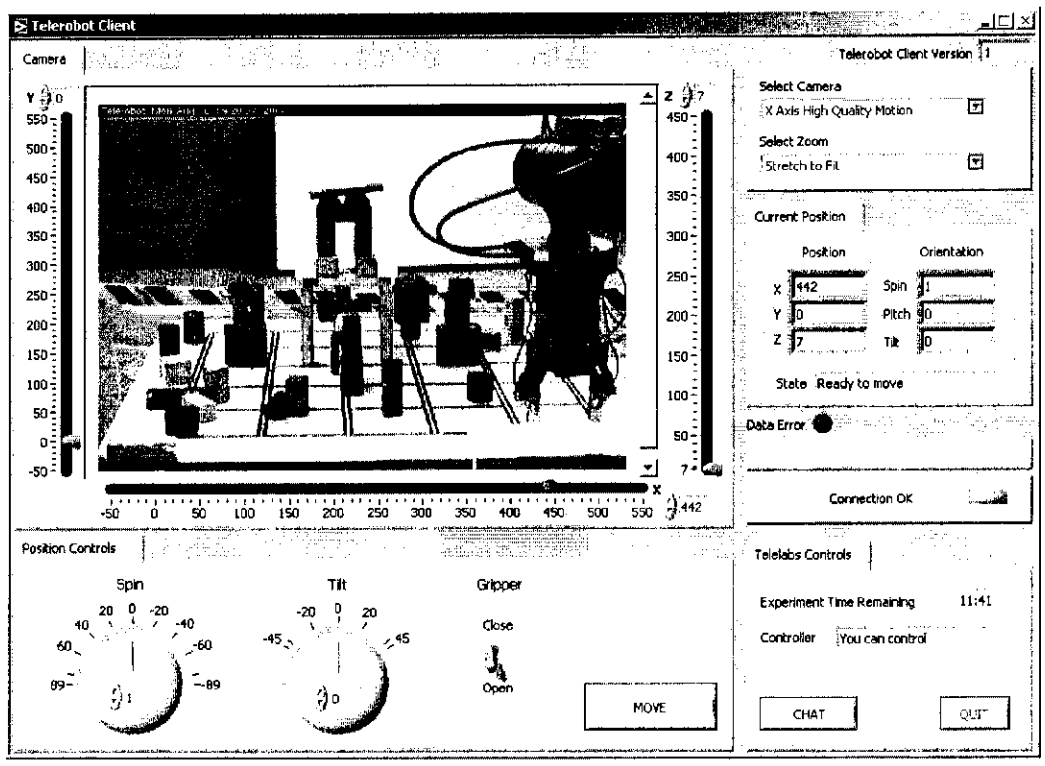


Figura 2.3: Interface de controle do robô (TeleRobot)

2.3.1 O Robô Xavier

O robô Xavier (Figura 2.4) pode aceitar comandos via Web, para se mover entre diferentes escritórios em um determinado prédio, enviando para Web as imagens obtidas durante o percurso percorrido.

Originalmente, esse experimento era apenas para testar um novo algoritmo de navegação autônoma. Depois, o robô passou a receber comandos para executar algumas tarefas, tais como, tirar uma foto ou contar uma piada.

Estima-se que mais de 30.000 requisições foram enviadas para o Xavier e que o robô teve que se locomover por mais de 210km para poder realizar as requisições enviadas (Simmons et al.).

Para fazer o controle do robô Xavier via Web, foi necessário o desenvolvimento de um sistema de navegação autônomo. Esse sistema de navegação será explicado a seguir.

2.3.1.1 Sistema de Navegação Autônomo

O sistema de navegação do Xavier é uma arquitetura modular constituída dos seguintes módulos: servo control, desvio de obstáculos, navegação e controle de trajetória. Cada módulo recebe uma informação dos módulos anteriores e gera um

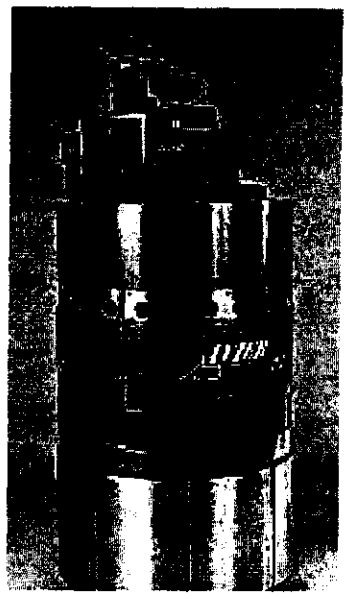


Figura 2.4: Xavier (Simmons et al.)

comando para os módulos posteriores (Simmons et al. 1997). Cada módulo do sistema de navegação é descrito a seguir.

1. **Servo Control**

   Este módulo é responsável pelo controle dos motores do robô. Ele controla a velocidade, direção e provê algumas informações sobre a posição do robô. Geralmente esse módulo é comercializado junto com o robô.

2. **Desvio de Obstáculos**

   Esse módulo mantém o robô em movimento na direção desejada, enquanto desvia de obstáculos estáticos e dinâmicos. Para essa tarefa é usado o método *Lane-Curvature* (Yong and Simmons 1998), que tenta encontrar "caminhos candidatos" para manter o robô na direção desejada, e usa o método *Curvature-Velocity* (Simmons 1996) para alternar entre os "caminhos" e desviar de obstáculos dinâmicos. Ambos os métodos são usados para mover o robô na maior velocidade possível, sem que haja risco do robô bater em algum obstáculo.

3. **Navegação**

   Esse módulo é responsável por movimentar o robô de um ponto para outro. Ele usa o modelo *Partially Observable Markov Decision Process*(POMDP) (Koenig et al. 1996)(Simmons and Koenig 1995) para manter uma alta probabilidade do robô conseguir ir do local onde ele se encontra, para o ponto meta.

4. **Planejamento de Trajetórias**

   Esse módulo define rotas eficientes baseadas em mapas topológicos, nas informações métricas obtidas e nas capacidades do robô. Ele usa uma abordagem *Decision-Theoretic* para escolher planos com maior expectativa, para a escolha da melhor trajetória. Por exemplo, se houver uma possibilidade razoável que o robô não possa ver uma intersecção do corredor, o planejador pode escolher um trajeto mais longo que evite passar por essa intersecção.

Todo o sistema de navegação do Xavier é implementado como sendo uma coleção de processos assíncronos, distribuídos em três computadores on-board no Xavier, que são integrados e coordenados usando a Arquitetura de Controle de Tarefas (Task Control Arquiteture(TCA)).

A TCA provê facilidades na comunicação inter-processos, na decomposição de tarefas, na sincronização de tarefas, na execução do monitoramento, tratamento de exceções e gerenciamento de recursos.

Com a TCA, os processos podem ser facilmente adicionados ou removidos do sistema, mesmo com ele sendo executado. A arquitetura de controle TCA é uma das arquiteturas mais utilizadas nessa área.

A Figura 2.5 mostra os módulos que compõem o sistema de navegação do robô Xavier.

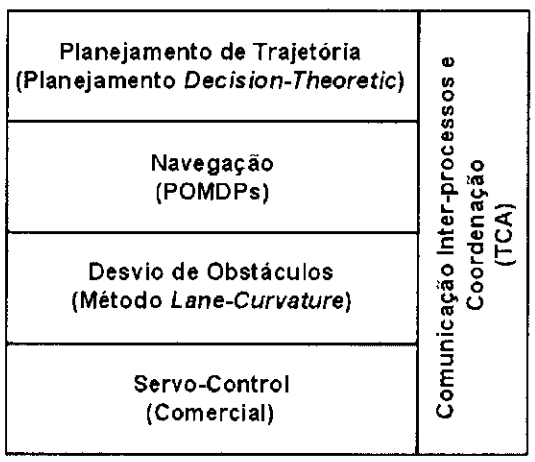


Figura 2.5: Módulos que compõem o sistema de navegação do robô Xavier (Simmons et al.)

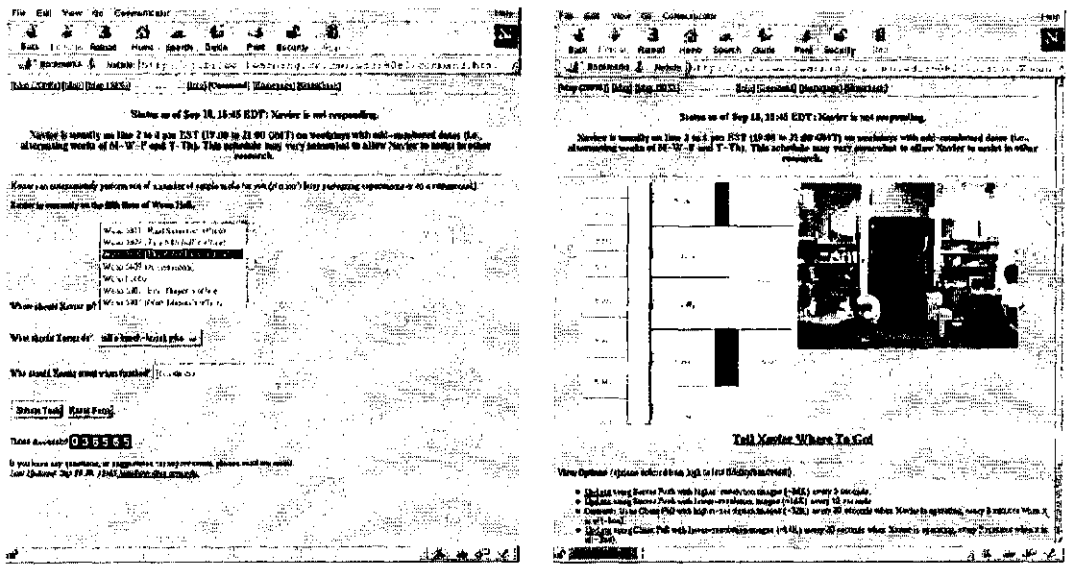
2.3.1.2 Interface Web

A interface Web é projetada com a intenção de tornar fácil a interação com o robô. A página Web com a interface de comando mostra o estado atual do robô Xavier e

provê uma lista de destinos para onde o robô pode ir, uma lista de tarefas que ele pode realizar em cada um dos destinos, e um espaço para colocar um endereço de e-mail. As tarefas que o Xavier pode executar em um determinado destino incluem tirar uma foto, dizer "olá" e contar uma piada.

Em adição à página Web com a interface de controle, existe uma página de monitoramento que inclui o estado atual do robô, um mapa do ambiente com zoom, e uma foto colorida do que o robô está "vendo".

Tanto o mapa, quanto a imagem da câmera, são enviados em formato *.gif*[4], e são atualizados em intervalos de 5-10 segundos. Na Figura 2.6, é mostrada a Interface de Controle do Robô Xavier (Figura 2.6a) e a Página de Monitoramento do Robô Xavier (Figura 2.6b).



(a) Interface de Controle do Robô Xavier (b) Página de Monitoramento do Robô Xavier

Figura 2.6: Interface Web do Robô Xavier (Simmons et al.)

A interface Web é implementada como sendo um módulo adicional no topo do sistema de navegação, mostrado na Figura 2.5, um módulo off-board[5] é adicionado para fazer o gerenciamento da página Web. Esse módulo é mostrado na Figura 2.7a.

O módulo de ordenação de tarefas é responsável por escalonar as tarefas a serem executadas pelo robô Xavier. Sendo assim, este módulo fica comandando o módulo de planejamento de trajetórias, que é encarregado de fazer o robô se deslocar do ponto de partida até o ponto de destino, centralizar o robô na porta se o destino for um escritório ou uma sala de aula, e executar a tarefa.

[4]Formato compactado de codificação de imagens, muito utilizado na Internet

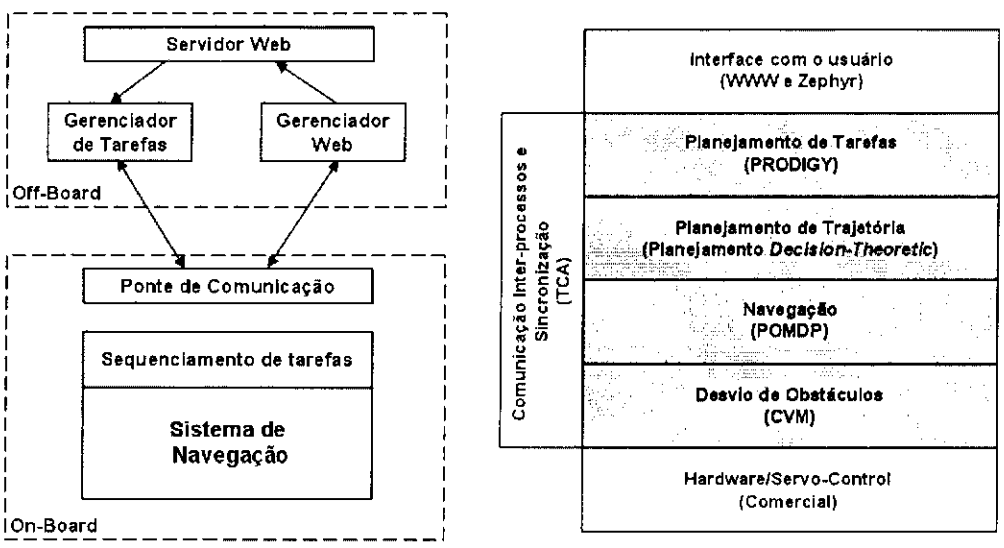
[5]os processos denominados on-board são controlados pelo computador acoplado ao robô e os processos denominados off-board são controlados por um computador externo ao robô.

O processo da ponte de comunicação, que também fica on-board, é responsável pela troca de informações entre os processos on-board e os processos off-board por meio de um rádio modem.

Este processo recebe as requisições e as tarefas a serem executadas dos processos off-board e as passa para o processo apropriado on-board. Na outra direção, as pontes de comunicação recebem as posições estimadas, planos de rota e imagens das câmeras dos processos on-board e os envia para os processos off-board.

A idéia de se ter um módulo de processamento on-board e outro off-board é que se o robô perder a conexão com os processos off-board, ele pode continuar executando a tarefa recebida anteriormente.

O gerenciamento da página Web consiste de dois processos off-board: o gerenciador de tarefas e o gerenciador Web. A arquitetura de funcionamento desses módulos pode ser vista na Figura 2.7a e a arquitetura de controle do robô Xavier com o módulo Web podem ser vistos na Figura 2.7b.



(a) Sistemas de controle on-board e off-board do robô Xavier

(b) Arquitetura de Controle do robô Xavier com o módulo de interface Web

Figura 2.7: Sistemas de controle On-Board e Off-Board e Arquitetura de Controle do robô Xavier com o módulo de interface Web (Simmons et al.)

O processo de gerenciamento de tarefas é responsável por adicionar as requisições em uma fila, enviar requisições para o módulo de ordenação de tarefas e enviar um e-mail para o usuário, confirmando que a tarefa requisitada por ele foi completada.

O processo de gerenciamento Web é responsável pela manutenção das páginas Web. Ele requisita informações sobre posição do robô e imagens das câmeras, cria uma imagem *.gif* mostrando o robô no mapa e cria uma nova página Web contendo

o estado do robô, mapa e as imagens da câmera. Ele também cria uma nova página de interface dependendo de qual andar do prédio em que o robô está.

Mesmo quando o robô está off-line, o gerenciador de tarefas e o gerenciador Web continuam funcionando. Quando uma tarefa é enviada para o gerenciador de tarefas e não é possível fazer uma conexão com o robô, o gerenciador de tarefas assume que o robô está off-line e coloca a requisição na fila de espera para ser realizada quando o robô ficar novamente on-line.

2.3.2 O Robô Minerva

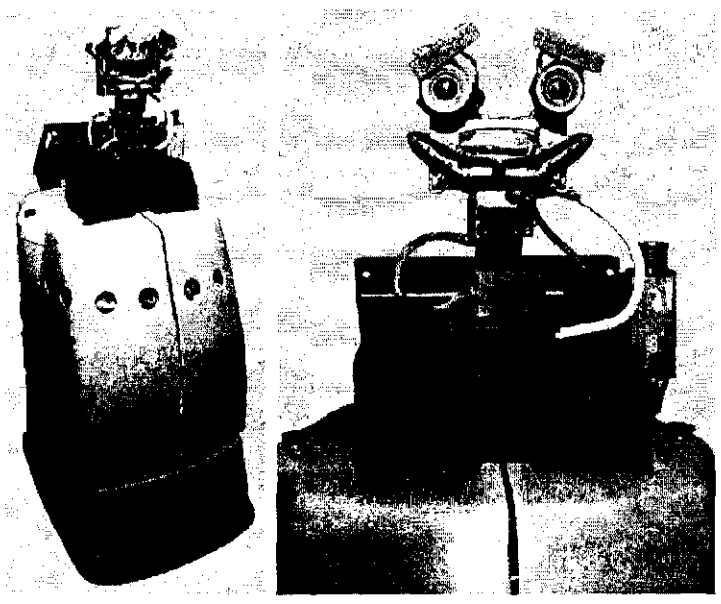


Figura 2.8: Minerva (Thrun et al. 1999)

O robô Minerva (Figura 2.8) foi projetado para interagir, entreter e educar pessoas em lugares públicos. Esse robô foi utilizado como guia turístico de um museu. O robô guiava as pessoas por um passeio no museu e explicava as atrações que estavam sendo vistas ao longo do caminho.

Durante um tempo o robô esteve trabalhando, com muito sucesso, no *Smithsonian's National Museum of American History*, em uma exibição do *Lemelson Center for Invention and Innovation* (Thrun et al. 1999).

Segundo Thrun et al. (1999), para a navegação em ambientes dinâmicos e interação com os humanos o software de controle do robô deve ser capaz de resolver os seguintes problemas:

- **Navegação em ambiente dinâmicos:** Lugares públicos estão cheio de pessoas. A abordagem adotada no robô Minerva provê meios para uma navegação segura e efetiva no meio da multidão.

- **Navegação em ambientes não modificados:** A não modificação do ambiente é necessária para a operação do robô. Apesar disso, o software habilita o robô a se adaptar ao ambiente, em caso de alterações.

- **Interação do Robô com Humanos:**[6] Um software foi especialmente desenvolvido para facilitar a interação com as pessoas que nunca interagiram com um robô antes. Para atrair a atenção das pessoas, a interface do robô usa padrões de interação parecidos com os que as pessoas usam quando interagem umas com as outras.

- **Tele presença virtual:** Uma interface Web possibilita que as pessoas ao redor do mundo possam monitorar o robô, controlar seus movimentos, e ver as imagens que o robô está vendo em tempo real.

2.3.2.1 Arquitetura Geral de Software

Os módulos que compõem a arquitetura de software do robô Minerva se comunicam entre si assincronamente e podem ser classificados em quatro grupos: os módulos de interface de hardware, os módulos de navegação, os módulos de interação e a interface Web.

No nível mais baixo, vários módulos comandam diretamente os sensores e atuadores do robô. Sobre os módulos de hardware estão os vários módulos de navegação que executam funções como mapeamento, localização, desvio de obstáculos e planejamento de trajetórias.

Os módulos de interação determinam o "estado emocional" do robô, controlando a direção da sua cabeça e determinando o que ele fala. Esse estado emocional do robô é definido pelo módulo de interação do robô com humanos e pode ser visto em detalhes em (Schulte et al. 1999).

A interface Web é constituída de módulos responsáveis por mostrarem as imagens e a posição do robô na Web e receber comandos dos usuários. Finalmente, os módulos de alto nível realizam as missões globais de agendamento e controle.

O sistema de navegação usado pelo robô Minerva é muito parecido com o usado pelo Xavier. Na verdade, ele é uma versão melhorada do software usado no robô Xavier. Uma das maiores diferenças entre o robô Minerva e o robô Xavier é que o robô Minerva, além dos módulos responsáveis pela navegação, apresenta um módulo de controle de alto nível, explicado a seguir:

[6] O software desenvolvido para a interação do robô com humanos é explicado detalhadamente em (Schulte et al. 1999)

2.3.2.2 Controle de Alto Nível

O módulo de Controle de Alto Nível do robô Minerva, realiza duas importantes tarefas (Thrun et al. 1999):

1. Durante a operação normal de todos os dias, ele agenda passeios e monitora suas execuções. O tempo aproximado de cada passeio é de 6 minutos, que é determinado pela duração média que as pessoas seguem o robô. O tempo médio do passeio depende criticamente do número e do comportamento das pessoas ao redor do robô.

2. O controlador de alto nível também monitora a execução dos passeios e muda o curso das ações quando uma exceção ocorre. Como exemplo, tem-se o controle da bateria, que se ficar abaixo de um nível crítico força o robô a parar o passeio e voltar para fazer o recarregamento da bateria. Uma exceção também é acionada quando a confiança nas rotinas de localização caem abaixo de um determinando nível crítico. Nesse caso, é necessário parar o passeio temporariamente e acionar a estratégia de re-localização.

2.3.2.3 Interface Web

Com o desenvolvimento do sistema remoto de controle desse robô foi possível que usuários remotos, localizados em qualquer parte do mundo, pudessem estabelecer uma tele-presença virtual no museu, usando a Internet. Para isso o robô Minerva tinha que estar conectado na Internet e podia ser acessado pelo site `http://www.cs.cmu.edu/~Minerva`.

Os visitantes além de controlar o robô Minerva, ainda tinham a possibilidade de fazer um passeio virtual pelo museu. Uma câmera com zoom permitia que os usuários da Web pudessem ver o robô Minerva e as pessoas próximas a ele.

O robô Minerva foi controlado predominantemente pelos visitantes do museu, que podiam selecionar passeios usando uma tela sensível ao toque, colocada em suas costas. Durante todo o passeio, os usuários da Web podiam ver as imagens gravadas pelas câmeras do robô Minerva, por uma outra câmera off-board e podiam ver a localização do robô mostrada no mapa.

Durante algumas semanas o museu ficou fechado para os visitantes e o robô Minerva foi controlado apenas pelos usuários da Web. Usando a interface mostrada na Figura 2.9, os usuários podiam agendar pontos para onde o robô deveria ir. Cada usuário podia escolher até cinco pontos por onde o robô deveria passar e assim "sentir" como se estivesse no museu fazendo um passeio.

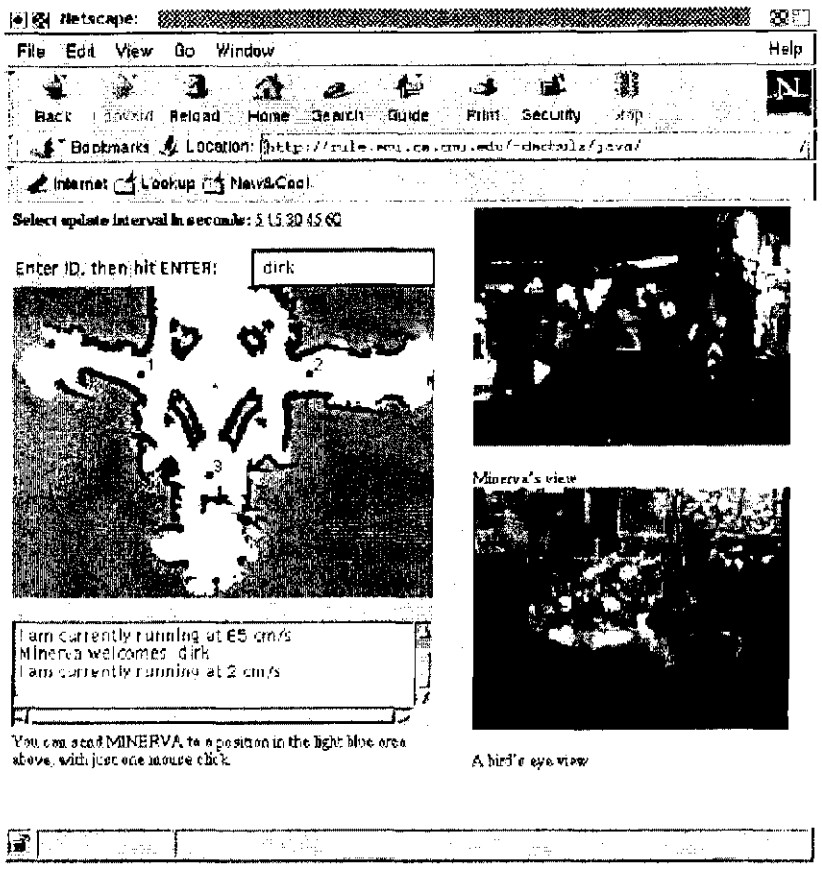


Figura 2.9: Interface Web Minerva (Thrun et al. 1999)

2.4 Pesquisas no Brasil

No Brasil, existem algumas iniciativas para a teleoperação de robôs, porém, ainda são poucas.

Um exemplo de teleoperação de robôs é mostrado no laboratório virtual desenvolvido pelo CenPRA[7] em Campinas. Uma descrição completa deste laboratório virtual pode ser encontrada em (Guimarães; et al. 2003).Esse laboratório virtual é construído com componentes de software e permite que usuários da Internet utilizem um robô, do tipo XR4000, de acordo com o seu nível de experiência em robótica. Os usuários com menos conhecimentos em robótica podem apenas interagir com o robô por meio de teleoperação. Os usuários mais avançados em robótica podem planejar e executar experimentos robóticos complexos no campo da navegação autônoma, mapeamento, fusão de sensores e planejamento de missões.

Alguns trabalhos com teleoperação de robôs podem ser encontrados no site do GRACO [8] (Grupo de Automação e Controle) da Universidade de Brasília.

[7]Centro de Pesquisas Renato Archer
[8]`http://www.graco.unb.br/`

Um destes trabalhos apresenta o desenvolvimento de um robô móvel controlado via Web e é descrito em detalhes em (Alvares and Jr. 2000). O robô pode ser controlado via Internet por meio de uma interface Java. Este controle é realizado por meio de um joystick virtual, onde o usuário define qual a direção que o robô deverá seguir. Durante o controle do robô o usuário visualiza o ambiente através de um sistema de captura de vídeo baseado em uma webcam.

2.5 Considerações Finais

Neste capítulo foram explicados alguns conceitos referentes a teleoperação (ou controle remoto) de robôs. Alguns dos principais robôs, existentes na literatura, que puderam ou ainda podem ser controlados pela Web foram apresentados. Dois robôs mundialmente conhecidos que puderam ser controlados via Web foram apresentados de forma detalhada. É o caso do robô Xavier (Simmons et al.) e mais recentemente do robô Minerva (Thrun et al. 1999).

O capítulo seguinte apresenta o sistema de navegação do robô. Esse sistema é responsável por todo o controle do robô. O módulo de navegação é composto pelos módulos de mapeamento, localização e planejamento de trajetórias. Esses módulos serão explicados de forma detalhada no capítulo seguinte, bem como todos os algoritmos utilizados em cada um desses módulos.

Capítulo

3

O Sistema de Navegação Autônomo

Na maioria dos casos os robôs móveis são projetados de forma a interagir com um meio específico e é a sua capacidade de lidar com mudanças no meio onde atua que o caracteriza como apto para realizar determinadas tarefas.

Para uma interação efetiva com o ambiente, o robô deve ser capaz de obter informações do mesmo. Para isto são utilizados sensores, dispositivos responsáveis pela captação de informações do ambiente, sendo que os mais comuns são: câmeras, sonares (que medem a distância de possíveis obstáculos através de pulsos acústicos) e sensores laser (faz medidas de distância a objetos a partir de radiação eletromagnética).

A navegação autônoma de um robô móvel consiste em um conjunto de tarefas que provê a capacidade do robô se locomover no ambiente. Um sistema de navegação pode envolver diversas tarefas, entre elas estão a localização, o mapeamento, a exploração do ambiente e o planejamento de trajetórias.

Um sistema para a navegação do robô Pioneer 1 existente no LABIC[1] foi proposto em (Bianchi 2003). Este sistema foi aperfeiçoado durante o desenvolvimento do presente trabalho e constitui um dos módulos do sistema que está sendo proposto. Os módulos de localização, mapeamento e controle de trajetórias são descritos a seguir para dar uma idéia melhor do funcionamento deste sistema de navegação.

3.1 Localização

Localização de robôs móveis é uma tarefa que consiste em estimar a posição do robô em um determinado ambiente. As informações para estimar a localização do robô são provenientes dos sensores do robô.

[1] Laboratório de Inteligência Computacional, locado no Instituto de Ciências Matemáticas e de Computação no campus da USP-São Carlos

Para isso o robô deve possuir um mapa do ambiente. Esse mapa pode tanto ser obtido automaticamente, por meio de exploração do ambiente, quanto ser obtido manualmente, ou seja, ser desenvolvido por seres humanos usando alguma ferramenta de projeto auxiliado por computador.

Localização de robôs é reconhecido como um dos problemas mais fundamentais em robótica (Cox and Wilfong 1990). As técnicas de localização desenvolvidas até o presente momento podem ser distinguidas de acordo com o tipo de problema abordado.

As técnicas *tracking* ou local tem como objetivo compensar os erros odométricos que ocorrem durante a navegação do robô. As técnicas globais são projetadas para estimar a posição do robô até sob incertezas globais. Estas técnicas são mais poderosas que as locais e podem enfrentar situações nas quais o robô provavelmente se depare com sérios problemas de posicionamento.

Existem várias abordagens entre os métodos que resolvem o problema de localização. Essas abordagens podem ser categorizadas em quatro dimensões: locais ou globais; para ambientes estáticos ou dinâmicos; ativas ou passivas; um único robô ou múltiplos robôs (Thrun et al. 1998). A seguir, as diferenças entre elas são brevemente descritas:

- **Métodos Locais Versus Globais**

  Abordagens locais para o problema de localização exigem que se tenha a priori a posição inicial exata do robô, pois elas apenas compensam os erros odométricos para manter coerente a informação da localização do robô. Com este tipo de técnica não é possível recuperar a posição do robô quando ocorre alguma falha de localização.

  Abordagens globais são mais robustas. Elas podem estimar a posição de um robô mesmo sem nenhuma informação a priori sobre sua localização. Essa abordagem possui uma maior capacidade de recuperação em caso de falhas. Algumas abordagens globais podem, até mesmo, resolver o problema do "robô seqüestrado", no qual o robô é retirado do seu local e posicionado em outro sem que ele seja informado sobre isso. Este problema é mais complexo do que a localização global sob total incerteza, pois nesse caso, ao invés de o robô não ter nenhuma informação sobre sua posição, ele possui uma informação errônea.

- **Ambientes Estáticos Versus Dinâmicos**

  Ambientes estáticos são aqueles onde apenas o robô se move. Todos os outros objetos que os sensores do robô detectam estão presentes no mapa. Em

contrapartida, ambientes dinâmicos podem possuir vários objetos móveis não presentes no mapa.

A maioria dos métodos de localização podem lidar apenas com ambientes estáticos, ou ambientes com poucos elementos dinâmicos, os quais são tratados como ruídos. Porém, existem algumas abordagens, como a proposta por (Fox et al. 1998), que usam filtros para eliminar as leituras de sensores que possuem alta probabilidade de representar objetos dinâmicos. Dessa forma, pode-se usar métodos para ambientes estáticos, mesmo em locais altamente populosos como museus e bibliotecas.

- **Métodos Passivos Versus Ativos**

  A localização passiva trata exclusivamente da estimativa da posição do robô baseando-se em um fluxo de dados provenientes dos sensores. Nesta abordagem, a movimentação do robô e a direção de seus sensores não podem ser controlados pela tarefa de localização. Nos métodos ativos, a rotina de localização tem o controle total ou parcial do robô. Dessa forma, a rotina de localização pode melhorar a eficiência e a robustez da localização.

- **Único Robô Versus Múltiplos Robôs**

  A abordagem mais comumente usada é a que envolve um único robô. Ela é conveniente pois todos os dados são coletados de um único robô e não existem problemas decorrentes de comunicação. Porém, em times de robôs, os métodos de localização não podem ignorar os outros robôs e o conhecimento de um robô pode ser usado para influenciar a crença de outro robô. Por exemplo, um robô X que sabe apenas que se encontra próximo a uma porta possui várias hipóteses sobre sua posição (cada hipótese corresponde a um lugar próximo a uma porta). Se X for detectado por outro robô Y que sabe exatamente a sua própria posição, X poderá eliminar as falsas hipóteses baseando-se nas informações fornecidas por Y.

Abordagens probabilísticas têm se tornado o paradigma dominante em problemas de robótica, por ser robusto e apresentar um alto nível de performance no mundo real. A idéia central desta abordagem é representar informações incertas através de probabilidades. Em automação de robôs móveis, tais incertezas podem ser encontradas em percepção e ação, pois robôs são inerentemente incertos sobre seu estado do ambiente, devido às limitações dos seus próprios sensores, e conseqüentemente devem ter capacidade de tomar decisões sobre tais estados de incerteza. Em robótica, alcançou-se notável sucesso com algoritmos probabilísticos quando comparados aos

algoritmos convencionais não-probabilísticos (Kaelbling et al. 1996)(Simmons and Koenig 1995).

Neste trabalho foi usado um método probabilístico para localização de robôs, o método de Localização de Markov (Fox et al. 1998). Existem várias formas de se implementar esse método, elas diferem principalmente em três pontos: a representação da crença do robô; nos procedimentos de atualização da crença; e nos modelos probabilísticos de ação e percepção. Neste trabalho foi usado o módulo de localização, implementado em (Bianchi 2003), que utiliza a técnica de Localização de Monte Carlo (Dellaert et al. 1999)(Thrun et al. 1998), descrita em 3.1.1.

3.1.1 Localização de Markov

O método de Localização de Markov é uma técnica global para estimar a posição do robô em seu ambiente. Este método utiliza uma estrutura probabilística para manter probabilidades de posições sobre um conjunto total de possíveis posições do robô. A Localização de Markov é um caso especial de estimativa probabilística de estados aplicada a localização de robôs móveis (Fox et al. 1999).

Seja $l = (x, y, \theta)$ uma posição no espaço de estados do robô, onde x e y são as coordenadas do robô e θ é o ângulo de orientação do robô. A distribuição $Bel(l)$ sobre todas os locais l expressa uma crença subjetiva do robô estar na posição l. Se o robô conhece a posição inicial, $Bel(l)$ é centrada na posição correta; se o robô não conhece a posição inicial , $Bel(l)$ é distribuída uniformemente para refletir a incerteza global do robô. Durante a operação do robô, a distribuição $Bel(l)$ é incrementalmente refinada.

A Localização de Markov aplica dois modelos probabilísticos diferentes para atualizar $Bel(l)$, um modelo de ação para incorporar os movimentos do robô e um modelo perceptual para atualizar a crença sobre as leituras dos sensores.

A movimentação do robô é modelada pela probabilidade condicional $p(l \mid l', a)$ especificando a probabilidade de que, quando a ação a for executada em l', leve o robô para a posição l. $Bel(l)$ é dada por:

$$Bel(l) \longleftarrow \sum_{l'} p(l \mid l', a) \cdot Bel(l'). \tag{3.1}$$

O termo $p(l \mid l', a)$ representa um modelo de cinemática do robô. Na implementação proposta por Fox, Burgard, and Thrun (1998), foi assumido que os erros de odometria[2] são normalmente distribuídos.

Seja s uma leitura de um sensor e $p(s \mid l)$ a probabilidade de se perceber s estando

[2]Processo de estimar o deslocamento de um veículo medindo a rotação dos eixos de suas rodas.

o robô na posição l. Então, as leituras dos sensores são integradas de acordo com a equação de atualização Bayesiana, $Bel(l)$, dada por:

$$Bel(l) \leftarrow \alpha p(s \mid l) Bel(l), \tag{3.2}$$

onde α é um fator de normalização que assegura que a soma de $Bel(l)$ seja 1 para todos os locais l.

Ambos os passos desta atualização Bayesiana são aplicáveis apenas se o problema for *Markoviano*, isto é, as leituras obtidas dos sensores são condicionalmente independentes das futuras leituras dada a localização do robô. Sendo assim, a hipótese de Markov assume que o mundo é estático. Apesar desta abordagem ter sido aplicada na prática até mesmo em ambientes que continham pessoas, os experimentos reportados em (Fox et al. 1998) mostra que ela não se adequa bem em ambientes dinâmicos.

Uma das formas de resolver esse problema consiste na aplicação de filtros, propostos por (Fox et al. 1998), que selecionam quais leituras dos sensores serão utilizadas efetivamente na localização. O objetivo é eliminar as leituras influenciadas por objetos dinâmicos.

Localização de Monte Carlo

O algoritmo de Monte Carlo (Algoritmo 3.1) representa a crença $Bel(s)$ por um conjunto de amostras s associadas a um fator numérico de importância p. Esse fator indica a probabilidade da amostra ser relevante na determinação da posição do robô.

A crença inicial é obtida gerando-se aleatóriamente N amostras da distribuição prévia $P(s_o)$, e atribuindo-se o fator de importância uniforme N^{-1} para cada amostra.

As leituras de sensores o e as ações a são processadas em pares. A localização de Monte Carlo constrói uma nova crença repetindo a seguinte seqüência de "suposições": primeiro, um estado aleatório de s é gerado a partir da crença atual, sob consideração de fatores de importância.

Para aquele estado s, Monte Carlo supõe um novo estado s' de acordo com o modelo de ação $P(s' \mid a, s, m)$ no qual m contém os dados do mapa. O fator de importância para esse novo estado s' recebe um valor proporcional à consistência perceptual desse estado, como medido por $P(o \mid s, m)$.

A nova amostra, juntamente com seu fator de importância, é memorizada e o laço básico é repetido. A geração de amostras é finalizada quando a soma total dos fatores de importância exceder um limite p_{max}, ou quando o próximo par $< o, a >$ é obtido. Finalmente, os fatores de importância são normalizados e o método de Monte Carlo retorna o novo conjunto de amostras definido como crença corrente.

Algoritmo 3.1 Localização de Monte Carlo (S, a, o) (Thrun et al. 2000)

$S' = 0$
$p_{sum} = 0$
enquanto $p_{sum} < p_{max}$ **faça**
  cric uma amostra aleatória $< s, p >$ de S de acordo com $p_1, \cdots, p_N$
  gere um s' aleatório de acordo com $P(s' \mid a, s, m)$
  $p' = P(o \mid s', m)$
  adicione $< s', p' >$ a S'
  $p_{sum} = p_{sum} + p'$
fim-enquanto
normalize os fatores de importância p em S'
retorne S'

3.2 Mapeamento

O mapeamento autônomo de ambientes é muito importante pois a tarefa de ensinar ao robô a sua localização requer um mapa detalhado do ambiente. A tarefa de mapeamento é considerada difícil de se conseguir manualmente ou por meio de plantas, haja visto que esses mapas não contém informações sobre móveis ou outros objetos pertencentes ao ambiente (Carvalho et al. 2004).

Para executar eficientemente missões complexas em ambientes internos, robôs autônomos devem ser capazes de aprender e manter modelos de seus ambientes. O processo de mapeamento de ambiente pode ser dividido em duas tarefas: síntese do mapa e exploração.

A síntese do mapa consiste em dada as observações do ambiente, construir uma representação do mesmo. A exploração é um processo ativo que visa controlar o robô de forma que ele receba leituras dos sensores correspondentes a todas as partes do ambiente.

O problema de adquirir modelos é difícil e está longe de ser resolvido. Os seguintes fatores impõem limites sobre o aprendizado do robô e o uso de modelos precisos (Thrun and Bucken 1996).

1. **Sensores:** Freqüentemente os sensores não são capazes de medir diretamente o fator de interesse. Por exemplo, as câmeras capturam cor, brilho e saturação da luz, considerando que para navegação, seria interessante uma pergunta como: Há uma porta em frente ao robô?

2. **Limite de percepção:** O grau de percepção da maioria dos sensores (sonar, câmera) está limitado a uma pequena área em volta do robô. Para adquirir informações gerais o robô deve explorar ativamente o ambiente.

3. **Ruído nos sensores:** As leituras dos sensores são normalmente distorcidas por ruídos. Freqüentemente, a distribuição destes ruídos não é conhecida (a distribuição é raramente Gaussiana).

4. **Deslocamento:** Os movimentos do robô são imprecisos e infelizmente erros odométricos são acumulados com o tempo. Por exemplo, um pequeno erro rotacional pode causar grandes efeitos subseqüentes aos erros de translação ao estimar a posição do robô.

5. **Complexidade e dinâmica:** Os ambientes são complexos e dinâmicos, sendo impossível manter modelos exatos.

6. **Tempo real:** O tempo de resposta freqüentemente requer um modelo simples e de fácil acesso. Por exemplo, um modelo CAD de alta precisão não é vantajoso, pois as tomadas de decisões não seriam geradas rapidamente.

As pesquisas em navegação de robôs móveis produziram dois grandes paradigmas para mapeamento de ambientes internos: o paradigma métrico (*grid-based*) e o paradigma topológico. Enquanto os métodos métricos (Elfes 1989)(Moravec 1988)(Borenstein and Koren 1991) produzem mapas precisos, sua complexidade freqüentemente dificulta um planejamento de trajetórias eficientes e resolução de problemas em grandes ambientes fechados.

Os mapas topológicos (Matarić 1994)(Kortenkamp and Weymouth 1994), por outro lado, podem ser usados de maneira mais eficiente para cálculo de trajetórias. Mesmo sendo precisos e consistentes, os mapas topológicos são freqüentemente difíceis de aprender e manter quando utilizados em grandes ambientes, particularmente se os dados momentâneos do sensor forem altamente ambíguos (Thrun 1998).

Ambos os paradigmas possuem pontos positivos e negativos, como pode ser observado na Tabela 3.1.

Lee (1996) classifica os tipos de mapas da seguinte forma:

- **Mapas de Locais Reconhecíveis:** O mapa consiste em uma lista de locais que podem ser reconhecidos pelo robô de forma confiável.
- **Mapas Topológicos:** Além dos locais reconhecíveis, o mapa guarda quais locais são conectados por caminhos atravessáveis.
- **Mapas Métrico-Topológicos:** São mapas onde a distância e o ângulo são adicionados a descrição do caminho.
- **Mapas Totalmente Métricos:** Os locais dos objetos são especificados em um sistema de coordenadas fixo.

Tabela 3.1: Vantagens (+) e desvantagens (–) das abordagens métrica e topológica (Thrun 1998).

Paradigma métrico	**Paradigma topológico**
+ fácil para construir, representar e manter	+ permite um planejamento eficiente, baixa complexidade de espaço (resolução depende da complexidade do ambiente)
+ reconhecimento de lugares (baseados na geometria) não possui ambigüidade e é independente do ponto de visualização	+ não precisa da determinação exata da posição do robô
+ facilita a computação dos menores caminhos	+ representação conveniente para soluções de problemas simbólicos, linguagem natural
– planejamento ineficiente, espaço consumido (resolução não depende da complexidade do ambiente)	– dificuldade para construir e manter em grandes ambientes se a informação dos sensores for ambígua
– requer a posição precisa do robô	– reconhecimento de lugares (baseado em marcos) freqüentemente ambíguos, sensível ao ponto de visualização
– interface pobre para a maioria das soluções de problemas simbólicos	– podem produzir caminhos não-ótimos

O método de mapeamento implementando em (Bianchi 2003) e utilizado nesse trabalho é chamado de Grades de Ocupação (*Occupancy Grid*) (Elfes 1989). Este método é classificado como um mapa totalmente métrico e será descrito a seguir.

Grades de Ocupação

A representação dos mapas no método de Grades de Ocupação é uma matriz de ocupação multidimensional que mapeia o espaço em células, onde cada célula armazena uma estimativa probabilística do seu estado. Os estados possíveis são ocupado ou livre.

Esse método trata do problema de construção de mapas consistentes a partir de sensores que apresentam ruído e outras incertezas. Este método baseia-se em um campo aleatório multidimensional que fornece estimativas das células em um espaço reticulado.

As estimativas do estado das células são obtidas por meio da interpretação das leituras de distâncias usando modelos probabilísticos dos sensores. Procedimentos de estimativa Bayesianos permitem a atualização incremental das Grades de Ocupação

usando leituras realizadas por vários sensores sobre múltiplos pontos de vista.

A variável de estado $s(C)$ associada a célula C da Grade de Ocupação é definida como uma variável aleatória discreta com dois estados possíveis, ocupada e vazia, denotadas por OCC e EMP. Dessa forma, a Grade de Ocupação corresponde a um campo aleatório de estados discretos e binários, onde $P[s(C) = OCC] + P[s(C) = EMP] = 1$.

No método de Grades de Ocupação, para entender os dados sensoriais (medidas de distância), é usado um modelo estocástico de sensores definidos por uma função probabilística de densidade, $p(r|z)$, que relaciona a entrada r ao parâmetro de espaço z correspondente a uma posição no mapa.

Esta função de densidade é subseqüentemente utilizada em um procedimento de estimativa Bayesiano para determinar as probabilidades de estado das células das Grades de Ocupação.

Partindo-se da estimativa atual do estado da célula C_i, $P[s(C_i) = OCC|\{r\}_t]$, baseada em observações $\{r\}_t = \{r_1, ..., r_t\}$ e dada uma nova observação, r_{t+1}, a atualização da estimativa é dada por:

$$P[s(C_i) = OCC|\{r\}_{t+1}] = \frac{p[r_{t+1}|s(C_i) = OCC]P[s(C_i) = OCC|\{r\}_t]}{\sum_{s(C_i)} p[r_{t+1}|s(C_i)]P[s(C_i)|\{r\}_t]} \tag{3.3}$$

Nesta formulação recursiva, a estimativa anterior do estado da célula , $P[s(C_i) = OCC|\{r\}_t]$, serve como uma estimativa priori e é obtida diretamente das Grades de Ocupação. Esta nova estimativa é então armazenada no mapa. Sendo assim, o mapa obtido é uma representação das probabilidades de ocupação de cada célula.

3.3 Planejamento de Trajetórias

Robôs móveis autônomos devem saber como chegar a um determinado destino apenas com as informações do ponto de partida e do destino. Para isso é necessário que o robô saiba a sua localização no ambiente e quais ações devem ser tomadas a partir da posição onde ele se encontra.

A localização do robô no ambiente é obtida por meio dos módulos de localização e mapeamento. Após saber a sua localização, o robô tem que decidir qual a melhor opção de comando a ser realizado de forma que ele possa se movimentar no ambiente, desviando de obstáculos, a fim de atingir o destino.

A determinação da seqüência de comandos a enviar aos atuadores para que o robô atinja sua meta, desviando de obstáculos e possivelmente objetivando otimizar o custo é realizada pelo método de planejamento de trajetória.

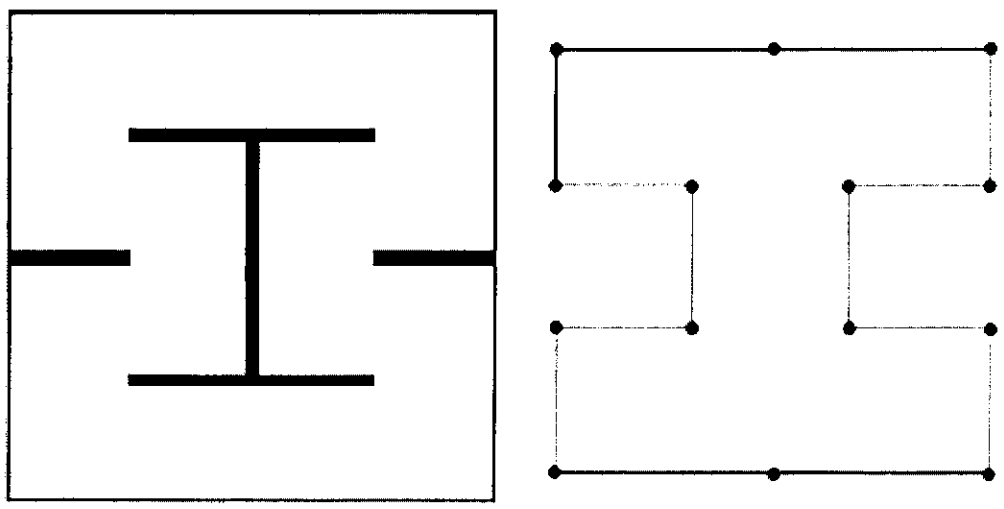
Os métodos para planejamento de trajetória estão divididos em duas categorias: topológicos e métricos (Murphy 2000).

Planejamento de Trajetórias Topológico

Freqüentemente pessoas usam diretivas para explicarem uma determinada rota para outras pessoas, portanto parece bem natural que um robô use um tipo de controle de trajetória baseado em diretivas do tipo "siga esse corredor, vire a esquerda no final do corredor e entre na segunda sala a direita". Mesmo sem um mapa, existe informação suficiente para um robô realizar a navegação, haja visto que essa navegação é baseada em marcos identificáveis e o robô sabe o que é um "corredor", "um final de corredor" e uma "sala".

Existem duas abordagens para representação de rotas:

1. Relacional: As técnicas relacionais são mais populares e podem ser obtidas por meio de uma representação de um grafo da memória espacial. A Figura 3.1 mostra um exemplo de uma representação relacional para planejamento topológico.



(a) Mapa de um ambiente em forma de H (b) Mapa topológico do ambiente H

Figura 3.1: Grafo de uma representação relacional para um planejamento topológico

2. Associativa: Técnicas relacionais tendem a focar a representação de um grafo da memória espacial. Técnicas associativas focalizam a união dos sensores com a localização de uma maneira que converte as informações dos sensores em direções para o robô seguir.

Navegação topológica depende da presença de marcos. Um marco é composto de uma ou mais características perceptualmente distinguíveis de um local ou objeto

de interesse. O planejamento de trajetórias resultante é normalmente uma rota que indica por quais marcos o robô deve passar.

Planejamento de Trajetórias Métrico

O planejamento métrico favorece técnicas que produzem uma trajetória ótima, enquanto o planejamento topológico se contenta em produzir uma rota com marcos identificáveis. Outra diferença é que as trajetórias métricas são usualmente decompostas em submetas, que são normalmente uma posição fixa ou uma coordenada (x, y).

3.3.1 Vector Field Histogram

O método *Vector Field Histogram* (Borenstein and Koren 1991) mantém um mapa métrico probabilístico simplificado baseado em malhas (semelhante ao das Grades de Ocupação). Porém, ao invés valores reais entre 0 e 1 representando a probabilidade de ocupação da célula, ele utiliza valores inteiros de 0 a 5.

Esse mapa difere de uma Grade de Ocupação pela forma como é obtido, pois para operar com a velocidade necessária para que o robô desvie de obstáculos em altas velocidades, a manutenção do mapa não pode gastar muito tempo de processamento.

Dessa forma, ele é obtido por uma técnica chamada amostragem rápida, onde cada célula inicia em 0 e é incrementada em 1 a cada vez que um sensor detecta um obstáculo em sua posição até que ela alcance o valor máximo 5, quando então não mudará mais de valor.

A cada ciclo do algoritmo, é realizada uma amostragem rápida e então, um histograma polar é criado, dividindo as regiões em torno do robô em k setores e medindo a densidade de ocupação de cada setor. Para isso, não é utilizado o mapa inteiro e sim uma janela do mapa chamada de "região ativa" cujo centro é o robô. Cada célula $c_{i,j}$ da região ativa é analisada e são atribuídos dois valores a cada uma, sua direção em relação ao centro do veículo é dada por

$$\beta_{i,j} = \tan^{-1} \frac{y_j - y_0}{x_i - x_0} \tag{3.4}$$

e sua magnitude é dada por

$$m_{i,j} = (c_{i,j})^2 [a - bd_{i,j}] \tag{3.5}$$

onde:

a, b	Constantes positivas.
$d_{i,j}$	Distância entre a célula (i,j) e o centro do robô.
$c_{i,j}$	Valor da célula (i,j).
$m_{i,j}$	Magnitude da célula para a densidade do setor.
x_0, y_0	Coordenadas da célula (i,j).
$\beta_{i,j}$	Direção da célula (i,j) ao centro do robô.

Para setores com largura angular α, a correspondência entre as células ativas e seu setor é dada por

$$k = INT(\beta_{i,j}/\alpha) \tag{3.6}$$

e a densidade de ocupação h_k para cada setor k do histograma polar é calculada por

$$h_k = \sum_{i,j} m_{i,j} \tag{3.7}$$

Devido a natureza discreta do histograma polar, uma função de suavização é aplicada para melhorar sua saída. Essa função é definida por:

$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} + \cdots + lh_k + \cdots + 2h_{k+l-1} + h_{k+l}}{2l+1} \tag{3.8}$$

Com o histograma polar pronto, é necessário determinar o ângulo Ω entre o robô e a meta. Então escolhe-se o setor k referente ao ângulo mais próximo de Ω que tenha densidade de ocupação menor do que uma constante pré-definida.

O ângulo referente a esse setor determina a direção do robô. A velocidade de translação, pode ser determinada com heurísticas em relação as densidades de ocupação próximas ao ângulo referente ao k escolhido.

No trabalho desenvolvido em (Bianchi 2003) foi implementado um módulo de planejamento de trajetória métrico utilizando o algoritmo *Vector Field Histogram*. Apesar do sistema realizar a navegação do robô satisfatóriamente, é necessário que o usuário faça o planejamento manual da trajetória do robô.

Esse planejamento é feito no momento em que o usuário deseja enviar o robô para algum ponto do mapa. Além do usuário ter que escolher o ponto para o qual o robô deve se locomover, ele também deve definir submetas, a fim de assegurar que o robô conseguirá se locomover até o ponto desejado pelo usuário.

Isto é feito, pois o método VFH apresenta algumas limitações que são discutidas e tratadas em (Ulrich and Borenstein 2000). Para minimizar esse problema, um planejador de trajetória de alto nível foi criado e será explicado a seguir.

3.3.2 Planejador de Trajetórias de Alto Nível

Como visto anteriormente, o planejamento métrico favorece técnicas que produzem uma trajetória ótima e estas trajetórias são decompostas em submetas, que nada mais são do que coordenadas (x, y). Porém, só com o planejamento métrico não é possível garantir que o robô conseguirá chegar a sua meta.

Uma forma de se contornar esse problema é criar um mapa topológico a partir do mapa métrico gerado. Esse mapa topológico pode ser gerado de forma automática (Thrun and Bucken 1996) ou manual.

Neste trabalho o mapa topológico está sendo gerado manualmente, a partir do mapa métrico obtido pelo módulo de mapeamento. A geração manual deste mapa não influencia no funcionamento do sistema, pois este mapa tem que ser gerado apenas uma vez para cada ambiente.

O mapa topológico gerado, corresponde a um grafo e, facilita o planejamento da trajetória do robô. Cada nó do grafo corresponde a uma coordenada (x, y) do mapa métrico.

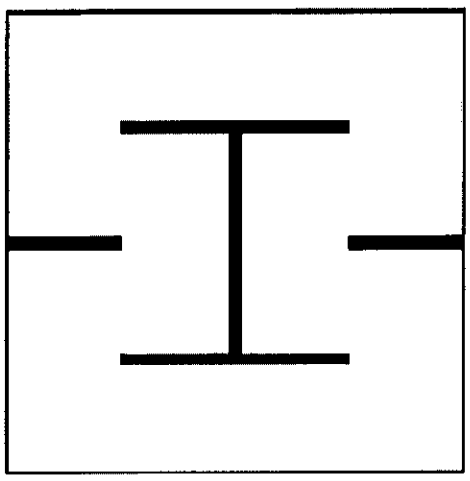
Sendo assim, o grafo pode ser visto como um conjunto de submetas para a navegação do robô e a melhor rota para o robô é definida utilizando-se um algoritmo de menor caminho em grafo. O algoritmo de menor caminho implementado neste sistema foi o algoritmo de Dijkstra (Johnsonbaugh 1997).

O planejador de alto nível faz o planejamento da trajetória de forma autônoma, sem necessitar da intervenção do usuário. Porém, a geração manual do mapa topológico é uma solução paliativa. Pretende-se no futuro implementar um módulo para geração automática deste mapa topológico. A Figura 3.2a mostra um mapa simulado em forma de H, a Figura 3.2b mostra o mapa métrico gerado pelo módulo de mapeamento e a Figura 3.2c mostra o mapa topológico criado manualmente sobre o mapa métrico.

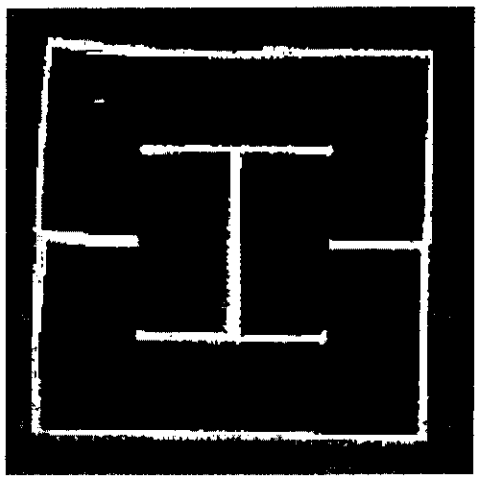
3.4 Considerações Finais

Neste capítulo foram apresentadas as técnicas utilizadas para a navegação autônoma do robô que está sendo utilizado neste trabalho. Esses algoritmos são responsáveis por prover ao robô a capacidade de se locomover no ambiente de forma segura. Essa capacidade é fundamental para um bom rendimento do sistema de controle remoto do robô. Com essas técnicas pretende-se alcançar um alto nível de autonomia, a fim de garantir que não haja intervenção humana no processo de realização da tarefa por parte do robô ou que essa intervenção seja quase inexistente.

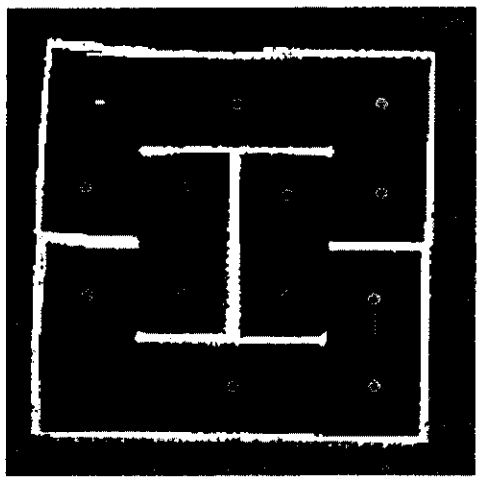
No próximo capítulo serão apresentados os módulos da interface Web. Esses módulos são responsáveis pela comunicação entre o usuário e o sistema de controle



(a) Mapa simulado em forma de H



(b) Mapa métrico



(c) Mapa topológico

Figura 3.2: Obtenção do mapa métrico utilizando o sistema de navegação do robô Pioneer 1 e visualização do mapa topológico construído a partir desse mapa métrico

central do robô.

Capítulo

4

A Interface Web

Neste capítulo, a interface Web desenvolvida será apresentada de forma detalhada. A interface Web é responsável por fazer o controle e o monitoramento do robô. Por meio da interface, o usuário pode enviar comandos para o robô, ver a atualização da posição do robô no mapa do ambiente e pode também ver imagens capturadas pela câmera instalada no robô. Todos esses recursos precisam funcionar em tempo real para que o usuário tenha a real sensação de estar controlando o robô. Na Seção 4.1 são descritos todos os módulos utilizados para o perfeito funcionamento da interface Web e na Seção 4.2 é descrito de forma detalhada o funcionamento desta interface.

4.1 Descrição da Interface

A interface Web é constituída de quatro módulos. Esses módulos são: módulo de comunicação, módulo de controle, módulo de visualização e o módulo de câmera. As funcionalidades desses módulos são descritas a seguir:

- **Módulo de Comunicação**

  Este módulo é responsável por realizar a comunicação da interface Web e o servidor de controle do robô. Esta comunicação é feita via Internet, por meio de sockets. Os sockets criam um canal de comunicação entre 2 computadores, com uma conexão do tipo cliente-servidor.

  Neste trabalho o servidor do socket é o computador, no qual estão funcionando os sistemas de controle do robô, enquanto o cliente é a interface Web. O servidor do socket fica esperando conexões em uma determinada porta. Enquanto o cliente socket, por sua vez, envia um pedido de conexão nesta porta e estabelece uma conexão com o servidor. O servidor socket pode receber, de forma simultânea, conexões de vários clientes.

Todas as informações necessárias para todo o funcionamento da interface são enviadas e recebidas por esse módulo.

- **Módulo de Controle**

  O módulo de controle é responsável por receber as requisições dos usuários. Neste trabalho, os usuários só podem enviar requisições do tipo "vá para tal ponto do mapa". Espera-se no futuro, que este sistema seja estendido para que possa receber algumas tarefas mais elaboradas, tal como, "vá para tal ponto do mapa e pegue algum objeto".

- **Módulo de Visualização**

  Este módulo faz a atualização da posição do robô no mapa da interface. Ele recebe do servidor de controle as coordenadas (x, y, θ) de posicionamento do robô, onde (x, y) são as coordenadas do robô no plano e θ é o ângulo de orientação do robô. Com base nessas coordenadas, o módulo atualiza o desenho do robô no mapa da interface. Esse processo é realizado continuamente e faz com que o usuário "veja" a trajetória realizada pelo robô para executar as tarefas. Essas coordenadas recebidas do módulo de localização do robô precisam ser convertidas para mostrar, de forma precisa, a localização do robô no mapa da interface.

- **Módulo de Câmera**

  Por meio da câmera CMOS NTSC sem fio acoplada ao robô, são capturadas imagens do ambiente por onde o robô está se locomovendo e essas imagens são mostradas na interface Web. Assim como o módulo de visualização atualiza a posição do robô no mapa da interface continuamente, o módulo de câmera faz a atualização das imagens do ambiente continuamente. A idéia desse módulo é fazer com que o usuário possa ver o ambiente com o qual o robô está interagindo. Infelizmente, devido a velocidade da Internet não é possível fazer essa atualização das imagens em tempo real, como se fosse um filme, por exemplo. Entretanto, o que se faz é mostrar uma seqüência de fotos, tiradas a cada 5-10 segundos.

Todos os módulos desta interface Web foram implementados, em uma applet, na linguagem Java[1]. Esta escolha se deve ao fato de Java ser uma linguagem amplamente utilizada em aplicações que rodam na Web e também por sua grande portabilidade, sendo portanto, independente do computador a ser utilizado. Um breve histórico e as principais características da linguagem Java são descritas a seguir.

[1] `http://java.sun.com/`

4.1.1 Linguagem Java[2]

A linguagem Java foi criada como parte de um grande projeto da Sun Microsystems[3], que pretendia criar uma nova geração de computadores portáteis inteligentes, capazes de se comunicar de muitas formas, ampliando suas potencialidades de uso. Para tanto decidiu-se criar também uma nova plataforma para o desenvolvimento desses equipamentos de forma que seu software pudesse ser portado para os mais diferentes tipos de equipamentos. A primeira escolha de uma linguagem de programação para tal desenvolvimento foi C++. Mas em conseqüência de uma lista cada vez maior de problemas com o C++ (principalmente vazamentos de memória e vários problemas de herança) a Sun resolveu criar uma nova linguagem de programação, surgindo assim a linguagem Java.

Como resultado de vários anos de pesquisa e desenvolvimento da Sun surgiu mais que uma simples linguagem de programação, surgiu um ambiente de desenvolvimento e execução de programas que provê todas as facilidades proporcionadas pela orientação a objetos, pela extrema portabilidade do código produzido, pelas características de segurança que esta plataforma oferece e finalmente pela facilidade de sua integração aos outros ambientes, destacando-se a Internet.

A arquitetura do Java não é nem radical nem totalmente nova. Resumindo, os aplicativos em Java são compilados em *bytecodes* independentes de arquitetura. Esses *bytecodes* podem então ser executados em qualquer plataforma que suporte um interpretador Java. O Java requer somente um arquivo fonte e um binário e, mesmo assim, é capaz de funcionar em diversas plataformas, tornando-o extremamente portável. As principais características da linguagem Java são descritas a seguir:

- **Orientada à Objetos**

  Java é uma linguagem puramente orientada à objetos pois, com exceção dos seus tipos primitivos de dados, tudo em Java são classes ou instâncias de uma classe. Java atende todos os requisitos necessários para uma linguagem ser considerada orientada à objetos que resumidamente são: oferecer mecanismos de abstração, encapsulamento e hereditariedade. Como os objetos encapsulam dados e funções relacionados em unidades coesas, é fácil localizar dependências de dados, isolar efeitos de alterações e realizar outras atividades de manutenção, e talvez o mais importante, as linguagens orientadas a objetos facilitam a reutilização.

[2]Esta Seção é inteiramente baseada na documentação do Java. Essa documentação pode ser obtida gratuitamente em `http://java.sun.com`

[3]`http://www.sun.com`

- **Independente de Plataforma**

  Java é uma linguagem independente de plataforma pois os programas Java são compilados para uma forma intermediária de código denominada *bytecodes* que utiliza instruções e tipos primitivos de tamanho fixo, ordenação *big-endian* e uma biblioteca de classes padronizada. Os *bytecodes* são como uma linguagem de máquina destinada a uma única plataforma, a máquina virtual Java (*JVM - Java Virtual Machine*), um interpretador de *bytecodes*. Pode-se implementar uma JVM para qualquer plataforma, assim temos que um mesmo programa Java pode ser executado em qualquer arquitetura que disponha de uma JVM.

- **Sem Ponteiros**

  Java não possui ponteiros, isto é, Java não permite a manipulação direta de endereços de memória e nem exige que os objetos criados sejam destruídos, livrando os programadores de uma tarefa complexa. Além disso, a JVM possui um mecanismo automático de gerenciamento de memória conhecido como *garbage collector*, que libera a memória alocada para objetos não mais referenciados pelo programa.

- **Portabilidade**

  A característica de neutralidade da arquitetura Java é o grande motivo pelo qual os programas em Java são portáveis. Outro aspecto da portabilidade envolve a estrutura ou os tipos de dados inerentes da linguagem, como inteiro, string e ponto flutuante.

  O compilador Java foi escrito com o próprio Java, enquanto seu ambiente de tempo de execução foi escrito em ANSI C e tem uma interface de portabilidade bem definida e concisa.

- **Performance**

  A linguagem Java foi projetada para ser compacta, independente de plataforma e para utilização em rede, o que levou a decisão de ser interpretada por meio do esquema de *bytecodes*. Como uma linguagem interpretada a performance é razoável, não podendo ser comparada a velocidade de execução de código nativo. Para superar essa limitação a JVM dispõe de um compilador *just in time* (JIT) que compilam os *bytecodes* para código nativo durante a execução, otimizando assim a execução e com isso há uma melhora significativa na performance dos programas em Java.

- **Segurança**

Considerando a possibilidade de aplicações obtidas através de uma rede, a linguagem Java possui mecanismos de segurança que podem, no caso de *applets* (que serão explicadas mais detalhadamente na Seção 4.1.2), evitar qualquer operação no sistema de arquivos da máquina alvo, minimizando problemas de segurança. Tal mecanismo é flexível o suficiente para determinar se uma applet é considerada segura especificando nesta situação diferentes níveis de acesso ao sistema alvo.

- **Permite Multithreading**

  A linguagem Java oferece recursos para o desenvolvimento de aplicações capazes de executar múltiplas rotinas concorrentemente e dispõe de elementos para a sincronização destas várias rotinas. Cada um destes fluxos de execução é denominado de *thread*, um importante recurso para aplicações mais sofisticadas.

- **Robustez**

  Quanto mais robusto um aplicativo, mais confiável ele será. Isso é desejável tanto para os desenvolvedores de software quanto aos consumidores. A maioria das linguagens OO, como o C++ e Java, possuem tipos bastante fortes. Isso significa que a maior parte da verificação de tipos de dados é realizada em tempo de compilação e não em tempo de execução. Isso evita muitos erros e condições aleatórias nos aplicativos. A linguagem Java, ao contrário do C++, exige declarações explícitas de métodos, o que aumenta a confiabilidade dos aplicativos.

- **Simplicidade**

  Um dos principais objetivos do projeto do Java foi criar uma linguagem o mais próxima possível do C++, para garantir sua rápida aceitação no mundo do desenvolvimento orientado a objetos. Outro objetivo do seu projeto foi eliminar os recursos obscuros e danosos do C++, que fugiam à compreensão e aumentavam a confusão que poderia ocorrer durante as fases de desenvolvimento, implementação e manutenção do software. O Java é simples porque é pequeno. O interpretador básico do Java ocupa aproximadamente 40 kBytes de RAM, excluindo-se o suporte a multitarefas e as bibliotecas padrão, que ocupam outros 175 kBytes. Mesmo a memória combinada de todos esses elementos é insignificante, se comparada a outras linguagens e ambientes de programação.

4.1.2 Applets

Uma applet é um programa Java que é executado por um browser quando é carregada a página da Internet que o contem. Sendo assim, uma applet é um programa Java destinado a ser utilizado por browsers e será transportado pela Internet tal como documentos HTML, imagens(gif e jpeg) e outros conteúdos típicos da rede.

Em função deste uso, as applets são construídas para serem programas pequenos e, por questões de segurança, obedecem critérios rígidos para que sua execução seja possível pelos browsers.

Como as applets são programas executados em ambientes gráficos das diversas plataformas que utilizam a Internet, sua construção é bastante semelhante a construção de programas que utilizem componentes ou recursos gráficos disponíveis nas bibliotecas gráficas do Java.

A programação e a compilação de uma applet é uma tarefa idêntica a criação de programas em Java: utiliza-se um editor para escrever o código e posteriormente esse código é compilado nos *bytecodes* correspondentes.

4.1.2.1 Funcionamento das Applets

Embora a construção de applets seja semelhante a criação de programas gráficos em Java, a compreensão do seu ciclo de vida auxilia bastante no entendimento de sua estrutura. As applets são aplicações especiais que possuem um modo particular de funcionamento:

1. Instanciação (*create*)
2. Inicialização (*init*)
3. Início (*start*)
4. Execução e renderização (*paint* e outros métodos)
5. Parada (*stop*)
6. Finalização ou destruição (*destroy*)

Este modo de funcionamento define o ciclo de vida de uma applet e pode ser visto na Figura 4.1.

É importante ressaltar que as implementações de applets não necessitam de todos esse métodos. Uma implementação mínima de uma applet pode ser realizada utilizando-se apenas o métodos **init** ou o método **paint**.

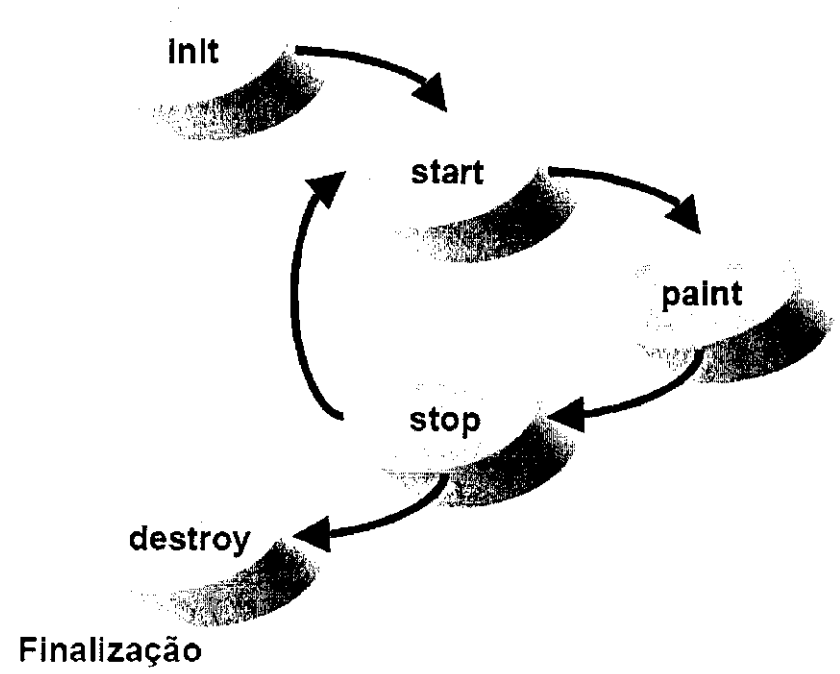


Figura 4.1: Ciclo de vida das applets

- O método ***init***

  O método init é invocado após a criação da applet e é utilizado pelos navegadores para iniciar a execução das applets. Esse método é executado apenas uma vez e é utilizado para a adição de componentes, recebimento de parâmetros de execução e outras atividades de preparação da applet.

  Na applet implementada neste sistema, o método init é responsável pela inicialização dos componentes que compõem a interface.

- O método ***start***

  O método start é acionado pelo browser logo após a execução do método init e continua sendo acionado sempre que a applet se torna visível.

  Na applet da interface, o método start é responsável por inicializar as threads que fazem a comunicação do servidor de controle do robô com a interface.

- O método ***paint***

  Este método é chamado toda vez que a applet necessita atualizar sua exibição. Isto ocorre quando a applet é exibida pela primeira vez, quando a janela do browser é movimentada ou redimensionada.

O método paint pode ser utilizado para a realização de operações de desenho na applet, ou seja, quando deseja-se tratar diretamente da renderização de conteúdo exibido pela applet, tal como no caso da construção de animações ou aplicações gráficas.

Para a applet implementada neste sistema, o método paint é responsável por desenhar o mapa e o robô no mapa, de acordo com as coordenadas recebidas do servidor de controle do robô. É responsável também por mostrar a foto capturada pelo robô na interface.

- O método ***stop***

  Este método é acionado toda vez que a applet deixa de ser visível. Isto significa que a applet, durante seu ciclo de vida, pode ter seus métodos start e stop acionado inúmeras vezes.

  Na interface implementada o método stop é responsável por fechar as conexões abertas para a comunicação entre a interface e o servidor de controle do robô.

- O método ***destroy***

  O método destroy é acionado logo após o método stop e sua função é a de descarregar todos os recursos utilizados durante a execução da applet.

4.1.2.2 Restrições das Applets

As applets foram planejadas para serem carregadas de um site remoto e então executadas localmente. Desta forma, para evitar que applets estranhas prejudiquem o sistema local onde estão sendo executadas, isto é, alterem, apaguem ou acessem informações armazenadas neste sistema, foram impostas algumas restrições ao funcionamento das applets.

As applets são executadas pelo browser e monitoradas por um gerenciador de segurança(denominado de *applet security manager*) que lança uma interrupção caso a applet viole alguma das regras de segurança impostas.

As applets podem realizar as seguintes tarefas: podem exibir imagens, podem executar sons, podem processar o acionamento do teclado e do mouse e podem se comunicar com o *host* de onde foram carregadas.

Mesmo com todas as restrições impostas pelo gerenciador de segurança das applets, este modelo admite funcionalidade suficiente para que uma applet exiba conteúdos diversos, realize operações de consistência de dados e envie ou receba informações necessárias para a realização de uma consulta a um banco de dados, um pedido de compra ou cadastro de informações.

Por outro lado as applets estão sujeitas as seguintes restrições: não podem executar programas localmente instalados, não podem se comunicar com outros *hosts*, exceto o de origem, não podem ler ou escrever no sistema de arquivos local e não podem obter informações do sistema onde operam, exceto sobre a máquina virtual Java sobre a qual operam.

Quando é necessário que uma applet realize alguma operação restrita, é possível "assinar" o código da applet, isto é, anexar um certificado de segurança que ateste a origem da applet e que permita a sua execução em um nível de restrição semelhante ao de programas comuns.

4.2 Funcionamento da Interface

Assim que a página que contém a applet da interface é aberta, o módulo de comunicação cria uma conexão de socket com o sistema de controle do robô. Esse canal de comunicação entre a interface e o sistema de controle do robô ficará aberto enquanto a applet estiver funcionando.

Após a comunicação ser estabelecida, o sistema de controle do robô envia para a interface o mapa topológico, em forma de grafo, que está sendo usado no planejamento de trajetórias do robô.

O próximo passo é receber do sistema de controle do robô as coordenadas (x, y, θ) do módulo de localização, as imagens que o robô está capturando e atualizar tanto a posição do robô no mapa da interface, quanto a imagem capturada pela câmera sem fio do robô Pioneer 1. As coordenadas da localização do robô são atualizadas a todo momento. As imagens da câmera são atualizadas em intervalos de 5-10 segundos. É importante ressaltar que as coordenadas enviadas pelo módulo de localização para a interface Web precisam ser convertidas para mostrar com a maior precisão possível a localização do robô no mapa da interface.

Quando um usuário deseja enviar alguma requisição de tarefa para o robô, basta ele clicar na posição do mapa para onde ele deseja enviar o robô e depois clicar no botão "Enviar Requisição".

A interface captura a posição do clique do usuário e verifica a distância entre a posição (x, y) do clique do mouse e todos os nós do grafo que correspondem ao mapa topológico do sistema central de controle do robô.

Ao encontrar o ponto (nó do grafo) mais próximo da posição do clique do usuário, a interface envia uma mensagem para o sistema de controle do robô com o nó do grafo determinado e o sistema se encarrega de enviar o robô para o ponto "mais próximo possível" dessa coordenada. Isto ocorre porque quando o robô se desloca ocorrem erros de odometria entre outras coisas que impedem que o robô chegue

exatamente na posição desejada.

O envio de requisições de tarefas para o robô pode ser feita a qualquer momento. Mesmo enquanto o robô estiver executando uma determinada tarefa os usuários podem continuar enviando requisições de tarefa para o robô.

Neste projeto foi utilizada uma estrutura de dados do tipo fila para ordenar todas as requisições recebidas pelo sistema de controle do robô, ou seja, as requisições são realizadas de acordo com a ordem de chegada, as primeiras requisições enviadas são as primeiras a serem executadas.

Por enquanto, a única tarefa aceita pelo sistema é do tipo "vá para tal ponto". O robô apenas se locomove do ponto onde se encontra para o ponto desejado pelo usuário. Pretende-se que no futuro, além do robô se locomover entre dois pontos ele possa realizar outros tipos de tarefas. Uma possibilidade seria a integração deste sistema com o sistema desenvolvido em (Quiles 2004). Tal sistema faz o reconhecimento de objetos e cores utilizando uma rede neural artificial. Assim, o robô poderia realizar tarefas do tipo "vá ate tal ponto e pegue o cubo azul".

A Figura 4.2 mostra um exemplo da interface de controle do robô com o grafo correspondente ao mapa topológico usado no sistema de controle do robô.

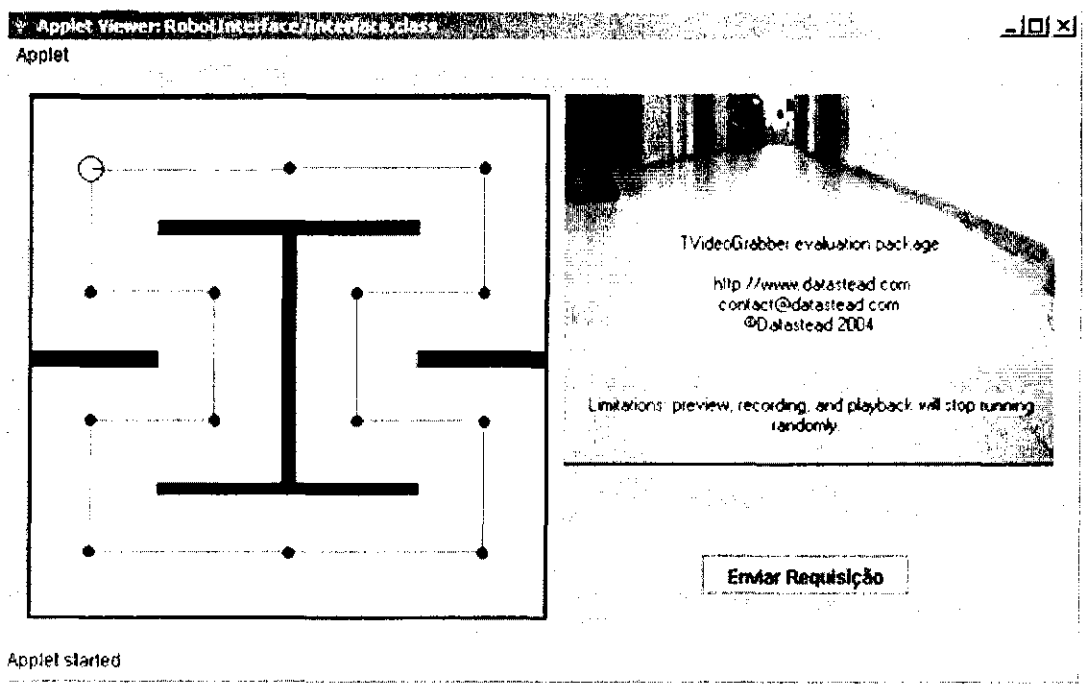


Figura 4.2: Visualização da interface de controle do robô com o grafo correspondente ao mapa topológico do sistema de controle do robô

4.3 Considerações Finais

Neste capítulo foram explicados, de forma detalhada, todos os módulos que compõem a interface de controle do robô via Web, que é responsável pela comunicação entre o usuário e o sistema. Foram explicados também o funcionamento da interface Web e as conexões necessárias para troca de informações entre a interface e o sistema central de controle do robô. Por fim, foi apresentado um histórico da linguagem Java e algumas características das Applets.

A integração de todos os módulos necessários para o funcionamento do sistema remoto para controle de robôs será explicada no capítulo seguinte. Os módulos que compõem o sistema já foram explicados detalhadamente no capítulo anterior. O módulo de navegação, que engloba o mapeamento, localização e planejamento de trajetórias foi explicado, de forma detalhada, no Capítulo 3 e o módulo de interface Web foi apresentado detalhadamente neste capítulo.

Capítulo

5

Descrição do Sistema

Neste capítulo será apresentada a arquitetura para integração dos módulos do sistema remoto para controle de robôs móveis via Web, bem como suas funcionalidades.

Os módulos que constituem o sistema foram explicados separadamente nos capítulos anteriores. O módulo responsável pela navegação do robô no ambiente foi apresentado no Capítulo 3 e o módulo de interface Web foi apresentado no Capítulo 4.

A arquitetura de teleoperação do sistema, o funcionamento do sistema e o hardware robótico utilizado neste projeto são descritos a seguir.

5.1 Arquitetura do Sistema de Teleoperação

A arquitetura de teleoperação deste sistema baseia-se no modelo de redes cliente-servidor. A Figura 5.1 mostra a arquitetura do sistema de teleoperação proposto neste trabalho.

Essa arquitetura de teleoperação funciona da seguinte forma: o módulo cliente da teleoperação se comunica com o módulo servidor por meio da Internet.

O módulo cliente é composto pela interface Web que recebe comandos do cliente e os envia para o módulo servidor. A interface Web também é responsável por receber informações do módulo servidor e fazer a atualização da posição do robô no mapa mostrado na interface.

O módulo servidor é composto pelo sistema de navegação e pelo sistema de captura de imagens do robô. Este módulo interage com o robô enviando comandos e recebendo as imagens capturadas ao longo do percurso e as leituras do sensor laser existente no robô.

O sistema central de controle do robô, responsável pela navegação do robô, foi

apresentado em detalhes no Capítulo 3. A interface Web, responsável por enviar os comandos do usuário e atualizar a posição do robô no mapa do ambiente, foi apresentada detalhadamente no Capítulo 4.

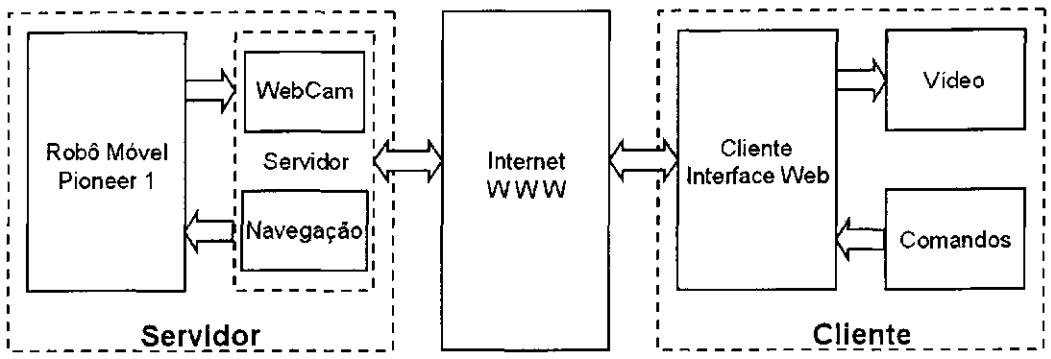


Figura 5.1: Arquitetura cliente-servidor do sistema de teleoperação

5.2 Funcionamento do Sistema

O sistema de controle do robô, explicado em detalhes no Capítulo 3, pode ser subdividido em dois outros sistemas: Sistema Central de Controle (SCC) e interface Web para o monitoramento do robô.

O SCC é composto por três módulos: módulo de mapeamento, módulo de localização e módulo de controle de trajetórias. O módulo de mapeamento tem a finalidade de modelar o ambiente, de forma que o robô possa se locomover e desviar dos obstáculos. A localização tem a finalidade de estimar o posicionamento do robô no ambiente. O módulo de controle de trajetórias consiste em fazer o deslocamento do robô de um ponto de partida a um ponto de chegada da melhor maneira possível.

A interface Web é responsável por enviar novas tarefas para o SCC, pelo monitoramento do robô em um mapa do ambiente e por mostrar imagens capturadas pelo robô durante a realização da tarefa.

A Figura 5.2 mostra a arquitetura do sistema proposto neste trabalho. O sistema funciona da seguinte maneira:

- O usuário envia uma requisição de tarefa para o robô por meio da interface Web. Essa requisição é enviada pela Internet, via sockets, para o SCC do robô.
- O SCC é responsável pela navegação do robô. Este sistema recebe a requisição enviada pelo usuário, por meio da interface, e decide qual a melhor ação, ou conjunto de ações, que o robô deverá seguir para executar a tarefa. A escolha da melhor ação a ser tomada pelo robô é feita pelo módulo de planejamento de trajetórias. A ação a ser tomada é enviada para o robô, via radio modem.

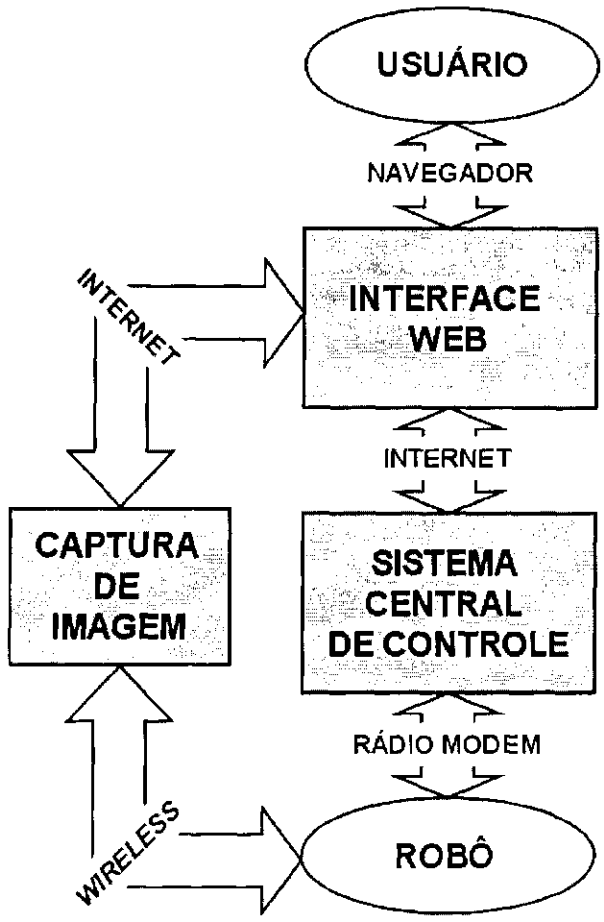


Figura 5.2: Arquitetura do Sistema de Controle do Robô

- O robô executa as ações determinadas e envia para o SCC as novas informações sobre sua nova posição no ambiente. Essas informações são obtidas pelo robô por meio do sensor laser. O sistema central de controle envia as novas informações sobre a posição do robô para a interface Web. A posição do robô no mapa da interface é atualizada. Durante todo o funcionamento do sistema, a câmera sem fio acoplada ao robô vai capturando imagens do ambiente. Essas imagens são atualizadas na interface pelo módulo de captura de imagem.

- O módulo de interface recebe as requisições enviadas pelos usuários. O usuário envia apenas a posição, no mapa do ambiente, para onde o robô deve se locomover. Este módulo também é responsável por mostrar ao usuário a posição do robô durante a execução da tarefa e as imagens capturadas durante o percurso percorrido pelo robô.

Para navegar pelo ambiente o SCC necessita do mapa métrico e topológico desse

ambiente. O mapeamento do ambiente deve ser realizado antes de se iniciar a utilização do sistema de teleoperação. A obtenção dos mapas do ambiente é realizada da seguinte maneira:

- Primeiramente, o robô realiza a exploração do ambiente e constrói o mapa métrico do ambiente. Neste trabalho, a exploração do ambiente é feita de forma manual, mas o mapa é construído autonomamente. Este mapa contém informações sobre o posicionamento dos objetos e das paredes no ambiente.

- Após o mapa métrico ser construído, é gerado um mapa topológico baseado neste mapa métrico. Este mapa topológico é obtido manualmente e é utilizado pelo planejamento de trajetórias. A rota a ser seguida pelo robô é determinada pelo módulo de planejamento de trajetórias, descrito detalhadamente em 3.3.

5.3 *Hardware Robótico*

Nesse projeto foi utilizado um robô Pioneer 1, da ActivMedia Robotics[1] mostrado na Figura 5.3. Este tipo de robô é muito utilizado para ensino e para pesquisa. As características desse robô são descritas a seguir:

- **Motores**

  O robô é movido por dois motores com movimentação por diferencial. Ele pode se mover a uma velocidade mínima, aproximada, de 60cm por segundo. Suas capacidades de odometria variam bastante dependendo da superfície por onde ele está se locomovendo. O erro de distância é de aproximadamente 1cm por metro; o erro de rotação é de aproximadamente 8^o por rotação.

- **Sensores**

  Este robô é equipado com 7 sonares, um em cada lado do robô e 5 na frente.

- ***Gripper***

  O robô é equipado também com um *gripper* que se movimenta para cima e para baixo e com 1 grau de liberdade. O *gripper* opera entre dois estados: aberto/abaixo e fechado/acima. Os sensores infra-vermelhos existentes na frente e atrás de cada pá podem sentir os objetos entre pás e qual pressão que deve ser exercida para pegar os objetos.

[1] `http://www.activmedia.com`

- **Radio-links**

  O software de controle do robô Pioneer é executado off-board e é passado para o robô de forma remota, por meio de um link de rádio de 9.600 bps. O software remoto recebe as informações do robô e dos sonares a uma freqüência de 10 Hz. A taxa máxima de envio de comandos para mudar a direção do robô é de apenas 2 Hz (ou menos) por limitações no controlador on-board do robô.

- **Medições dos Sonares**

  Os sonares do Pioneer exibem as características típicas de qualquer sonar; eles raramente medem a distância do objeto mais próximo. Os sonares sempre retornam valores que são imprecisos e, não necessariamente denotam a distância correta. Os sonares medem o tempo decorrido entre a emissão e a recepção de uma onda sonora. Para objetos lisos, a chance de receber um eco depende do ângulo entre o sonar e o objeto que refletirá a onda sonora. Quando as ondas sonoras acertam a parede frontalmente, muito provavelmente elas retornam na direção dos sensores; entretanto, quando as ondas sonoras acertam a parede em um ângulo desfavorável, provavelmente elas são refletidas para longe dos sensores e não podem ser detectadas.

- ***Proximity Laser Scanner***

  Devido aos sonares não apresentarem medições satisfatórias para as tarefas de mapeamento e localização, foi necessário encontrar uma solução para esse problema. A solução encontrada foi a aquisição de um sensor laser denominado PLS (Proximity Laser Scanner). O sensor é capaz de rastrear o ambiente com um ângulo de visão de 180^o, resolução angular de $0{,}5^o$, distância máxima de 50m e erro de 131mm para distâncias $\leq$ 4m.

- **Câmera sem fio**

  Para este projeto foi acoplada ao robô uma câmera sem fio para capturar imagens durante toda a locomoção do robô. Esta câmera apresenta resolução 330 linhas, lentes de 6mm, ângulo de visão 56^o e iluminação de 2 lux. A câmera que foi adquirida para o projeto é uma câmera CMOS NTSC sem fio.

- **Radio Modem**

  Para poder controlar o robô de forma remota é necessário que as informações obtidas pelo laser sejam enviadas, via rádio modem, para o computador que controla o robô. Neste projeto está sendo utilizado um transceiver RF do tipo

BiM2[2]. O BiM2 é um rádio transceiver usado para aplicações que necessitam de uma transferência bidirecional, em alta velocidade, e possui uma alcance de aproximadamente 200m. O rádio trabalha a uma freqüência de 433.92MHz, velocidade de transferência de até 160 kbps e voltagem entre 3 e 5V.

- **Simulador Saphira**

O robô dispõe de um ambiente para simulação denominado Saphira que permite que códigos em C sejam anexados à biblioteca deste ambiente. Este ambiente apresenta um comportamente similar ao robô e assim auxiliam os testes dos algoritmos implementados para o robô real.

O Saphira é uma arquitetura para o desenvolvimento de aplicações robóticas, desenvolvido e mantido pelo *Artificial Intelligence Center* do *Stanford Research Institute*. Ele possui uma biblioteca de rotinas que permite a construção de programas em linguagem C/C++ para o controle de robôs móveis como o Pioneer 1, fornecendo uma abstração de alto nível do hardware robótico provendo informações sensoriais, como leituras de sonares, *encoders* e recebendo comandos como "vire 30^o".

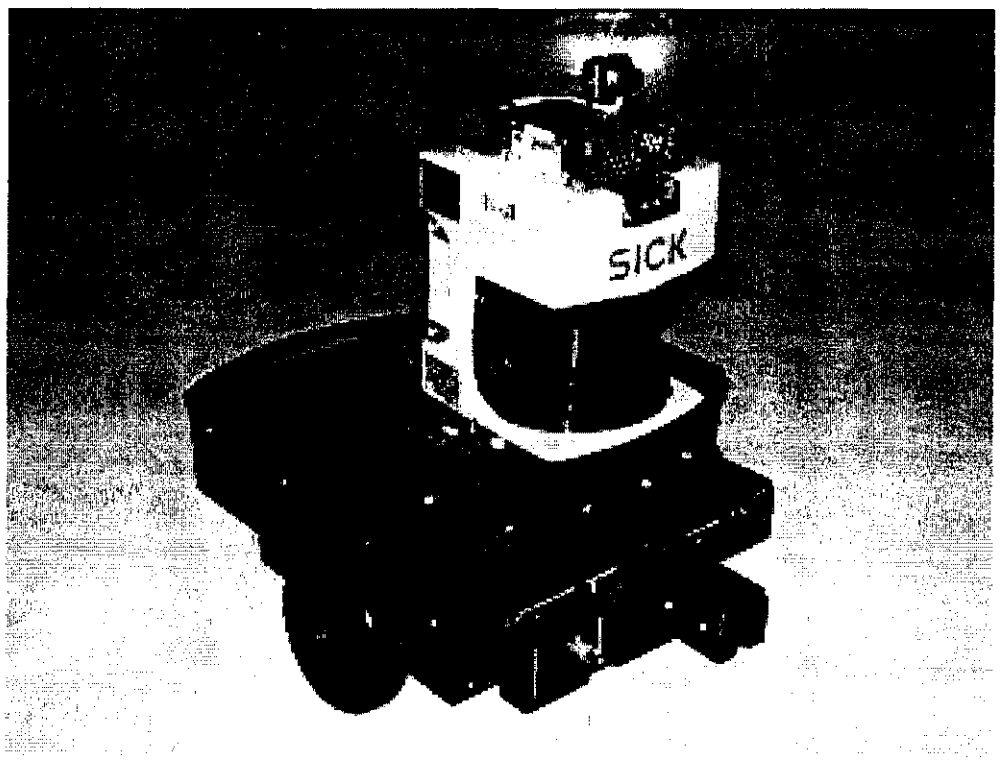


Figura 5.3: Robô Pionner 1 equipado com o sensor laser SICK PLS

[2]http://www.radiometrix.co.uk/products/bim2.htm

5.4 Considerações Finais

Neste capítulo foram apresentadas a arquitetura do sistema proposto neste trabalho e as conexões necessárias entre os módulos que compõem o sistema. As funcionalidades do sistema foram detalhadas e explicadas para um melhor entendimento do projeto.

No próximo capítulo serão mostrados os testes e experimentos realizados para validar o sistema remoto para controle de robôs móveis via Web

Capítulo

6

Experimentos Realizados

Neste capítulo são apresentados alguns testes realizados para a validação do sistema proposto neste trabalho. Os testes foram realizados em 3 etapas distintas. Na primeira etapa foram realizados testes para verificar o funcionamento do sistema de navegação. Na segunda etapa, foram realizados testes para validar o funcionamento do sistema de controle do robô, juntamente com a interface web. Esses testes foram realizados usando o simulador do robô. Finalmente, na terceira etapa, foram realizados os testes de integração de todos os módulos do sistema e verificação do seu funcionamento utilizando o robô Pioneer 1.

Todos os experimentos realizados são apresentados e discutidos a seguir.

6.1 Testes do Sistema de Navegação

O sistema de navegação utilizado neste projeto foi desenvolvido e testado em (Bianchi 2003). Apesar disso, alguns problemas deveriam ser tratados para que esse sistema de navegação pudesse ser utilizado neste projeto.

O maior problema encontrado foi na utilização da técnica de planejamento de trajetórias. O planejamento de trajetórias era feito de forma manual. Para solucionar esse problema foi desenvolvido um módulo de plenajamento de trajetórias de alto nível. Esse módulo de planejamento de trajetórias é explicado em detalhes na Seção 3.3.2.

Com isso, o problema do planejamento de trajetórias foi contornado e para testar o sistema de navegação foi criado um sistema que simulava um cliente e enviava requisições randômicas para o sistema central de controle do robô.

O sistema central de controle ficou operando por algumas horas, sem interrupção, recebendo várias requisições diferentes e executou todas as requisições recebidas. Durante este tempo o sistema de navegação executou 10.000 requisições de tarefas, sempre conseguindo realizar as tarefas enviadas. Dessa forma, foram validados os

módulos de mapeamento, localização e planejamento de trajetórias que constituem o sistema de navegação do robô. Não ocorreram problemas durante os teste e o robô conseguiu navegar pelo ambiente.

6.2 Testes Simulados

Após a validação do sistema de navegação, era necessária a validação da comunicação entre o sistema central de controle do robô e a interface web. Este teste foi realizado para verificar se o robô conseguiria realizar as tarefas com os dados enviados pelo usuário por meio da interface.

Para analisar o desempenho do robô e a comunicação entre o robô e a interface web desenvolvida, foram escolhidos dois ambientes. O comportamente do robô nesses ambientes foi simulado utilizando-se o simulador Saphira, explicado na Seção 5.3.

O primeiro ambiente escolhido para o teste simulado pode ser visto na Figura 6.1. Este ambiente foi escolhido pois os obstáculos presentes nele formam a figura de um "H". Este formato de ambiente é considerado um dos mais difíceis para a navegação de robôs móveis autônomos, pois contém dois "U"s, que dependendo da técnica que está sendo utilizada, pode acarretar em pontos de mínimos locais (Koren and Borenstein 1991)(Faria and Romero 2004).

Um outro ambiente foi desenvolvido para simular um escritório. Este ambiente é importante para avaliar a capacidade do robô em se locomover em ambientes semelhantes aos presentes no mundo real. A Figura 6.2 mostra este ambiente simulado.

Nos dois casos os testes foram realizados da seguinte maneira: a interface web foi aberta em quatro computadores diferentes para simular quatro usuários diferentes. Cada usuário podia enviar uma requisição para o sistema central de controle do robô, por meio da interface web.

Nas Figuras 6.1 e 6.2, são mostradas as quatro requisições enviadas pelos usuários, isto é, os pontos marcados no mapa do ambiente representam as várias requisições apresentadas ao sistema. Esses pontos representam os diferentes locais para onde o robô deverá se dirigir.

As requisições são recebidas pelo sistema central de controle do robô e colocadas em uma fila. A primeira requisição recebida é a primeira a ser executada e assim sucessivamente.

Na Figura 6.3 são mostrados os caminhos realizados pelo robô para executar a requisição enviada pelo usuário 1 na interface web e no sistema central de controle do robô, para os dois casos de teste. O sistema sempre escolhe o menor caminho para executar as requisições enviadas pelos usuários. Esse menor caminho é escolhido com base no mapa topológico gerado sobre o mapa métrico do sistema central de

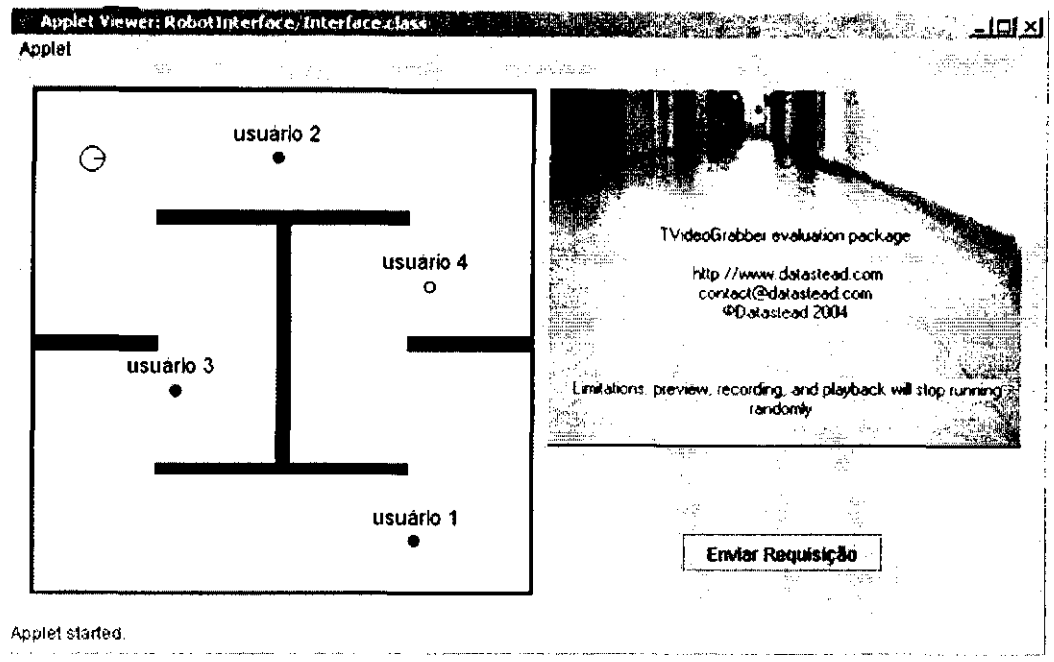


Figura 6.1: Exemplo de uma mapa em forma de "II" com quatro requisições de usuários diferentes

controle do robô.

Após terminar a execução da requisição do usuário 1, o sistema verifica se existem mais requisições na fila e, caso exista, envia os comandos necessários para o robô executar a próxima requisição da fila.

Na Figura 6.4, são mostrados os caminhos percorridos pelo robô na interface web e no sistema central de controle do robô, tanto para o ambiente em forma de "H", quanto para o ambiente simulado de um escritório.

Durante todo o tempo utilizado pelo robô para executar as requisições de tarefas enviadas pelos usuários, a interface web faz a atualização da posição do robô no mapa do ambiente e a atualização das imagens recebidas da câmera acoplada ao robô. Neste caso, como o teste foi realizado no simulador a imagem é apenas para ilustrar a interface. Os caminhos percorridos pelo robô para executar a tarefa do usuário 3 são mostrados na Figura 6.5.

É importante ressaltar que enquanto o robô está executando uma determinada tarefa, os usuários podem enviar outras requisições de tarefas para o robô. O sistema recebe as requisições de tarefa e as coloca na fila de requisições. Enquanto a fila de requisições não estiver vazia o sistema continua controlando o robô para realizar todas as requisições pendentes. Na Figura 6.6 são mostrados os caminhos percorridos pelo robô, relativo a requisição apresentada pelo usuário 4, para os dois casos de teste, tanto na interface, quanto no sistema central de controle do robô.

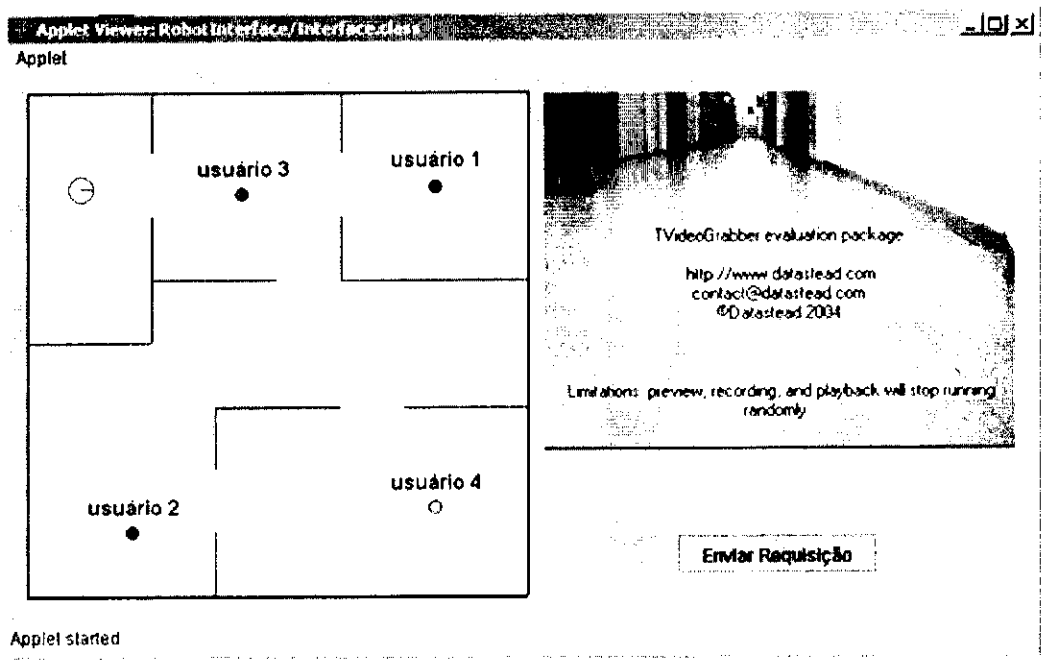


Figura 6.2: Exemplo de um mapa fictício de um escritório com requisições de quatro usuários diferentes

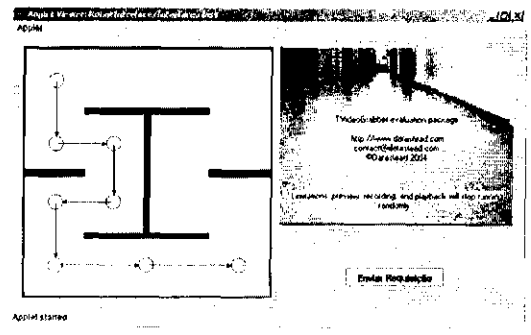
Foram realizados vários testes nos mesmos moldes dos descritos nesta seção e o sistema não apresentou nenhum problema. Tanto a interface quanto o sistema central de controle do robô não apresentaram nenhum tipo de erro, isto é, se comunicaram perfeitamente.

O próximo passo é testar o funcionamento do sistema com o robô real. Os resultados obtidos nos testes com o robô real são descritos na próxima seção.

6.3 Testes com o Robô

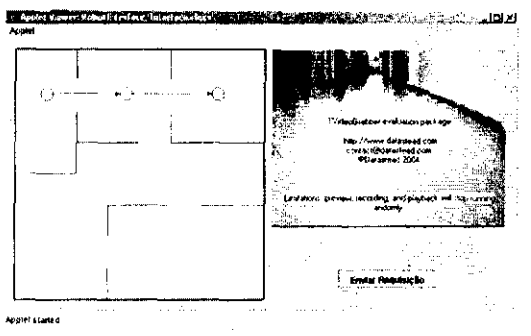
Com os testes realizados anteriormente, o sistema de navegação autônoma e a comunicação entre a interface e o sistema central de controle foram validados. A última etapa dos testes trata da integração do sistema central de controle do robô, a interface web e o robô Pioneer 1. Este teste trata a integração de todos os módulos do sistema e verificação do seu funcionamento em tempo real.

Devido a problemas na construção e implementação dos rádios modem, problemas esses que são detalhados na seção 6.4, foi necessário adotar uma solução paliativa para realização dos testes. A solução adotada faz uso de um adaptador de rede sem fio USB acoplado a um notebook. Esse notebook tem a função de ler as informações do sensor laser e enviá-las, por meio do adaptador de rede sem fio, para o sistema de controle central do robô. Com isso, pode-se simular o funcionamento dos radios



(a) Interface web com um mapa simulado em forma de "H"

(b) Sistema Central de Controle do Robô



(c) Interface web com um mapa simulado de um escritório

(d) Sistema Central de Controle do Robô

Figura 6.3: Caminho realizado pelo robô para executar a requisição do usuário 1 nos dois casos de teste

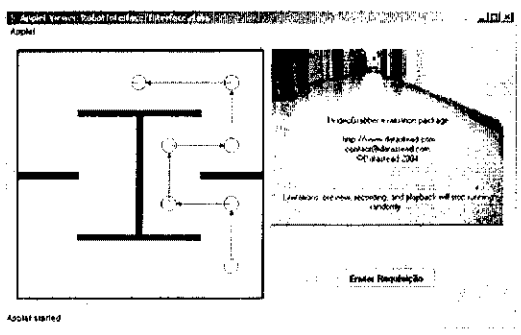
modem a fim de testar o sistema.

Para iniciar o teste é necessário realizar o mapeamento do ambiente com o qual o robô irá interagir. Neste caso, foi mapeado apenas uma parte do corredor do prédio de laboratórios do ICMC-USP. O mapeamento do corredor não pode ser completo devido à problemas no hardware de comunicação. Tanto a câmera sem fio, quanto o rádio modem, responsável por controlar o robô, paravam de responder a uma determinada distância do servidor.

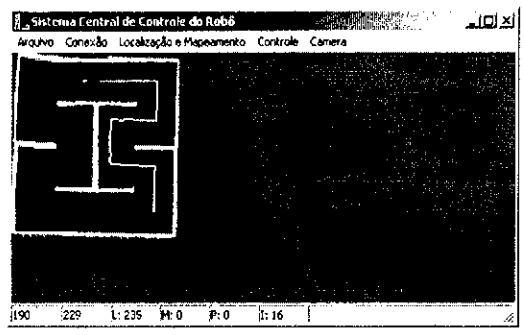
A planta do prédio de laboratórios do ICMC pode ser visto na Figura 6.7. A parte marcada na figura corresponde a área mapeada pelo robô para este teste. O mapa métrico obtido após o mapeamento do ambiente pelo sistema central de controle do robô é mostrado na Figura 6.8.

Após a obtenção do mapa métrico pelo sistema é necessário a construção do mapa topológico. O mapa topológico é construído manualmente sobre o mapa métrico. A Figura 6.9 mostra o mapa topológico gerado para esse caso de teste.

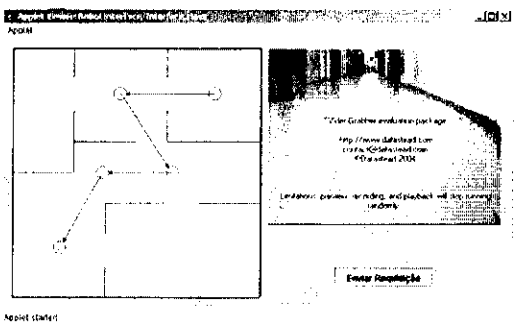
O próximo passo é criar um mapa correspondente ao ambiente real com o qual o robô vai interagir e colocá-lo na interface. Este mapa será utilizado para mostrar



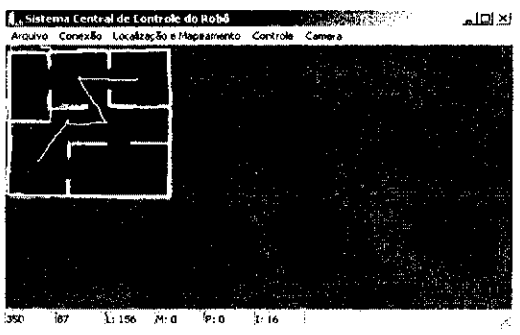
(a) Interface web com um mapa simulado em forma de "H"



(b) Sistema Central de Controle do Robô



(c) Interface web com um mapa simulado de um escritório



(d) Sistema Central de Controle do Robô

Figura 6.4: Caminho realizado pelo robô para executar a requisição do usuário 2 nos dois casos de teste

ao usuário, por meio da interface, o posicionamento correspondente do robô no ambiente e as imagens que o robô vai capturando ao longo do percurso realizado para a realização da tarefa. A interface com o mapa do ambiente real que o robô está interagindo pode ser visto na Figura 6.10.

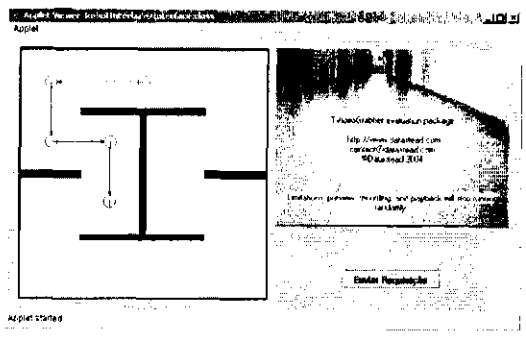
Terminados todos esses passos, o sistema está pronto para ser controlado, por qualquer pessoa, por meio da interface web.

Várias requisições foram enviadas para o robô e em todos os casos o sistema conseguiu realizar as requisições dos usuários de forma correta.

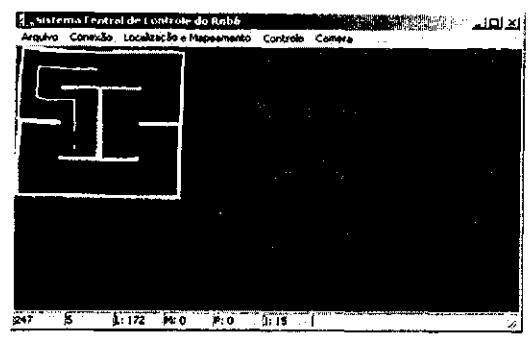
Apesar do robô ter apresentado um bom desempenho nas tarefas que lhe foram propostas, algumas limitações foram encontradas no sistema e são citadas a seguir.

6.4 Limitações do Sistema

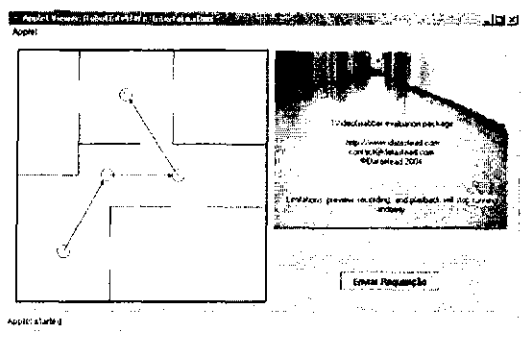
O sistema tal como está possui um alto nível de autonomia, mas alguns melhoramentos, que precisam ainda ser realizados, foram detectados a medida que o sistema foi sendo desenvolvido e são apontados abaixo.



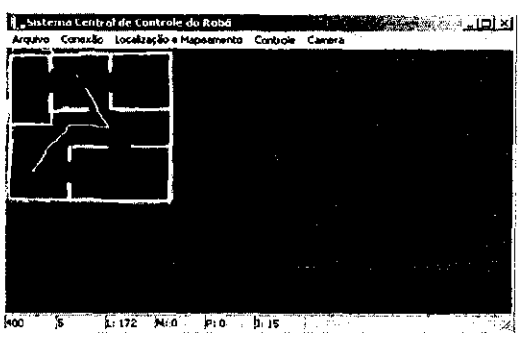
(a) Interface web com um mapa simulado em forma de "H"



(b) Sistema Central de Controle do Robô



(c) Interface web com um mapa simulado de um escritório



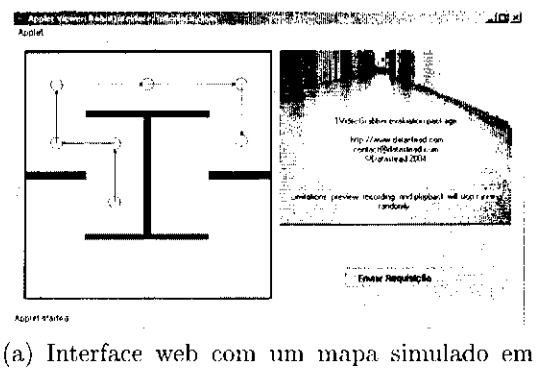
(d) Sistema Central de Controle do Robô

Figura 6.5: Caminho realizado pelo robô para executar a requisição do usuário 3 nos dois casos de teste

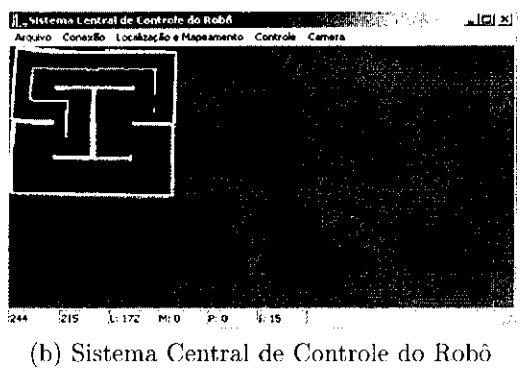
- **Problemas com rádio modem.** A proposta inicial era a de contar com dois rádios modem do tipo BiM2, um instalado no robô, que enviaria as informações do laser para o servidor central de controle, e outro instalado no servidor central de controle, responsável pela transmissão de dados entre o servidor de controle e o robô.

  No entanto, foram encontradas inúmeras dificuldades para colocar os dois rádios em operação. Uma delas foi a falta de conhecimento em hardware que fez com que esse tópico do trabalho dependesse da ajuda de terceiros. Uma outra dificuldade encontrada foi na inicialização da comunicação entre os dois rádios modem. Para inicializar a transmissão de dados é necessária a sincronização entre os rádios modem. Muitos problemas ocorriam nessa sincronização e os dados eram transmitidos incorretamente.

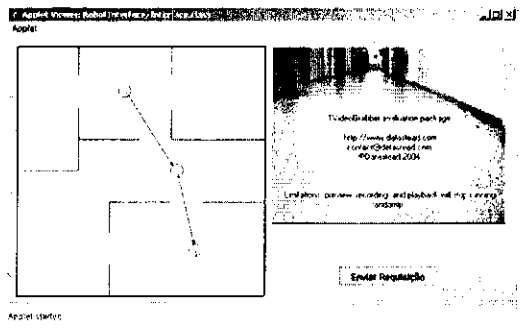
  Observou-se que seria necessário a utilização de um microcontrolador para fazer o envio dos parâmetros necessários para configurar o laser e posteriormente receber os dados da porta serial do laser e enviar para o computador.



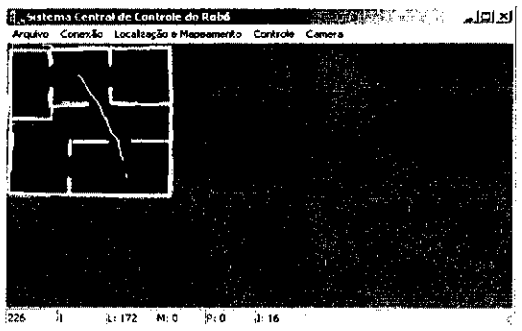
(a) Interface web com um mapa simulado em forma de "H"



(b) Sistema Central de Controle do Robô



(c) Interface web com um mapa simulado de um escritório



(d) Sistema Central de Controle do Robô

Figura 6.6: Caminho realizado pelo robô para executar a requisição do usuário 4 nos dois casos de teste

Devido aos erros ocorridos, uma solução paliativa foi desenvolvida. Optou-se por utilizar um adaptador wireless USB. Com este adaptador, a comunicação sem fio está sendo simulada entre o laser acoplado ao robô e o computador. Esta função seria exercida pelos rádios modem.

A espera pela construção dos rádios modem e os erros ocorridos na tentativa de uso desses rádios ocasionaram um atraso no projeto.

- Durante o desenvolvimento do projeto foi verificado que a carga da bateria tem autonomia de apenas para duas horas de funcionamento. Isto certamente inviabiliza que o robô fique disponível na Internet por períodos longos. Uma solução é a construção de uma doca de recarregamento para o robô.

- A exploração do ambiente para a construção do mapa métrico foi realizada de forma manual, isto é, de um trecho para o outro o robô foi movimentado por meio de comandos expressos do operador do robô. O operador guia o robô fazendo a exploração do ambiente. Durante essa exploração o módulo de mapeamento vai criando o mapa do ambiente. O ideal seria que o robô fizesse

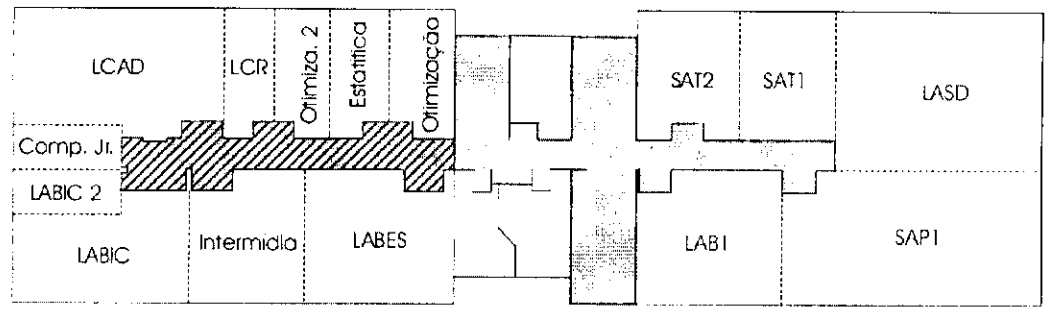


Figura 6.7: Planta do prédio de laboratórios do ICMC

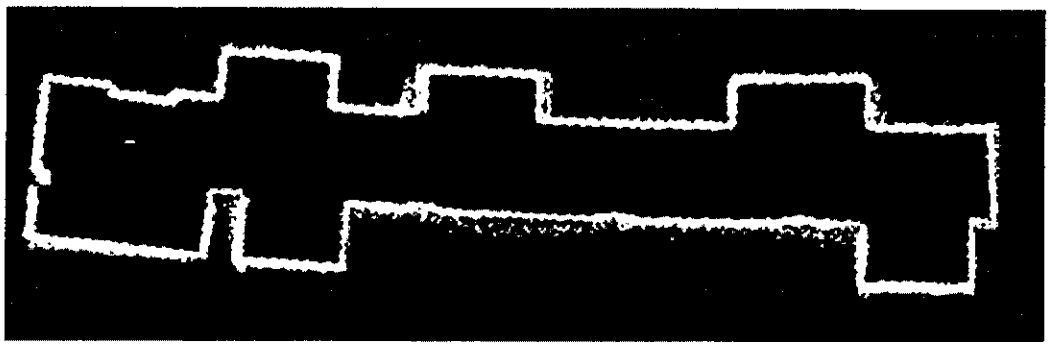


Figura 6.8: Mapa métrico obtido do mapeamento do corredor do prédio de laboratórios do ICMC

a exploração de forma automática, uma vez que existem métodos para realizar esta tarefa. Isto daria maior autonomia para o sistema de controle do robô.

- A geração do mapa topológico, a partir do mapa métrico, foi feita de forma manual. Um grafo foi construído com base no mapa métrico, obtido de forma automática pelo módulo de mapeamento descrito na Seção 3.2. Para a construção do grafo foram escolhidos pontos considerados importantes no ambiente que o robô está interagindo. O ideal seria que o robô construisse o mapa

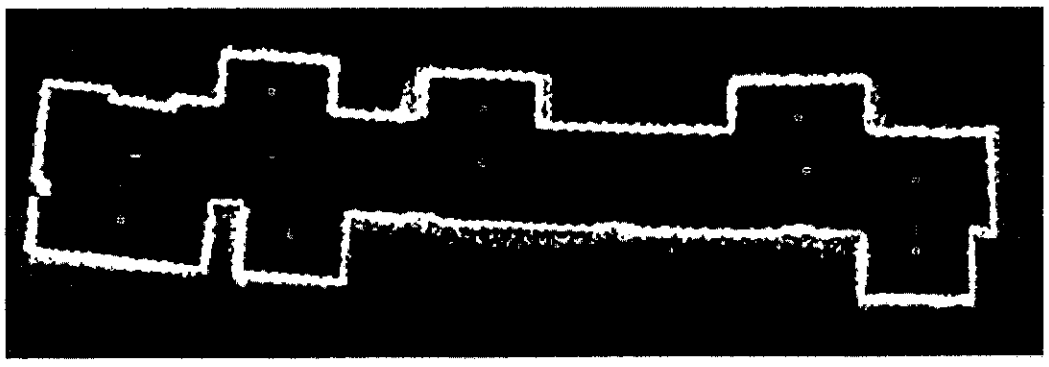


Figura 6.9: Mapa topológico criado manualmente sobre o mapa métrico

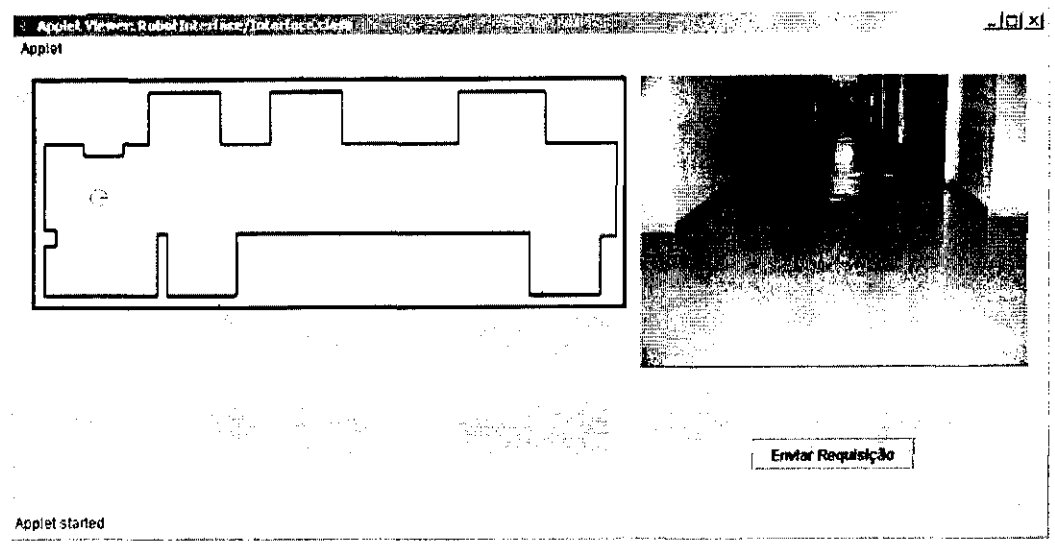


Figura 6.10: Interface web com o mapa correspondente ao corredor do prédio de laboratórios do ICMC

topológico de forma automática, uma vez que existem métodos para isto. A construção automática do mapa topológico daria uma maior flexibilidade para o sistema em qualquer ambiente.

Todas estas limitações foram levantadas e são apontadas como propostas para trabalhos futuros, apresentados em mais detalhes no Capítulo 7.

Uma descrição sucinta do sistema proposto nesta dissertação também pode ser encontrada em (Barbosa et al. 2005).

6.5 Considerações Finais

Neste capítulo todos os experimentos realizados para validar o sistema remoto para controle de robôs móveis via web foram apresentados. Os testes foram realizados em três etapas. Primeiramente, foram feitos testes para validar as modificações realizadas no sistema de navegação. Na segunda etapa, foram realizados testes de integração da interface web com o sistema central de controle do robô. Estes testes foram realizados em um ambiente de simulação do robô real. A terceira etapa de testes foi realizada fazendo a integração de todos os módulos anteriormente testados. A comunicação entre todos os módulos de controle e o robô foi realizada em tempo real. Os resultados dos testes foram satisfatórios e nenhum problema foi detectado. O desempenho do sistema remoto para controle de robôs móveis pela web foi muito bom e o sistema está pronto para ser usado em tempo real.

No próximo capítulo algumas conclusões são apresentadas e alguns trabalhos futuros são discutidos. A conclusão deste trabalho abre um grande leque de possi-

bilidades e uma gama enorme de trabalhos que poderão ser desenvolvidos pelo grupo de Computação Bioinspirada do ICMC-USP.

Capítulo

7

Conclusão e Trabalhos Futuros

Neste trabalho foi desenvolvido um sistema para controle remoto de robôs móveis via Web. Foi utilizado como base o sistema desenvolvido em (Bianchi 2003), que é constituído por um módulo de mapeamento, localização e planejamento de trajetórias. Porém, esse sistema teve que ser aperfeiçoado para que o robô pudesse ter a autonomia necessária para ser controlado pela Web. Esse aperfeiçoamento consistiu em criar um módulo de planejamento de trajetórias de alto nível. A construção desse módulo de planejamento de trajetórias foi necessária pois o planejamento de trajetórias era realizado, de forma manual, pelo próprio usuário. O usuário escolhia submetas nas quais o robô deveria passar até atingir a meta desejada. O módulo de planejamento de trajetórias de alto nível realiza o planejamento de trajetórias de forma automática. Dessa forma, o usuário fornece apenas o ponto meta ao qual o robô deve se locomover.

Além disto, uma interface Web foi desenvolvida para permitir que usuários possam interagir com o sistema de controle do robô através da Intenet. Essa interface foi desenvolvida em linguagem Java e permite ao usuário visualizar o caminho percorrido pelo robô desde um ponto de partida até o ponto de chegada requisitado pelo usuário. Para a visualização das imagens capturadas pelo robô foi acoplada ao robô uma câmera sem fio do tipo CMOS NTSC.

Foram realizados testes para verificar a comunicação entre o robô e a interface e testes para validar as funcionalidades da interface propriamente dita. Os testes foram feitos, inicialmente, em nível de simulação e numa segunda etapa, em um robô do tipo Pioneer 1.

Todos os testes realizados mostraram que o sistema funciona adequadamente para o controle do robô via Web.

A maior contribuição deste trabalho é possibilitar que un usuário possa controlar um robô por meio da Web. Isso abre muitas perspectivas de utilização desse projeto

no futuro. Outras contribuições e perspectivas para esse projeto são citadas a seguir:

- Mostrar o controle do robô, via Web, em palestras, aulas, seminários e com isso ajudar a mostrar na prática os algoritmos ensinados em cursos e aulas de robótica.

- Qualquer pessoa que tenha acesso a Internet poderá controlar o robô de maneira simples. Dessa forma, pretende-se popularizar a área de robótica, mostrando as várias técnicas utilizadas no controle do robô e permitindo que pessoas sem conhecimento em robótica possam realizar o controle de um robô.

- O sistema também pode ser utilizado para tarefas como entrega de documentos entre vários locais de um determinado ambiente, entrega de peças numa fábrica, entrega de medicamentos num hospital, sendo controlado via Internet, isto é, sem a necessidade do usuário saber técnicas de controle de robôs. O usuário precisará apenas saber como utilizar a interface de controle do robô via Web.

Apesar do sistema para controle do robô via Web ter aumentado as possibilidades de utilização do robô Pioneer 1 do LABIC, alguns melhoramentos podem e devem ser realizados para permitir que o robô tenha uma maior autonomia e também para deixá-lo funcionando de forma on-line na Web. Essas sugestões de trabalhos futuros são descritas a seguir.

7.1 Trabalhos Futuros

Com base no desenvolvimento deste trabalho, pode-se vislumbrar inúmeros projetos e melhoramentos a serem realizados. Espera-se que este trabalho seja aperfeiçoado cada vez mais e que o robô Pioneer 1 possa ser controlado, de forma online, por qualquer pessoa ao redor do mundo.

Abaixo algumas melhorias e alguns projetos futuros são apresentados.

1. **Laboratório Virtual**

   O sistema proposto neste trabalho será incorporado ao projeto de Aprendizagem Eletrônica (TIDIA-Ae), que está sendo desenvolvido pelo grupo de Intermídia do ICMC-USP.

   Os principais objetivos do Projeto de Aprendizagem Eletrônica (TIDIA-Ae) são a pesquisa e o desenvolvimento na área de tecnologia da informação voltada para especificação, projeto e implementação de ferramentas aplicáveis à área de Educação a Distância (EaD).

O TIDIA-Ae também prevê o desenvolvimento de um ambiente de aprendizagem eletrônica (Ae), que servirá como base para a criação de novas ferramentas de EaD. O desenvolvimento desse ambiente deverá ser fundamentado em uma arquitetura baseada em componentes, facilitando, assim, sua elaboração, implementação, manutenção e principalmente, sua evolução, permitindo que novas funcionalidades sejam acrescidas ao longo do tempo.

Pretende-se, com a integração deste trabalho com o projeto TIDIA-Ae, a utilização do robô em um laboratório virtual. Este laboratório virtual contará com o robô Pioneer conectado a uma *whiteboard*. Por meio desta, o professor poderá interferir no mapa, obtido pelo robô, colocando obstáculos virtuais e rodar diversos experimentos correspondentes às mudanças efetuadas neste ambiente (mapa) virtual.

2. **Melhoramentos para aumentar o grau de autonomia do sistema**

   Com o decorrer do trabalho, foram percebidos alguns empecilhos para que o robô pudesse ficar efetivamente on-line. Percebeu-se a necessidade de uma grande equipe envolvida neste projeto e é importante que ocorra a inclusão de outros pesquisadores no grupo, a fim de, desenvolver as funcionalidades necessárias para que o robô Pioneer 1 possa ficar disponível, e ser controlado remotamente, durante a maior parte do dia na Internet.

   Para deixar o sistema efetivamente on-line são necessários alguns módulos que provêem uma maior autonomia ao robô. Esses módulos são responsáveis por fazer a recuperação do sistema em caso de falhas e são explicados a seguir:

   - **Tratamento da conexão:** Todas as informações sobre localização, mapeamento e planejamento de trajetórias do robô estão no servidor central de controle do robô. Em caso de perda de conexão com o servidor, o robô deixa de ter todas as informações necessárias para se locomover no ambiente. As informações não são perdidas do servidor, mas sem a conexão o robô não obtém os comandos necessários para se locomover de forma segura no ambiente e acaba ficando a "deriva".

     A construção de um módulo de tratamento da conexão se faz necessário para que esse tipo de problema não ocorra, ou ocorra apenas esporadicamente. É impossível garantir 100% de confiabilidade, pois, o sistema depende dos serviços de rede do local com o qual estará interagindo. Porém,

pretende-se que esse percentual de falhas causadas pela perda de conexão seja muito baixo e não prejudique na execução das tarefas do robô.

- **Doca de carregamento:** O robô utiliza uma bateria de 12V que possui uma autonomia média de 3 horas. Já está sendo estudada a construção de uma doca para o recarregamento do robô. Atualmente, o robô é usado e quando sua bateria acaba ele tem que ser recarregado manualmente. Para que o sistema possa ser utilizado de forma on-line, faz-se necessária a recarga automática da bateria do robô.

  Um sistema de monitoramento do nível de carga da bateria do robô deverá ser construído. Sendo assim, pode-se monitorar a carga da bateria do robô e enviá-lo automaticamente para a doca de recarregamento quando a sua bateria estiver com um nível baixo de carga. O sistema deverá informar para o usuário que o robô está temporariamente fora de serviço. Após a recarga total da bateria o robô volta às suas atividades normais, até que uma nova parada para recarregamento da bateria seja necessária.

- **Rádio Modem:** Durante o desenvolvimento do projeto algumas dificuldades técnicas foram surgindo. Alguns problemas de comunicação entre os rádio modens apareceram e não puderam ser sanados. Espera-se que em breve esses problemas possam ser corrigidos e que os rádio modens possam ser instalados, de maneira adequada, tanto laser do robô, quanto no servidor de controle central.

  A utilização dos rádios modem é necessária para aliviar o peso carregado pelo robô. O robô Pioneer 1 é um robô de pequeno porte e atualmente é necessário que o robô carregue um peso de aproximadamente 5kg. Este peso é proveniente do sensor laser e do notebook. A utilização do notebook ainda é necessária devido ao não funcionamento dos rádio modens. Neste projeto o notebook está funcionando apenas como uma placa de rede sem fio para o envio dos dados provenientes do sensor laser para o sistema de controle central do robô.

- **Fusão de sensores:** Notou-se que apenas com os sinais obtidos do sensor laser o robô não detecta paredes constituídas por vidros, tais como, portas de vidro. Nesse caso, o mapeamento não reflete fielmente o ambiente e existe a necessidade de se considerar também os sinais recebidos dos sonares. Sendo assim, se faz necessário a utilização de uma abordagem de fusão de sensores para minimizar esses tipos de erros. Uma possível abordagem de fusão de sensores a ser implementada é explicada em (Dam 1998).

3. **Métodos exploratórios automáticos**

   Para que o robô possa realizar o mapeamento do ambiente é necessário que ele faça uma exploração por esse ambiente. No presente projeto essa exploração do ambiente é feita de forma manual. Isso não é um grande problema, haja visto que o robô precisa explorar o ambiente apenas uma vez para realizar o mapeamento. Entretanto, espera-se que o robô tenha cada vez mais autonomia e consiga realizar todas as tarefas necessárias para seu deslocamento em ambientes de forma automática.

   Um método de exploração automática de ambientes utilizando funções harmônicas (Prestes et al. 2002) foi analisado e deverá ser implementado futuramente.

4. **Obtenção automática do mapa topológico**

   O mapa topológico (explicados na Seção 3.2) é utilizado para facilitar o planejamento de trajetórias. Neste sistema o mapa topológico está sendo construído manualmente sobre o mapa métrico obtido pelo robô. Esta é uma solução paliativa e bastante aceitável, visto que o mapa topológico precisa ser criado apenas uma vez para um determinado ambiente.

   No entanto, a obtenção automática do mapa topológico pode ser conseguida por meio de técnicas de geometria computacional ou esqueletização de imagens. Durante esse trabalho foi estudada a técnica de diagramas de Voronoi. Uma implementação dessa técnica é explicada em (Thrun and Bucken 1996) e será usada futuramente para assegurar que o robô tenha cada vez mais autonomia nas suas tarefas.

Referências Bibliográficas

Alvares, A. J. and L. S. J. R. Jr. (2000). Desenvolvimento de um robô móvel autônomo teleoperado via internet. In *I CONEM - Congresso Nacional de Engenharia Mecânica.*

Barbosa, A. W., R. E. Bianchi, and R. A. F. Romero (2005). Control of mobile robots via internet. In *18th COBEM - International Congress of Mechanical Engineering.*

Bianchi, R. E. (2003). Sistema de navegação de robôs móveis autônomos para o transporte de documentos. In *Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.*

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

Buhmann, J. M., W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun (1995). The mobile robot RHINO. *AI Magazine 16*(2), 31 38.

Carvalho, A. C. P. L. F., A. C. B. Delbem, R. A. F. Romero, E. V. Simões, and G. P. Telles (2004). Computação bioinspirada. In *Capítulo de livro da XXIII JAI - Jornada de Atualização em Informática - JAI'2004, Anais do XXIV Congresso da SBC.*

Cox, I. J. and G. T. Wilfong (Eds.) (1990). *Autonomous Robot Vehicles.* Springer Verlag.

Dam, J. V. (1998). *Environment Modelling for Mobile Robots: Neural Learning for Sensor Fusion.* Ph. D. thesis, Universiteit van Amsterdam, Faculteit WINS, Amsterdam, The Netherlans.

Dellaert, F., D. Fox, W. Burgard, and S. Thrun (1999). Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Elfes, A. (1989). *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Ph. D. thesis, Carnegie Mellon University.

Faria, G. and R. A. F. Romero (2004). Estratégia para futebol de robôs baseada em campos potenciais. In *Anais da SBC 2004 - XXIV Congresso da Sociedade Brasileira de Computação*.

Fox, D., W. Burgard, and S. Thrun (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems 25*, 195 207.

Fox, D., W. Burgard, and S. Thrun (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research 11*, 391 427.

Fox, D., W. Burgard, S. Thrun, and A. B. Cremers (1998). Position estimation for mobile robots in dynamic environments. In *AAAI/IAAI*, pp. 983 988.

Grange, S., T. W. Fong, and C. Baur (2000, April). Effective vehicle teleoperation on the world wide web. In *International Conference on Robotics and Automation*. IEEE.

Guimarães;, E. G., A. T. Maffeis;, R. P. Pinto;, C. A. Miglinski;, E. Cardozo;, M. Bergerman;, and M. F. Magalhães (2003). Real - a virtual laboratory built from software components. In *Proceedings of the IEEE*, Volume 46.

Johnsonbaugh, R. (1997). *Discrete Mathematics*. Prentice Hall.

Kaelbling, L. P., A. R. Cassandra, and J. A. Kurien (1996). Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Koenig, S., R. Goodwin, and R. Simmons (1996). Robot navigation with markov models: A framework for path planning and learning with limited computational resources.

Koren, Y. and J. Borenstein (1991, April). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Conference on Robotics and Automation*, Sacramento, California, pp. 1398 1404.

Kortenkamp, D. and T. Weymouth (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial intelligence*, Menlo Park, pp. 979 984. AAAI: AAAI Press/MIT Press.

Lee, D. (1996). *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Robot*. Cambridge University Press.

Matarić, M. J. (1994). Interaction and intelligent behavior. Technical Report AI-TR-1495, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA.

Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 61–74.

Murphy, R. R. (2000). *Introduction to AI Robotics*. MIT Press.

Prestes, E., P. M. Engel, M. Trevisan, and M. A. Idiart (2002, julho). Exploration method using harmonic functions. *Robotics and Autonomous Systems 40*(1), 25–42.

Quiles, M. G. (2004). Sistema de visão baseado em redes neurais artificiais para o controle de robôs móveis. In *Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo*.

Ribeiro, C., A. Reali, and R. Romero (2001). Robôs móveis inteligentes: Princípios e técnicas. In *Capítulo de livro da I Jornada de Atualização em Inteligência Artificial - JAIA'2001, Anais do XXI Congresso da SBC*.

Schulte, J., C. Rosenberg, and S. Thrun (1999). Spontaneous short-term interaction with mobile robots in public places.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance.

Simmons, R., J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan. Xavier: An autonomous mobile robot on the web. *IEEE Robotics and Automation Magazine*.

Simmons, R. and S. Koenig (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI–95*, Montreal, Canada, pp. 1080–1087.

Simmons, R. G., R. Goodwin, K. Z. Haigh, S. Koenig, J. O'Sullivan, and M. M. Veloso (1997). Xavier: Experience with a layered robot architecture. *ACM Magazine Intelligence*.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence 99*(1), 21–71.

Thrun, S., M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz (1999). MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Thrun, S., M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. R. Rosenberg, N. Roy, J. Schulte, and D. Schulz (1999). MINERVA: A tour-guide robot that learns. In *KI - Kunstliche Intelligenz*, pp. 14–26.

Thrun, S. and A. Bucken (1996). Integrating grid-based and topological maps for mobile robot navigation. In *AAAI/IAAI, Vol. 2*, pp. 944 950.

Thrun, S., W. Burgard, and D. Fox (1998). *Probabilistic Robotics.*

Thrun, S., D. Fox, W. Burgard, and F. Dellaert (2000). Robust Monte Carlo localization for mobile robots. Technical report, Carnegie Mellon University.

Ulrich, I. and J. Borenstein (2000). VFH*: Local obstacle avoidance with lookahead verification. In *IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 2505–2511.

Yong, N. and R. Simmons (1998). The lane-curvature method for local obstacle avoidance. In *Proceedings Intelligent Robots and Systems(IROS), Victoria Canada, 1998.*

Zhai, S. and P. Milgram (1991). A telerobotic virtual control system. In *Proceedings of SPIE, vol.1612, Cooperative Intelligent Robotics in Space II, Boston, 311-320.*