Um sistema integrado para navegação autônoma de robôs móveis

Janderson Rodrigo de Oliveira

Um sistema integrado para navegação autônoma de robôs móveis

Janderson Rodrigo de Oliveira

Orientadora: ***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 - Ciências de Computação e Matemática Computacional.

USP – São Carlos
Janeiro/2010

AGRADECIMENTOS

À Universidade de São Paulo, pela oportunidade de realização do curso de mestrado.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior e à Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio financeiro para a realização desta pesquisa.

Ao Instituto de Ciências Matemáticas e de Computação, por colocar à disposição o Laboratório de Aprendizado de Robôs.

À Prof[a]. Dr[a]. Roseli Aparecida Francelin Romero, pela orientação ao longo desse projeto de mestrado.

Ao Edson Aparecido Storolli Filho, pela constante ajuda na solução dos diversos problemas que surgiram durante esse trabalho.

À minha família e amigos, pelo apoio sempre presente.

“Não sou nada.

Nunca serei nada.

Não posso querer ser nada.

À parte a isso, tenho em mim todos os sonhos do mundo”.

Fernando Pessoa

RESUMO

OLIVEIRA, J. R. **Um sistema integrado para navegação autônoma de robôs móveis**. 2010. 100 f. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010.

O mapeamento de ambientes é um dos maiores desafios para pesquisadores na área de navegação autônoma. As técnicas existentes estão divididas em dois importantes paradigmas, o mapeamento métrico e o topológico. Diversos métodos de mapeamento que combinam as vantagens de cada um desses paradigmas têm sido propostos. Este projeto consiste na adaptação e extensão de um sistema integrado para navegação autônoma de robôs móveis através do aperfeiçoamento da interface e também da incorporação de uma técnica de mapeamento topológico. Para isso, a técnica conhecida como Grade de Ocupação, utilizada em geral para mapeamento métrico é combinada com um método de esqueletização de imagens para a realização do mapeamento topológico. Além disso, filtros morfológicos matemáticos de erosão e abertura, adequados a ambientes reais, foram utilizados, visando reduzir a influência de ruídos na abordagem proposta, uma vez que devido a ruídos inerentes as leituras sensoriais obtidas pelo robô, o mapa topológico gerado apresenta diversas linhas topológicas desnecessárias, dificultando consequentemente a tarefa de navegação autônoma. Vários experimentos foram executados para verificar a eficiência da combinação de técnicas proposta, tanto em nível de simulação quanto em um robô real. Os resultados obtidos demonstraram que a técnica de esqueletização de imagens combinada ao mapeamento métrico do ambiente é uma forma simples e viável de se obter as linhas topológicas do espaço livre do ambiente. A aplicação dos filtros morfológicos matemáticos demonstrou ser eficiente para a criação de mapas topológicos livres de ruído, uma vez que elimina grande parte das linhas topológicas geradas em conseqüência dos ruídos dos sensores do robô.

Palavras-chave: Robôs móveis, mapeamento topológico, método de esqueletização de imagens, filtros morfológicos matemáticos.

ABSTRACT

OLIVEIRA, J. R. **A mobile robot autonomous navigation integrated system**. 2010. 100 f. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010.

The environment mapping has been a great challenge of many researchers of the autonomous navigation area. There are two important paradigms for mapping, metric and topological mapping. Diverse mapping methods that combine the advantages of each paradigm have been proposed. This project consists to adaptation and extension of a mobile robots autonomous navigation integrated system by improving the interface and incorporation of a topological mapping technique. For this, the technique known as Occupation Grid for metric mapping is combined with an image skeletonization method used for topological mapping. This work also aims to propose a set of mathematical morphology filters to generation of topological maps suitable for real environments, seeking to reduce influence of noises in performed mapping. The topological map generated through this combination presents several unnecessary topological lines, due noises inherent to the own robot ability of capturing sensor signals, hindering consequently the task of autonomous navigation. Several experiments were performed to verify the efficiency of the proposed approach. The results obtained demonstrate that image skeletonization technique combined with the metric mapping is a simple and feasible method for obtaining the topological lines corresponding to free space of the environment. The application of the mathematical morphology filters demonstrated to be a useful method to the creation of topological maps free of noises, since it eliminates most of the topological lines generated in consequence of noise in the sensors.

Keywords: Mobile robots, topological mapping, image skeletonization method, mathematical morphology filters.

LISTA DE FIGURAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

"Não quero ser o início de um fim e nem o fim de um começo,

quero ser o início de um começo sem fim".

Autor desconhecido

A necessidade da criação de sistemas e do desenvolvimento de novas tecnologias que visam o auxílio gradativo ao trabalho humano é uma vertente em ascensão na humanidade. Como pré-requisito para este paradigma tais sistemas devem ser capazes de executar suas tarefas de forma que não haja interferência externa. Sistemas que possuem esta característica são denominados sistemas independentes. Nesta classe, tem-se o problema de navegação autônoma de robôs móveis, que consiste no desenvolvimento de mecanismos de tomada de decisão para um ou mais robôs em um ambiente arbitrário, no qual devem realizar tarefas de forma autônoma. Em outras palavras, um sistema de navegação autônoma deve ser capaz de realizar duas tarefas: primeiro, guiar um robô sem nenhuma interferência externa, e segundo, definir ações para o mesmo, tais como atingir alvos, desviar de obstáculos e explorar o ambiente (CALVO; FIGUEIREDO, 2003; CAZANGI; FIGUEIREDO, 2002; CRESTANI et al., 2002; BORENSTEIN et al., 1996).

Segundo Bensalem et al. (BENSALEM et al., 2009), um sistema robótico deve ser capaz de trabalhar com uma ampla variedade de informação, como por exemplo: perceber o ambiente, planejar suas ações, lidar com ambientes dinâmicos e com incertezas em tempo real, interagir inteligentemente com humanos e outros sistemas robóticos, e garantir sua própria segurança, assim como das pessoas ao seu redor.

A pesquisa na área de navegação autônoma de robôs móveis provê basicamente três desafios fundamentais, sendo eles: a localização, o mapeamento de ambientes, e o

planejamento de trajetórias. A localização de robôs móveis consiste na tarefa de estimar a posição de um robô, por meio de informações obtidas de seus sensores, em um mapa do ambiente no qual o robô está inserido. O mapeamento tem o objetivo de criar, também através dos dados adquiridos de seus sensores, um modelo espacial do ambiente do robô. Por fim, o planejamento de trajetória é responsável por enviar comandos motores ao robô fazendo com que ele navegue de forma segura, desviando de obstáculos, até um local determinado.

1.1 MOTIVAÇÃO

Existe uma grande variedade de aplicações para robôs móveis autônomos nos mais diversos cenários, tais como: silvicultura, espaço, reatores nucleares, desastres ambientais, indústrias, e escritórios. As tarefas executadas nesses ambientes são geralmente perigosas para os seres humanos e remotamente localizadas (GARRIDO et al., 2008). Sendo assim tarefas em potencial para robôs móveis autônomos incluem serviços de manutenção, entrega e vigilância.

Algumas abordagens de sistemas de navegação consideram que o ambiente (topologia e eventos) é totalmente conhecido, ou seja, o sistema de navegação tem o conhecimento do mapa do ambiente e, além disso, todos os possíveis eventos pertencem a um número finito de classes conhecidas (ambientes estáticos e controlados). Neste caso, o sistema de navegação pode ser programado com antecedência, com poucos parâmetros adaptáveis.

Há grandes expectativas para aplicações de robótica em que o ambiente seja totalmente desconhecido. Neste caso, a autonomia é uma das características essenciais para sistemas de controle e, em particular, para sistemas de navegação (CRESTANI et al., 2002; ANTSAKLIS, 1995).

Smithers (SMITHERS, 1997) apresentou uma definição abrangente para o termo “autonomia”, quando aplicado a um sistema de controle artificial. De acordo com esta definição, ”autonomia” se refere não apenas à capacidade de ação e decisão independente de influências externas, mas sobretudo à capacidade de autoprodução dos mecanismos que geram a ação. Ou seja, um sistema autônomo deve ser capaz de sintetizar suas próprias leis de ação sobre o ambiente.

O presente trabalho visa contribuir com o desenvolvimento de pesquisas ligadas ao tema de mapeamento. A técnica de mapeamento empregada até então[1] (BIANCHI, 2002; BARBOSA, 2006) possui um consumo de espaço grande e, além disso, ela é independente da complexidade do ambiente, ou seja, ambientes simples e complexos ocupam praticamente o mesmo espaço. Isso ocorre porque a representação do meio através dessa técnica deve ser detalhada o suficiente para capturar cada detalhe relevante do mundo. Contudo, os mapas gerados por essa abordagem são fáceis de construir, representar e manter. De maneira que, a geometria do mapa gerado corresponde diretamente à geometria do ambiente, facilitando dessa forma tarefas de localização do robô. Por outro lado, mapas ditos topológicos são geralmente muito mais compactos, uma vez que eles dependem diretamente da complexidade do ambiente. Além disso, os mapas topológicos não requerem a determinação exata da posição do robô e eles permitem também a obtenção de interfaces mais naturais para instruções humanas. Sendo assim, torna-se necessário combinar as informações obtidas por cada tipo de mapeamento em uma solução mais eficiente para lidar com ambientes reais amplos e complexos.

1.2 OBJETIVOS

Os objetivos do presente trabalho podem ser definidos através de duas vertentes distintas, sendo elas: a adaptação de um sistema de navegação autônoma, existente no Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), para uma nova plataforma mais robusta; e o aperfeiçoamento da autonomia desse mesmo sistema, buscando-se para esse propósito explorar a aplicação de uma técnica de mapeamento topológico no sistema desenvolvido.

Este sistema é inicialmente idealizado para a tarefa de entrega de documentos no ICMC. No entanto, ele poderá ser utilizado posteriormente em outras tarefas, tais como entrega de peças e/ou objetos num ambiente de chão de fábrica, segurança de ambientes e coleta de lixo, incluindo também tarefas relacionadas ao ensino da robótica, através da disponibilização via Web dos algoritmos implementados e testados.

Para a consecução dessa meta os seguintes objetivos específicos foram abordados nesse projeto:

[1] Mapeamento métrico

- Adaptação do sistema para a plataforma do robô Pioneer 3 – DX, pois a atual plataforma (Pioneer 1) não se encontra completamente em estado viável de operação, devido principalmente ao desgaste de seus componentes. Esse desgaste ocasiona vários erros de navegação que se acumulam gradativamente ao longo da trajetória do robô. É importante salientar que o sistema como um todo ainda não foi totalmente adaptado para a plataforma do Pioneer 3 – DX, contudo diversas etapas foram alcançadas visando essa mudança;
- Realização de uma atualização no módulo responsável pela Interface Web, tornando-o efetivamente operacional. Inicialmente esse módulo apresentava diversos problemas causados pela mudança de versão da plataforma de desenvolvimento da Interface;
- Exploração e incorporação de uma técnica de mapeamento topológico à abordagem métrica existente. Deve-se destacar que o mapeamento topológico realizado apresentou algumas características (linhas topológicas) indesejáveis. Essas características são oriundas de ruídos pertencentes à abordagem de mapeamento métrico, sendo assim também se tornou necessário o estudo de mecanismos que permitissem um tratamento dos mapas topológicos gerados. Os mecanismos de tratamento utilizados foram os filtros morfológicos matemáticos.

A principal contribuição do presente trabalho consiste especificamente na junção da técnica de mapeamento métrico implementada em (BIANCHI, 2002; BARBOSA, 2006), denominada Grade de Ocupação, com o método de mapeamento topológico explorado nessa dissertação. O método topológico proposto em (KO et al., 2004) não explora a união do método topológico com uma técnica métrica específica, ao invés disso, ele ilustra genericamente a construção de mapas topológicos baseados em esqueletização de imagens a partir de um mapa métrico qualquer. Por isso, uma das contribuições do presente trabalho é a junção da técnica Grade de Ocupação com o método de esqueletização de imagens proposto por Ko (KO et al., 2004), visando combinar as informações obtidas por cada tipo de mapeamento em uma solução mais eficiente, buscando lidar com ambientes reais amplos e complexos. Além disso, a aplicação de filtros morfológicos matemáticos é outra importante contribuição que permitiu a construção de mapas topológicos de maior qualidade. Por fim, é importante destacar que o sistema desenvolvido, fruto desta dissertação, alcançou uma maior

autonomia e robustez através da adaptação gradativa para a plataforma do Pioneer 3 – DX. Finalmente, deve-se salientar que a união das técnicas topológica e métrica além de gerar mapas mais robustos e intuitivos quando comparados com abordagens unicamente métricas também irá contribuir futuramente para pesquisas visando a melhoria do método de localização implementado no sistema de navegação, maiores detalhes sobre como as linhas topológicas podem contribuir com a tarefa de localização podem ser encontrados na Seção 6.1.

1.3 ORGANIZAÇÃO DO TEXTO

O presente trabalho está organizado da seguinte forma. No Capítulo 2, são discutidos os principais conceitos da área da robótica no contexto da Inteligência Artificial, assim como são apresentados alguns trabalhos importantes relacionados ao tema dessa dissertação. No Capítulo 3, uma descrição detalhada dos métodos de localização, mapeamento e planejamento de trajetórias utilizados no sistema de navegação autônoma é realizada, assim como são apresentados os conceitos básicos relativos aos filtros morfológicos explorados nesse trabalho. No Capítulo 4, o estado inicial do sistema é sucintamente detalhado, apresentando-se os diversos módulos que o compõem. No Capítulo 5, são apresentados os principais resultados obtidos com a consecução desse projeto. Por fim, no Capítulo 6, são descritas as conclusões e trabalhos futuros decorrentes do presente estudo.

CAPÍTULO

2

A ROBÓTICA NA INTELIGÊNCIA ARTIFICIAL

"Compreender que há outros pontos de vista é o início da sabedoria".

Campbell

Neste capítulo são abordados alguns aspectos referentes à robótica sob a ótica da Inteligência Artificial. Importantes conceitos são elucidados e os principais desafios da área são brevemente discutidos.

2.1 ROBÔS

O termo "robô" é uma palavra de origem Eslava e significa "trabalhador". Em 1921, esta palavra foi popularizada na peça *RUR* (*Rossum's Universal Robots*), escrita pelo dramaturgo tchecoslovaco Karel Kapek. Nesta peça, as máquinas se revoltam contra a raça humana, matando seus mestres e dominando o planeta. Curiosamente, estas máquinas eram uma raça de trabalhadores construídas de um repositório de partes biológicas, e não de partes mecânicas e eletrônicas como atualmente é concebido o "conceito" de "robô". Em 1950, Isaac Asimov publicou seu livro *I Robot*, o qual descrevia os robôs como máquinas humanóides inteligentes, projetadas especificamente para servir a raça humana seguindo um conjunto de três leis básicas que delineavam as regras sociais da interação entre os robôs e os humanos

(MCKERROW, 1986). É importante salientar que embora os seres humanos atualmente conceituem os robôs através de padrões mecatrônicos e antropomórficos, isto na realidade não é uma obrigação. Na maioria das implementações robóticas suas características estruturais são projetadas para melhor se adequarem às funções que os mesmos irão desempenhar, e não simplesmente projetadas segundo a estrutura física humana.

Desta forma, torna-se necessária uma definição precisa do que é um robô. Arkin (ARKIN, 1998) e Schlussel (SCHLUSSEL, 1985) definem um robô como um equipamento manipulador re-programável e multifuncional, projetado para mover peças, ferramentas, materiais ou dispositivos especializados através de movimentos variáveis programados, visando à execução de um determinado conjunto de tarefas. Entende-se por "equipamento manipulador" a capacidade de uma determinada máquina de modificar sistematicamente o ambiente no qual se encontra. O termo "multifuncional" indica que o robô poderá desempenhar mais de uma função caso seja necessário. Finalmente, a palavra "re-programável" indica que os padrões de atuação da máquina podem ser redefinidos, porém uma vez que um padrão seja definido ele será seguido minuciosamente até que um novo padrão seja estabelecido.

Analisando-se esta definição nota-se que ela se aplica mais precisamente às aplicações industriais, em que o foco de atuação dos equipamentos está na precisão e na contínua realização de movimentos específicos nas diversas linhas de montagem. Assim, pela definição anterior, robôs são programados para executar movimentos fixos, sem que haja a necessidade de se monitorar, ou ainda representar, o ambiente no qual estão inseridos. Então, torna-se necessário uma nova definição para um tipo diferente de robô. Um tipo que evoluiu da simples execução sistemática de um conjunto pré-definido de ações, para máquinas capazes de agir de forma autônoma, ou seja, capazes de "reconhecer" por conta própria o ambiente no qual se encontram e, conseqüentemente, definir seus próprios cursos de ação. Este último tem a denominação de robô inteligente.

2.1.1 Robôs Inteligentes

Brady (BRADY, 1985), citado por McKerrow (MCKERROW, 1986), define a robótica como:

"A robótica é a conexão inteligente entre a percepção e a ação."

Esta afirmativa oferece um aspecto totalmente novo acerca do que é a natureza robótica. Ela simula, ou até mesmo cria, certo grau de "inteligência" nos robôs, os quais agora, a partir desta afirmação, são capazes de perceber e se situar no mundo e de agir "conscientemente" no seu ambiente.

Desta maneira, pode-se definir um robô como um sistema autônomo que existe no mundo físico e que pode sentir o ambiente e agir sobre ele (o ambiente) de forma a atingir uma meta específica (MATARIĆ, 2007).

Segundo Matarić (MATARIĆ, 2007) um robô autônomo realiza uma determinada ação com base em suas próprias decisões, e sem interferência humana. O fato dele existir no mundo físico implica que um robô deverá lidar com as leis e desafios físicos, sendo esta característica uma das questões chaves da área da robótica. Robôs que existem em computadores são apenas simulações, e como tais, eles não precisam lidar verdadeiramente com as propriedades do mundo físico, uma vez que simulações não são tão complexas quanto o mundo real.

Sentir o ambiente significa necessariamente que um robô deve possuir mecanismos para perceber o ambiente, como por exemplo, sensores. Se um sistema não pode "sentir" então ele não pode ser considerado um robô, uma vez que ele não poderá responder ao que acontece ao seu redor. Realizar ações como resposta às entradas sensoriais com o objetivo de atingir o que se deseja é uma parte fundamental da natureza robótica. Porém, deve-se destacar que as possíveis ações realizadas por uma máquina no mundo físico podem assumir diversas formas diferentes (mover objetos, se locomover, explorar uma área, entre outras) e esta é uma das principais razões do campo de estudo da robótica ser tão vasto (MATARIĆ, 2007).

2.1.2 Robôs Teleoperados

Há ainda um outro fator que pode ser acoplado à natureza robótica, conhecido como teleoperação. A teleoperação pode ser vista como o processo pelo qual um operador humano

controla uma máquina à distância (*tele* significa "longe" em Grego) (BARBOSA, 2006). Esta característica foi, inicialmente, idealizada para o uso em tarefas que representam risco à vida humana, como por exemplo, em situações que envolvam a detecção de explosivos, resgate de pessoas em áreas de perigo, ambientes desfavoráveis ou prejudiciais (i.e., exposição a altas temperaturas, radiação, agentes contaminadores, etc.).

Deve-se notar que com a teleoperação é possível que um operador humano exerça remotamente uma ação física qualquer em um determinado ambiente, sendo que esta ação é limitada apenas pelas capacidades inerentes ao robô a ser manipulado. Além disso, o operador também pode receber informações sensoriais por parte da máquina, geralmente através de dados visuais, sonoros ou táteis.

Em geral, sistemas de teleoperação de robôs móveis são constituídos por: uma parte local, onde um operador humano, denominado teleoperador, controla um serviço manipulador; uma parte remota, onde o robô móvel interage com o mundo físico; e finalmente, por um canal de comunicação que liga ambas as partes (SLAWIÑSKI et al., 2007). Pode-se utilizar inclusive a Internet como canal de comunicação entre as partes local e remota do sistema de teleoperação (BARBOSA et al., 2006).

Matarić (MATARIĆ, 2007) não considera os robôs teleoperados como "verdadeiros" robôs, pois para ele verdadeiros robôs devem agir de forma totalmente autônoma. Porém, analisando-se cuidadosamente estas duas características, teleoperação e autonomia, percebe-se que elas podem coexistir em diferentes níveis de intensidade. Zhai e Milgram (ZHAI; MILGRAM, 1991), por exemplo, classificam os sistemas teleoperados nos seguintes graus de autonomia:

1º. Controle manual sem auxílio do computador (robô);

2º. Controle manual com relevante auxílio computacional;

3º. Controle supervisionado com a maior parte do controle sendo realizada pelo operador humano.

4º. Controle supervisionado com a maior parte do controle sendo realizada pelo computador.

5º. Controle completamente automático, sendo que o operador apenas observa o robô realizar suas tarefas.

Segundo a classificação acima, o sistema que está sendo aperfeiçoado nesta dissertação se encontra no quinto nível de autonomia, uma vez que o robô móvel tratado aqui deve agir de forma completamente autônoma, recebendo requisições de tarefas do operador via Web. É importante salientar que o operador não controla o robô, ele apenas envia uma tarefa que deve ser realizada por ele, e este deverá agir autonomamente para conseguir executar a tarefa.

A seguir são apresentados, resumidamente, alguns dos principais casos de robôs móveis autônomos que são, ou já foram, controlados via Internet.

2.1.3 Robôs Móveis Autônomos

Simmons et al. (SIMMONS et al., 1997) apresentaram o robô Xavier, que pôde receber requisições via Web para se locomover entre os escritórios de um determinado prédio. Além disso, o robô pôde enviar ao operador as imagens obtidas durante o seu percurso. Durante o período em que esteve on-line este robô viajou mais de 110 Quilômetros e recebeu mais de 2500 requisições de navegação através de sua interface WWW.

A arquitetura de controle deste robô é dividida em quatro camadas distintas e independentes, sendo elas: camada de desvio de obstáculos, navegação, controle de trajetória, e planejamento de tarefas.

A camada de desvio de obstáculos mantém o robô na direção desejada, enquanto desvia de obstáculos estáticos (cadeiras, mesas, lixeiras, etc.) e dinâmicos (pessoas). Ela providencia, em tempo real, velocidade alta e movimentação segura considerando-se a dinâmica do robô. Sua função objetivo combina segurança, velocidade e progresso ao longo do caminho desejado.

A camada de navegação guia o robô de uma determinada posição para uma outra posição alvo. Esta camada usa o modelo *Partially Observable Markov Decision Process* (KOENIG et al., 1996; SIMMONS; KOENIG, 1995) para manter a distribuição de probabilidades das posições do robô através do tempo, e assim definir as ações necessárias para se mover da origem até a meta.

A camada de controle de trajetória determina como se mover eficientemente de uma localização para outra, ou ainda, ela define rotas eficientes a serem percorridas com base em mapas topológicos e informações métricas obtidas pelos sensores do robô (o mapeamento de ambientes será amplamente discutido na Seção 3.1.2).

Finalmente, a última camada consiste no planejamento de tarefas, que integra as novas requisições de forma assíncrona em um novo plano, priorizando as tarefas e, oportunamente, atingindo as tarefas pertinentes. Ou seja, é a camada responsável por escalonar todas as tarefas recebidas.

Outro caso bastante conhecido é o do robô interativo Rhino (BURGARD et al., 1998; BUHMANN et al., 1995), o qual foi construído para auxiliar e entreter pessoas em lugares públicos, tais como museus. Sua principal tarefa foi realizar um *tour* interativo, oferecendo múltiplas explicações (gráficas e sonoras) sobre as várias exposições ao longo do caminho. Em Maio de 1997, durante um período de seis dias, Rhino "guiou" mais de 2000 visitantes pelo museu *Deutsches Museum Bonn* na Alemanha e estima-se que mais outras 2000 pessoas acompanharam e controlaram sua operação pela Web.

A arquitetura do Rhino consiste de doze módulos, sendo que uma visão geral desses módulos é sumariamente descrita a seguir: os módulos de interface (sonares, câmera e fala) controlam a comunicação básica entre os componentes de hardware do robô. No topo destes módulos, há uma rotina de desvio rápido de obstáculos que é responsável por analisar as medidas dos sonares para evitar possíveis colisões. Um módulo de visão segmenta as imagens obtidas de uma câmera para encontrar objetos alvos e obstáculos que bloqueiem o caminho do robô. Além disso, mapas métricos e topológicos são construídos utilizando-se uma proposta baseada em redes neurais. O fluxo de controle do Rhino é monitorado por um planejador de tarefas e por uma interface central do usuário.

Finalmente, um último exemplo de robô autônomo teleoperado, também utilizado como guia turístico em museus, deve ser mencionado: o robô Minerva (THRUN et al., 2000; THRUN et al., 1999a; THRUN et al., 1999b).

Minerva foi inicialmente projetado para avaliar o uso da robótica probabilística em situações de alta complexidade, ou seja, para testar o uso de robôs probabilísticos no mundo real. No Outono de 1998, Minerva foi inserido em um dos maiores museus dos EUA, o *Smithsonian Museum of American History*, em Washington. A tarefa do robô era atrair a atenção das pessoas e explicar as várias exposições ao longo do museu.

O robô foi controlado por uma aplicação genérica voltada para a navegação robótica e interação humano-robô, sendo que esta aplicação buscou resolver os seguintes problemas (THRUN et al., 1999b):

- Navegação em ambientes dinâmicos: lugares públicos estão, na maioria das vezes, ocupados por uma grande quantidade de pessoas e geralmente os comportamentos dessas pessoas não são necessariamente cooperativos. Então, a abordagem adotada pelo robô providencia mecanismos de navegação segura e efetiva no meio de multidões.
- Navegação em ambientes não modificados: nenhuma modificação no ambiente foi necessária para a operação do robô. O sistema foi capaz de se adaptar a qualquer alteração no ambiente.
- Interação humano-robô: a plataforma foi especialmente projetada para interagir com pessoas, ou multidões, que nunca foram expostas a um robô anteriormente. Para atrair a atenção, a interface do robô usa padrões de interação similares aos encontrados quando as próprias pessoas interagem umas com as outras. O robô Minerva utilizava, por exemplo, os seguintes padrões, olhar para a pessoa mais próxima, utilizar diferentes expressões faciais (felicidade, tristeza, raiva), empregar diferentes opções de fala (“Venha.”, “Você gosta de robôs?”), entre outras estratégias.
- Tele presença virtual: uma interface Web possibilitou que qualquer pessoa ao redor do mundo monitorasse o robô, controlasse seu movimento, e assistisse as imagens gravadas por suas câmeras.

A arquitetura de software do Minerva consiste de aproximadamente 20 módulos distribuídos que se comunicam de forma assíncrona. Estes módulos 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 de interface atuam diretamente nos sensores e atuadores do robô (lasers, sonares, câmeras, motores, face, unidade de fala, display sensível a toque, servidor, entre outros). Sobre eles, há vários módulos de navegação que executam

funções de mapeamento, localização, desvio de obstáculos, controle de trajetórias, e planejamento da missão global.

Os módulos de interação determinam o "estado emocional" do robô, a direção da sua cabeça e a definição de sua forma de comunicação (fala ou sinais sonoros). Finalmente, a interface Web consiste dos módulos responsáveis por apresentar as informações referentes ao robô, tais como as imagens obtidas por suas câmeras e sua posição no museu. Além disso, estes módulos também são responsáveis por receber os comandos de usuários conectados via Web.

2.2 PARADIGMAS E ARQUITETURAS

De acordo com Bianchi (BIANCHI, 2002), um paradigma robótico é um conjunto de concepções que caracteriza a estrutura da "inteligência" em robôs. Estas concepções são definidas através de três características básicas e o modo como estas primitivas são organizadas é que estabelece um tipo de paradigma robótico. As três primitivas básicas são: *sentir*, *planejar* e *agir*. *Sentir* é o processo pelo qual um robô captura, através de seus sensores, a informação do ambiente no qual está inserido. *Planejar* é a tarefa de receber informação e a partir desta definir um curso de ação apropriado. E por último, *agir* é o processo de executar efetivamente os comandos responsáveis pelos atuadores do robô (motores e dispositivos que modificam o ambiente de forma ativa). Murphy (MURPHY, 2000) afirma que a partir destas três características básicas há os seguintes paradigmas robóticos: hierárquico, reativo e híbrido (reativo/deliberativo).

O paradigma robótico hierárquico foi o primeiro a ser idealizado. Nele as primitivas básicas estão organizadas de forma totalmente seqüencial. Ou seja, inicialmente é necessário capturar dados do ambiente e usá-los para construir um modelo de mundo global (*sentir*). Em seguida, o modelo de mundo global deve ser interpretado, e a partir dele deve-se estabelecer as ações necessárias para alcançar uma tarefa alvo (*planejar*). Por fim, uma vez que ações já foram previamente definidas, os comandos que irão controlar os atuadores do robô são executados (*agir*). Como pode ser observada pela Figura 2.1 esta organização é cíclica, uma vez que já se tenha sentido, planejado e agido, todo o processo é reiniciado.

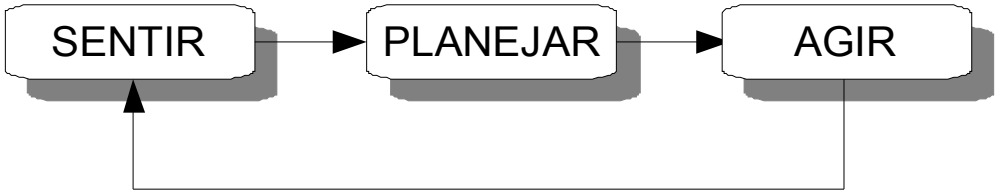


Figura 2.1 – Paradigma hierárquico

A principal carência encontrada nesta organização é o tempo de resposta para se verificar a validade de uma ação. Como os processos de planejamento e criação de modelo de mundo são complexos e computacionalmente caros existe uma espera razoável para se perceber o efeito de uma ação realizada pelo robô em um ciclo anterior.

Para tentar resolver este problema um novo paradigma foi proposto, o paradigma reativo. Nele a primitiva de planejamento foi totalmente eliminada com o objetivo de aproximar o máximo possível a "ação" da "percepção" (Figura 2.2). Brooks (BROOKS, 1991) afirma que:

"O melhor modelo de mundo é ele próprio."

Seguindo esta filosofia o paradigma reativo foi proposto para ser orientado a "elos" que ligam a percepção à ação. Estes elos são unidades comportamentais básicas que se combinam para formar comportamentos independentes. Em outras palavras, vários comandos motores independentes são gerados em resposta a alguma entrada perceptual e estes devem ser integrados posteriormente para realizar uma ação.

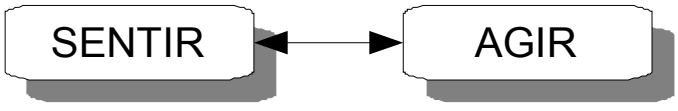


Figura 2.2 – Paradigma reativo

Porém, um novo problema surge desta abordagem. Muitas tarefas requerem uma representação de mundo mínima, tais como o mapeamento e a localização. Tais tarefas são

cruciais para a navegação autônoma de robôs móveis em percursos que envolvam longas distâncias durante longos períodos de tempo (BIANCHI, 2002).

A alternativa encontrada para solucionar este fato foi elaborar uma metodologia híbrida dos dois paradigmas anteriores. No paradigma híbrido, a primitiva *planeja*r ainda está presente, contudo não mais estruturada de forma seqüencial entre *sentir* e *agir* (Figura 2.3). Neste paradigma o planejamento é realizado em um passo e as tarefas de *sentir* e *agir* são executadas simultaneamente em outro. Desta forma, *planejar* não faz parte do ciclo de percepção-ação, sendo que esta primitiva só irá interferir no processo global quando obtiver algum resultado relevante de planejamento para a aplicação.

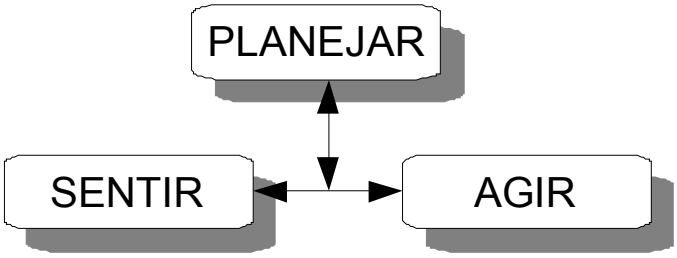


Figura 2.3 – Paradigma híbrido

Uma vez que os paradigmas robóticos já foram definidos, serão apresentadas agora algumas arquiteturas que podem ser seguidas para o desenvolvimento de uma aplicação robótica. Neste contexto, uma arquitetura robótica pode ser compreendida como o projeto de construção de robôs a partir de blocos comuns de software (ARKIN, 1998).

Como arquitetura do paradigma hierárquico pode-se citar o *Nested Hierchical Controller* (MURPHY, 2000). Esta arquitetura estabelece a organização seqüencial das primitivas *sentir*, *planejar* e *agir*. Contudo, o processo de planejamento do NHC é dividido em três agentes, ou módulos, sendo eles: o planejador da missão, o navegador e o piloto. O planejador é módulo responsável por criar, ou receber de um operador humano, uma missão a ser realizada. A missão então é entregue ao navegador, que define um curso de ação para atingir a meta. Esse curso de ação é definido como um conjunto de segmentos de reta que devem ser repassados, um a um, ao piloto. Finalmente, o módulo piloto, após receber um segmento de reta, determina qual é a ação necessária para realizar a tarefa recebida, e então envia os comandos necessários para o controlador de baixo nível, que os traduz nos sinais de controle apropriados.

No paradigma reativo a principal diferença entre suas arquiteturas está no modo como os comportamentos independentes são organizados e combinados para se obter o comportamento geral. Como por exemplo, na arquitetura *Subsumption* (ARKIN, 1998) os comportamentos são estruturados em camadas e a combinação deles é realizada ou por inibição ou por supressão. Durante a inibição, o comportamento da atividade inibidora substitui a saída da atividade inibida. Na supressão, o comportamento dito supressor substitui a entrada da atividade suprimida pela saída da atividade supressora.

Como exemplo de arquitetura do paradigma híbrido pode-se mencionar a Saphira (MURPHY, 2000). Essa arquitetura busca conciliar três princípios chaves: coordenação, coerência e comunicação. Coordenação porque um robô deve coordenar seus objetivos globais de longo prazo, além de coordenar seus sensores e atuadores. Coerência pela necessidade de se manter modelos de mundo globais fiéis à realidade. E comunicação pela interação com seres humanos e outros robôs. Esta arquitetura usa um planejador reativo que é capaz de receber comandos em linguagem natural e interpretá-los em tarefas de navegação e rotinas de percepção.

2.3 NAVEGAÇÃO AUTÔNOMA

A pesquisa na área de navegação autônoma de robôs móveis em ambientes não estruturados (ambientes comuns que não foram modificados para facilitar o uso de robôs) provê desafios muito interessantes para a área da Ciência da Computação. Dentre estes desafios, a localização, o mapeamento de ambientes e o planejamento de trajetória estão entre os mais fundamentais.

A localização de robôs móveis consiste basicamente da tarefa de estimar a posição de um robô móvel, por meio de informações obtidas de seus sensores, no mapa do ambiente no qual o robô está inserido. Deve-se destacar que a localização é um dos componentes mais importantes de um sistema de navegação de robôs móveis autônomos, como pode ser percebido pela afirmação de Cox (COX, 1991):

> *"Usar informação sensorial para localizar o robô em seu ambiente é o problema mais fundamental para dar capacidades autônomas a um robô móvel."*

O mapeamento tem o objetivo de criar, também através dos dados adquiridos de seus sensores, um modelo espacial do ambiente do robô. Para criar este mapa, o robô deve explorar o ambiente ao seu redor. Contudo, ele pode realizar essa exploração tanto sob o controle de um ser humano quanto de forma completamente autônoma. O mapeamento também é considerado uma tarefa importante, pois a localização requer um mapa detalhado do ambiente. O que pode ser difícil de se conseguir manualmente ou por meio de plantas, uma vez que estas geralmente não contém informações sobre móveis e outros objetos pertencentes ao ambiente.

O planejamento de trajetória é responsável por enviar comandos motores ao robô fazendo com que ele navegue de forma segura, desviando de obstáculos, até um local determinado. Além disso, o problema do controle de trajetória também está relacionado à definição de qual é a melhor rota que liga a posição atual do robô com a sua meta.

Maiores detalhes sobre localização, mapeamento e planejamento de trajetória são discutidos no Capítulo 3.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram elucidados sumariamente os principais conceitos sobre o que são robôs inteligentes, assim como os paradigmas e algumas arquiteturas robóticas. Três dos principais robôs móveis encontrados na literatura que puderam ser controlados via Web foram apresentados, sendo eles: o robô Xavier, o robô Rhino e mais recentemente o robô Minerva.

O capítulo seguinte é dedicado a descrever detalhadamente os principais aspectos do sistema de navegação autônoma de um robô móvel, apresentando uma explicação aprofundada dos métodos de localização, mapeamento e planejamento de trajetórias.

CAPÍTULO

3

ASPECTOS TEÓRICOS GERAIS

"Ao entender esta escritura, jogue-a

fora. Se não conseguir entendê-la

jogue-a fora. Insisto na sua liberdade".

Jack Kerovav

Esse capítulo aborda os principais aspectos teóricos envolvidos nesse trabalho. Sendo assim, ele elucida todas as técnicas utilizadas para o desenvolvimento do sistema autônomo explorado nesse projeto de Mestrado. De modo geral, os aspectos teóricos discutidos se referem às áreas de navegação autônoma e de filtros morfológicos matemáticos.

3.1 NAVEGAÇÃO AUTÔNOMA

Nesta seção são abordados os conceitos chaves da navegação autônoma de um sistema robótico. São tratados aspectos relativos à localização, mapeamento e planejamento de trajetórias de robôs móveis.

3.1.1 Aspectos da Localização

Localização de robôs móveis é o processo de se estimar a posição de um robô dentro do ambiente no qual está inserido (OLSON, 2000). Para fazê-lo é necessário que o robô utilize um mapa que modele o mundo ao seu redor e estime através deste modelo sua posição corrente e sua orientação. Sendo assim, a habilidade de estimar sua posição, através de seus sensores, é uma das pré-condições básicas para a autonomia de robôs móveis (BURGARD et al., 1996; WOLF et al., 2005; ANDREASSON et al., 2007).

O mapa do ambiente pode ser obtido de duas formas diferentes: ou automaticamente pelo próprio robô, que seria responsável por explorar previamente o ambiente; ou manualmente projetado por um ser humano, que definiria as características do mapa utilizando alguma ferramenta de projeto auxiliado por computador.

Segundo a referência anterior, as técnicas de localização, para serem consideradas robustas, devem apresentar os seguintes pré-requisitos:

- A técnica deve ser capaz de lidar com incertezas, uma vez que os sensores são geralmente imperfeitos e os mapas que modelam o ambiente são, quase sempre, imprecisos;
- O método deve ser capaz de lidar com ambigüidades. Por exemplo, um escritório típico contém vários locais que não podem ser diferenciados apenas com uma única medida sensorial (i.e., corredores);
- Por fim, a técnica deve permitir a integração de leituras sensoriais de diferentes tipos de sensores, pois a fusão sensorial melhora a confiança do sistema, compensando possíveis ruídos e resolvendo ambigüidades quando necessário.

Atualmente, há várias abordagens, ou paradigmas, que solucionam o problema da localização. Thrun et al. (THRUN et al., 2000b), citado por Bianchi (BIANCHI, 2002), definem que esses paradigmas podem ser categorizados em quatro dimensões distintas: locais ou globais; para ambientes estáticos ou dinâmicos; passivas ou ativas; um único robô ou múltiplos robôs. A seguir, na Seção 3.1.1.1, as principais características desses paradigmas são descritas.

3.1.1.1 Paradigmas

Segundo a categorização proposta em (THRUN et al., 2000b), os paradigmas de localização podem ser descritos como:

- Métodos Locais Versus Métodos Globais

  Os métodos locais realizam apenas um rastreamento da posição do robô, ou seja, dada uma posição inicial, os métodos locais compensam os erros odométricos[2] que vão ocorrendo enquanto o robô se movimenta e mantém uma estimativa da localização correta do mesmo durante o percurso. Caso o robô venha a perder, em algum instante, sua localização, os métodos locais não conseguem se recuperar dessa falha, uma vez que o erro de posição é gradativamente acumulado durante o percurso de sua trajetória (SOHN; KIM, 2008).

  Os métodos globais podem localizar o robô em seu ambiente partindo da total incerteza, ou seja, não é necessário informar a sua posição inicial e caso sua localização correta seja perdida em algum ponto, os métodos globais podem se recuperar e continuar a estimar a posição correta do robô. Os métodos globais mais robustos podem resolver um tipo de problema ainda maior, conhecido como o *problema do robô seqüestrado*, no qual o robô é transportado de um local para outro sem ser informado dessa manobra. Este problema é considerado mais complexo que a localização global, sob total incerteza, visto que ao invés do robô não possuir nenhuma informação sobre sua posição ele possui uma informação incorreta (BARBOSA, 2006).

- Ambientes Estáticos Versus Dinâmicos

  Ambientes estáticos são aqueles em que o único agente que se move é o próprio robô. Desta forma, todos os outros objetos detectados pelos sensores do robô já estão presentes no mapa do ambiente. Por outro lado, ambientes dinâmicos são

[2] Erros odométricos: erros causados por imperfeições mecânicas que fazem com que os sensores introspectivos (sensores que medem a rotação das rodas) não consigam rastrear com precisão as distâncias percorridas pelo robô.

aqueles que possuem outros elementos móveis (pessoas, animais, outros robôs, etc.), que não estão presentes no mapa.

A grande maioria dos métodos de localização lida apenas com ambientes estáticos, ou ainda com ambientes que possuem poucos agentes dinâmicos, os quais podem ser tratados como ruído. Porém, existem algumas abordagens que usam filtros para eliminar as leituras de sensores que possuem uma probabilidade alta de representarem elementos dinâmicos (FOX et al., 1998a). Deste modo, é possível utilizar métodos de localização para ambientes estáticos mesmo em locais densamente "populosos", como bibliotecas e museus (THRUN et al., 2000a; THRUN et al., 1999a; THRUN et al., 1999b).

- Métodos Passivos Versus Ativos

  A localização passiva se baseia exclusivamente, durante a estimativa da posição do robô, no fluxo de dados provenientes de seus sensores. A movimentação e a direção dos sensores do robô, nesta abordagem, não podem ser controladas pelo processo de localização. Em contrapartida, nos métodos ditos ativos, a função de estimativa de posição tem o controle parcial, ou até mesmo total, do robô, criando dessa forma, uma oportunidade única para a melhoria da eficiência e robustez do método de localização.

- Único Robô Versus Múltiplos Robôs

  A metodologia mais usualmente utilizada é a que envolve apenas um robô. Sua principal vantagem é que todos os dados são coletados em uma única plataforma, e portanto, não existem problemas decorrentes de comunicação. Contudo, em aplicações que envolvam times de robôs, os métodos de localização que se utilizam dos dados sensoriais dos múltiplos robôs oferecem a vantagem de aproveitamento do conhecimento de um robô para influenciar a crença de outro (MURTRA et al., 2008).

Uma outra característica extremamente relevante na localização de robôs móveis é a natureza do ambiente, podendo essa tarefa ser direcionada para ambientes internos (*indoor*) ou para ambientes externos (*outdoor*). Cobano et al. (COBANO et al., 2008) afirmam que tipos diferentes de informação são necessários para cada ambiente. Segundo os autores, para a localização *indoor*, consegue-se criar mapas do ambiente usando apenas as informações

obtidas pelos sensores do robô. Enquanto que na localização *outdoor* isso não é geralmente possível, uma vez que ambientes naturais são usualmente desconhecidos.

Além do que foi discutido até o presente momento, tem-se que o padrão dominante nos diversos paradigmas robóticos é o uso de abordagens probabilísticas, uma vez que elas apresentam um alto nível de desempenho no mundo real (BARBOSA, 2006). A idéia central da abordagem probabilística é representar as informações incertas através de probabilidades. De acordo com Lee e Chung (LEE; CHUNG, 2006), as razões dessas incertezas incluem: erros sistemáticos, incluindo erro de modelagem dos parâmetros cinemáticos; erros não-sistemáticos, incluindo derrapagem das rodas; erros de leituras dos sensores e erros causados por obstáculos dinâmicos e mudanças no ambiente. É importante salientar que os algoritmos probabilísticos alcançaram considerável sucesso quando comparados aos métodos convencionais não-probabilísticos (SIMMONS; KOENIG, 1995; THRUN et al., 2000b).

A técnica de localização utilizada nesse trabalho é derivada do método probabilístico conhecido como Localização de Markov (FOX, 1998; FOX et al., 1998b; FOX et al., 1999). Há várias formas de se implementar este método, sendo que elas diferem entre si nos seguintes pontos: na representação da crença do robô, nos procedimentos de atualização da crença e nos modelos probabilísticos de ação e percepção. Sendo assim, será usada a técnica Localização de Monte Carlo (DELLAERT et al., 1999; THRUN et al., 2001; KÖSE; AKIN, 2007; KOOTSTRA; BOER, 2009), implementada em (BIANCHI, 2002; BARBOSA, 2006).

3.1.1.2 Localização de Markov

Localização de Markov é uma técnica probabilística que mantém múltiplas hipóteses sobre a posição do robô. Cada hipótese possui um peso associado, que representa um fator probabilístico numérico. A idéia principal desta abordagem é calcular uma distribuição de probabilidades sobre todos os possíveis locais do ambiente.

Seja $l = < x, y, \theta >$ uma posição no espaço de estados, onde x e y são coordenadas cartesianas do robô e θ é a sua orientação. Além disso, tem-se também que a distribuição *Bel(l)*, sobre todos os locais l, expressa a crença subjetiva do robô em estar na posição l. Caso a posição inicial seja conhecida, *Bel(l)* é centrada na posição correta, caso contrário, ela é uniformemente distribuída, refletindo a incerteza de sua posição. Desde modo, assim que o robô inicia sua operação, *Bel(l)* é atualizada incrementalmente.

A Localização de Markov aplica dois diferentes modelos probabilísticos para atualizar *Bel(l)*: um modelo de ação, responsável por incorporar os movimentos do robô; e um modelo perceptual, destinado a atualizar a crença do robô baseando-se na entrada sensorial.

A movimentação do robô é modelada pela probabilidade condicional *p(l | l', a)*, que determina a probabilidade de que uma ação *a*, quando executada em *l'*, leve o robô para *l*. *Bel(l)* é então atualizada segundo a teoria das cadeias de Markov, dada por:

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

O termo *p(l | l', a)* representa um modelo da cinemática do robô. Na implementação do sistema proposto em (BIANCHI, 2002; BARBOSA, 2006), foi assumido que os erros de odometria são uniformemente distribuídos.

As leituras dos sensores são integradas à crença *Bel(l)* segundo uma fórmula de atualização Bayesiana. Sendo assim, seja *s* uma leitura de sensor e *p(s | l)* a probabilidade de se perceber *s*, dado que o robô está na posição *l*. Então, *Bel(l)* pode ser atualizada de acordo com a regra:

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

em que α é um fator de normalização que garante que o somatório de *Bel(l)*, para todo *l*, totalize 1.

Deve-se mencionar que ambos os passos de atualização só são aplicáveis se o problema for *Markoviano*, ou seja, se as leituras sensoriais passadas forem condicionalmente independentes de leituras futuras, dada a localização do robô.

Logo, a hipótese de Markov assume que o mundo é estático. Na prática, essa abordagem já foi aplicada com sucesso até mesmo em ambientes que continham elementos dinâmicos, e que desta forma violavam a hipótese de Markov. Contudo, os experimentos reportados em (FOX et al., 1998b) demonstram que ela não se aplica bem a ambientes densamente povoados.

Uma das formas de resolver o problema anterior consiste em aplicar filtros que selecionam quais leituras dos sensores serão utilizadas, efetivamente, na localização. Objetiva-se dessa maneira eliminar as leituras influenciadas por objetos dinâmicos. Fox et al. (FOX et al., 1998a), por exemplo, propõem e descrevem dois filtros, o filtro de entropia e o

filtro de novidade. O filtro de entropia utiliza medidas que só aumentam a certeza da crença atual do robô, enquanto que o filtro de novidade apenas descarta medidas que, quando verificadas em relação ao mapa, sejam menores do que o esperado.

3.1.1.3 Localização de Monte Carlo

O algoritmo Localização de Monte Carlo (MCL – do inglês *Monte Carlo Localization*), demonstrado no Algoritmo 3.1, expressa a crença *Bel(s)* através de um conjunto de amostras *s,* que são associadas a um fator numérico de importância *p*. Esse fator indica a probabilidade da amostra ser relevante para a determinação da posição do robô. Tem-se que a crença inicial pode ser obtida gerando-se N amostras aleatórias da distribuição prévia $P(s_o)$, e atribuindo-se para cada amostra o fator de importância uniforme N-1.

Localização de Monte Carlo*(S,a,o)* – Extraído de: (BIANCHI, 2002)
1: $S' = 0$ 2: $p_{sum} = 0$ 3: **enquanto** $p_{sum} < p_{max}$ **faça** 4: crie uma amostra aleatória $< s, p >$ de S de acordo com $p_1, ..., p_N$ 5: gere um s' aleatório de acordo com $P(s' \mid a,s,m)$ 6: $p' = P(o \mid s',m)$ 7: adicione $< s', p' >$ a S' 8: $p_{sum} = p_{sum} + p'$ 9: **fim-enquanto** 10: normalize os fatores de importância p em S' 11: **retorne** S'

Algoritmo 3.1 – Algoritmo MCL

Leituras de sensores *o* e ações *a* são processadas em pares. Então, o método de Monte Carlo constrói uma nova crença repetindo o seguinte processo: inicialmente, um estado aleatório *s* é gerado a partir da crença atual, considerando-se os fatores de importância. Para esse estado *s*, supõe-se um novo estado s_o segundo o modelo de ação $P(s_o \mid a, s, m)$, em que os dados do mapa estão representados em *m*. O fator de importância para esse novo estado s_o recebe um valor proporcional à consistência perceptual desse estado, como medido por $P(o \mid s, m)$. Em seguida, a nova amostra, juntamente com seu fator de importância, é memorizada e o laço básico é repetido.

Finaliza-se a geração de amostras quando a soma total dos fatores de importância exceder um limite estabelecido, ou quando o próximo par $< o, a >$ é obtido. Por fim, os

fatores de importância são normalizados e o algoritmo de Monte Carlo retorna o novo conjunto de amostras como sendo a crença corrente.

Este método possui a desvantagem de descartar, ocasionalmente, hipóteses corretas, podendo assim não se recuperar de falhas de localização global. Porém, essa deficiência pode ser resolvida adicionando-se amostras aleatórias, matematicamente justificadas, uma vez que o robô pode ser seqüestrado com uma pequena probabilidade.

Por outro lado, ele possui uma série de vantagens: eficiência, pois a computação tem seu foco em regiões de alta probabilidade; robustez a ruídos e a mudanças que afetem a disponibilidade de recursos computacionais; e finalmente, ele pode se adaptar aos recursos computacionais através da alteração do número de amostras utilizadas pelo método, contudo vale ressaltar que a quantidade de amostras afeta diretamente a qualidade da solução obtida.

Atualmente, há uma tendência de se utilizar sensores baseados em visão nos métodos de localização. Segundo Spacek e Burbridge (SPACEK; BURBRIDGE, 2007), entre todos os sensores disponíveis a um robô, a visão provê o maior ganho de informação com a maior precisão, sendo que a visão omnidirecional oferece informações detalhadas de todo o ambiente e é amplamente adequada para o uso na localização de robôs móveis.

A localização de Monte Carlo orientada à visão tem se demonstrado uma técnica bastante promissora (WOLF et al., 2005; MENEGATTI et al., 2006; LINAKER; ISHIKAWA, 2006; ANDREASSON et al., 2007). Segundo esses autores, um dos diferenciais do uso de sensores do tipo visão é a sua passividade, ou seja, eles não interferem na utilização de outros tipos de sensores.

De acordo com Menegatti et al. (MENEGATTI et al., 2006) a combinação de visão omnidirecional com o algoritmo de Monte Carlo providencia um sistema de localização robusto contra oclusões e falhas de localização, capaz de explorar as transições naturais de cores que existem no ambiente.

Por fim, deve-se destacar que uma vez que a Localização de Monte Carlo calcula as probabilidades de todas as amostras aleatórias, o custo computacional do método cresce à medida que o número de amostras também cresce. Nesse caso, um longo período durante a fase de atualização do algoritmo irá provocar uma diferença significativa entre a posição estimada do robô e a sua real posição. Sendo assim, esses erros de localização tornar-se-ão maiores à medida que a velocidade de navegação aumenta, limitando dessa maneira a velocidade do robô ao tempo computacional do método de localização.

3.1.2 Aspectos do Mapeamento

Segundo Thrun (THRUN, 2002), o mapeamento robótico tem sido uma área de pesquisa bastante ativa na robótica, e na IA, nas últimas duas décadas. O mapeamento robótico corresponde ao problema de adquirir um modelo espacial do ambiente físico através de robôs móveis. Assim, o problema do mapeamento é geralmente considerado um dos mais importantes problemas para se construir, verdadeiramente, robôs móveis autônomos.

O problema de adquirir modelos espaciais é difícil e está longe de ser completamente resolvido. Os seguintes fatores impõem limitações práticas na habilidade do robô de aprender e de utilizar precisamente os modelos gerados (THRUN; BÜCKEN, 1996; THRUN, 1998; THRUN, 2002):

- Sensores: freqüentemente os sensores não são capazes de medir diretamente o fator de interesse (tal como a localização exata de obstáculos);
- Limitações perceptuais: o grau de percepção da maioria dos sensores é limitado a uma pequena área em volta do robô. Para adquirir informações globais, o robô precisa explorar ativamente seu ambiente.
- Ruído sensorial: as leituras dos sensores são geralmente distorcidas por ruídos.
- Deslocamento: a movimentação do robô não é precisa, uma vez que os erros odométricos se acumulam com o tempo.
- Complexidade e dinamismo: o ambiente do robô é complexo e dinâmico, tornando impossível manter um modelo espacial exato.
- Requisições em tempo real: o tempo de resposta para requisições, normalmente, requer um modelo simples e de fácil acesso.

Nos últimos anos, vários estudos (TOVAR et al., 2006; GRISETTI et al., 2007; CASTELLANOS et al., 2007; KIM et al., 2008) têm se dedicado a resolver um problema conhecido como Mapeamento e Localização Simultâneos (SLAM – do inglês *Simultaneous Localization and Mapping*). De acordo com Blanco et al. (BLANCO et al., 2008), a formulação do problema SLAM pode ser definida como: construir alguma forma de

representação espacial do mundo por meio de uma seqüência de dados capturados pelo robô, assumindo ainda que não haja nenhuma informação *a priori* do ambiente, e simultaneamente localizar o robô usando essa representação.

A tarefa de mapeamento de ambiente consiste de dois processos distintos: síntese de mapa e exploração. O processo de síntese de mapa é passivo e se caracteriza, dada as observações do ambiente, pela construção de um modelo do mesmo. Enquanto que a exploração é um processo ativo que visa controlar o robô para que ele explore, com seus sensores, todo o ambiente.

O mapeamento de ambientes internos se expressa através de dois grandes paradigmas: o mapeamento métrico (MORAVEC, 1988; ELFES, 1989; BORENSTEIN; KOREN, 1991; PRESTES E SILVA JR. et al. 2004) e o mapeamento topológico (MATARIĆ, 1994; KORTENKAMP; WEYMOUTH, 1994; ROQUE; DOERING, 2005).

3.1.2.1 Paradigmas

De acordo com o apresentado em (THRUN, 2002) paradigmas métricos representam o ambiente através de grades de ocupação espaçadas uniformemente, como ilustrado na Figura 3.1. Cada célula da grade pode indicar a presença ou não de um obstáculo na região correspondente do ambiente.

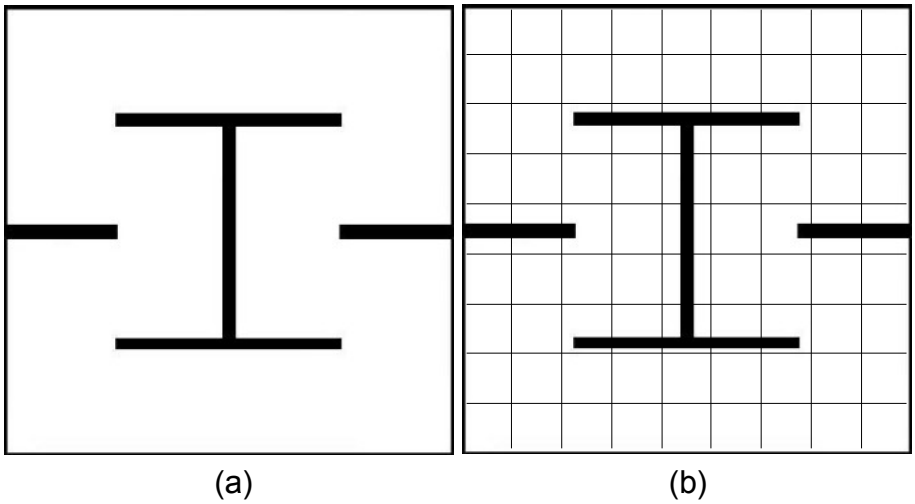


Figura 3.1 - (a) Ambiente não mapeado; (b) Mapeamento métrico

Propostas topológicas representam o ambiente do robô através de grafos, onde os nós desses grafos correspondem a situações, locais ou marcos distintos, como pode ser observado pela Figura 3.2. Caso exista um caminho direto que ligue dois pontos distintos do ambiente ele será representado através de um arco que conectará os dois nós correspondentes no grafo.

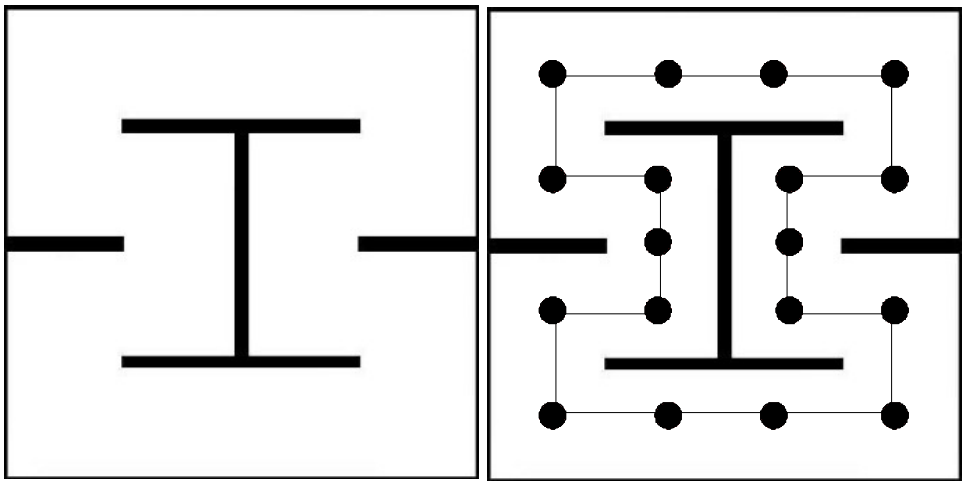


Figura 3.2 - (a) Ambiente não mapeado; (b) Mapeamento topológico

Conforme elucidado por Thrun (THRUN, 2002), os mapas métricos capturam as propriedades geométricas do ambiente, enquanto que os mapas topológicos descrevem a conectividade entre os diferentes locais, ou seja, mapas topológicos representam o ambiente como uma lista de locais significantes que são conectados através de arcos.

Os métodos métricos podem gerar mapas bastante precisos, contudo sua complexidade dificulta, geralmente, um planejamento de trajetórias eficiente. Além de dificultar também a resolução de problemas em grandes ambientes internos. As abordagens topológicas, em contrapartida, podem ser utilizadas eficientemente na determinação de trajetórias. Porém, também se torna difícil manter e gerar mapas topológicos em grandes ambientes, principalmente se os dados sensoriais momentâneos forem substancialmente ambíguos.

Na Seção 3.1.2.2 será apresentado um método de mapeamento métrico probabilístico, conhecido como Grade de Ocupação[3] (MULLANE et al., 2009; NOYKOV; ROUMENIN,

[3] do inglês *Occupancy Grids*

2007; BIRK; CARPIN, 2006; ELFES, 1989), e na seção subseqüente, Seção 3.1.2.3, será apresentado o mapeamento topológico. Segundo Ranganathan et al. (RANGANATHAN et al., 2006) propostas probabilísticas têm obtido grande sucesso ao lidar com as incertezas associadas aos mecanismos de controle (atuadores) e percepção (sensores) do robô, o que do contrário tornaria a construção do mapa do ambiente um processo muito frágil.

3.1.2.2 Mapeamento Métrico

Um dos métodos mais amplamente utilizado para mapeamento métrico é o método Grade de Ocupação proposto por Elfes (ELFES, 1989). O método Grade de Ocupação usa uma representação probabilística e reticulada da informação para modelar o ambiente. A Grade de Ocupação é definida como um campo aleatório multidimensional que contém estimativas estocásticas dos estados das células em um espaço reticulado.

A representação dos mapas na Grade de Ocupação é uma matriz de ocupação multidimensional (normalmente 2D ou 3D), que mapeia o espaço em células, de tal forma que cada célula armazena uma estimativa probabilística de seu estado, sendo os possíveis estados: ocupado ou livre.

As estimativas do estado de cada célula são obtidas através da interpretação das leituras de distâncias utilizando modelos probabilísticos dos sensores. Procedimentos de estimativa Bayesianos permitem a atualização incremental da Grade de Ocupação por meio de leituras realizadas, por vários sensores, sobre múltiplos pontos de vista.

A variável de estado $s(C)$, associada à célula C da Grade de Ocupação, pode ser definida como uma variável aleatória discreta que possui dois estados distintos, ocupada e vazia, denotadas aqui por OCC e EMP. Dessa maneira, a Grade de Ocupação corresponde a um campo aleatório de estados discretos e binários. Uma vez que os estados das células são exclusivos e exaustivos, $P[s(C) = OCC] + P[s(C) = EMP] = 1$.

Para interpretar as medidas de distâncias (dados sensoriais), é usado um modelo estocástico do sensor. Definido por uma função de densidade de probabilidade na forma $p(r \mid z)$ que relaciona a leitura r ao parâmetro de espaço z referente a uma posição do mapa. Esta função de densidade é subsequentemente utilizada em um procedimento de estimativa Bayesiano para determinar as probabilidades de estado das células da Grade de Ocupação.

Partindo-se da estimativa atual do estado da célula C_i, $P[s(C_i) = OCC \mid \{r\}_t]$, baseada nas observações $\{r\}_t = \{r_1,..., r_t\}$ e dada uma nova observação r_{t+1}, a estimativa atualizada é dada por[4]:

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

A partir dessa formulação recursiva, a estimativa anterior do estado da célula, $P[s(C_i) = OCC \mid \{r\}_t]$, serve como priori e é obtida diretamente da Grade de Ocupação. Então, a nova estimativa é armazenada na matriz e desde modo, obtém-se um mapa que expressa as probabilidades de ocupação de cada célula.

3.1.2.3 Mapeamento Topológico

A idéia chave do mapeamento topológico é simples, mas muito efetiva: o espaço-livre do mapa é particionado em um pequeno número de regiões, separadas por "linhas críticas". Essas linhas críticas correspondem a passagens estreitas, tais como as entradas de uma sala. O mapa particionado é então convertido em um grafo isomórfico, de modo que os nós correspondam às regiões do ambiente e os arcos correspondam às conexões entre regiões vizinhas. Este grafo gerado é precisamente o mapa topológico.

Geralmente, o mapa topológico é gerado através de técnicas baseadas em diagramas ou grafos de Voronoi (THRUN 1998; CHOSET; NAGATANI, 2001; BOADA et al., 2004; KALRA et al., 2009). Rawlinson e Jarvis (RAWLINSON; JARVIS, 2008b) afirmam que métodos baseados em diagramas de Voronoi generalizados são os mais usualmente utilizados para a geração de mapas híbridos que contenham informações métricas e topológicas em modelos paralelos. Tipicamente, o diagrama de Voronoi generalizado é computado a partir dos dados globais de uma grade de ocupação, uma vez que a principal vantagem desse método é o planejamento topológico eficiente.

[4] Note que uma célula C_i na Grade de Ocupação corresponde à célula l_i de coordenadas x_i e y_i no espaço de estados descrito na Seção 3.1.1.2. A mudança de notação visa apenas facilitar a formulação das equações elucidadas a partir da atual seção.

3.1.2.4 Grafos de Voronoi

A informação topológica é uma abstração do ambiente em termos de nós representando lugares discretos e linhas conectando essas regiões. O mapa topológico pode ser gerado através de técnicas baseadas em grafos de Voronoi ou por métodos de esqueletização de imagens (KWON et al., 2006). Sendo assim, o conceito geral do algoritmo baseado em grafos de Voronoi segue abaixo (THRUN, 1998; CHOSET; NAGATANI, 2001; BOADA et al., 2004):

1º. *Thresholding*: inicialmente, cada valor na malha de ocupação deve ser comparado com um *threshold*. As células cujo valor de ocupação está abaixo do limiar definido pelo *threshold* serão consideradas como espaço-livre (denotado por C). Todos os demais pontos serão considerados ocupados (denotados por $\overline{C}$);

2º. Construção do Diagrama de Voronoi: para cada ponto no espaço-livre $\langle x, y \rangle \in C$, há um ou mais pontos próximos no espaço ocupado $\overline{C}$. Esses pontos serão denominados pontos-base de $\langle x, y \rangle$, e a distância entre $\langle x, y \rangle$ e os pontos-base será denominada *abertura de* $\langle x, y \rangle$. O Diagrama de Voronoi é o conjunto de pontos no espaço-livre que possui pelo menos dois pontos-base diferentes, separados equidistantemente;

3º. Definição dos pontos críticos: a principal idéia por trás do particionamento do espaço-livre é encontrar os chamados "pontos críticos". Pontos críticos $\langle x, y \rangle$ são pontos no Diagrama de Voronoi que minimizam localmente a abertura. Em outras palavras, cada ponto crítico $\langle x, y \rangle$ possui as seguintes propriedades: primeiro, ele é parte do Diagrama de Voronoi; e segundo, a abertura de todos os pontos em uma vizinhança ε de $\langle x, y \rangle$ não é menor do que a abertura do próprio ponto crítico;

4º. Definição das linhas críticas: linhas críticas são obtidas pela conexão de cada ponto crítico com seus pontos-base. Pontos críticos possuem exatamente dois pontos-base (de outro modo eles não seriam o mínimo local da função de abertura). As linhas críticas particionam o espaço-livre em regiões disjuntas;

5º. Construção do grafo topológico: o particionamento é mapeado em um grafo isomórfico. Sendo que, cada região corresponde a um nó no grafo topológico, e cada linha crítica a um arco.

Segundo Kwon et al. (KWON et al., 2006), os métodos baseados em grafos de Voronoi são robustos para vários ambientes e podem ser estendidos para espaços de maiores dimensões. Contudo, os mapas gerados por esses métodos criam alguns nós e linhas de percurso que são desnecessárias para a navegação. Por outro lado, as técnicas de esqueletização de imagens não geram esse tipo de informação, e ainda assim se mantêm robustas a ruídos sensoriais e a diversos tipos de ambientes. De acordo com esse autor, os métodos de esqueletização de imagens apresentam um melhor desempenho do ponto de vista de eficiência computacional e de capacidade de representação das várias características geométricas do ambiente.

3.1.2.5 Esqueletização de Imagens

Os métodos de esqueletização de imagens são algoritmos utilizados para processamento de imagens, o qual é usado para detectar o "esqueleto" de imagens (BAXES, 1994), como ilustrado pela Figura 3.3. No contexto deste trabalho, o método de esqueletização de imagens será responsável por definir as linhas topológicas de um mapa métrico.

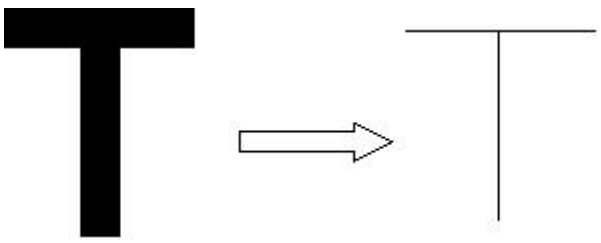


Figura 3.3 – Conceito de esqueletização de imagens

A utilização de linhas topológicas, como percursos de navegação, pode ser justificada em parte pelo campo de atuação dos sensores. Se o robô navegar ao longo de caminhos eqüidistantes dos diversos obstáculos, os quais correspondem às linhas topológicas, ele terá

maior oportunidade de capturar mais informações sobre o ambiente do que se percorresse outros caminhos. Segundo Kwon et al. (KWON et al., 2006) esse aumento de informação pode conduzir a uma melhora no desempenho do MCL.

O princípio geral da construção do mapa topológico baseado em esqueletização de imagens é descrito a seguir. Inicialmente, o robô coleta os dados do ambiente através de seus sensores. A probabilidade de ocupação de cada célula do mapa métrico é então atualizada baseada na fórmula de atualização Bayesiana. A cada amostragem, baseado nos dados capturados, a Grade de Ocupação local e o seu respectivo mapa topológico local são construídos. Assim, o mapa topológico é constantemente modificado à medida que o mapa é atualizado. O método de esqueletização de imagens proposto por Ko et al. (KO et al., 2004) e implementado nesse trabalho é denominado *Thinning*.

3.1.2.6 Descrição do Método *Thinning*

O método *Thinning* considera um conjunto de nove células, como pode ser observado pela Figura 3.4, representadas por valores binários. Se a célula estiver ocupada será representada pelo valor 1, caso contrário, pelo valor 0.

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Figura 3.4 – Célula central p_1 e suas 8 células vizinhas

Considerando a célula central p_1 como referência e assumindo que ela está ocupada, o funcionamento da técnica de esqueletização de imagens pode ser descrito através da verificação de oito condições divididas igualmente em dois passos, sendo esses:

[Passo 1]

1. $2 \leq N(p_1) \leq 6$;
2. $S(p_1) = 1$;
3. $p_2 \cdot p_4 \cdot p_6 = 0$;
4. $p_4 \cdot p_6 \cdot p_8 = 0$.

em que:

$N(p_1)$ é o número de células ocupadas ao redor de p_1 (i.e., $N(p_1) = p_2 + p_3 + p_4 + p_5 + p_6 + p_6 + p_8 + p_9$);

$S(p_1)$ é o número de mudanças de 0 para 1 na seqüência $p_2, p_3, ..., p_9$.

[Passo 2]

5. $2 \leq N(p_1) \leq 6$;
6. $S(p_1) = 1$;
7. $p_2 \cdot p_4 \cdot p_8 = 0$;
8. $p_2 \cdot p_6 \cdot p_8 = 0$.

Se as oito células vizinhas de p_1 satisfazem as condições de um desses passos, então a célula central é convertida em uma célula vazia, pois ela não faz parte do esqueleto da imagem a ser gerada. Quando os passos 1 e 2 são realizados sequencialmente, as células mais externas, correspondentes ao contorno do objeto, são eliminadas (ou seja, convertidas em células vazias). Esse processo é repetido até que restem somente as células ocupadas pertencentes ao esqueleto da imagem. Contudo, como no mapeamento topológico o que se deseja é a linha mediana do espaço livre e não os esqueletos dos objetos (i.e, obstáculos) as células vazias são representadas por "1" e as células ocupadas por "0".

A Figura 3.5 ilustra dois exemplos de uso do método *Thinning*. Os estados das células representadas pela cor cinza são iguais a 1 (EMP) e os estados das células representadas pela cor branca são iguais a 0 (OCC). Considerando novamente a célula C_1 como referência e assumindo que seu estado é igual a 1, de acordo com as oito condições do método tem-se que na Figura 3.5(a) o estado da célula C_1 não será alterado. Uma vez que $N(C_1)$ é menor do que

2, não satisfazendo a primeira condição do Passo 1. Por outro lado, na Figura 3.5(b) o estado da célula C1 será modificado para 0, pois todas as condições do método são satisfeitas.

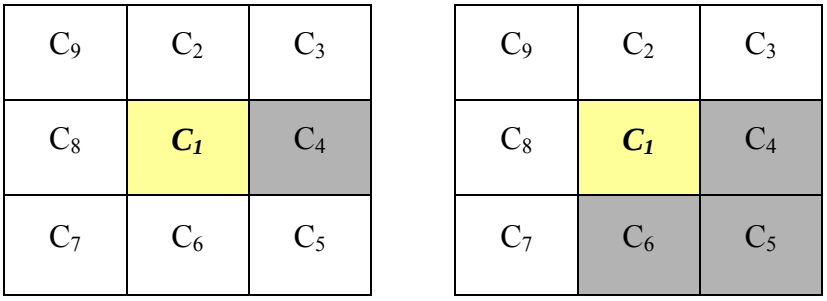


Figura 3.5 – Exemplo do método de esqueletização

O Algoritmo 3.2 abaixo ilustra os princípios gerais do método de esqueletização *Thinning* combinado com a técnica da grade de ocupação. Para realizar a implementação desse método no sistema de navegação autônoma alguns ajustes foram necessárias para adequar o método à estrutura do sistema. Dentre estes ajustes, deve-se mencionar principalmente que as alterações dos estados das células durante o processo de esqueletização se dão através de duas etapas distintas. Em uma primeira etapa, busca-se apenas descobrir quais são as células pertencentes ao mapa métrico que deverão ter seus estados alterados para gerar o mapa topológico. Em seguida, em uma segunda etapa, os estados das células são efetivamente alterados. Isso se torna necessário porque como o processo de esqueletização verifica a vizinhança de todas as células do mapa, a alteração do estado de qualquer célula afeta diretamente a avaliação das células pertencentes a sua vizinhança. Sendo assim, tornou-se necessário realizar as etapas de verificação e atualização em fases completamente disjuntas. A variável C representa a matriz contendo os valores dos estados das células da grade de ocupação.

Método de Esqueletização de Imagens(C)

```
VARIAVÉIS
    i, j, continue, NC, MC: Inteiro;

INÍCIO
REPITA
continue = 0;
```

PARA i = (BordaSuperior+1) **ATÉ** BordaInferior **FAÇA**
PARA j = (BordaEsquerda+1) **ATÉ** BordaDireita **FAÇA**

SE ($s(C_{ij})$) **ENTÃO**
NC = $s(C_{i-1j}) + s(C_{i-1j+1}) + s(C_{ij+1}) + s(C_{i+1j+1}) + s(C_{i+1j}) + s(C_{i+1j-1}) + s(C_{ij-1}) + s(C_{i-1j-1})$;

Calcule a quantidade de mudanças de "0" para "1" ao redor da célula C_{ij} (variável MC);

SE (((NC >=2) **E** (NC <= 6)) **E** (MC == 1) **E** $(s(C_{i-1j})*s(C_{ij+1})*s(C_{i+1j}) == 0)$ **E** $(s(C_{ij+1})*s(C_{i+1j})*s(C_{ij-1}) == 0)$) **ENTÃO**
$s(C_{ij}) = 0$;
continue = 1;
FIM-SE

SE (((NC >=2) **E** (NC <= 6)) **E** (MC == 1) **E** $(s(C_{i-1j})*s(C_{ij+1})*s(C_{ij-1}) == 0)$ **E** $(s(C_{i-1j})*s(C_{i+1j})*s(C_{ij-1}) == 0)$) **ENTÃO**
$s(C_{ij}) = 0$;
continue = 1;
FIM-SE

FIM-SE

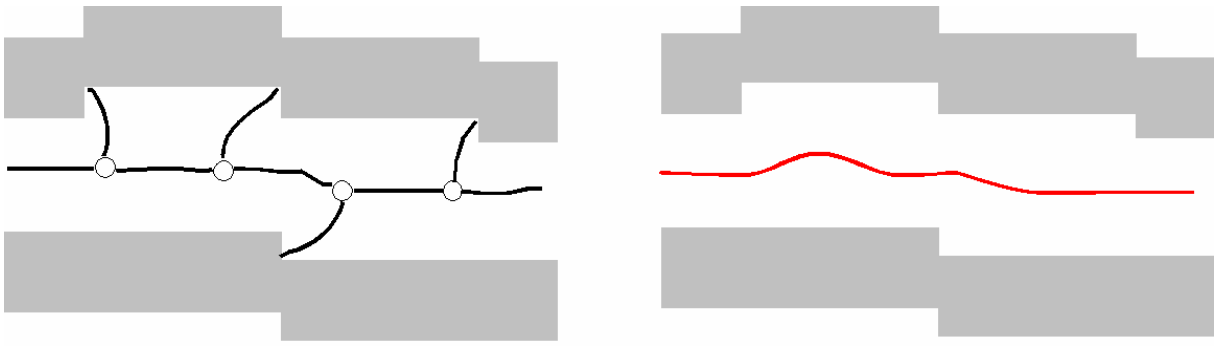
FIM-PARA
FIM-PARA

ENQUANTO (continue == 1);
FIM.

Algoritmo 3.2 – Algoritmo do método de esqueletização

É importante notar que o mapa topológico também pode ser gerado através de técnicas baseadas em grafos de Voronoi. De acordo com Kwon et al. (KWON et al., 2006), os métodos baseados em Grafos de Voronoi Generalizados (GVG) são robustos para vários ambientes e podem ser estendidos a espaços de maiores dimensões. Contudo, os mapas gerados através desses métodos podem gerar linhas que conectam pontos de junção (ou vértices do grafo de Voronoi) a pontos presentes em regiões côncavas do mapa, Figura 3.6(a). Por outro lado, como ilustrado na Figura 3.6(b), as técnicas de esqueletização não geram este tipo de informação. Dessa forma, os métodos de esqueletização são robustos a ruídos sensoriais e podem ser aplicados a diversos tipos de ambiente. Segundo o autor, de modo geral, a técnica de esqueletização de imagens apresenta um desempenho melhor do ponto de vista da

eficiência computacional e da capacidade de representação das diversas características geométricas do ambiente.



(a) (b)

Figura 3.6 – Linhas topológicas (a) baseadas em GVG;

(b) baseadas em esqueletização

Atualmente, modelos híbridos (BLANCO et al., 2009; RIZZINI; CASELLI, 2009; RAWLINSON; JARVIS, 2008a) que combinam as informações obtidas pelo mapeamento métrico e pelo mapeamento topológico têm sido propostos como uma solução promissora para lidar com ambientes reais amplos e complexos. Usualmente, propostas híbridas alocam mapas geométricos locais (adequados para os algoritmos de localização) nos nós de uma representação topológica (adequada para tarefas de planejamento de trajetórias).

3.1.3 Aspectos do Planejamento de Trajetórias

A localização de um robô dentro do ambiente no qual está inserido é realizada pelos módulos de localização e mapeamento. Contudo, uma vez que ele tenha conhecimento de sua atual posição é de fundamental importância que o robô defina quais as ações necessárias para se atingir a posição alvo (BARBOSA, 2006).

A definição dos comandos que serão enviados aos atuadores do robô, para que ele atinja sua meta, é realizada pelo módulo de planejamento de trajetórias. Esta tarefa será

realizada considerando-se a otimização do percurso escolhido e o desvio seguro de possíveis obstáculos.

Murphy (MURPHY, 2000) afirma que os métodos para planejamento de trajetórias podem ser caracterizados através de duas abordagens, o planejamento topológico e o planejamento métrico. O planejamento de trajetórias topológico se utiliza de diretivas sobre marcos presentes no ambiente (e, consequentemente, no mapa), tais como "siga em frente neste corredor" e "vire a direita no primeiro pilar". Sendo assim, a navegação topológica depende exaustivamente da presença de marcos e o planejamento resultante é um percurso que descreve por quais marcos o robô deverá passar. As trajetórias geradas pelo planejamento métrico usualmente são decompostas em submetas, normalmente uma posição fixa ou uma coordenada (x,y), com o objetivo de se obter um percurso ótimo.

A seguir é apresentada uma técnica de planejamento de trajetórias conhecida como *Vector Field Histogram* (BORENSTEIN; KOREN, 1991; KOREN; BORENSTEIN, 1991; ULRICH; BORENSTEIN, 2000).

3.1.3.1 Vector Field Histogram

O método *Vector Field Histogram* mantém um mapa métrico probabilístico simplificado baseado em malhas (similar à Grade de Ocupação), contudo ao invés de utilizar valores reais entre zero e um para representar a probabilidade de ocupação de cada célula, ele utiliza valores inteiros de zero a cinco.

Esse mapa também difere de uma Grade de Ocupação pela maneira como é gerado, pois para operar com a rapidez necessária para que se consiga desviar de obstáculos em velocidades elevadas, a manutenção do mapa não pode gastar muito tempo de processamento. Sendo assim, ele é obtido através de uma técnica chamada amostragem rápida, onde cada célula é inicializada em zero e é incrementada em um a cada vez que um sensor detecta um obstáculo em sua posição, até que ela alcance o valor máximo cinco, quando então não mudará mais seu valor.

A cada iteração do algoritmo, é realizada uma amostragem rápida. Desta forma, um histograma polar é criado, de tal forma que as regiões em torno do robô são divididas em k setores contendo uma densidade de ocupação própria. Para fazê-lo, não se utiliza a totalidade do mapa, mas sim uma janela chamada de "região ativa" cujo centro é o robô. Então, cada

célula c_{ij} da região ativa é analisada, atribuindo-se dois valores a cada uma: primeiro, sua direção em relação ao centro do veículo, dada por:

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

e segundo, sua magnitude, dada por:

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

em que:

a, b	Constantes positivas.
d_{ij}	Distância entre a célula (i,j) e o centro do robô.
c_{ij}	Valor da célula (i,j), em um dado instante.
m_{ij}	Magnitude da célula para a densidade do setor.
x_0, y_0	Coordenadas do centro do robô.
x_i, y_j	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:

$$k = INT(\beta / \alpha) \tag{3.6}$$

A densidade de ocupação h_k de cada setor k do histograma é calculada por:

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

Finalmente, devido à natureza discreta do histograma polar, deve-se utilizar uma função de suavização para melhorar a saída do mesmo, definida por:

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

A partir do histograma criado, torna-se necessário calcular o ângulo Ω entre o robô e a meta. Para isso, escolhe-se o setor *k*, referente ao ângulo mais próximo de Ω, que possua densidade de ocupação menor do que uma constante pré-definida. Desta forma, o ângulo referente a esse setor determina a direção do robô. Enquanto que, a velocidade de translação pode ser calculada através de heurísticas que se utilizam das densidades de ocupação próximas ao ângulo referente ao *k* escolhido.

3.2 FILTROS MORFOLÓGICOS MATEMÁTICOS

A tarefa de adquirir uma representação espacial do ambiente possui algumas limitações práticas herdadas da própria natureza do problema. Dentre estas limitações destaca-se os ruídos oriundos dos sensores acoplados ao robô, influenciando diretamente na obtenção de um modelo espacial exato do ambiente robótico. Sendo assim, nesta seção é apresentado um conjunto de filtros morfológicos matemáticos que visam reduzir a influência de ruídos durante o processo de mapeamento do ambiente.

3.2.1 Aspectos Gerais

O campo da morfologia matemática foi originado do trabalho dos matemáticos J. Serra e G. Matheron em meados de 1960 e tornou-se uma importante base teórica para a criação de ferramentas úteis na área de processamento e análise de imagens. De fato, a morfologia matemática tem sido aplicada a uma ampla variedade de problemas práticos, tais como, por exemplo, pré-processamento de imagens, associação de padrões, detecção e decomposição de formas, e filtragem de ruídos (COSTA; CÉSAR, 2001).

Os algoritmos baseados em morfologia matemática são criados através da combinação de um conjunto de operações simples, permitindo a criação de técnicas capazes de resolver problemas de processamento de imagens complexos, tais como os problemas encontrados na área de robôs móveis autônomos.

No contexto da navegação autônoma de robôs móveis utilizando a técnica Grade de Ocupação para mapeamento métrico, os filtros morfológicos matemáticos podem ser facilmente descritos através da mudança do estado s(C) de alguma célula da grade de ocupação. Sendo que essa mudança leva em consideração determinadas regras que dependem do padrão das células vizinhas. Os filtros utilizados neste trabalho são os filtros de erosão e de abertura. Cada filtro é executado sobre cada célula do mapa, utilizando o padrão original das células.

De acordo com Costa e César (COSTA; CÉSAR, 2001) a erosão morfológica é amplamente aplicada em problemas de processamento de imagens e análise de formas. Segundo o autor embora a erosão possa ser utilizada para a redução de alguns tipos de ruídos, tais como os encontrados em regiões pequenas de imagens, o filtro morfológico de abertura é mais usualmente adotado para este propósito.

O filtro de erosão remove células do espaço livre do mapa ou, em outras palavras, muda o estado de uma célula que é originalmente EMP (*empty*) para OCC (*occupied*). O propósito é remover células livres que não deveriam estar no mapa. O tipo mais simples de erosão é remover (mudar para OCC) qualquer célula que seja vizinha a uma célula pertencente ao espaço ocupado (células que originalmente já são OCC). O filtro de erosão pode ser representado pela Equação 3.9, onde a variável θ representa um valor de limiar que varia de 1 a 8 e o estado inicial da célula C_1 é igual a EMP.

$$s(C_1)_{EMP} = \begin{cases} OCC \rightarrow se\ N(C_1) \geq \theta \\ EMP \rightarrow \text{caso contrário} \end{cases} \tag{3.9}$$

O processo de erosão remove uma camada de células ao redor da periferia de todas as regiões livres do mapa (Figura 3.7), provocando um encolhimento das dimensões dessas regiões. De acordo com a Figura 3.7 as células em branco representam o espaço ocupado e as células em preto representam o espaço livre. As células na cor cinza representam as células que tiveram seus estados alterados devido ao filtro de erosão, ou seja, originalmente essas células eram EMP, mas após a aplicação do filtro de erosão seus estados foram alterados para OCC.

O encolhimento das dimensões das regiões livres pode eventualmente causar a ruptura de uma região em várias outras partes. O filtro de erosão é capaz de remover completamente

células ou linhas estranhas que representem ruídos, desde que estas tenham uma largura máxima de uma ou duas células.

Uma vez que utilizar a erosão para remover linhas de ruídos provoca o encolhimento das regiões do mapa, às vezes é necessário restaurar as dimensões do mapa (tanto quanto possível). Para realizar essa restauração é utilizado o filtro de abertura, que é definido como a combinação do filtro de erosão seguido por um outro operador morfológico matemático chamado dilatação (DOUGHERTY; LOTUFO, 2003).

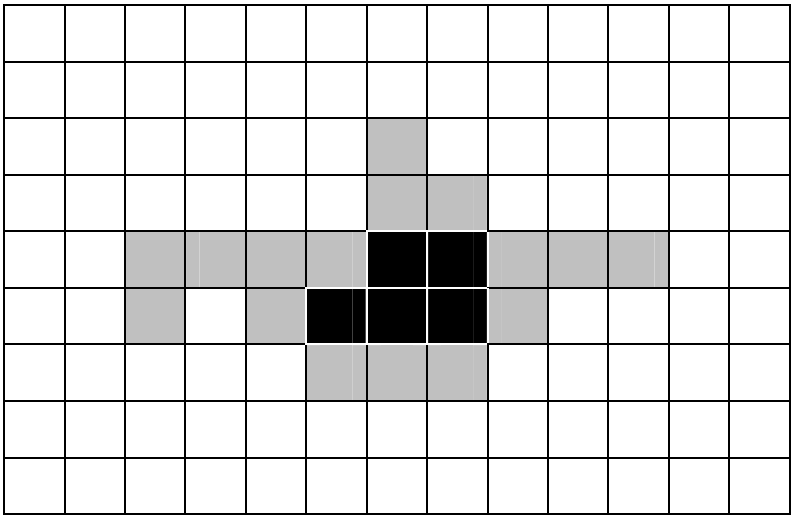


Figura 3.7 – Filtro de erosão

A dilatação é uma operação que expande as regiões de um mapa ou imagem de entrada. A Equação 3.10 ilustra o filtro de dilatação, deve-se notar que a operação (8 – $N(C_1)$) representa o número células com estado OCC, uma vez que $N(C_1)$ é a quantidade de células EMP ao redor de C_1.

$$s(C_1)_{OCC} = \begin{cases} EMP \rightarrow se\ (8\text{-}N(C_1)) \geq \theta \\ OCC \rightarrow \text{caso contrário} \end{cases} \tag{3.10}$$

Como pode ser observado pela Figura 3.8, ao invés de remover células do mapa a dilatação adiciona (altera para EMP) qualquer célula do espaço ocupado que seja vizinha a uma célula pertencente ao espaço livre (células que originalmente já são EMP). As células em branco representam o espaço ocupado e as células em preto representam o espaço livre. As células na cor cinza representam as células que tiveram seus estados alterados, ou seja,

originalmente essas células eram OCC, mas após a aplicação do filtro de erosão seus estados foram alterados para EMP.

O processo de dilatação irá adicionar uma camada de células ao redor da periferia de todas as regiões do mapa, o que irá provocar um crescimento das dimensões das regiões, podendo provocar a união de áreas distintas em uma única região. De modo geral a dilatação também provoca o preenchimento de pequenas lacunas dentro do espaço livre do mapa.

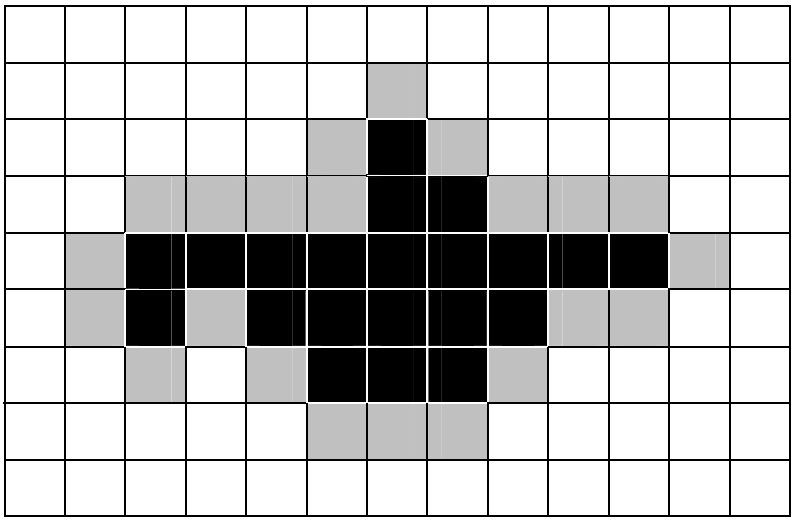


Figura 3.8 – Filtro de dilatação

É importante enfatizar que células e linhas isoladas que foram completamente removidas pelo filtro de abertura não provocam a adição de novas células. Elas são permanentemente apagadas do mapa. Devido a esse fato o filtro de abertura é o operador mais geralmente utilizado para remover ruídos de imagens binárias.

De acordo com Russ (RUSS, 2007) existem vários parâmetros que podem ser utilizados para ajustar os filtros de erosão e de abertura, contudo os principais são o número de iterações e o padrão de vizinhos, ou seja, as regras empregadas para avaliar a adição ou remoção de células no mapa.

Sobre o padrão de vizinhos, os filtros de erosão e abertura clássicos discutidos anteriormente nessa seção alteram o estado de uma célula OCC ou EMP se esta é vizinha de outra célula com estado oposto. Geralmente, a vizinhança nesse contexto corresponde às oito células vizinhas de uma célula central C_1 qualquer, embora alguns sistemas possam lidar com um padrão de vizinhos de quatro células. Outra proposta é contar o número de células vizinhas com estado oposto à célula central, comparar este número com algum valor de limiar, e apenas alterar o estado da célula se o número de vizinhos for maior ou igual ao valor

de limiar. O efeito de diferentes coeficientes para o limiar é a alteração da taxa de crescimento e encolhimento das regiões do mapa e o controle, em certa medida, da isotropia do resultado.

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo, foram abordados os principais aspectos referentes à localização, ao mapeamento de ambientes e ao planejamento de trajetórias de um sistema de navegação autônomo. Na localização de robôs móveis foi discutido o algoritmo conhecido como Localização de Monte Carlo, baseado na técnica Localização de Markov. Em seções subseqüentes, foram tratados os principais conceitos relativos ao mapeamento de ambientes. Apresentando-se tanto técnicas de mapeamento métrico (Grade de Ocupação) quanto de mapeamento topológico (Grafos de Voronoi e esqueletização de imagens). Também foi apresentado o algoritmo *Vector Field Histogram*, responsável por planejar e controlar as possíveis trajetórias que podem ser seguidas pelo robô.

Além disso, no final deste capítulo foram elucidados os principais aspectos sobre os filtros morfológicos matemáticos utilizados nessa dissertação. Os filtros de erosão e de abertura abordados visam à redução da influência de ruídos no desenvolvimento do mapeamento topológico do ambiente. Garantindo, dessa maneira, a obtenção de mapas topológicos de qualidade e que sejam aplicáveis aos mais diversos cenários.

O capítulo seguinte é dedicado a detalhar o estado inicial do sistema de navegação autônoma explorado nesse trabalho. Sendo assim, será apresentada uma descrição aprofundada dos módulos de localização, mapeamento e planejamento de trajetórias que compõem o sistema.

CAPÍTULO

4

O SISTEMA DE NAVEGAÇÃO AUTÔNOMA INICIAL

"Não está na natureza das coisas que o homem realize um descobrimento súbito e inesperado; a ciência avança passo a passo e cada homem depende do trabalho dos seus predecessores".

Sir Ernest Rutherford

Neste capítulo são descritos sumariamente os aspectos gerais do sistema de navegação autônoma para robôs móveis desenvolvido em (BIANCHI, 2002; BARBOSA, 2006). Devido a importância deste sistema para o presente trabalho, será descrito o seu funcionamento e apresentado os módulos que o compõe. No final desse capítulo, Seção 4.5, é realizada uma breve análise sobre esse sistema, implementado originalmente sobre a plataforma do Pioneer 1, quando observado sob a ótica da mudança (do sistema) para o Pioneer 3 – DX.

4.1 DESCRIÇÃO DO SISTEMA

O sistema de navegação autônoma desenvolvido por Bianchi (BIANCHI, 2002) e Barbosa (BARBOSA, 2006) é composto de vários módulos. Uma visão geral da arquitetura desse sistema pode ser vista na Figura 4.1. Para enviar comandos motores ao robô e receber os dados odométricos é utilizada a API do Saphira 6.2, um software de controle para robôs

móveis (KONOLIGE, 1997). Sendo assim, os módulos do sistema são: Hardware Robótico, Interface, Comunicação com o PLS, Gerador de Logs, Localização, Mapeamento e, finalmente, Navegação. Uma breve descrição de cada módulo é descrita a seguir:

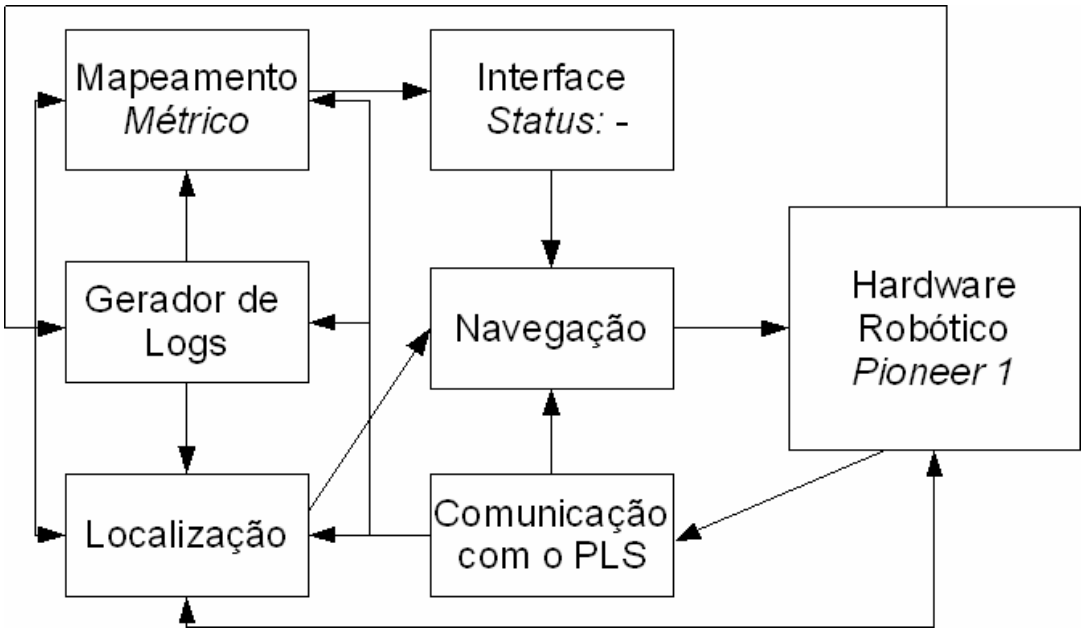


Figura 4.1 – Descrição do sistema (Pioneer I)

- Hardware Robótico: o sistema foi inicialmente implementado sobre a plataforma do robô Pioneer I, mostrado na Figura 4.2. Este robô é dotado de sete sonares, rodas com movimentação por diferencial, rádio para a comunicação serial sem fio e uma garra para capturar objetos. Um sensor de proximidade com medição por laser de alta precisão SICK PLS (*Proximity Laser Scanner*) é utilizado para obter os dados do ambiente. Um notebook é utilizado para comportar os algoritmos e fazer a interface entre o PLS e o robô, controlando o mesmo. Vale lembrar que, atualmente, a plataforma do Pioneer 1 não se encontra completamente em estado viável de operação, devido ao desgaste de seus componentes. Esse desgaste ocasiona vários erros de navegação que se acumulam gradativamente ao longo da trajetória do robô, característica esta que afeta diretamente nas tarefas de localização e planejamento de trajetórias;
- Interface: este módulo faz a interação com o usuário. Ele apresenta ao usuário o mapa do ambiente no qual o usuário poderá marcar os pontos pelos quais o robô deverá passar e pontos onde o robô deverá parar. Nos pontos onde o robô

pára, ele deve esperar até que uma tecla seja pressionada, ou que um determinado tempo limite se esgote para que ele prossiga para o próximo ponto. Contudo, deve-se destacar novamente, que inicialmente o módulo de interface Web do sistema não se encontrava operacional, uma vez que esse módulo apresentava diversos problemas causados pela mudança de versão da plataforma de desenvolvimento da Interface.

- Comunicação com PLS: este módulo faz a comunicação entre o notebook e o PLS por meio de um protocolo definido sobre uma conexão serial do tipo RS-232. Através deste módulo, os outros módulos têm acesso às leituras das distâncias do ambiente;
- Gerador de Log: este módulo gera um arquivo contendo registros das medidas do PLS, a data e hora em que foram capturadas e a informação odométrica do robô. Este arquivo de log é utilizado para realizar experimentos comparativos de mapeamento e localização em casos em que se deseja testar vários parâmetros dos algoritmos em condições idênticas, realizando vários experimentos com o mesmo log;
- Localização: usando o algoritmo de localização Monte Carlo descrito na Seção 3.1.1.3, este módulo faz o rastreamento da posição do robô relativa a seu ambiente;
- Mapeamento: este módulo é o responsável por gerar mapas métricos de ambientes. A abordagem de mapeamento implementada integra o método de mapeamento Grade de Ocupação, descrito na Seção 3.1.2.2 com o método Localização Monte Carlo. Nessa abordagem, o mapeamento é realizado de modo incremental, alternando-se ciclos de mapeamento com ciclos de localização. Inicialmente, o robô executa um ciclo de mapeamento, gerando um pequeno trecho do mapa e o robô se desloca a uma distância pequena determinada. Então, o robô executa um ciclo de localização e com base na amostra de maior consistência perceptual com o trecho de mapa previamente gerado, o mapa é atualizado, incluindo mais um trecho. Então, o robô se desloca mais uma vez e o ciclo se repete continuamente até que o ambiente todo esteja mapeado;
- Navegação: com base nos dados recebidos do PLS e da posição do robô fornecida pelo módulo de localização, este módulo emite comandos de controle ao robô que levam o mesmo aos pontos definidos pelo usuário no

módulo de interface, desviando de obstáculos utilizando o método *Vector Field Histogram* descrito na Seção 3.1.3.1.

4.2 PIONEER I

O robô Pionner I é uma plataforma móvel montada sobre um eixo de duas rodas (Figura 4.2), permitindo rotações e movimentos nas direções para frente e para trás. O robô se locomove através de dois motores com movimentação por diferencial, sendo que a velocidade máxima alcançada pelo mesmo é de 60 centímetros por segundo.

Figura 4.2 – Pioneer I

O Pioneer I possui sete sonares ultra-sônicos, cinco deles estão localizados na parte frontal do robô, um na lateral esquerda e outro na lateral direita. O tempo necessário para qualquer um dos sonares realizar uma leitura é de aproximadamente 40 milisegundos. Os sonares freqüentemente recuperam valores que não são precisos, ou seja, retornam medidas que não denotam a distância correta. Os sonares quantificam o tempo decorrido entre a emissão e recepção de uma onda sonora. Quando a onda sonora atinge frontalmente um obstáculo muito provavelmente ela retornará na direção dos sonares. Contudo, quando a onda sonora atinge um obstáculo por meio de um ângulo desfavorável, ela provavelmente será refletida para longe dos sensores e, conseqüentemente, não poderá ser detectada.

Para solucionar este problema utilizou-se um sensor laser denominado *Proximity Laser Scanner* (PLS), ilustrado na Figura 4.3. O sensor pode rastrear o ambiente através de um ângulo de visão de 180 graus, resolução angular de 0,5 graus, distância máxima de 50 metros e erro de 131 milímetros para distâncias menores do que quatro metros.

Figura 4.3 – Laser PLS

Por fim, tem-se que o software responsável pelo controle do robô móvel é executado *off-board* e é repassado para o Pioneer I de forma remota, por meio de um *link* de rádio de 9,600 bps.

4.3 SAPHIRA

O ambiente Saphira, desenvolvido pela MobileRobots Inc., é uma plataforma voltada exclusivamente para o desenvolvimento de aplicações robóticas. Projetado e mantido pelo Artificial Intelligence Center do Stanford Research Institute, o ambiente dispõe de uma biblioteca de funções que visa à construção de programas em C/C++ para o controle de robôs móveis. A ferramenta é compatível com Microsoft Windows 95/NT, FreBSD, Linux e Unix, permitindo uma abstração de alto nível do hardware robótico. Sendo assim, a plataforma provê informações sensoriais, como leituras de sonares, e recebe comandos de manipulação dos movimentos do robô, como por exemplo, “gire 15 graus para a direita”.

Além disso, o ambiente Saphira também possui um simulador para robôs Pioneer I, denominado SRIsim, no qual é possível monitorar ambientes estáticos e acompanhar os

movimentos realizados pelo robô. O SRIsim é um ambiente de simulação especializado para serviços da MobileRobots.

4.4 INTERFACE WEB

A interface Web do sistema desenvolvida em (BARBOSA, 2006), Figura 4.4, é constituída de três módulos implementados em uma *applet* da linguagem Java. Sendo que, *applet* é um programa que é executado por um *browser* quando é carregada a página da Internet que o contém.

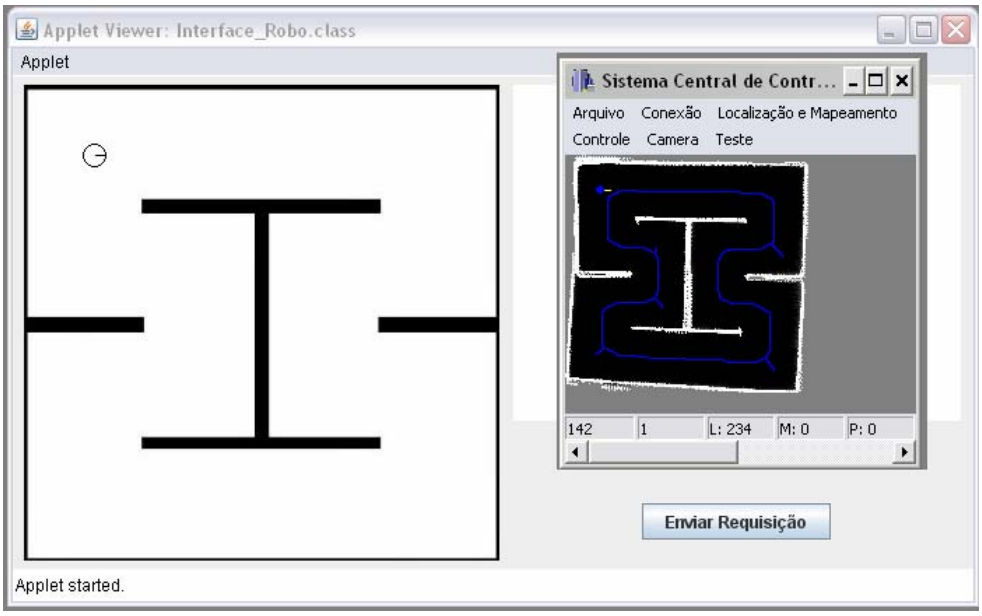


Figura 4.4 – Interface Web

Os módulos da interface são: módulo de comunicação, módulo de controle e módulo de visualização. Cada um desses módulos é descrito a seguir:

- Módulo de Comunicação: este é o módulo responsável pela comunicação (via Internet) entre a interface Web e o servidor de controle do robô. Todas as informações que são necessárias para o funcionamento da interface são enviadas e recebidas por esse módulo;
- Módulo de Controle: é o modulo responsável por receber as solicitações dos usuários. Atualmente, os usuários podem apenas enviar solicitações de mudança posicional do robô (por exemplo, “vá para o ponto x do mapa”);

- Módulo de Visualização: este módulo atualiza a posição do robô no mapa da interface. Ele recebe as coordenadas (x, y, θ) do posicionamento do robô, onde (x, y) são as coordenadas do robô no plano e θ é o ângulo de orientação do mesmo, e então realiza a atualização do desenho da trajetória do robô no mapa da interface;

Como pode ser observado pela Figura 4.5 o funcionamento geral do sistema é descrito através das seguintes diretrizes. Inicialmente, o usuário envia uma solicitação de tarefa para o robô por meio da interface Web. O sistema recebe essa requisição e decide qual a ação necessária para que o robô consiga executar a tarefa. Em seguida, a ação a ser realizada é enviada para o Pioneer 1 via radio modem. O robô então executa as ações determinadas e, finalmente, envia as novas informações sobre sua posição para a interface Web.

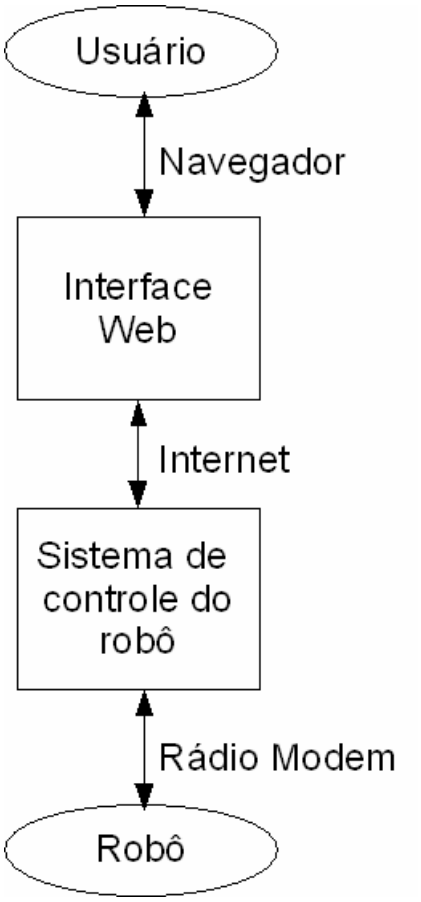


Figura 4.5 – Funcionamento geral do sistema

A comunicação entre processos de software é uma tarefa indispensável, uma vez que os sistemas estão mais complexos e interligados com outros sistemas. O mecanismo mais utilizado para possibilitar a troca de informação entre aplicações é chamado de socket. Na

Seção 4.4.1 são apresentados os conceitos gerais sobre como o socket pode ser utilizado para fazer a comunicação entre a interface Web e o sistema de controle do robô, conforme elucidado por Barbosa (BARBOSA, 2006).

4.4.1 Sockets – Linguagem Java

A linguagem Java oferece dois modos de conexão através de sockets que funcionam sobre o protocolo IP (*Internet Protocol*), o modo orientado à conexão e o modo orientado à datagrama.

O modo orientado à conexão funciona sobre o protocolo TCP (*Transmission Control Protocol*) e oferece serviços confiáveis, ou em outras palavras, suporta serviços sem perdas de dados na rede e garante a ordenação dos pacotes. Além disso, oferece a possibilidade de utilização de fluxos de dados (*DataStreams*). Contudo, uma desvantagem do modo TCP/IP é que esse protocolo é mais lento, devido às verificações de dados realizadas, e o comportamento do servidor é diferente do comportamento do cliente.

O modo orientado a datagrama funciona sobre o protocolo UDP (*User Datagram Protocol*) e oferece serviços não confiáveis, ou seja, durante a transmissão as mensagens podem ser perdidas e não há garantias quanto à ordenação dos pacotes. No caso do UDP/IP cada mensagem é tratada independentemente e sua principal vantagem é a velocidade. Geralmente esse protocolo é empregado em aplicações onde a aquisição de dados atrasados possui um resultado mais negativo do que a obtenção de dados incorretos, como exemplo dessas aplicações pode-se citar a tarefa de transmissão de vídeo ao vivo através da Web.

A interface Web desenvolvida por Barbosa (BARBOSA, 2006) se comunica com o sistema central de controle do robô pelo protocolo TCP/IP, uma vez que os dados transmitidos entre o sistema de controle do robô e a interface Web não podem ser perdidos. Basicamente, os dados que são enviados entre essas duas aplicações são os pontos pelos quais o robô poderá se locomover e as requisições enviadas pelo usuário. No Algoritmo 4.1**,** é apresentado um exemplo de conexão TCP/IP baseado no cliente utilizando sockets da linguagem Java.

Cliente Java – TCP

```
#1 import java.io.*;
#2 import java.net.*;
#3
#4 class ClienteTCP{
#5
#6    public static void main (String argv[]) throws Exception
#7    {
#8        String frase;
#9        String fraseModificada;
#10       BufferedReader doUsuario =
#11          new BufferedReader(new InputStreamReader(System.in));
#12       Socket socketCliente = new Socket("nomeHosp", 6789);
#13       DataOutputStream paraServidor =
#14          new DataOutputStream(socketCliente.getOutput());
#15       BufferedReader doServidor =
#16          new BufferedReader(new InputStreamReader(socketCliente.getInputStream()));
#17       frase = doUsuario.readLine();
#18        paraServidor.writeBytes(frase + '\n');
#19        fraseModificada = doServidor.readLine();
#20        System.out.println("Do Servidor: " + fraseModificada);
#21        socketCliente.close();
#22    }
#23 }
```

Algoritmo 4.1 – Exemplo de conexão TCP/IP – Cliente

Algumas características importantes do Algoritmo 4.1 são descritos a seguir de acordo com o respectivo número de linha. Em seguida, no Algoritmo 4.2, é apresentado um outro exemplo de conexão TCP/IP, mas desta vez baseado no servidor.

#1 – Contém a classe para Streams I/O.

#2 – Contém a classe para suporte a rede.

#10, #11 – Cria fluxo de entrada.

#12 – Cria o socket do cliente, conexão ao servidor.

#13, #14 – Cria fluxo de saída ligado ao socket.

#15, #16 – Cria fluxo de entrada ligado ao socket.

#18 – Envia linha ao servidor.

#19 – Lê linha do servidor.

Servidor Java – TCP

```
#1 import java.io.*;
#2 import java.net.*;
#3
#4 class ServidorTCP{
#5    public static void main (String argv[]) throws Exception
#6    {
#7        String fraseCliente;
#8        String fraseMaisculas;
#9        ServerSocket socketRecepcao = new ServerSocket(6789);
#10       while(true){
#11          Socket socketConexao = socketRecepcao.accept();
#12          BufferedReader doCliente =  new
#13            BufferedReader(new InputStreamReader(socketConexao.getInputStream()));
#14          DataOutputStream paraCliente =
#15            new DataOutputStream(socketConexao.getOutputStream());
#16          fraseCliente = doCliente.readLine();
#17          fraseMaisculas = fraseCliente.toUpperCase() + '\n';
#18           paraCliente.writeBytes(fraseMaisculas);
#19       }
#20   }
#21 }
```

Algoritmo 4.2 – Exemplo de conexão TCP/IP – Servidor

Algumas características importantes do Algoritmo 4.2 são descritos a seguir de acordo com o respectivo número de linha.

#1 – Contém a classe para Streams I/O.

#2 – Contém a classe para suporte a rede.

#9 – Cria socket para recepção na porta 6789.

#11 – Aguarda, no socket para recepção, o contato do cliente.

#12, #13 – Cria fluxo de entrada ligado ao socket.

#14, #15 – Cria fluxo de saída ligado ao socket.

#16 – Lê linha do socket.

#18 – Escreve linha para o socket.

#19 – Final do laço while. Volta ao início e aguarda conexão de outro cliente,

É importante salientar que, nesse trabalho, além da tarefa de restauração da interface Web descrita na Seção 4.4 e implementada por Barbosa (BARBOSA, 2006), também buscou-se projetar e desenvolver uma nova interface Web com novas funcionalidades e que atendesse satisfatoriamente as recentes características do sistema de navegação, como por exemplo, o mapeamento topológico. Na Figura 4.6, está ilustrada a nova interface Web.

De modo geral, os novos recursos incluídos nesta nova interface Web estão descritos a seguir. Inicialmente, foram inseridas algumas informações sobre o estado do sistema ao longo de sua execução, como a velocidade e o ângulo em que se encontra o robô, além disso, também foi incluído o campo *status*, responsável por indicar qual a requisição que está sendo executada pelo robô naquele exato momento. Tem-se ainda a inclusão da característica *zoom*, possibilitando uma melhor visualização do mapa do ambiente na interface Web. Por fim, outro importante recurso implementado foi a opção “Visualizar pontos de Alcance”. A partir do momento que o mapa topológico do ambiente é definido no sistema central de controle do robô, o usuário pode solicitar ao sistema que sejam gerados os pontos de alcance. Os pontos de alcance são definidos e utilizados pelo sistema central de controle para efetuar as requisições de percurso de um usuário qualquer.

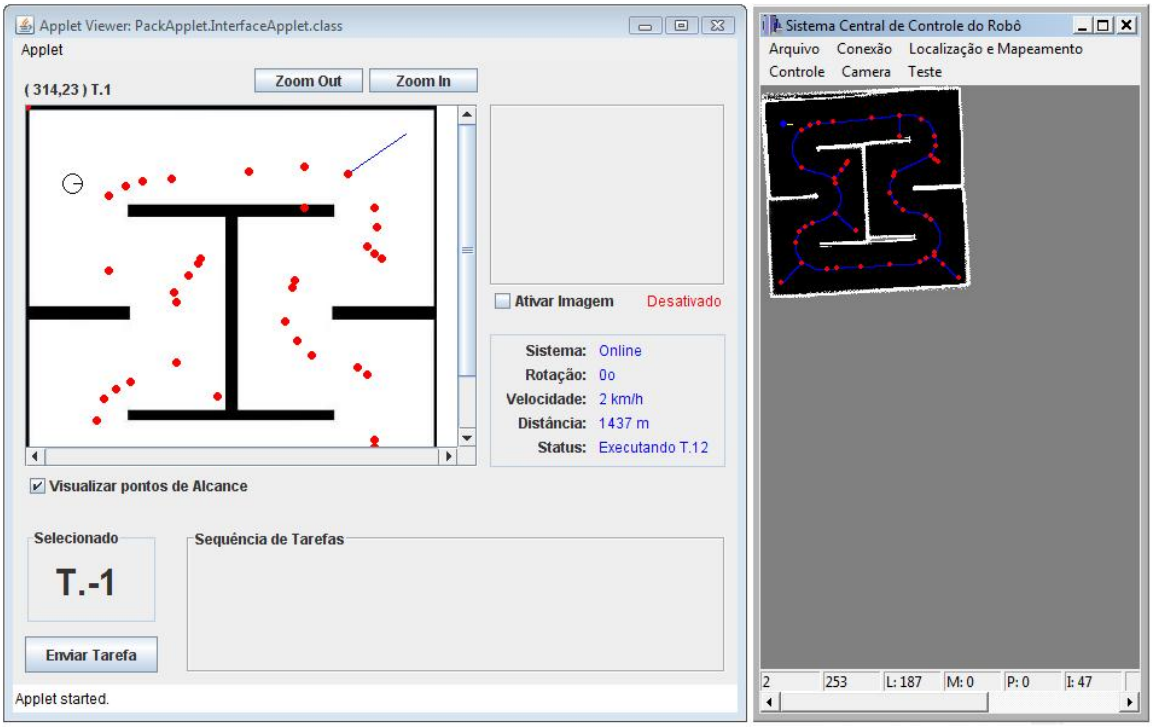


Figura 4.6 – Nova interface Web

Os pontos de alcance são utilizados pelo sistema central de controle para calcular qual é entre esses pontos o mais próximo ao ponto solicitado pelo usuário. Uma vez que esse ponto de alcance é definido, o sistema guiará o robô até esse ponto e não até a exata posição solicitada pelo usuário, limitando consequentemente a locomoção do robô apenas as linhas encontradas pelo mapeamento topológico. Assim, através da opção "Visualizar pontos de alcance" esses pontos também se tornam visíveis na interface Web, oferecendo ao usuário uma percepção mais clara e consistente das requisições a serem enviadas ao robô.

4.5 ANÁLISE DE UMA NOVA PLATAFORMA

Essa seção dedica-se a analisar sobre alguns importantes aspectos referentes a mudança do sistema desenvolvido na plataforma do Pioneer 1 para o robô Pioneer 3 – DX. Sendo assim, inicialmente, são descritas as características do novo Pioneer e ao longo da seção são apresentadas algumas considerações sobre como essa mudança afeta o sistema geral.

O robô Pionner 3 – DX, assim como o Pioneer 1, é uma plataforma móvel montada sobre um eixo de duas rodas (Figura 4.7), permitindo rotações e movimentos nas direções para frente e para trás. O robô se locomove através de dois motores com movimentação por diferencial, podendo carregar até 23 Kg.

Figura 4.7 – Pioneer 3 – DX

Os sonares estão posicionados um em cada lado do robô, e seis na face frontal. O tempo necessário para qualquer um dos sonares realizar uma leitura é de aproximadamente 40 milisegundos. O alcance de sensibilidade de cada sonar varia de 10 centímetros até cinco metros.

Esta plataforma foi idealizada para ser totalmente compatível com as suas versões anteriores, incluindo-se neste caso o Pioneer 1, bem como o seu software de controle de robôs móveis - ARCOS (*Advanced Robot Control & Operations Software*) - também foi projetado para suportar e executar os programas clientes desenvolvidos para as plataformas do Pioneer 1 e 3. Ambas as plataformas, Pioneer 1 e 3, utilizam uma arquitetura cliente-servidor para o controle dos robôs móveis. Nessa arquitetura, as aplicações servidoras (ARCOS no Pioneer 3 – DX e PSOS[5] no Pioneer 1) são responsáveis por gerenciar todos os detalhes de baixo nível do sistema de navegação autônoma, incluindo a operação dos motores, ativação dos sonares, aquisição das leituras sensoriais, controle das rodas e realização dos comandos enviados por uma aplicação cliente separada (sistema proposto nesse trabalho).

[5] Pioneer Server Operating System

Com esta arquitetura cliente/servidor, não é necessário conhecer os detalhes de operação de uma plataforma robótica em particular. Tem-se também que o software ARCOS é compatível com a plataforma do Pioneer 1, implementando os mesmos comandos utilizados na versão original do sistema. Além disso, o ARCOS oferece novas funcionalidades para o sistema, melhoria no desempenho da plataforma e fornece informações adicionais sobre o estado do robô.

Uma importante limitação do sistema como um todo, encontrada no início do presente trabalho, está relacionada aos aplicativos necessários para o seu funcionamento. De modo geral, dois aplicativos essenciais (o ambiente Saphira e o software responsável pela configuração do laser) possuem suas execuções limitadas ao notebook original dos trabalhos de (BIANCHI, 2002; BARBOSA, 2006). Contudo esse notebook não suporta as necessidades atuais dos projetos desenvolvidos para o sistema de navegação autônoma, restringindo inclusive o uso do sistema ao robô Pioneer 1. Portanto, para se realizar a adaptação do sistema para a plataforma do Pioneer 3 – DX é necessário, primeiramente, migrar os aplicativos mencionados anteriormente para computadores mais atuais, que possam suportar as novas exigências do sistema.

4.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma descrição geral do atual estado do sistema de navegação autônoma para robôs móveis construído no ICMC/USP. Cada um dos módulos que compõem o sistema foi sumariamente elucidado, assim como o robô e os softwares utilizados foram detalhadamente descritos. Além disso, foi realizada uma breve análise entre as plataformas do Pioneer 1 e Pioneer 3 – DX.

No próximo capítulo serão discutidos os principais resultados obtidos com o presente projeto de Mestrado. Portanto, serão descritas as modificações realizadas nos módulos do sistema, assim como dar-se-á uma visão geral das características do mapeamento topológico realizado pelo sistema.

CAPÍTULO

5

RESULTADOS E DISCUSSÃO

"Não deves acreditar nas respostas.

As respostas são muitas e a tua pergunta é única e insubstituível".

Mário Quintana

Neste capítulo são apresentados os resultados obtidos neste projeto de Mestrado. São abordadas as seguintes temáticas: metodologia de trabalho, descrição da nova arquitetura do sistema, resultados e discussão sobre o mapeamento topológico, e por fim, resultados e discussão sobre a aplicação dos filtros morfológicos matemáticos.

5.1 METODOLOGIA

A implementação do sistema proposto foi desenvolvida na linguagem C/C++ utilizando a IDE de desenvolvimento Borland C++ Builder 5.0. Os experimentos foram executados tanto em ambientes simulados quanto em ambientes reais. Os testes em ambientes reais foram realizados no robô Pioneer 1 da MobileRobots Inc, existente no Laboratório de Aprendizado de Robôs do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo. O sistema sensorial empregado no robô consiste em um sensor laser PLS SICK instalado na face frontal do robô.

Os experimentos em ambientes simulados foram realizados no simulador SRIsim, o qual é um ambiente de simulação especializado para serviços da MobileRobots. No SRIsim, os ambientes podem ser definidos por características geométricas. De modo geral, o funcionamento desse simulador é descrito a seguir. Uma vez que as características geométricas de um ambiente sejam definidas, o simulador envia todos os sinais sensoriais correspondentes a cada situação do robô simulado nesse ambiente. Para minimizar a distância entre simulação e realidade, ruídos artificiais são incorporados aos sinais sensoriais.

A implementação das técnicas de mapeamento e dos filtros morfológicos foi desenvolvida através de três ciclos distintos e bem definidos. Primeiramente, os estados das células da grade de ocupação são adequadamente calculados baseados nas leituras sensoriais obtidas pelo robô. Em seguida, os filtros morfológicos matemáticos são executados sobre o mapa métrico obtido. Finalmente, quando essa tarefa é concluída, a técnica de esqueletização de imagens é realizada, gerando-se assim as linhas topológicas. Desse modo, a funcionalidade do sistema proposto é baseada em ciclos de atualização da grade de ocupação alternados com ciclos de definição das linhas topológicas.

5.2 DESCRIÇÃO DO NOVO SISTEMA

Uma visão geral da arquitetura do novo sistema pode ser vista na Figura 5.1. Considerando-se a tarefa de adaptação do sistema os seguintes módulos foram estudados e atualizados: Hardware Robótico e Interface. Em contrapartida, considerando-se o processo de ampliação, tem-se que o módulo de Mapeamento foi devidamente explorado e modificado. As modificações e/ou atualizações de cada módulo estão sintetizadas a seguir:

- Hardware Robótico: Como discutido no Capítulo 4, a adaptação do sistema para a plataforma do Pioneer 3 – DX exige a migração de dois importantes aplicativos para computadores mais atuais, sendo eles o Saphira e o software de configuração do laser. Até o presente momento, esses dois softwares foram migrados com êxito para novos computadores. Contudo analisando-se os arquivos de configuração da versão do ambiente Saphira utilizado até então (versão 6.2) e de acordo com os testes realizados, essa aplicação não foi totalmente projetada para controlar o Pioneer 3 – DX, levando-se em

consideração que ela foi desenvolvida em um momento anterior a criação do Pioneer 3 – DX pela empresa fabricante do robô. Sendo assim, o Pioneer 3 não responde adequadamente a todos os comandos motores enviados pelo Saphira 6.2. Considerando-se esta limitação inesperada, uma vez que a empresa fabricante alega total compatibilidade entre seus produtos, atualmente está sendo estudada a substituição desse ambiente no sistema por uma versão mais recente do mesmo aplicativo, Saphira versão 8.4.1. De acordo com os arquivos de configuração da versão 8.4.1, o novo Saphira deverá ser capaz de controlar adequadamente o Pioneer 3, uma vez que os parâmetros de configuração desse robô estão presentes nessa versão do Saphira. Outra alternativa visualizada para lidar com essa limitação é a substituição do Saphira 6.2 pelo ambiente de controle ARIA. Embora o ARIA[6] seja um ambiente diferente, ele é mais robusto, além de ser o aplicativo de controle mais recente fornecido pela empresa fabricante do robô. É importante salientar que o sistema desenvolvido através do presente trabalho permanece compatível com a plataforma do robô Pioneer 1, de forma que todos os testes de mapeamento realizados em ambientes reais foram realizados através dessa plataforma;

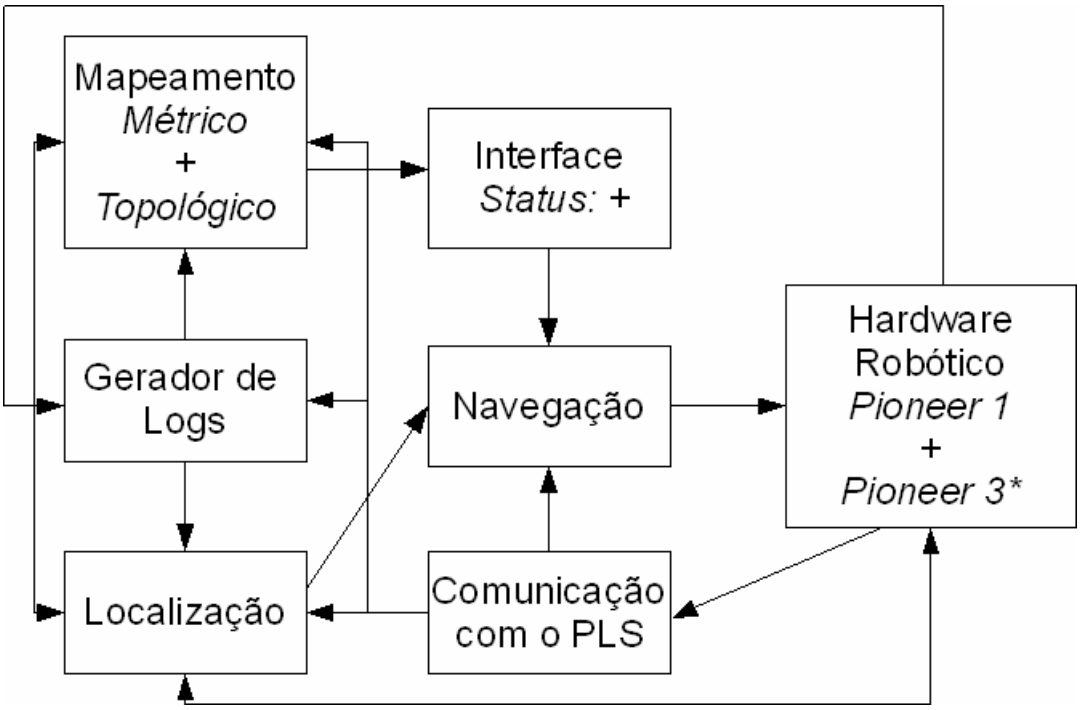


Figura 5.1 - Descrição das modificações do sistema

[6] O ambiente ARIA é a evolução do Saphira, ambos foram projetados pela mesma empresa visando controlar os robôs da linha Pioneer.

- Interface: este módulo foi adequadamente revitalizado. O módulo de Interface faz a interação com o usuário e apresenta ao mesmo o mapa do ambiente, permitindo ao usuário marcar os pontos onde o robô deverá parar. O funcionamento exato da interface está descrito na Seção 4.4. Como salientado anteriormente, o robô Pioneer 3 – DX ainda não está em operação, sendo assim os testes de viabilização da interface foram realizados apenas através do simulador SRIsim. Para os testes realizados, uma máquina cliente enviou ao servidor, responsável pelo simulador robótico, uma solicitação de percurso em um determinado mapa e o simulador foi responsável por controlar toda a movimentação do robô nesse mapa;
- Mapeamento: o módulo de Mapeamento foi modificado de forma a incorporar o mapeamento topológico à abordagem métrica implementada em trabalhos anteriores. O processo completo de mapeamento é descrito a seguir. Inicialmente, um mapa de estados é definido pela técnica métrica Grade de Ocupação, em seguida filtros morfológicos de erosão e abertura são aplicados sobre o mapa obtido e, por fim, o mapa topológico é gerado através da aplicação de um método de esqueletização de imagens.

5.3 INTERFACE WEB

Para validar a comunicação entre o sistema central de controle do robô e a interface Web foram realizados alguns testes para verificar se o robô conseguiria realizar as tarefas com os dados enviados pelo usuário por meio da interface. O comportamento do robô nesses testes foi simulado utilizando o SRIsim do ambiente Saphira.

Os testes foram realizados do seguinte modo. Como pode ser observado na Figura 5.2[7], inicialmente, o sistema central de controle do robô é executado em uma máquina qualquer, gerando-se assim o mapa híbrido do ambiente simulado. Em seguida, a interface Web é aberta em um computador diferente daquele que possui o sistema de controle central do robô.

[7] Deve-se salientar que nas figuras dessa seção as janelas da applet da interface e do sistema central de controle foram agrupadas em uma única imagem apenas para facilitar a visualização, uma vez que ambos programas são executados em computadores diferentes.

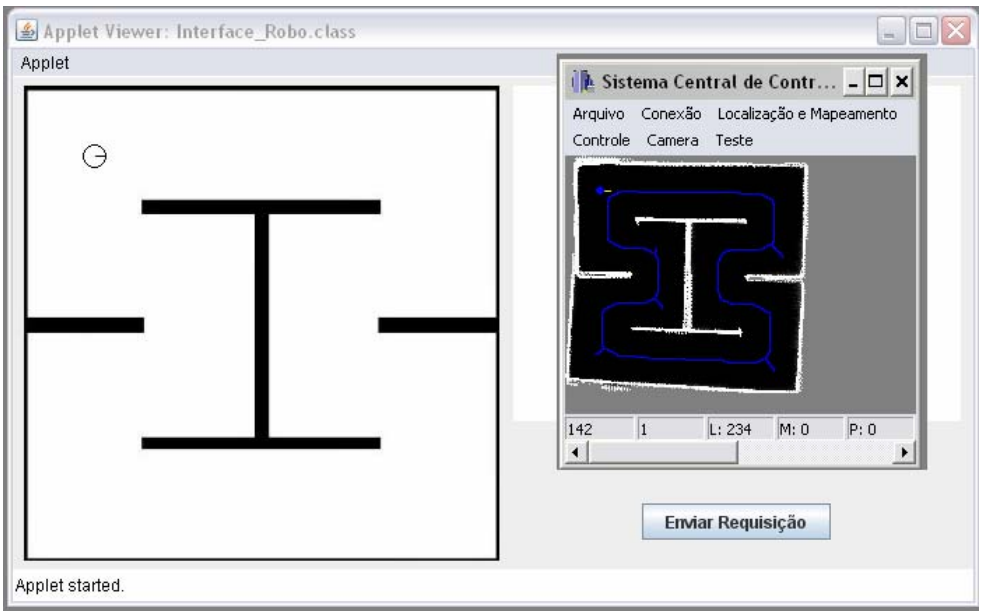


Figura 5.2 – Interface Web fase 1

Em seguida, de acordo com a Figura 5.3, o usuário define, através da interface Web, algum ponto no mapa para o qual ele deseja que o robô se locomova. Por exemplo, o ponto vermelho na Figura 5.3. O usuário então envia essa requisição de percurso para o sistema central de controle do robô.

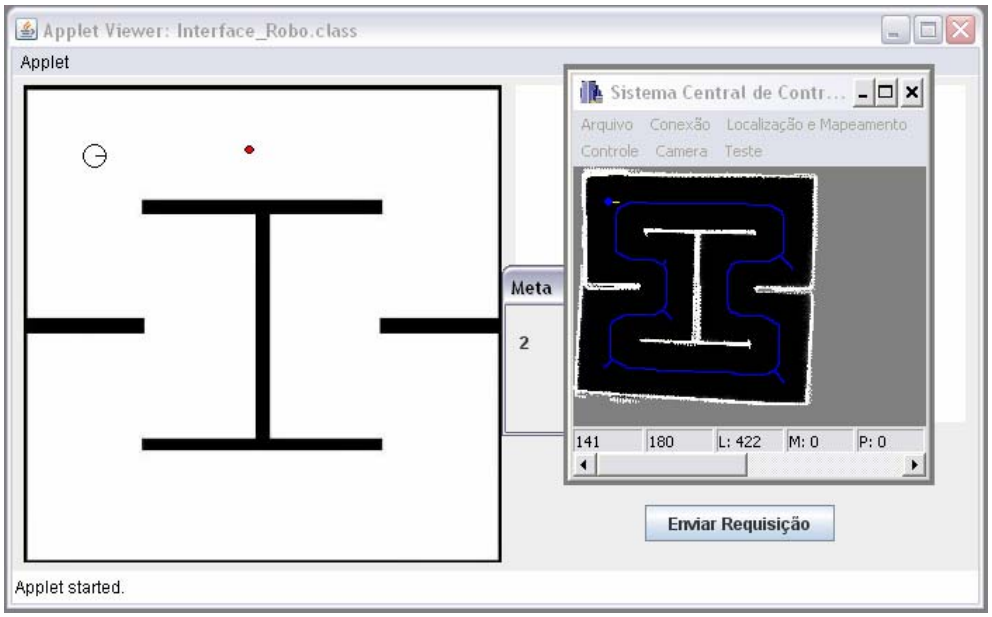


Figura 5.3 – Interface Web fase 2

Através da comunicação entre a interface Web e o sistema central de controle, o robô gradativamente navega até o primeiro ponto solicitado, como ilustrado na Figura 5.4, sempre atualizando sua posição tanto na interface quanto no sistema de controle. Ao atingir umas das requisições, o robô espera que alguma tecla seja pressionada para se locomover até o próximo ponto de percurso solicitado.

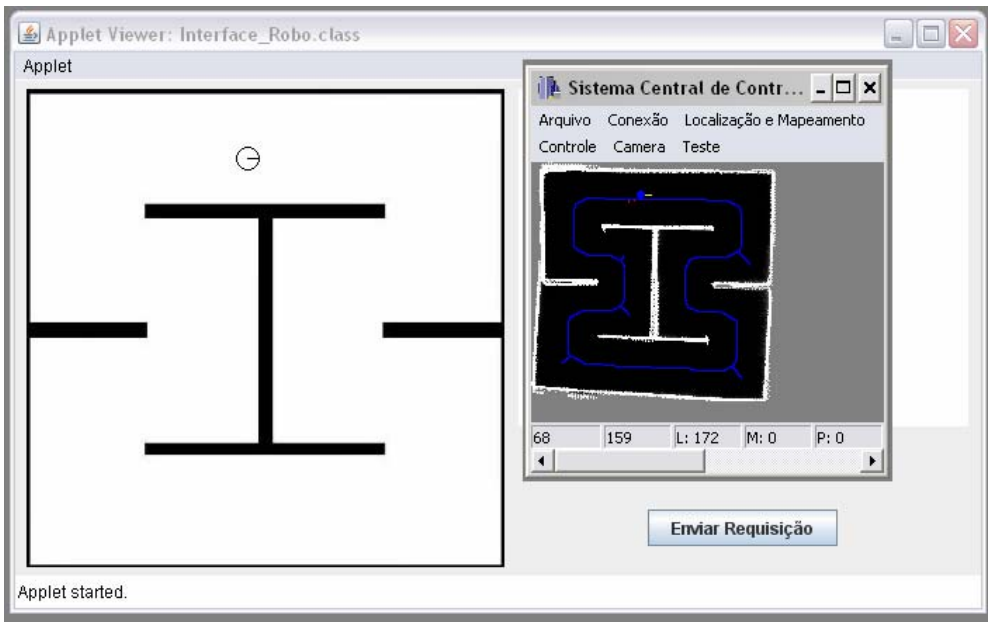


Figura 5.4 – Interface Web fase 3

É importante ressaltar que enquanto o robô executa uma determinada requisição, os usuários podem enviar outras tarefas de percurso para o robô. O sistema recebe as requisições e as coloca em uma fila de requisições. Enquanto a fila de requisições não estiver vazia o sistema continua controlando o robô para realizar todas as requisições pendentes.

5.4 Método de Esqueletização de Imagens

A proposta de mapeamento hibrida desenvolvida foi validada através da geração de mapas topológicos a partir de diferentes ambientes. Os ambientes considerados diferiam entre

si nas seguintes características: tamanho, forma e complexidade. Na Figura 5.5, é apresentado um dos vários ambientes simulados utilizados.

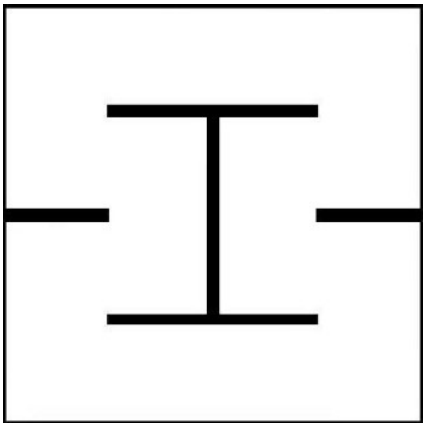


Figura 5.5 – Mapa H

A geração do mapa topológico global é realizada a partir do mapa métrico global, sendo assim, é necessário que o mapa métrico do ambiente, simulado ou real, seja conhecido a *priori*. Na Figura 5.6, é apresentada a construção incremental do mapa topológico a partir da técnica de esqueletização de imagens para o caso do ambiente simulado da Figura 5.5. Essa construção é realizada por partes, uma vez que o robô atualiza gradualmente a grade de ocupação enquanto explora o ambiente. De acordo com a Figura 5.6, tem-se que as linhas verdes representam o mapa topológico. Os obstáculos, detectados no ambiente, são representados pela cor branca. O espaço livre é indicado pela cor preta. O robô está ilustrado pelo ponto azul e, por fim, os pontos vermelhos representam o grau de confiança computado pelo robô para sua localização no mapa. As regiões na cor cinza representam áreas não exploradas ou desconhecidas.

Na Figura 5.6(a), é apresentado o primeiro estágio do mapeamento, no qual o robô captura as primeiras leituras sensoriais do espaço livre sem nenhum conhecimento prévio. Como pode ser observado, a região próxima ao robô é adequadamente convertida no seu mapa topológico correspondente, contudo a região mais distante ao robô ainda não está bem definida, uma vez que ela é limitada pelo alcance dos sinais sensoriais obtidos pelo robô. Nesse caso, os estados dessas células na grade de ocupação (OCC ou EMP) ainda não foram definidos.

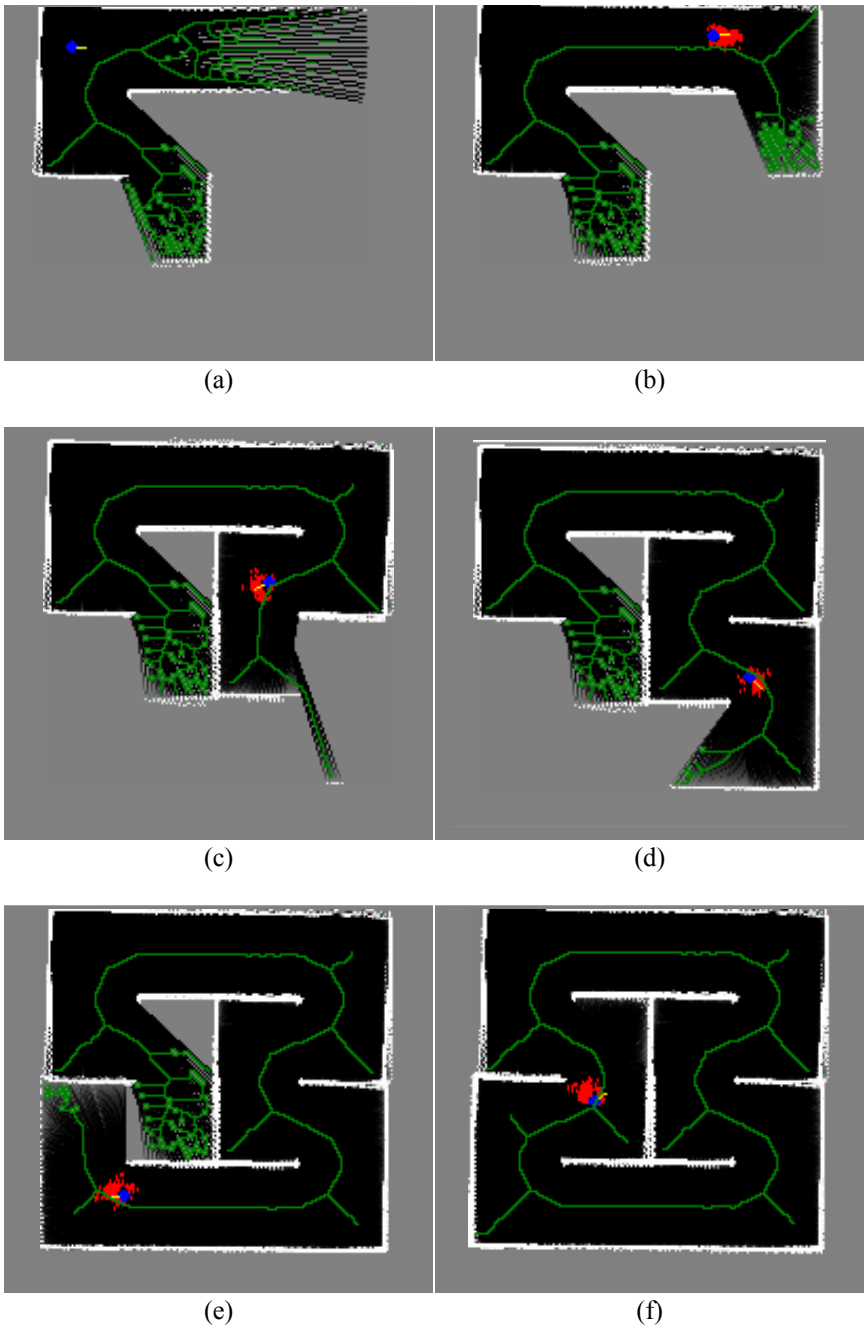


Figura 5.6 – Atualização incremental do mapa topológico a partir do mapa métrico

A medida que o robô explora a região superior do mapa, Figuras 5.6(b) e (c), os estados de suas respectivas células na grade de atualização são atualizados. Dessa forma, as linhas topológicas para essas regiões são novamente computadas e adequadamente corrigidas. Finalmente, quando o robô executa a exploração completa do ambiente, o mapa topológico global é concluído, como está ilustrado na Figura 5.6(f).

Para validar essa técnica híbrida de mapeamento em uma situação real, o sistema proposto foi aplicado em um ambiente dinâmico utilizando o robô Pioneer 1. A Figura 5.7 apresenta o mapa obtido através da exploração do corredor do prédio do Laboratório de Aprendizado de Robôs. As regiões R1 e R2 representam áreas não exploradas presentes no mapa, devido ao abrir e fechar de portas durante o processo de mapeamento.

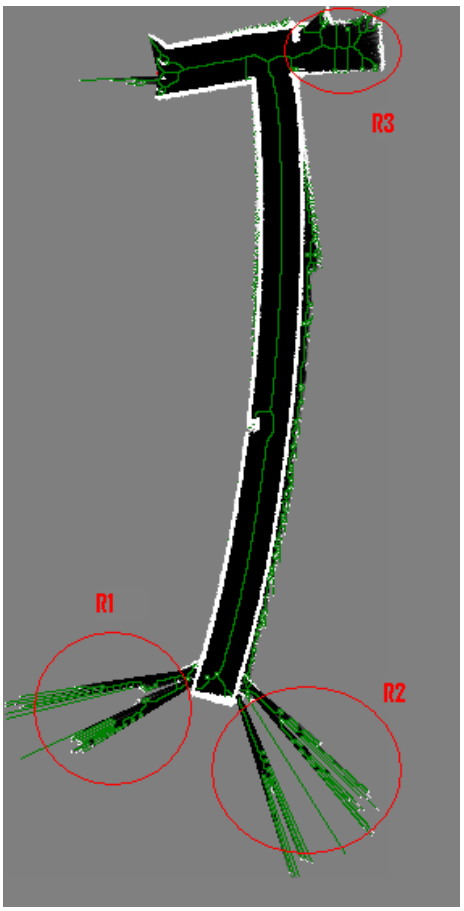


Figura 5.7 – Linhas topológicas

A técnica de esqueletização de imagens construiu satisfatoriamente as linhas topológicas de um ambiente real. Contudo, como pode ser notado no mapa da Figura 5.7**,** algumas linhas topológicas que inadequadas (principalmente na região R3) são geradas. Essa característica é proveniente do fato de que algumas células na grade de ocupação podem conter estimativas incorretas sobre suas taxas de ocupação. Isto é devido, por exemplo, aos ruídos obtidos dos sensores do robô. Os ruídos externos à área interna do mapa não afetam inteiramente a qualidade do mapa construído, pois essas regiões não pertencem ao campo de atuação do robô.

De acordo com Ko et al. (KO et al., 2004), o Grafo de Voronoi Generalizado gera linhas que conectam pontos de junção a pontos presentes em regiões côncavas do mapa. Contudo, essas linhas de borda são desnecessárias para a navegação. Por outro lado, o mapa topológico baseado em um método de esqueletização de imagens não gera esse tipo de linha. Portanto, ele é mais eficiente sob a ótica da navegação autônoma de robôs móveis. A ausência das linhas de borda também foi observada nos testes executados neste trabalho.

5.5 FILTROS MORFOLÓGICOS MATEMÁTICOS

A aplicação dos filtros morfológicos matemáticos foi validada também através da geração de mapas topológicos a partir de diferentes ambientes. Novamente, os ambientes considerados diferiam entre si em tamanho, forma e complexidade. Na Figura 5.8, é apresentado um dos ambientes mapeados em sua representação métrica. Este mapa específico contem características relevantes para a aplicação dos filtros morfológicos matemáticos, tais como mobília ao redor do ambiente, regiões não exploradas pelo robô e diversos ruídos na região externa do mapa.

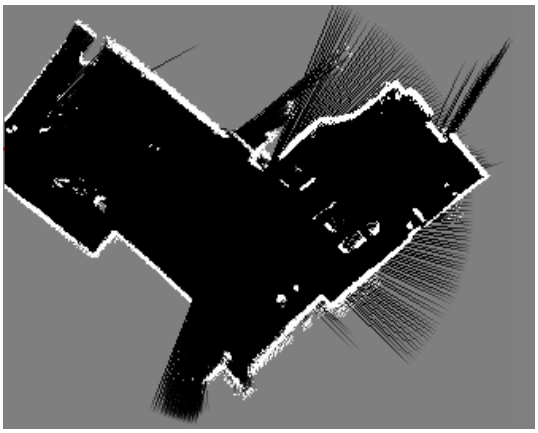
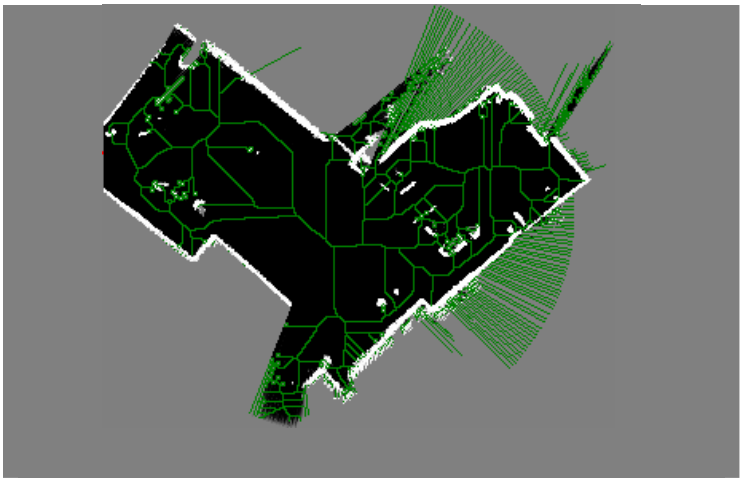


Figura 5.8 – Mapa da entrada da biblioteca

Na Figura 5.9, são ilustradas as linhas topológicas obtidas através do método de esqueletização de imagens baseado na técnica *Thinning* com e sem a aplicação dos filtros de erosão e de abertura.



(a) Sem filtros

(b) Filtro de erosão

(c) Filtro de abertura

Figura 5.9 – Linhas topológicas com filtros

Na Figura 5.9(a), é apresentado o mapa obtido sem a utilização dos filtros morfológicos matemáticos. Nas Figuras 5.9(b) e 5.9(c), são apresentados os mapas híbridos com a aplicação dos filtros de erosão e de abertura, respectivamente. Em ambos os casos, a quantidade de iterações foi igual a 1 e o número de células vizinhas analisadas foi configurado também em 1. Em outras palavras, se o número de células vizinhas com estado oposto à célula de referência for maior ou igual a 1, então o estado da célula central será alterado de acordo com o filtro usado.

Como ilustrado na Figura 5.9, os ruídos na região externa do mapa são quase completamente eliminados. Contudo, em certas aplicações pode ser necessário eliminar completamente todas as linhas topológicas externas ao mapa. Incrementar o número de iterações utilizadas pelos filtros morfológicos é um dos métodos viáveis para eliminar essas linhas externas.

Na Figura 5.10 pode ser visto, por exemplo, o mapa topológico obtido através da aplicação do filtro de erosão com quatro iterações. Como observado, o mapa gerado possui algumas linhas topológicas segmentadas. Porém, em certas tarefas da navegação autônoma o usuário pode desejar manter o mapa topológico tão contínuo quanto for possível.

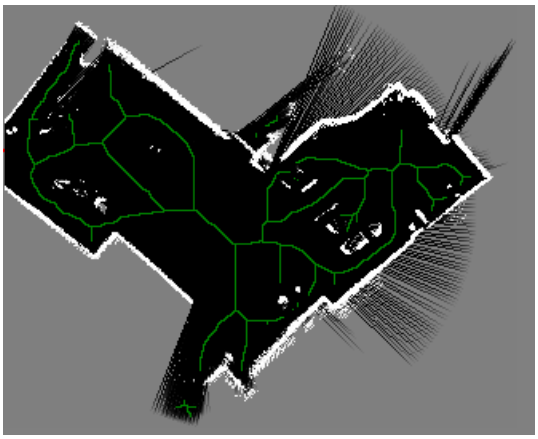


Figura 5.10 – Erosão (4 iterações)

Uma alternativa eficiente para eliminar ruídos e manter o mapa contínuo, restaurando as linhas topológicas segmentadas, é a aplicação do filtro de abertura. Na Figura 5.11, é ilustrado o mapa com um filtro de abertura de cinco iterações e número de células vizinhas analisadas configurado em 1.

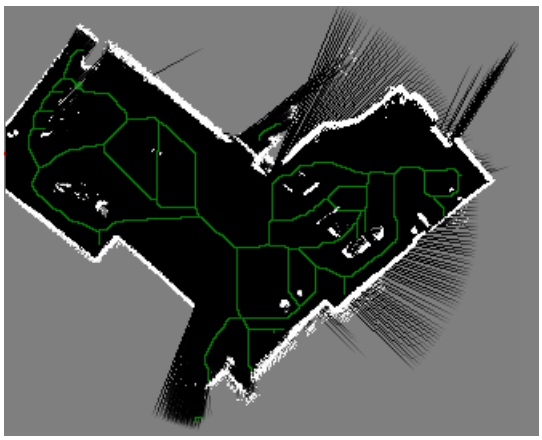


Figura 5.11 – Abertura (5 iterações)

De acordo com a aplicação outro possível requisito de um sistema de navegação autônoma é a redução da quantidade de linhas topológicas internas, definindo linhas topológicas mais concisas para o planejamento de trajetórias. A variação do número de células vizinhas analisadas demonstrou ser um método para controlar a quantidade de linhas internas. A Figura 5.12 apresenta o mapa obtido através da aplicação do filtro de abertura com cinco iterações e número de células vizinhas analisadas também igual a cinco.

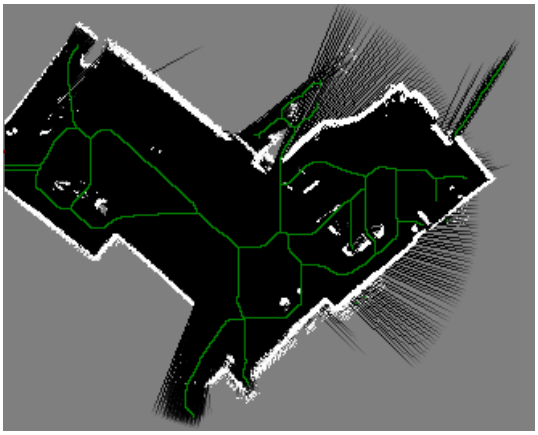


Figura 5.12 – Abertura (5 iterações e 5 células vizinhas)

De acordo com os testes executados, observa-se que escolher um número grande de células analisadas, tal como de 6 a 8, seleciona apenas células isoladas com ruído, não interferindo nas áreas maiores do mapa. Por exemplo, escolhendo um número de células

vizinhas igual a 8, apenas células isoladas terão seus estados alterados. Os filtros de erosão e abertura com o valor de número de células igual a 6 ou 7 são capazes de remover linhas unitárias de células sem afetar qualquer outra parte do mapa. Um valor de 1 a 5 é capaz de mudar mais significativamente a morfologia do mapa.

5.6 CONSIDERAÇÕES FINAIS

Nesse capítulo foi apresentado o estado atual do sistema de navegação autônoma do Laboratório de Aprendizado de Robôs pertencente ao Instituto de Ciências Matemáticas e de Computação da USP-São Carlos. Além disso, os principais resultados decorrentes do mapeamento topológicos foram elucidados e discutidos.

De acordo com estes testes, o mapa topológico baseado em um método de esqueletização de imagens não gera linhas que conectam pontos de junção a pontos presentes em regiões côncavas do mapa. Portanto, ele se torna mais eficiente sob a ótica da navegação autônoma de robôs móveis. Além disso, os testes relacionados aos filtros morfológicos matemáticos demonstraram que a aplicação de filtros de erosão e de abertura são alternativas eficientes para melhorar a qualidade dos mapas topológicos, principalmente quando estes possuem suas representações afetadas por ruídos oriundos dos estados das células da grade de ocupação obtidos do mapeamento métrico. Uma descrição desses resultados também pode ser encontrada em (OLIVEIRA; ROMERO, 2009; OLIVEIRA et al., 2009).

CAPÍTULO

6

CONCLUSÃO E TRABALHOS FUTUROS

"Sabemos o que somos, mas não o que podemos ser".

William Shakespeare

Nesse projeto um sistema de navegação autônoma para robôs móveis foi estendido para incorporar um método de mapeamento topológico, visando melhorar o desempenho de um sistema integrado de controle robótico. Este sistema foi desenvolvido por Bianchi (BIANCHI, 2002) e ampliado com uma interface Web, implementada por Barbosa (BARBOSA, 2006). O sistema de controle é constituído por módulos que lidam com tarefas relacionadas à localização, planejamento de trajetórias e mapeamento de ambientes. A interface Web, por sua vez, é responsável por apresentar ao usuário o mapa do ambiente no qual poderá definir pontos pelos quais o robô deverá passar.

Tendo como metas a adaptação do sistema para a plataforma do robô Pioneer 3 – DX, robô recentemente adquirido pelo Laboratório de Aprendizado de Robôs (LAR) do ICMC/USP; a atualização do módulo responsável pela interface Web, tornando-o efetivamente operacional; e a incorporação de uma técnica de mapeamento topológico à abordagem métrica implementada em trabalhos anteriores. O sistema desenvolvido neste trabalho atende a dois princípios chaves distintos: adaptação e ampliação. Adaptação porque inicia o processo de adequação do sistema desenvolvido até então para a plataforma do robô Pioneer 3 – DX, e ampliação porque incorpora novas funcionalidades e recursos ao sistema.

Quanto à adaptação, a interface Web desenvolvida por Barbosa (BARBOSA, 2006), foi devidamente restaurada. Além disso, parte dos aplicativos necessários para a mudança do sistema para a plataforma do Pioneer 3 – DX foi adequadamente migrada para computadores que comportam as atuais exigências do projeto.

Quanto à ampliação, uma abordagem híbrida foi proposta para a construção de mapas topológicos a partir de mapas métricos utilizando um método de esqueletização de imagens. O método de esqueletização *Thinning* utilizado foi proposto por Ko (KO et al., 2004) e ele mostrou ser adequado para a tarefa de mapeamento topológico, uma vez que é capaz de construir mapas topológicos a partir de uma grade de ocupação qualquer. Vários experimentos foram executados para validar esta proposta híbrida, demonstrando que nossa proposta é simples e eficiente. Ela pode ser aplicada em ambientes com diferentes formas e complexidades, sendo uma alternativa robusta para construção de mapas topológicos.

Além disso, nesse trabalho, dois filtros morfológicos matemáticos também foram propostos, visando aperfeiçoar o processo de construção dos mapas topológicos. Os filtros utilizados foram os de erosão e de abertura. O sistema proposto lida com situações nas quais ruídos oriundos das leituras sensoriais impedem a construção de um mapa métrico adequado, implicando na obtenção de mapas topológicos inviáveis. Dessa forma, torna-se necessária a utilização de filtros que sejam capazes de eliminar linhas incorretas, obtidas em função da imprecisão dos sensores, presentes no mapa topológico. De modo geral, o filtro de erosão apresentou uma maior capacidade de eliminar os ruídos do mapa topológico, contudo essa maior capacidade pode provocar um efeito considerado indesejado, a fragmentação das linhas topológicas. Por outro lado, o filtro de abertura demonstrou ser um método mais apropriado para se manter a consistência das linhas topológicas. Sendo assim, cada um dos filtros se mostrou mais adequado para um tipo de situação, que é definida pelas necessidades da aplicação do usuário.

6.1 TRABALHOS FUTUROS

Como trabalho futuro, pretende-se realizar uma análise detalhada comparando o método de esqueletização de imagens com as técnicas baseadas em digramas de Voronoi generalizados, visando obter uma visão mais ampla sobre as vantagens e desvantagens de cada abordagem na geração de mapas topológicos.

Além disso, uma vez que a Localização de Monte Carlo calcula as probabilidades de todas as amostras aleatórias, aumentando o custo computacional do método à medida que o número de amostras cresce, buscar-se-á explorar estratégias que visam à redução do número de amostras do filtro de partículas sem degradar o desempenho da Localização de Monte Carlo. Nessa situação, um longo período durante a fase de atualização do algoritmo irá provocar uma diferença significativa entre a posição estimada do robô e a sua real posição. Sendo assim, os erros de localização tornar-se-ão maiores à medida que a velocidade de navegação aumenta, limitando dessa maneira a velocidade do robô ao tempo computacional do método de localização.

Uma das estratégias possíveis para lidar com esse problema é a utilização das linhas topológicas durante o processo de localização. A estratégia visualizada é exigir que o robô navegue exclusivamente ao longo das linhas topológicas obtidas pelo método de esqueletização de imagens. Dessa maneira, o movimento do robô se restringirá apenas as regiões das linhas topológicas, o que garantirá que as amostras capturadas pelos sensores do robô serão apenas da vizinhança dessas linhas, e não do espaço livre inteiro. A utilização de linhas topológicas, como percursos de navegação, também pode ser justificada pelo campo de atuação dos sensores. Se o robô navegar ao longo de caminhos eqüidistantes dos diversos obstáculos, os quais correspondem às linhas topológicas, ele terá maior oportunidade de capturar mais informações sobre o ambiente do que se percorresse outros caminhos.

Outra proposta a ser explorada como um trabalho futuro é a fusão de informações obtidas de diferentes tipos de sensores, integrando múltiplas medidas com diferentes precisões para criar um *framework* de localização mais robusto e efetivo. Atualmente, alguns estudos afirmam que a fusão de sonares e sensores a laser é amplamente mais efetiva, para se atingir uma localização global, do que a utilização de cada um desses sensores individualmente.

Finalmente, a adaptação do sistema à plataforma do robô Pioneer 3 – DX deverá ser concluída, permitindo que futuramente o projeto de navegação autônoma alcance seu principal objetivo, a tarefa de entrega de documentos no ICMC, podendo inclusive ser utilizado posteriormente em outras tarefas, tais como a entrega de peças e/ou objetos, a coleta de lixo e o monitoramento de ambientes.

REFERÊNCIAS BIBLIOGRÁFICAS[8]

ANDREASSON, H.; TREPTOW, A.; DUCKETT, T. Self-localization in non-stationary environments using omni-directional vision. **Robotics and Autonomous Systems**, vol. 55, p. 541 – 551, 2007.

ANTSAKLIS, P. J. Scientific visualization: techniques and applications. **IEEE Control Systems Magazine**, vol. 15(3), p. 5 – 80, 1995.

ARKIN, R. **Behavior-Based Robotics**. MIT Press, 1998.

BARBOSA, A. W. **Sistema remoto para controle de robôs móveis via Web**. Dissertação de mestrado. ICMC/USP - São Carlos/SP, 2006.

BARBOSA, A. W.; BIANCHI, R. E.; ROMERO, R. A. F. Control of Mobile Robots Via Internet. **ABCM Symposium Series in Mechatronics**, vol. 2, p. 281 – 288, 2006.

BAXES, G. A. **Digital Image Processing**. John Wiley & Sons, 1994.

BENSALEM, S.; GALLIEN, M.; INGRAND, F.; KAHLOUL, I.; THANH-HUNG, N. Designing Autonomous Robots – Toward a More Dependable Software Architecture. **IEEE Robotics & Automation Magazine**, vol. 16 (1), p. 67 – 77, 2009.

BIANCHI, R. E. **Sistema de navegação de robôs móveis autônomos para o transporte de documentos**. Dissertação de mestrado. ICMC/USP - São Carlos/SP, 2002.

BIRK, A.; CARPIN, S. Merging Occupancy Grid Maps From Multiple Robots. In **Proceedings of the IEEE**, vol. 94 (7), p. 1384 – 1397, 2006.

BLANCO, J. L.; FERNÁNDEZ-MADRIGAL, J. -A.; GONZÁLEZ, J. Toward a Unified Bayesian Approach to Irbid Metric-Topological SLAM. **IEEE Transactions on Robotics**, vol. 24 (2), p. 259 – 270, 2008.

[8] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

BLANCO, J. L.; GONZÁLEZ, J.; FERNÁNDEZ-MADRIGAL, J.-A. Subjective local maps for hybrid metric-topological SLAM. **Robotics and Autonomous Systems**, vol 57, p. 64 – 74, 2009.

BOADA, B. L.; BLANCO, D.; MORENO, L. Symbolic Place Recognition in Voronoi-Based Maps by Using Hidden Markov Models. **Journal of Intelligent and Robotic** *Systems*, vol. 39, p. 173 – 197, 2004.

BORENSTEIN, J.; KOREN, Y. The Vector field histogram – fast obstacle avoidance for mobile robots. **IEEE Transactions on Robotics and Automation**, vol. 7(3), p. 278 – 288, 1991.

BORENSTEIN, J.; EVERETT, B.; FENG, L. **Navigating Mobile Robots: Systems and Techniques**. K. Peters Ltd, 1996.

BRADY, M. Artificial Intelligence and Robotics. **Artificial Intelligence**, vol. 26, p. 79 – 121, 1985.

BROOKS, R. A. Intelligence without reason. In **Proceedings of the 12th International Joint Conference on Artificial Intelligence**, p. 569 – 595, 1991.

BUHMANN, J.; BURGARD, W.; CREMERS, A. B.; FOX, D.; HOFMANN, T.; SCHNEIDER, F.; STRIKOS, J.; THRUN, S. The Mobile Robot Rhino. **AI Magazine**, vol. 16(1), 1995.

BURGARD, W.; FOX, D.; HENNIG, D.; SCHMIDT, T. **Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids**, 1996.

BURGARD, W.; FOX, D.; THRUN, S. **Active Mobile Robot Localization***,* 1997.

BURGARD, W.; CREMERS, A. B.; FOX, D.; HÄHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W.; THRUN, S. **Experiences With an Interactive Museum Tour-Guide Robot**, 1998.

CALVO, R.; FIGUEIREDO, M. Extensão da Lei de Hebb e Aprendizagem por Reforço em Redes Neurais Aplicadas à Sistema de Navegação Autônoma. In **XXIII Congresso da Sociedade Brasileira de Computação**, Encontro Nacional de Inteligência Artificial, Campinas, São Paulo. Anais do IV ENIA, 2003.

CALVO, R. **Arquitetura híbrida inteligente para navegação autônoma de robôs**. Dissertação de mestrado. USP - São Carlos/SP, 2007.

CASTELLANOS, J. A.; MARTINEZ-CANTIN, R.; TARDÓS, J. D.; NEIRA, J. Robocentric map joining: Improving the consistency of EKF-SLAM. **Robotics and Autonomous Systems**, vol. 55, p. 21 – 29, 2007.

CAZANGI, R.; FIGUEIREDO, M. Simultaneous emergence of conflicting basic behaviors and their coordination in an evolutionary autonomous navigation system. In **Proceedings of 2002 Congress on Evolutionary Computation**, Honolulu - Hawaii, USA: CEC'02, Congress on Evolutionary Computation, 2002.

CHOSET, H.; NAGATANI, K. Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization without Explicit Localization. **IEEE Transactions on Robotics and Automation**, vol. 17 (2), p. 125 – 137, 2001.

COBANO, J. A.; ESTREMERA, J.; GONZALEZ DE SANTOS, P. Location of legged robots in outdoor environments. **Robotics and Autonomous Systems**, vol. 56, p. 751 – 761, 2008.

COSTA, L. F.; CÉSAR Jr., R. M. **Shape Analysis and Classification – Theory and Practice**. Editora CRC, 2001.

COX, I. J. Blanche: an experiment in guidance and navigation of an autonomous robot vehicle. **IEEE Transactions on Robotics and Automation**, vol. 7(2), p. 193 – 204, 1991.

CRESTANI, P. R.; FIGUEIREDO, M.; ZUBEN, F. V. A hierarchical neuro-fuzzy approach to autonomous navigation. In **Proceedings of 2002 International Joint Conference on Neural Networks**, Honolulu - Hawaii, USA: IJCNN'02, International Joint Conference on Neural Networks, 2002.

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. Monte Carlo Localization for mobile robots. In **Proceedings of the IEEE International Conference on Robotics and Automation**, 1999.

DOUGHERTY, E. R.; LOTUFO, R. A. **Hands-on Morphological Image Processing**. Tutorial Texts in Optical Engineering, vol. TT59. Editora SPIE, 2003.

ELFES, A. **Occupancy Grids: A Probabilistic for Robot Perception and Navigation**. Tese Ph.D., Carnegie Mellon University, 1989.

FOX, D. **Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation**. Universidade de Bonn, Alemanha. Tese de Doutorado, 1998.

FOX, D.; BURGARD, W.; THRUN, S.; CREMERS, A. B. Position estimation for mobile robots in dynamic environments. In **AAAI/IAAI**, p. 983 – 988, 1998a.

FOX, D.; BURGARD, W.; THRUN, S. Active Markov localization for Mobile robots. **Robotics and Autonomous Systems**, vol. 25, p. 195 – 207, 1998b.

FOX, D.; BURGARD, W.; THRUN, S. Markov Localization for mobile robots in dynamic environments. **Journal of Artificial Intelligence Research**, vol. 11, p. 391 – 427, 1999.

GARRIDO, S.; MORENO, L.; BLANCO, D. Exploration of a cluttered environment using Voronoi Transform and Fast Marching. **Robotics and Autonomous Systems**, 2008.

GRISETTI, G.; TIPALDI, G. D.; STACHNISS, C.; BURGARD, W.; NARDI, D. Fast and accurate SLAM with Rao-Blackwellized particle filters. **Robotics and Autonomous Systems**, vol. 55, p. 30 – 38, 2007.

KALRA, N.; FERGUSON, D.; STENTZ, A.. Incremental reconstruction of generalized Voronoi diagrams on grids. **Robotics and Autonomous Systems**, vol. 57, p. 123-128, 2009.

KIM, C.; SAKTHIVEL, R.; CHUNG, W. K. Unscented FastSLAM: A Robust and Efficient Solution to the SLAM Problem. **IEEE Transactions on Robotics**, vol. 24 (4), p. 808 – 820, 2008.

KO, B. Y.; SONG, J. B.; LEE, S. Real-time Building of Thinning-Based Topological Map with Metric Features. In **Proceedings of International Conference on Intelligent Robots and Systems**, p. 1524 – 1529, 2004.

KOENIG, S.; GOODWIN, R.; SIMMONS, R. **Robot navigation with Markov models: A framework for path planning and learning with limited computational resources**, 1996.

KONOLIGE, K. COLBERT: A language for reactive control in saphira. In **German Conference on Artificial Intelligence**, Freiburg, 1997.

KOREN, Y.; BORENSTEIN, J. **Analysis of Control Methods for Mobile Robot Obstacle Avoidance**. 1991.

KORTENKAMP, D.; WEYMOUTH, T. Topological mapping for mobile robots using a combination of sonar and vision sensing. In **Proceedings of the Twelfth National Conference on Artificial Intelligence**. AAAI – IAAI / MIT Press, p. 979 – 984, 1994.

KÖSE, H.; AKIN, H. L. The Reverse Monte Carlo localization algorithm. **Robotics and Autonomous Systems**, vol. 55, p. 480 – 489, 2007.

KOOTSTRA, G.; BOER, B. Tackling the premature convergence problem in Monte-Carlo localization. **Robotics and Autonomous Systems**, vol. 57, p. 1107 – 1118, 2009.

KWON, T.; YANG J.; SONG, J.; CHUNG, W. Efficiency improvement in Monte Carlo localization through topological information. In **2006 IEEE/RSJ IROS - Int. Conf. on Intelligent Robots and Systems**, p. 424 – 429, 2006.

LEE, D.; CHUNG, W. Discrete-Status-Based Localization for Indoor Service Robots. **IEEE Transactions on Industrial Electronics,** vol. 53(5), p. 1737 – 1746, 2006.

LINAKER, F.; ISHIKAWA, M. Real-time appearance-based Monte Carlo localization. **Robotics and Autonomous Systems**, vol. 54, p. 205 – 220, 2006.

MATARIĆ, M. J. **The Robotics Primer**. MIT Press, 2007.

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

MCKERROW, P. J. **Introduction to Robotics**. Addison Wesley, 1986.

MENEGATTI, E.; PRETTO, A.; SCARPA, A.; PAGELLO, E. Omnidirectional Vision Scan Matching for Robot Localization in Dynamic Environments. **IEEE Transactions on Robotics**, vol. 22 (3), p. 523 – 535, 2006.

MORAVEC, H. P. Sensor fusion in certainty grids for mobile robots. **AI Magazine**, p. 61 – 74, 1988.

MULLANE, J.; ADAMS, M. D.; WIJESOMA, W. S. Robotic Mapping Using Measurement Likelihood Filtering. **The International Journal of Robotics Research**, vol. 28, p. 172 – 190, 2009.

MURPHY, R. R. **Introduction to AI Robotics**. MIT Press, 2000.

MURTRA, A. C.; TUR, J. M. M.; SANFELIU, A. Action evaluation for mobile robot global localization in cooperative environments. **Robotics and Autonomous Systems**, vol. 56, p. 807 – 818, 2008.

NOYKOV, S.; ROUMENIN, C. Occupancy grids building by sonar and mobile robot. **Robotics and Autonomous Systems**, vol. 55, p. 162 – 175, 2007.

OLSON, C. F. Probabilistic self-localization for mobile robots. **IEEE Transactions On Robotics And Automation**, vol. 16(1), p. 55 – 66, 2000.

OLIVEIRA, J. R.; ROMERO, R. A. F. Image Skeletonization Method applied to generation of Topological Maps. In **Proceedings of Latin American Robotics Symposium**, Valparaiso/Chile, 2009.

OLIVEIRA, J. R.; FILHO, E. A. S.; ROMERO, R. A. F. Mathematical Morphology Filters applied to an Image Sketonization Method to generation of Topological Maps. In **Proceedings of International Conference on Computer Sciences and Converge Information Technology**, Seoul/Korea, 2009.

PRESTES E SILVA Jr., E.; IDIART, M. A. P.; TREVISAN, M.; ENGEL, P. M. Autonomous Learning Architecture for Environmental Mapping. **Journal of Intelligent and Robotics Systems**, vol. 39, p. 243 – 263, 2004.

RANGANATHAN, A.; MENEGATTI, E.; DELLAERT, F. Bayesian Inference in the Space of Topological Maps. **IEEE Transactions on Robotics**, vol. 22 (1), p. 92 – 107, 2006.

RAO, R. N.; FUENTES, O. Learning navigational behaviors using a predictive sparse distributed memory. In **From Animals to Animals: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior**, p. 382 – 390, 1996.

RAWLINSON, D.; JARVIS, R. Topologically-directed navigation. **Robotica**, vol. 26, p. 189 – 203, 2008a.

RAWLINSON, D.; JARVIS, R. Ways to Tell Robots Where to Go – Directing Autonomous Robots Using Topological Instructions. **IEEE Robotics & Automation Magazine**, p. 27 – 36, 2008b.

RIZZINI, D. L.; CASELLI, S. Metric-topological maps from laser scans adjusted with incremental tree network optimizer. **Robotics and Autonomous Systems**, vol. 57, p. 1036 – 1041, 2009.

ROQUE, W. L.; DOERING, D. Trajectory planning for lab robots based on global vision and Voronoi roadmaps. **Robotica**, vol. 23, p. 467 – 477, 2005.

RUSS, J. C. **The Image Processing – Handbook**. Editora CRC, 5ª. Edição, 2007.

SCHLUSSEL, K. Robotics and Artificial Intelligence across the Atlantic and Pacific. **IEEE Transactions on Industrial Electronics**. IE-30, 3, p. 244 – 251, 1985.

SIMMONS, R.; KOENIG, S. Probabilistic robot navigation in partially observable environments. In **Proceedings of IJCAI-95**, Montreal/Canada, p. 1080 – 1087, 1995.

SLAWIÑSKI, E.; MUT, V. A.; POSTIGO, J. F. Teleoperation of Mobile Robots With Time-Varying Delay. **IEEE Transactions on Robotics**, vol. 23(5), p. 1071 – 1082, 2007.

SMITHERS, T. Autonomy in robots and others agents. **Brain and Cognition**, vol. 34, p. 88 – 106, 1997.

SOHN, H. J.; KIM, B. K. An Efficient Localization Algorithm Based on Vector Matching for Mobile Robots Using Laser Range Finders. **Journal of Intelligent and Robotic Systems**, vol. 51, p. 461 – 488, 2008.

SPACEK, L.; BURBRIDGE, C. Instantaneous robot self-localization and motion estimation with omnidirectional vision. **Robotics and Autonomous Systems**, vol. 55, p. 667 – 674, 2007.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. **Artificial Intelligence**, vol. 99, p. 21 – 71, 1998.

THRUN, S. **Robotic Mapping: A Survey**. Carnegie Mellon University, CMU-CS-02-111, 2002.

THRUN, S.; BEETZ, M.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A. B.; DELLAERT, F.; FOX, D., HÄHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. **Journal of Robotics Research**, 2000a.

THRUN, S.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A. B.; DELLAERT, F.; FOX, D.; HÄHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. MINERVA: A tour-guide robot that learns. In **KI – Kunstliche Intelligenz**, p. 14 – 26, 1999a.

THRUN, S.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A. B.; DELLAERT, F.; FOX, D.; HÄHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. MINERVA: A Second-Generation Museum Tour-Guide Robot. In **Proceedings of the IEEE International Conference on Robotics and Automation**, 1999b.

THRUN, S.; BÜCKEN, A. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In **Proceedings of the Thirteenth National Conference on Artificial Intelligence**, 1996.

THRUN, S.; FOX, D.; BURGARD, W. **Probabilistic Robotics**. Early Draft, 2000b.

THRUN, S.; FOX, D.; BURGARD, W.; DELLAERT, F. Robust Monte Carlo Localization for mobile robots. **Artificial Intelligence**, vol. 128, p. 99 – 141, 2001.

TOVAR, B.; MUÑOZ-GÓMEZ, L.; MURRIETA-CID, R.; ALENCASTRE-MIRANDA, M.; MONROY, R.; HUTCHINSON, S. Planning exploration strategies for simultaneous localization and mapping. **Robotics and Autonomous Systems**, vol. 54, p. 314 – 331, 2006.

ULRICH, I.; BORENSTEIN, J. VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In **Proceedings of the IEEE International Conference on Robotics and Automation**, p. 2505 – 2511, 2000.

ZHAI, S.; MILGRAM, P. A telerobotic virtual control system. In **Proceedings of SPIE**, vol. 1612, Cooperative Intelligent Robotics in Space II, Boston, p. 311 – 320, 1991.

WOLF, J.; BURGARD, W.; BURKHARDT, H. Robust Vision-Based Localization by Combining an Image-Retrieval System with Monte Carlo Localization. **IEEE Transactions on Robotics**, vol. 21(2), p. 208 – 216, 2005.