Mapeamento de ambientes externos utilizando robôs móveis

Alberto Yukinobu Hata

Mapeamento de ambientes externos utilizando robôs móveis

Alberto Yukinobu Hata

Orientador: ***Prof. Dr. Denis Fernando Wolf***

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
Março/2010

Agradecimentos

Primeiramente agradeço ao professor Denis Fernando Wolf pela oportunidade dada para trabalhar junto ao seu grupo de pesquisa, pelos seus ensinamentos científicos, pela sua compreensão e atenção por este trabalho, e acima de tudo, pela sua amizade.

Agradeço também aos amigos do LRM (Laboratório de Robótica Móvel) pela assistência oferecida para a execução dos experimentos e a companhia ao longo dos anos. Não poderia também deixar de agradecer as diversas amizades criadas com os alunos do curso de pós-graduação.

Por fim, agradeço o apoio financeiro concedido pela FAPESP (Fundação de Amparo à Pesquisa do Estado de SP) pelo processo 2008/02204-4 e ao INCT-SEC (Instituto Nacional de Ciência e Tecnologia - Sistemas Embarcados Críticos) pelos processos 573963/2008-8, 08/57870-9 e 08/09713-1.

Resumo

A robótica móvel autônoma é uma área relativamente recente que tem como objetivo a construção de mecanismos capazes de executar tarefas sem a necessidade de um controlador humano. De uma forma geral, a robótica móvel defronta com três problemas fundamentais: mapeamento de ambientes, localização e navegação do robô. Sem esses elementos, o robô dificilmente poderia se deslocar autonomamente de um lugar para outro. Um dos problemas existentes nessa área é a atuação de robôs móveis em ambientes externos como parques e regiões urbanas, onde a complexidade do cenário é muito maior em comparação aos ambientes internos como escritórios e casas. Para exemplificar, nos ambientes externos os sensores estão sujeitos às condições climáticas (iluminação do sol, chuva e neve). Além disso, os algoritmos de navegação dos robôs nestes ambientes devem tratar uma quantidade bem maior de obstáculos (pessoas, animais e vegetações).

Esta dissertação apresenta o desenvolvimento de um sistema de classificação da navegabilidade de terrenos irregulares, como por exemplo, ruas e calçadas. O mapeamento do cenário é realizado através de uma plataforma robótica equipada com um sensor laser direcionado para o solo. Foram desenvolvidos dois algoritmos para o mapeamento de terrenos. Um para a visualização dos detalhes finos do ambiente, gerando um mapa de nuvem de pontos e outro para a visualização das regiões próprias e impróprias para o tráfego do robô, resultando em um mapa de navegabilidade. No mapa de navegabilidade, são utilizados métodos de aprendizado de máquina supervisionado para classificar o terreno em navegável (regiões planas), parcialmente navegável (grama, casacalho) ou não navegável (obstáculos). Os métodos empregados foram, redes neurais artificais e máquinas de suporte vetorial. Os resultados de classificação obtidos por ambos foram posteriormente comparados para determinar a técnica mais apropriada para desempenhar esta tarefa.

Abstract

Autonomous mobile robotics is a recent research area that focus on the construction of mechanisms capable of executing tasks without a human control. In general, mobile robotics deals with three fundamental problems: environment mapping, robot localization and navigation. Without these elements, the robot hardly could move autonomously from a place to another. One problem of this area is the operation of the mobile robots in outdoors (e.g. parks and urban areas), which are considerably more complex than indoor environments (e.g. offices and houses). To exemplify, in outdoor environments, sensors are subjected to weather conditions (sunlight, rain and snow), besides that the navigation algorithms must process a larger quantity of obstacles (people, animals and vegetation).

This dissertation presents the development of a system that classifies the navigability of irregular terrains, like streets and sidewalks. The scenario mapping has been done using a robotic platform equipped with a laser range finder sensor directed to the ground. Two terrain mapping algorithms has been devolped. One for environment fine details visualization, generating a point cloud map, and other to visualize appropriated and unappropriated places to robot navigation, resulting in a navigability map. In this map, it was used supervised learning machine methods to classify terrain portions in navigable (plane regions), partially navigable (grass, gravel) or non-navigable (obstacles). The classification methods employed were artificial neural networks and support vector machines. The classification results obtained by both were later compared to determine the most appropriated technique to execute this task.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

Há muito tempo, o homem vem desenvolvendo ferramentas para tornar as suas tarefas mais seguras e rápidas. No dia a dia, por exemplo, os trabalhos domésticos, dispõem de diversos instrumentos que, se não existissem, tornariam esse serviço demorado e desgastante. Os meios de transporte são um exemplo claro de mecanismos que facilitam a locomoção do homem de um local para outro em pouco tempo. Entretanto, essas ferramentas necessitam de alguém para operá-los. Assim, primeiramente é necessário que o futuro usuário passe por uma fase de aprendizagem para poder manipular esses instrumentos. Esse requisito, em muitos casos (em especial, quando o equipamento envolve partes eletro-mecânicas), torna o seu uso não imediato. Além disso, a exigência do contato direto do manipulador com essa ferramenta não o deixa completamente seguro. Com isso, seria interessante automatizar essas ferramentas de forma a adquirirem um alto grau de independência do homem.

A robótica móvel autônoma é uma área da robótica que tem como objetivo desenvolver dispositivos (robôs) com a capacidade de executarem tarefas sem a necessidade da intervenção humana. Para isso, esta área investiga métodos que tornam o comportamento do robô o mais próximo possível do ser humano. As pesquisas na robótica móvel envolvem além da Computação, estudos de diversas outras áreas como Engenharia Elétrica, Engenharia Mecância, Física, Matemática, Estatística, entre outros. Diversas propostas de robôs móveis têm sido apresentadas, entre elas destacam-se aqueles utilizados nas explorações espaciais, serviços domésticos, atividades agrícolas e serviços militares (Murphy, 2000).

Apesar das diversas propostas de aplicações para a robótica móvel, esta ainda é uma área relativamente nova (Thrun *et al.*, 2005), sendo que seus primeiros robôs datam do período da segunda guerra mundial (Sandler, 1991). Além disso, por diversos motivos, relativamente poucos robôs são comercializados. Uma das razões é o alto custo para a sua produção. Outro fator é que não há uma garantia de que os algoritmos implementados irão funcionar em todas as circunstâncias, principalmente nas imprevistas. Dessa forma, a maioria dos robôs comerciais, até o momento, trabalha apenas em ambientes controlados como no interior de casas ou dentro de fábricas, pois existe um menor número de elementos a serem considerados. Por outro lado, robôs que atuam em ambientes externos como, por exemplo, em uma rua, deverão identificar as regiões transitáveis, reconhecer os objetos que são obstáculos e aqueles que não são, além da necessidade de correção da leitura dos sensores prejudicada por fatores naturais.

Para entender melhor o problema para possibilitar a atuação de um robô no ambiente externo, podemos citar a competição DARPA *Grand Challenge* (Thrun *et al.*, 2006), organizado pela agência militar norte-americana DARPA. Este desafio compreendeu em uma corrida realizada entre diversos carros autônomos não-tripulados, sendo que a equipe vencedora é recompensada com um prêmio em dinheiro. A primeira edição desse evento, ocorrido em 2004, consistia em atravessar o deserto de Mojave (EUA) com 240 km de extensão, utilizando um veículo autônomo. No entanto, nenhum dos 107 robôs (carros) inscritos completou a prova. No ano seguinte, ocorreu a sua segunda edição, observando-se um grande progresso das equipes, sendo que dentre os 195 robôs inscritos, 5 conseguiram concluir a prova. Em 2007, foi lançado o DARPA *Urban Challenge*, em que diferentemente dos desafios anteriores, neste, as provas foram efetuadas em ambientes urbanos do estado da Califórnia (EUA). Dessa forma, os veículos autônomos tinham que seguir as regras de trânsito local para cumprir as tarefas. Nessa edição, dentre os 55 inscritos, 3 carros conseguiram cumprir todas as provas. Isto mostra claramente o avanço das técnicas de controle de robôs móveis, uma vez que o problema de automatização não está no hardware, mas sim no software do robô. Por outro lado, percebe-se que esta não é uma tarefa simples, sendo necessário um maior amadurecimento nas pesquisas dessa área.

1.1 Motivação

Um dos ramos que vem explorando a atuação de robótica em ambientes externos é o setor automobilístico. Existem inúmeras vantagens de se automatizar totalmente um automóvel. Uma delas é a possibilidade da pessoa encarregar a tarefa árdua de dirigir horas e horas ao próprio carro. Outra vantagem é a possível redução no número de acidentes de trânsito, que consequentemente poderia diminuir os gastos com reparos de veículos e atendimento às vítimas. Além

disso, muitos deficientes físicos também se beneficiariam com essa tecnologia, utilizando-o para se locomoverem com maior facilidade e dispensando a presença de outra pessoa para essa tarefa. Essa mesma idéia poderia ser estendida para a aviação e embarcações. Assim, independente do meio de transporte, não seria mais necessário o seu controle pelo homem, mas apenas pelo sistema de navegação do robô.

O uso de robôs autônomos poderia também ser ampliado a outras aplicações como, por exemplo, na atuação de veículos em regiões contaminadas por produtos tóxicos, mapeamento de áreas com risco de desabamento e também em atividades militares. Com isso, pessoas que trabalham em locais de risco ficariam encarregadas simplesmente de atribuir um conjunto de tarefas para o veículo, eliminando riscos de acidentes.

O Laboratório de Robótica Móvel do ICMC-USP em parceria com o INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos), por meio do financiamento da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) e do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) tem participado de um projeto para automatização de um carro. O veículo está equipado com sensores laser, câmera, GPS e IMU. Nesse sentido, o sistema de mapeamento proposto pode contribuir significativamente para o desenvolvimento de um sistema de navegação veicular autônomo nacional.

1.2 Objetivos

Este trabalho teve como objetivo principal, o desenvolvimento de algoritmos que auxiliam na atuação de robôs autônomos em terrenos externos semi-estruturados ou desestruturados. O sensoriamento do ambiente externo pelo robô é realizado através de um sensor laser.

Basicamente foram desenvolvidos dois conjuntos de algoritmos, um para realizar o mapeamento do ambiente e outro para a classificação dos terrenos. O algoritmo de mapeamento gera representações computacionais do cenário de forma que o robô possa identificar os elementos presentes ao seu redor. Enquanto que no algoritmo de classificação é realizada a detecção das porções do terreno que são consideradas navegáveis para o robô e aquelas que não são. Para a classificação, foram utilizados dois métodos de aprendizado supervisionado, sendo estes, redes neurais artificiais e máquinas de suporte vetorial. Para o desenvolvimento e simulação dos programas, além do controle do robô real para a coleta de dados do ambiente, foi utilizada a biblioteca Player/Stage.

O sistema de navegação desenvolvido poderá futuramente ser aplicado aos veículos terrestres, tornando possível a locomoção dos mesmos sem a necessidade do controle humano, facilitando no acesso a ambientes desestruturados considerados inacessíveis ou perigosos.

1.3 Organização

Os capítulos seguintes da monografia abordarão os seguintes conteúdos:

- Capítulo 2: Modelos de sensores mais utilizados na robótica móvel.
- Capítulo 3: Principais conceitos da robótica móvel, sendo eles, mapeamento, localização e navegação.
- Capítulo 4: Técnicas para navegação e mapeamento em ambientes externos utilizados atualmente.
- Capítulo 5: Ambiente de simulação e programação de robôs móveis Player/Stage adotado neste trabalho.
- Capítulo 6: Métodos de aprendizado de máquina utilizados no projeto.
- Capítulo 7: Desenvolvimento e os resultados obtidos do projeto.

CAPÍTULO

2

Sensores

As tarefas essênciais de um robô móvel, tais como: a estimação da sua posição, a construção do mapa local e o deslocamento de um local para outro, não seriam possíveis se o robô não pudesse interagir com o ambiente. O robô, da mesma forma como qualquer ser vivo, necessita de determinados mecanismos para capturar informações do seu meio e posteriormente gerar algum comportamento específico (Murphy, 2000). Por isso, para que o robô móvel possa atuar em qualquer ambiente, é fundamental que seja estabelecido algum método de percepção. Os mecanismos responsáveis pelo sistema de observação do robô são os sensores. As ações do robô são calculadas através das medições dos sensores e pela extração de informações significativas dessas medições (Siegwart e Nourbakhsh, 2004). Atualmente existem inúmeros sensores disponíveis no mercado, sendo que cada um deles está direcionado a um propósito específico. No decorrer desta seção serão abordados alguns dos sensores mais utilizados na robótica móvel.

2.1 Modelo Comportamental do Robô

O processo de aquisição de dados do ambiente e a sua transformação em ação é resumido por meio da Figura 2.1. Existem três fases distintas envolvidas nesse processo. Na primeira etapa, o sensor captura sinais do ambiente e transforma-os em dados interpretáveis para o computador ou para o ser humano, sendo nesse caso, referido como “observação”. Em seguida, por meio

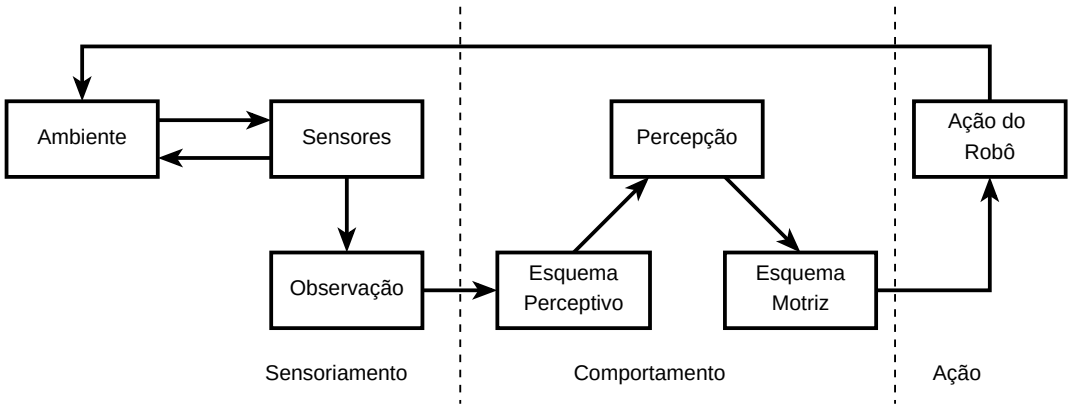


Figura 2.1: Processo de interação com o ambiente por meio de sensores.

do esquema perceptivo, esses dados são convertidos em informações úteis ao robô, como posição de objetos ou velocidade do robô, na qual é chamado de "percepção". As informações da percepção são associadas aos comandos do robô, como por exemplo, virar à esquerda devido à detecção de uma parede. Por fim, o conjunto de comandos é executado durante a fase denominada "ação" (Murphy, 2000).

2.2 Classificação dos Sensores

De acordo com (Murphy, 2000) e (Siegwart e Nourbakhsh, 2004), os sensores podem ser classificados de duas formas. A primeira classificação leva em conta o meio em que é efetuada a medição, sendo ela dividida em sensores proprioceptivos e exteroceptivos.

- Proprioceptivo: Os sensores medem valores de componentes internos do robô. Por exemplo, velocidade, carga da bateria, inclinação e altura do robô.
- Exteroceptivo: Os sensores adquirem dados do meio externo ao robô, ou seja, do ambiente ao seu redor. Por exemplo, temperatura, iluminação e distância dos obstáculos.

A segunda forma de classificação tem em vista a forma como os sensores capturam os sinais, podendo ser um sensor passivo ou ativo:

- Passivo: Sensores que recebem apenas energia do meio. Por exemplo, câmeras, microfones e termômetros.
- Ativos: Sensores que emitem energia no ambiente para realizar as medições. Por exemplo, lasers, sonares e infravermelho.

2.3 Odômetros

Os odômetros são os tipos de sensores mais utilizados na robótica, principalmente pelo fato de possuírem um baixo custo comercial, boa precisão em curtas distâncias e apresentarem alta taxa de amostragem (Borenstein *et al.*, 1997). Esses sensores possibilitam realizar a medição da velocidade do robô e da distância percorrida. Para aumentar a sua precisão, normalmente os odômetros são utilizados juntamente com outros sensores, pois, a longo prazo, o sistema de estimação da posição inevitavelmente acumula erros. Os erros de estimativa podem ser causados por diversas razões, dentre elas citamos as principais (Borenstein e Feng, 1996):

- Atrito da superfície: Dependendo do local em que o robô está se movendo, as rodas realizam mais ou menos revoluções para um mesmo espaço a ser percorrido. Isto é causado pela variação na força de atrito gerada através da fricção entre a roda e o chão. As superfícies lisas possuem a tendência de causarem maior derrapagem nas rodas. Por isso, prefere-se o uso de superfícies mais ásperas para se reduzir os erros de medição.
- Diferença entre os diâmetros das rodas do robô: Uma pequena variação no tamanho das rodas pode causar um pequeno desvio angular na trajetória do robô que poderá se acumular ao longo do tempo.

De uma forma geral, os erros de odometria são classificados em dois tipos quanto à localização da fonte geradora do erro (Borenstein e Feng, 1996):

- Erros sistemáticos: São erros causados pelas imperfeições físicas do robô (exemplo: diâmetro das rodas diferentes).
- Erros não-sistemáticos: Erros causados pela interação das rodas do robô com o solo (exemplo: derrapagem das rodas no gelo).

Os mecanismos mais comuns para o funcionamento do odômetro utilizam *encoders* ópticos (Figura 2.2). O *encoder* é formado por um disco que possui vários furos dispostos em distâncias regulares em sua extremidade. O disco está junto à roda do robô e, assim, os dois giram na mesma velocidade. No momento em que a roda gira, um feixe de luz atravessa os furos do disco, gerando uma onda quadrada que representa a freqüência de leitura desse feixe. A partir da análise da onda quadrada, são estimadas a velocidade e a distância percorrida do robô.

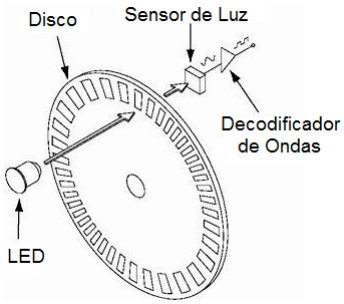


Figura 2.2: Representação de um encoder óptico (adaptado de (Borenstein *et al.*, 1997)).

2.4 Dispositivos GPS

Um recurso inicialmente exclusivo dos militares, os dispositivos GPS (do inglês, *Global Positioning System*) estão ganhando cada vez mais espaço na robótica, sendo amplamente aplicados na localização de robôs móveis. O GPS calcula a sua posição geográfica triangulando-se em relação a no mínimo quatro satélites, determinando a localização através das diferenças no tempo existentes para a chegada das informações dos satélites (Borenstein *et al.*, 1996).

Em geral, a precisão do GPS está em torno de 15m. A precisão do GPS pode ser influenciada pelo número de sinais de satélites recebidos, a disposição geométrica dos satélites e a existência de construções próximo ao receptor (Balaguer e Carpin, 2008). Porém, existe um método denominado DGPS (do inglês, *Differential* GPS) no qual associa dois receptores GPS, aumentando-se a precisão para 10cm. O primeiro sensor é fixado em um ponto cuja posição é conhecida e apenas o segundo GPS é utilizado pelo usuário. Dessa forma, os erros acumulados no GPS fixo podem ser determinados e em seguida descontados do segundo GPS (Borenstein e Feng, 1996)(Murphy, 2000).

Uma grande desvantagem destes sensores é a impossibilidade de utilização em ambientes internos. Analogamente aos aparelhos celulares, o seu uso pode ser comprometido devido à perda dos sinais causada pelas estruturas das construções.

2.5 Sensor Laser

Os sensores laser ou lidar (do inglês, *light detection and ranging*) permitem coletar dados do ambiente por meio de repetitivas emissões de radiação eletromagnética ao meio (Figura 2.3(a)). Para cada pulso de laser emitido pelo sensor, é medido o tempo decorrido entre a sua saída e a sua respectiva captura pelo sensor. Com esse tempo, é calculada a distância dos objetos à sua frente (Borenstein *et al.*, 1997).

(a) (b)

Figura 2.3: (a) Funcionamento de um sensor laser (adaptado de (Siegwart e Nourbakhsh, 2004)). (b) Modelo de sensor laser SICK LMS 200.

Sensores tais como o SICK LMS 200, mostrado na Figura 2.3(b), apresentam boa precisão dos dados coletados. Com eles, pode-se ter uma resolução angular de 0,5°, com alcance de até 81,90m, coletando 180 pontos (dados) do ambiente a uma freqüência de no máximo 75Hz. Uma característica deste sensor que deve ser levada em consideração é que este realiza apenas leituras planares, sendo assim, chamados de sensores 2D. Para cada ponto capturado são retornados dois valores: a distância e o ângulo do ponto em relação à direção de emissão do feixe do laser. Existem modelos de sensores lasers 3D, porém são muito mais caros e, em geral, não apresentam uma precisão tão boa quanto aos sensores 2D. Por serem dispositivos que emitem luz, estes sensores são incapazes de detectar superfícies que não refletem perfeitamente a luz, como no caso de vidros.

2.6 Sonares

Tendo um princípio de funcionamento similar ao do sensor laser, o sonar (do inglês, *sound navigation and ranging*) ou sensor ultrasônico utiliza o som para detectar os objetos que estão à sua volta. O sonar, como mostrado na Figura 2.4, possui uma membrana que vibra de acordo com pulsos de uma onda, emitindo um sinal sonoro com uma determinada freqüência, gerando-se, um "cone"que varre uma área do ambiente. O som refletido, ou ecoado, é captado pela mesma membrana, gerando um sinal que é amplificado e posteriormente filtrado. Quando o sinal recebido é muito fraco, considera-se o som captado como sendo uma forma de ruído (Borenstein *et al.*, 1996). Caso contrário, o sinal é validado e computado o tempo gasto para sair e voltar à membrana. A partir desse valor, pode-se determinar a distância do objeto ao sensor. Como os sonares possuem o campo de visão em formato de cone e retornam apenas a distância radial até

Figura 2.4: Modelo de um sonar. A parte circular corresponde à membrana e o circuito que se interliga é encarregado de analisar o som capturado pelo sensor (Murphy, 2000).

o obstáculo mais próximo, não são capazes de informar o ângulo em que o objeto foi detectado (Borenstein e Koren, 1991). Um modo de contornar esse problema é o uso associado de vários sonares para se ter uma melhor estimação da localização do objeto.

Apesar de serem dispositivos baratos e apresentarem uma razoável faixa de alcance, os sonares apresentam algumas limitações (Murphy, 2000). Um dos problemas que pode ocorrer é a reflexão do som em superfícies com ângulos agudos, fazendo com que o som não volte ao sonar. Este fato é denominado reflexão especular ou regular. O ideal seria refletir o som em superfícies planas e paralelas em relação à membrana do sonar.

2.7 Considerações Finais

Os sensores são uns dos dispositivos mais fundamentais da robótica. É por meio deles que os robôs conseguem capturar as informações do ambiente. Cada tipo de sensor possui um método específico para a coleta de dados do meio, sendo assim, a escolha do sensor implica diretamente no modo como o robô irá interagir com o ambiente.

Este capítulo apresentou alguns dos sensores mais utilizados na robótica, sendo eles: odômetro, sensor laser e sensor sonar. Foram descritos o funcionamento, vantagens e desvantagens de cada um. No Capítulo 3 serão apresentados métodos computacionais fundamentais que possibilitam a atuação do robô no ambiente utilizando os dados coletados pelos sensores.

CAPÍTULO

3

Algoritmos Fundamentais da Robótica Móvel

A robótica móvel autônoma é uma subárea da robótica que tem como objetivo desenvolver dispositivos capazes de se deslocarem sem o controle humano para cumprir determinadas tarefas. Dentre as principais vantagens dos robôs autônomos está a aplicação industrial com a substituição parcial ou total do homem em trabalhos pesados e arriscados, aumentando-se a produtividade e a segurança do trabalhador (Thrun, 2003). A área militar tem investido muito na robótica, com a construção de veículos autônomos para evitar a perda de grande quantidade de soldados durante as suas operações (Bishop, 2000) (Rasmussen, 2002). Com o barateamento dos componentes eletrônicos, vem se observando o crescimento do número de robôs domésticos disponíveis no mercado. Exemplos de robôs à venda estão: robôs aspiradores, robôs cortadores de grama, máquinas agrícolas autônomas, além de inúmeros robôs com fins de entretenimento.

Sendo uma área multidisciplinar, a robótica móvel envolve problemas de diversas áreas como a Física, a Matemática, a Engenharia e principalmente a Computação. Esta última compromete-se com a criação de programas de forma a tornar o robô capaz de tomar as decisões corretas para a execução das tarefas. Em outras palavras, a Computação tem como objetivo desenvolver a parte "inteligente" do robô. O sistema inteligente consiste no controle do robô, que é composto por três elementos: sensoriamento, planejamento e ação (Gat, 1998). O

processo de sensoriamento permite ao robô capturar informações do ambiente para calcular a posição do mesmo e dos obstáculos à sua volta. O planejamento permite determinar as rotas a serem percorridas pelo robô utilizando os dados do ambiente. E na ação, o robô irá executar as operações mecânicas necessárias para seguir as rotas estabelecidas na etapa de planejamento.

Neste capítulo serão apresentados conceitos básicos sobre mapeamento, localização e navegação. Estes assuntos são muito importantes para compreender as principais idéias envolvidas no projeto, as quais são mapeamento e navegação. Dessa forma, aqui ainda não será focada a atuação dos robôs móveis em ambientes externos, mas apenas em ambientes internos. Com isso, o objetivo deste capítulo é a de introduzir o conceito de robótica móvel. Assim, não serão necessariamente apresentadas as técnicas e os métodos mais inovadores, porém, os mais importantes.

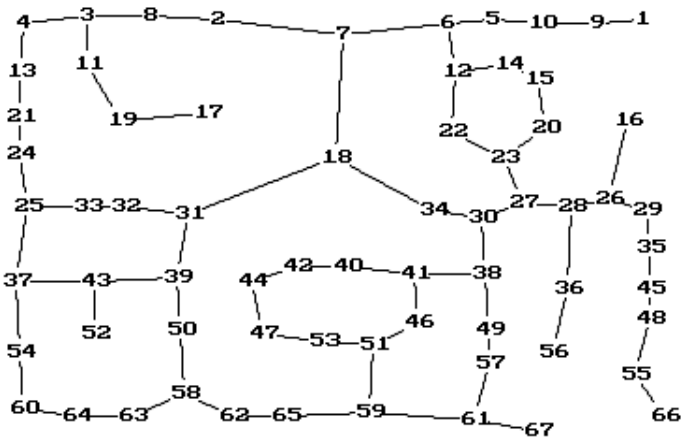
3.1 Mapeamento

Uma das funções fundamentais do robô para que possa cumprir corretamente suas tarefas, tanto em ambientes internos como nos externos, é a aquisição de informações espaciais do ambiente em que se situa através da construção de mapas do local. Sem esses mapas, o cálculo da posição relativa dos objetos ao redor do robô e também a determinação das rotas a serem seguidas seriam mais difíceis e imprecisas (Kortenkamp *et al.*, 1998). Com isso, é importante que o robô consiga realizar corretamente o mapeamento do local, pois mapas imprecisos podem implicar na falha de interpretação do posicionamento do robô, causando a execução imperfeita de suas operações. Para o processo de mapeamento, é essencial que o robô esteja equipado com sensores. Dentre os sensores mais utilizados para esta tarefa estão: sensores laser, câmeras, odômetros, dispositivos GPS e sonares. Os principais sensores são apresentados no capítulo sobre sensores (Capítulo 2).

Geralmente, os mapas utilizados pelos robôs são classificados, de acordo com a sua estrutura, em dois tipos: métricos ou topológicos (Murphy, 2000) (Thrun, 2003). Os mapas métricos normalmente são representados através de um plano dividido em células de tamanhos iguais, que também é chamada de grade (ou *grid*). Cada célula representa uma porção fixa do espaço e armazena um valor que indica o estado desta célula como, por exemplo, a presença ou a ausência de um obstáculo (Thrun, 1998). Na Figura 3.1(a) é ilustrado um exemplo de mapa métrico, sendo que os pontos escuros representam obstáculos e os claros, regiões livres. O mapa métrico mais comum é a grade de ocupação que será visto mais à frente.

No caso dos mapas topológicos, diferentemente do método anterior, o ambiente é representado por meio da conexão entre pontos de referência (Thrun, 2003). Assim, nesse mapa, são utilizados grafos para a representação do ambiente, onde os nós representam determinados

(a) Mapa Métrico



(b) Mapa Topológico

Figura 3.1: Tipos de mapas utilizados na robótica (Thrun, 2003).

locais (por exemplo, cômodos de uma casa) e as arestas são utilizadas para indicar a existência de algum caminho entre esses lugares (Figura 3.1(b)). Os mapas métricos possuem a vantagem de serem mais robustos, pois garantem uma localização precisa e apresentam um bom funcionamento em locais fechados. Por outro lado, os mapas topológicos, por serem mais compactos (em termos de armazenamento), facilitam o planejamento rápido de rotas do robô (Congdon *et al.*, 1993) (Thrun e Bücken, 1996). Cada uma destas representações será vista com mais detalhes nas próximas subseções.

3.1.1 Grades de Ocupação

O mapeamento por grades de ocupação (também conhecido como *occupancy grid* ou *certainty grid*) foi introduzido por Hans Moravec e Alberto Elfes em 1985 (Moravec e Elfes, 1985) e até hoje é uma das representações métricas mais utilizadas (Thrun *et al.*, 2000). A idéia central das grades de ocupação é gerar os mapas levando-se em conta a imprecisão e o ruído presente nas medições feitas pelos sensores dos robôs (Thrun *et al.*, 2005).

Nesta técnica utiliza-se uma área reticulada, onde cada célula armazena um valor binário, indicado por m, que especifica se esse campo está ocupado ($m = 1$) ou livre ($m = 0$). Dessa forma, conhecendo-se de antemão a posição x e o conjunto das medições z realizadas pelo robô, a grade de ocupação utiliza essas informações para calcular a probabilidade a posteriori de todas as células do mapa. A probabilidade é calculada por meio da seguinte expressão:

$$p(m|z, x) \tag{3.1}$$

Em muitos casos, a dimensão do mapa pode ser muito grande, chegando a milhares de células. Em um mapa com n células e considerando que cada uma pode armazenar dois valores (uma para indicar que a célula está vazia e outra para indicar que está ocupada), o número de mapas diferentes que poderá ser representado será 2^n. Isso torna o problema extremamente complexo para valores de n muito grandes. Para contornar essa restrição, o cálculo da probabilidade a posteriori é divido em subproblemas menores. Sendo assim, ao invés de calcular diretamente a probabilidade do mapa, calcula-se a probabilidade de cada uma das células, multiplicando-as no final (Thrun *et al.*, 2005). Entretanto, deve-se observar que essa simplificação não permite mais assumir dependência entre as células.

Outro conceito importante existente na grade de ocupação é o "modelo inverso de sensor" (ou *inverse sensor model*). Esse modelo expressa a probabilidade de uma célula estar ocupada por um obstáculo a partir das leituras realizadas pelo sensor, existindo um modelo específico para cada tipo de sensor (Thrun, 2003). Por exemplo, no modelo de um sonar, as células que se encontram na faixa do arco do cone têm o valor da probabilidade de ocupação incrementada, enquanto que a probabilidade de ocupação das células internas ao cone de leitura é decrementada. Na Figura 3.2 é apresentado o modelo de sensor inverso de um sonar, onde a probabilidade de ocupação das células é representada por meio de cores. As cores mais escuras remetem a uma probabilidade de ocupação alta e cores mais claras o contrário.

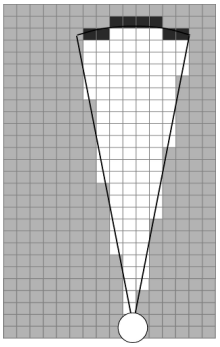


Figura 3.2: Modelo de sensor inverso de um sonar. Células claras denotam regiões livres e células escuras, áreas ocupadas (Thrun, 2003).

3.1.2 Mapas Topológicos

Ao contrário dos mapas métricos, os mapas topológicos são as representações que mais se assemelham à maneira de como o homem se orienta em um ambiente desconhecido. Nesses mapas, os elementos do ambiente são utilizados como pontos de referência para a localização do robô. Assim, objetos como portas, cadeiras e marcações podem ser aproveitados para a orientação do robô.

De acordo com o objeto utilizado, os pontos de referência podem ser naturais ou artificiais:

- Pontos de referência naturais: Compreende-se do conjunto de elementos que a princípio não foram construídos para serem utilizados como marcos de referência. Tais elementos podem ser casas, monumentos, prédios e árvores.
- Pontos de referência artificiais: Consiste nas características adicionadas aos objetos de forma a oferecer suporte ao reconhecimento do ponto de referência. Esses elementos são facilmente encontrados no trânsito, como os adesivos refletivos nas pistas e placas de trânsito com figuras e cores para facilitar a identificação delas.

Como os mapas topológicos dependem apenas de características do ambiente, a estrutura do mapa é constituída basicamente dos pontos de referência e das ligações entre eles. Com o intuito de facilitar a representação computacional, os mapas são construídos por meio de grafos, onde os nós representam os pontos de referência e as arestas indicam a interligação entre os pontos de referência. Na Figura 3.3 é ilustrado o mapa topológico de um escritório. As salas e os corredores são representados por meio de nós no grafo e as arestas indicam a existência de uma passagem de um local para outro. Em comparação com os mapas métricos, os mapas topológicos têm a vantagem de serem mais compactos, pois armazenam poucos elementos do cenário em sua estrutura. Porém, estes últimos, devido à sua representação simplificada, não

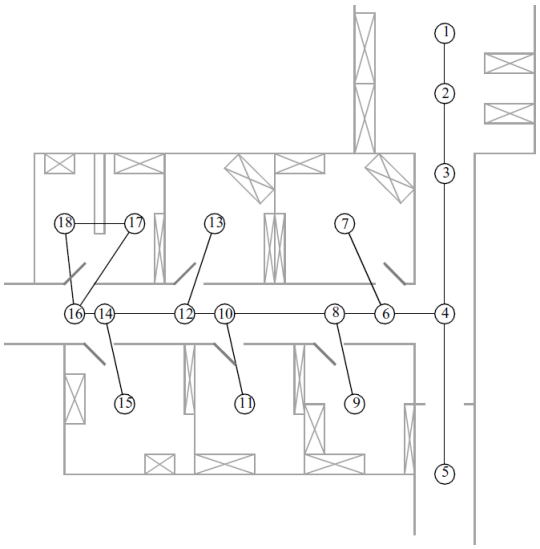


Figura 3.3: Mapa topológico de um escritório. Os nós representam os cômodos e as arestas indicam a existência de uma passagem entre os nós (Siegwart e Nourbakhsh, 2004).

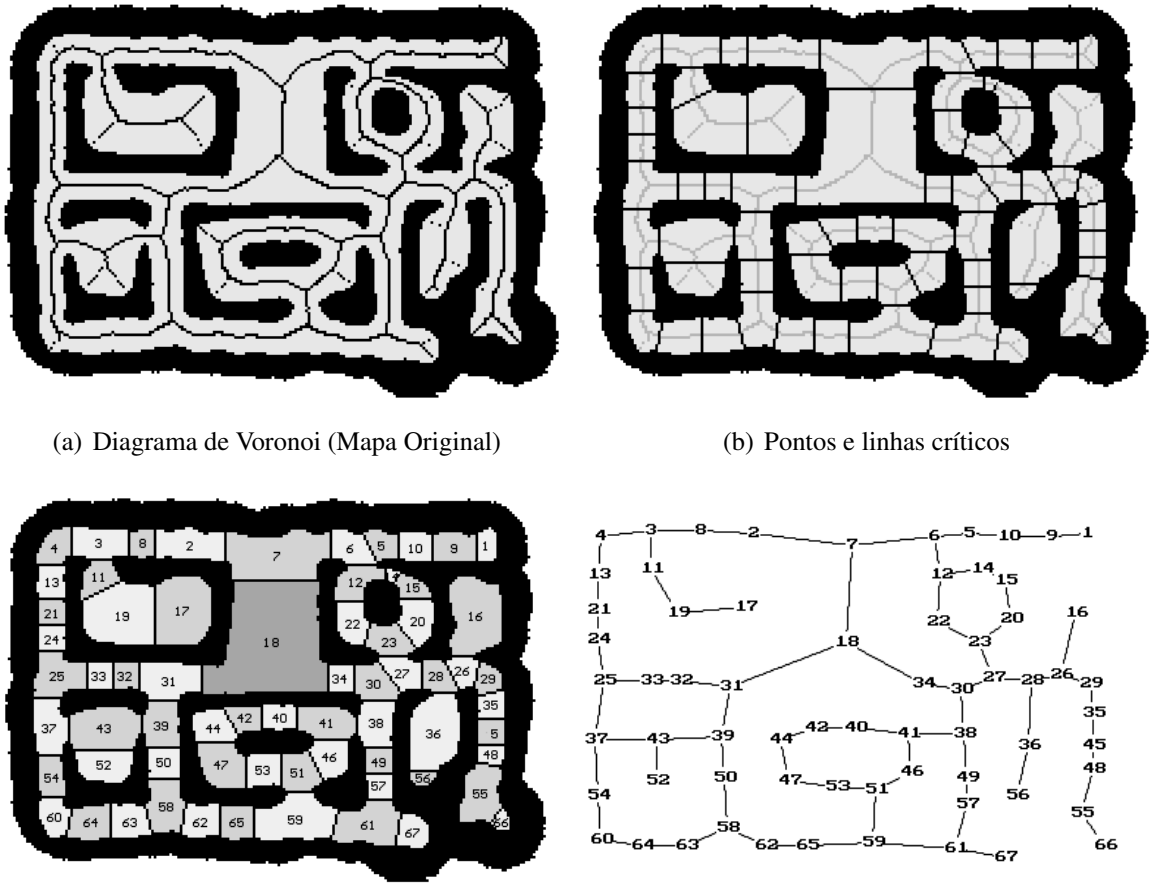
retratam a geometria do ambiente e, com isso, não possibilitam determinar com precisão a localização do robô no ambiente.

Existem diversos trabalhos sobre técnicas de criação de mapas topológico na literatura como (Kuipers e tai Byun, 1991), (Mataric, 1990) e (Mata *et al.*, 2003). Em (Thrun *et al.*, 1998b) é apresentado um método para conversão dos mapas métricos em mapas topológicos, resultando em representações mais compactas. Esta técnica consiste em seccionar os espaços livres do mapa métrico para determinar os nós do mapa. A Figura 3.4 ilustra os resultados decorrentes dos passos realizados nesse método. A redução da quantidade de informações para a descrição do ambiente pode ser constatado pela comparação do mapa métrico original(Figura 3.4(a)) com o mapa topológico resultante (Figura 3.4(d)).

3.2 Localização

O problema de localização de robôs móveis não é o foco deste trabalho, porém, é um assunto importante diretamente relacionado com o processo de navegação e mapeamento. Como será visto, vários métodos de localização dependem dos mapas do ambiente para estimar a posição do robô de forma precisa.

Analogamente aos seres humanos, para um robô autônomo chegar a um local do mapa é necessário conhecer a sua localização. Em teoria, tendo-se o mapa prévio do ambiente, o problema de localização do robô poderia ser resolvido com métodos de transformação de coordenadas. Nesse processo, as coordenadas locais obtidas pelo odômetro seriam transformadas



(a) Diagrama de Voronoi (Mapa Original)

(b) Pontos e linhas críticos

(c) Divisão em regiões

(d) Mapa final

Figura 3.4: Passos para a criação de um mapa topológico a partir de um mapa métrico (Thrun *et al.*, 1998b).

em coordenadas relativas ao mapa (coordenadas globais). Entretanto, como normalmente a leitura realizada pelo odômetro é afetada por ruídos (mais detalhes no Capítulo 2.3), é necessário o uso de técnicas de localização que levem em conta esse fator.

Uma das soluções amplamente utilizadas é associar informações provindas de outros sensores ao odômetro para aumentar a precisão da localização (Thrun, 1998) (Siegwart e Nourbakhsh, 2004). O uso conjunto de sensores exteroceptivos (por exemplo, sensor laser) com o odômetro possibilita a utilização de técnicas que associam as leituras atuais com as anteriores para se estimar a posição atual. Pelo fato das leituras dos sensores serem armazenadas nos mapas, é essencial que esses últimos não contenham erros para evitar falhas no processo de localização. De uma forma geral, o problema de localização pode ser classificado em três tipos (Thrun *et al.*, 2005) (Borenstein *et al.*, 1996) (Burgard *et al.*, 1996):

- Rastreamento da Posição (ou *Position Tracking*): Nesse problema, considera-se que a posição inicial do robô em relação ao mapa (posição global) é conhecida. Entretanto, com a movimentação do robô, a localização é prejudicada devido aos erros de leitura do sensor, como por exemplo, o odometro. O problema de rastreamento da posição tem como foco estimar esses erros, de forma a calcular corretamente a posição local do robô. Normalmente os erros são aproximados por meio de uma distribuição unimodal (por exemplo, distribuição normal).

- Localização Global (ou *Global Localization*): Ao contrário do problema de rastreamento da posição, na localização global o robô não conhece a sua posição inicial no mapa. Dessa forma, são buscadas estratégias para se estimar a posição inicial do robô, considerando inclusive, o problema de rastreamento da posição.

- Problema do Robô Raptado (ou *Kidnapped Robot Problem*): Esse problema é uma extensão da localização global. No problema do robô raptado, assume-se que a qualquer momento, o robô poderá ser transferido para outra posição sem ser informado. Caso isso ocorra, o robô deverá detectar a alteração em sua posição e logo em seguida, calcular a sua nova localização. Em termos práticos, a ocorrência dessa situação é mínima. Porém, esse problema pode ser utilizado para verificar a capacidade de recuperação do robô em falhas repentinas, de forma que possa manter o sistema de localização intacto.

Atualmente existem diversas técnicas que possibilitam estimar precisamente a posição do robô. A maioria delas tem foco nos problemas de rastreamento da posição. Algumas das abordagens podem ser encontradas em (Schiele e Crowley, 1994). Em relação ao problema de localização global, muitos focam o uso de métodos estatísticos (Thrun *et al.*, 2000). Entre eles destacam-se: Filtro de Kalman (Smith *et al.*, 1990), método de Monte Carlo (Dellaert *et al.*, 1999) e método de *grid* (Fox, 1998).

Com o intuito de ilustrar o funcionamento do processo de localização, é representado na Figura 3.5 o funcionamento do algoritmo de localização por método de Markov. A primeira figura (item a) mostra um robô percorrendo um corredor, mas não se sabe a sua posição. O gráfico da função $bel(x)$ (ou função *belief*) representa a estimação da posição calculada pelo robô. No momento em que encontra uma porta (item b), o algoritmo de localização logo conclui que o robô está em uma das três portas possíveis. Quando o robô não observa mais nenhum ponto de referência, o valor de $bel(x)$ diminui (item c). Ao identificar a próxima porta, o algoritmo mostra que há uma grande probabilidade do robô estar na segunda porta, já que esta é a mais próxima da primeira (item d). Por fim, quando o robô se locomove, $bel(x)$ é reduzido novamente devido à incerteza de sua posição (item e).

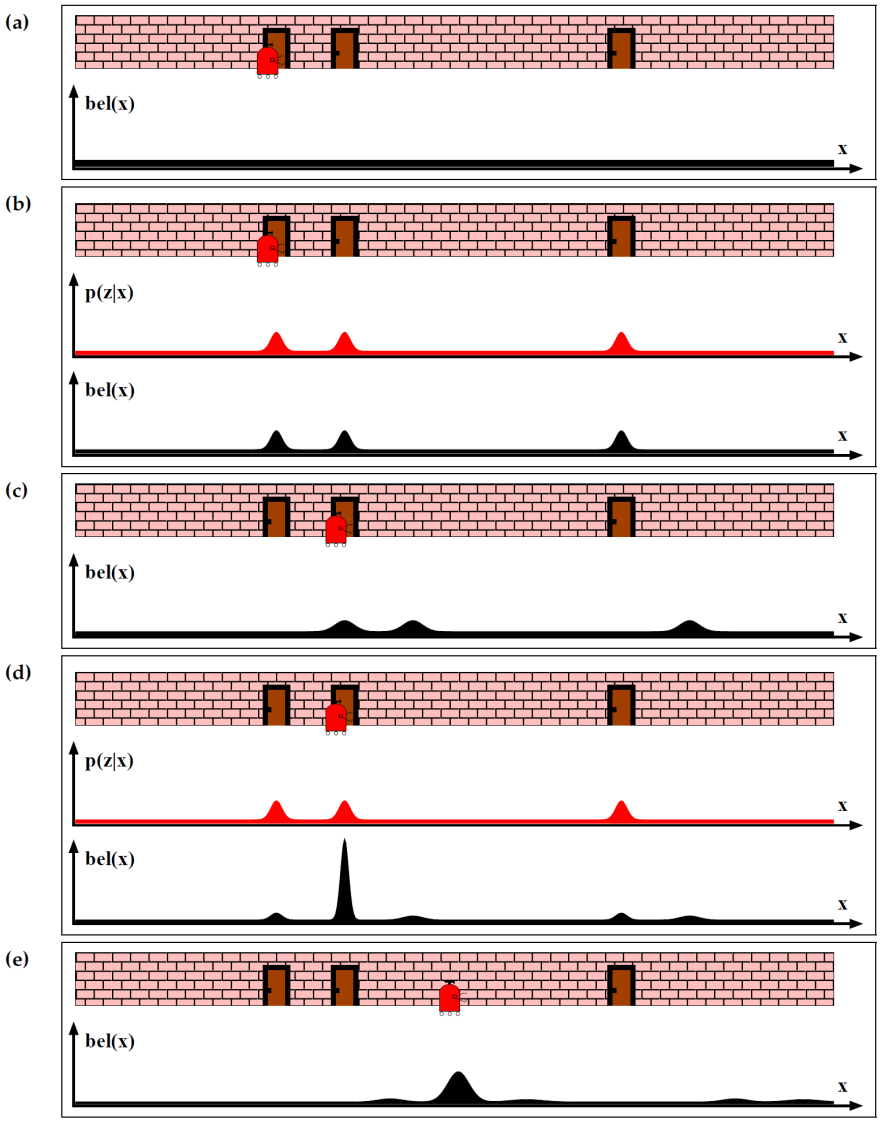


Figura 3.5: Representação do funcionamento do algoritmo de localização por Markov (Thrun, 2003).

3.3 Navegação

Como visto na Seção 3.1 e na Seção 3.2, os métodos de mapeamento e de localização são algoritmos básicos necessários para permitir a interação do robô com o ambiente. Entretanto, o uso exclusivo desses métodos não é suficiente para o robô calcular as rotas que levem aos seus locais de destino. Dessa forma, os métodos de navegação são introduzidos para definir estratégias que permitam ao robô deslocar de um ponto a outro do mapa (Siegwart e Nourbakhsh, 2004). Para determinar essas rotas, podem-se priorizar certas restrições, como por exemplo, chegar ao destino em menor tempo possível ou gastar o mínimo de combustível (Murphy, 2000).

De acordo com (Lin *et al.*, 2006) o processo de navegação é composto pelos seguintes componentes:

- Mapeamento
- Localização
- Planejamento de rotas
- Desvio de obstáculos

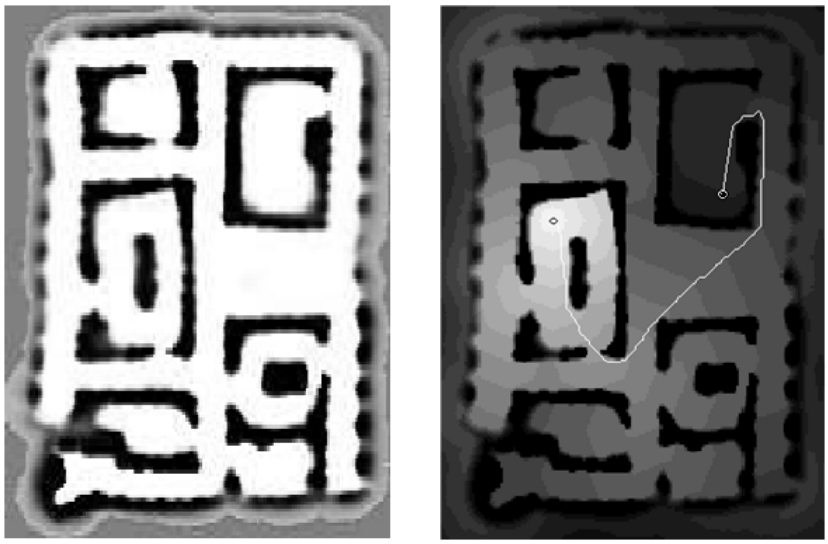
O problema de planejamento de rotas discute metodologias para se estabelecer os melhores caminhos que levam o robô de um ponto A até B. Os algoritmos de planejamento de rotas são divididos, de acordo com o mapa utilizado, em duas categorias: os que utilizam mapas topológicos e os que utilizam mapas métricos (Heero, 2006). Obtidos os caminhos, o algoritmo de desvio de obstáculos é utilizado para determinar as operações mecânicas que o robô deverá tomar, de forma que trafegue na rota sem colidir com obstáculos.

3.3.1 Planejamento de Rotas em Mapas Topológicos

Como os mapas topológicos estão estruturados por meio de grafos, podem-se aproveitar algoritmos de busca em grafos para gerar os caminhos de menor custo. Nesses algoritmos, dado um grafo de entrada, um vértice de partida e um vértice de chegada, será gerado na saída, um conjunto de vértices e arestas do grafo que interliga o vértice de partida ao vértice de chegada, sendo que a soma dos pesos dessas arestas deverá ser o menor possível. Alguns dos algoritmos de busca que podem ser utilizados são: Dijkstra, Floyd, Warshal, A* e programação dinâmica (Thrun *et al.*, 1998b).

3.3.2 Planejamento de Rotas em Mapas Métricos

Existem inúmeros algoritmos para navegação em mapas métricos. Uma das técnicas mais utilizadas consiste na conversão do mapa métrico em um grafo, considerando cada célula como um nó e as arestas, as conexões entre as células. A partir disso, podem-se aplicar algoritmos de busca de caminhos em grafos como A* ou D*. Existem também outros que são derivados destes, como *Incremental* A*, *Focussed* D*, D* *lite* e *Delayed* D* (Heero, 2006). Na Figura 3.6 é ilustrado o resultado de um método de planejamento de rotas proposto em (Thrun *et al.*, 1998a). Esse método tem como base a programação dinâmica e o algoritmo A* para gerar as rotas mais curtas. O caminho é gerado escolhendo-se sempre as células de menor custo.



(a) (b)

Figura 3.6: (a) Mapa original. (b) Caminho de custo mínimo obtido pela técnica apresentada por (Thrun *et al.*, 1998a).

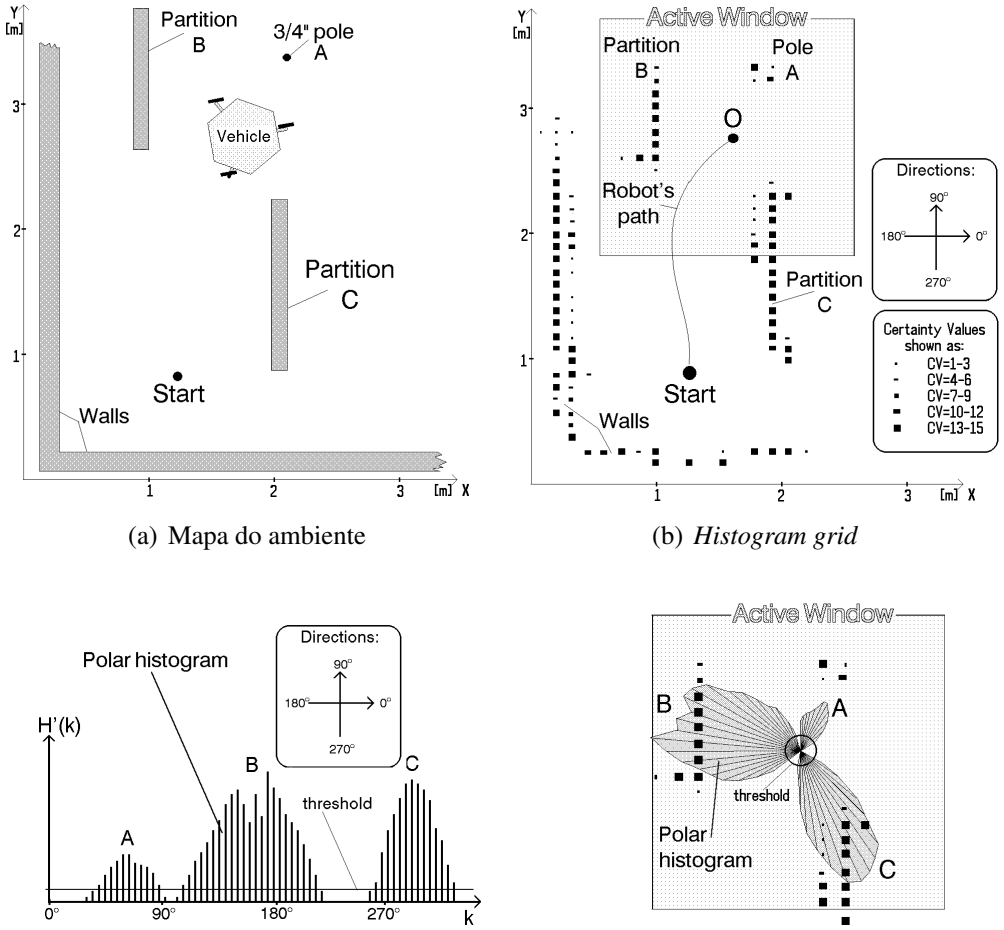
3.3.3 Desvio de Obstáculos

Os métodos de desvio de obstáculos permitem que o robô percorra da maneira mais segura possível a trajetória definida no planejamento de rotas. Basicamente as rotinas que compõem o desvio de obstáculos controlam a velocidade e orientação do robô, fazendo com que, por exemplo, o robô não realize conversões muito bruscas a fim de evitar colisões com as paredes ou outros obstáculos. Os métodos de planejamento de ações e de desvio de obstáculos podem ser classificados em dois grupos: globais e locais (ou reativos) (Siegwart e Nourbakhsh, 2004).

- Globais: Neste método, são determinados todos os movimentos a serem efetuados pelo robô no trajeto, antes de iniciar a navegação do robô. Assim, esta abordagem depende que os mapas estejam livres de erros e de imprecisões para que o algoritmo seja eficiente. E como, o cálculo é realizado em uma única etapa para toda a trajetória, dependendo da sua extensão, pode-se ter um custo computacional elevado.

- Locais: Diferentemente da abordagem global, este método avalia apenas porções do mapa que estão próximas ao robô para se gerar os movimentos. Dessa forma, apresenta um baixo custo computacional. Porém, as técnicas locais para desvio de obstáculos têm o risco de caírem em mínimos locais, principalmente em regiões em formato de "**U**".

Dentre os métodos de desvio de obstáculos globais podemos citar road-maps, decomposição celular (ou *cell decomposition*) e campos potenciais (ou *potential fields*). Os métodos de desvio



(a) Mapa do ambiente

(b) *Histogram grid*

(c) *Polar histogram*

(d) *Polar histogram* sobreposto ao *histogram grid*

Figura 3.7: Detecão de obstáculos por VFH (Borenstein e Koren, 1991).

de obstáculos locais, normalmente apresentam dois estágios distintos (Fox *et al.*, 1997). A primeira consiste em adquirir a direção do movimento e na segunda, são gerados os comandos que irão movimentar o robô na direção especificada (Borenstein e Koren, 1990) (Khatib, 1986). Outro estágio levantado em (Fox *et al.*, 1997) é o cálculo da inércia do robô para permitir, por exemplo, a realização de conversões em curvas estreitas em altas velocidades.

Em relação às técnicas de detecção de obstáculos locais, podemos citar o método de detecção de bordas (ou *edge-detection*) (Borenstein e Koren, 1991). Neste, como o próprio nome diz, o robô toma como referência a posição das bordas verticais existentes no cenário para contorná-los. Uma desvantagem presente neste método é a necessidade do robô ter que permanecer parado em frente ao obstáculo para coletar informações. Outra restrição é a necessidade de sensores precisos para efetuar a análise do ambiente. Outra técnica amplamente utilizada é o VFH (do inglês, *Vector Field Histogram*) (Borenstein e Koren, 1991). Nesta técnica, o algo-

ritmo conta com um mapa de histogramas, denominado *histogram grid* que mantém a confiança da existência de um obstáculo em um determinado local (Figura 3.7(b)). Pode-se perceber que esse mapa possui características similares às grades de ocupação. A partir do *histogram grid*, constrói-se outro gráfico denominado *polar histogram*, o qual tem como objetivo restringir a quantidade de informações do primeiro mapa para focar apenas nos obstáculos ao redor do robô (Figura 3.7(c)). Dado um limiar, o robô escolhe as regiões do gráfico desse valor para escolher seu próximo passo (Figura 3.7(d)).

Outro algoritmo amplamente utilizado são os campos potenciais (ou *potencial fields*), pois são fáceis de serem implentados (Borenstein e Koren, 1990). Nesse método, cada obstáculo do mapa possui uma força repulsiva e o destino, uma força atrativa ao robô. Deste modo, o robô é tratado como um ponto no mapa influenciado pelos campos atrativos e repulsivos e, por meio, deles é obtido o modelo do movimento do robô (Figura 3.8). Entretanto esse método nem sempre garante as melhores soluções, pois existe a possibilidade de encontrar mínimos locais (Pimenta *et al.*, 2006). Para contornar esse problema pode-se associar a essa técnica as transformadas de frente de ondas (Batavia e Nourbakhsh, 2000).

A abordagem proposta em (Thrun *et al.*, 1998a) denominada de janela dinâmica (ou *dynamic window*) propõe uma busca de controles admissíveis em um espaço de velocidades do robô.

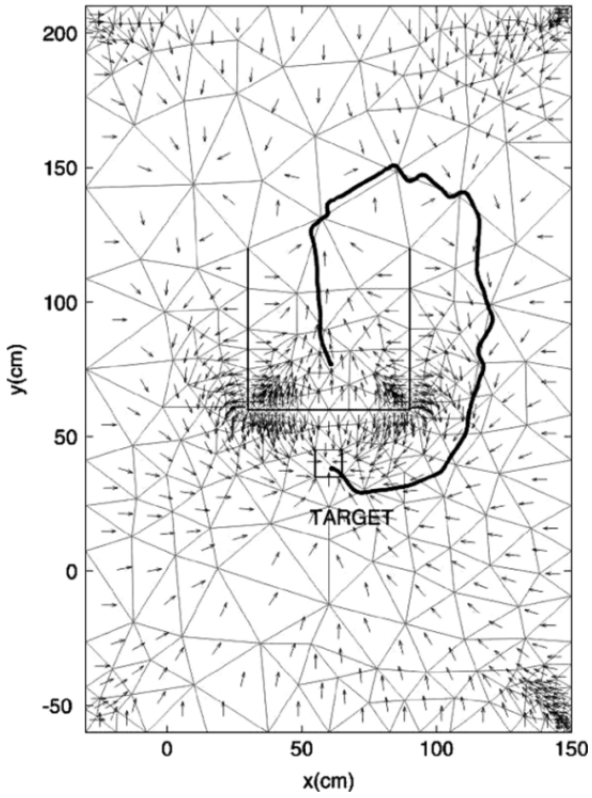
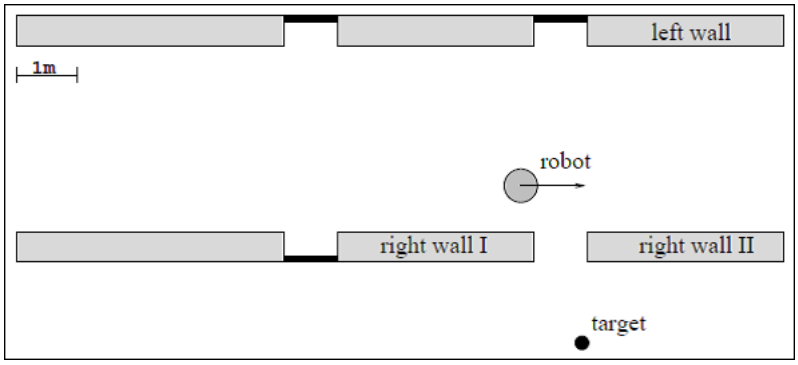
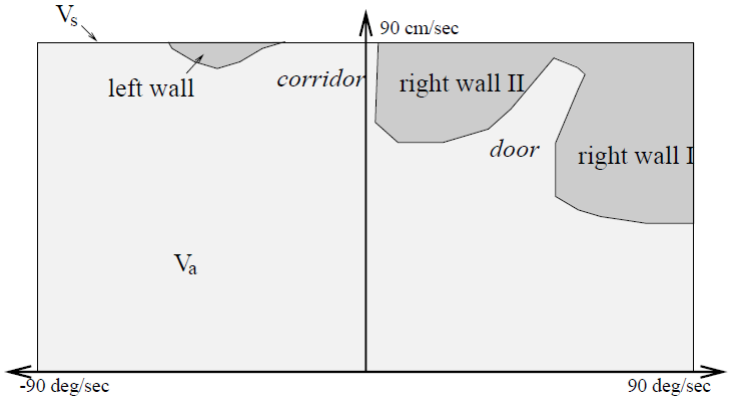


Figura 3.8: Rota gerada através dos campos potenciais (Borenstein e Koren, 1990).



(a)



(b)

Figura 3.9: (a) Cenário percorrido pelo robô. (b) Espaço de velocidade da visão local do robô (Thrun *et al.*, 1998a).

Essa técnica permite que robôs trafeguem a altas velocidades, tomando vantagem sobre outros métodos. O espaço de velocidade é um gráfico contendo a representação local do ambiente, sendo que dado um ponto para o qual se deseja se deslocar é obtida a velocidade linear e angular para que o robô possa se locomover com segurança até esse local. Além disso, como o seu algoritmo não apresenta uma complexidade elevada, pode-se garantir a eficiência do método. A Figura 3.9 mostra um exemplo do espaço de velocidade de um corredor.

3.4 Considerações Finais

Os algoritmos apresentados são essenciais para a robótica móvel, na medida que sem eles não seria possível realizar a navegação autônoma dos robôs. Um algoritmo de navegação ideal conta com quatro componentes: mapeamento, localização, planejamento de rotas e desvio de obstáculos. O mapeamento tem como função gerar informações espaciais sobre o ambiente ao redor do robô. Na localização, a posição do robô é determinada considerando-se os erros obtidos

na leitura de seus sensores. O algoritmo de planejamento de rotas calcula o melhor trajeto a ser percorrido pelo robô para se chegar ao seu destino. E por fim, no desvio de obstáculos são determinados os movimentos necessários de forma que o robô trafegue pelo trajeto sem colidir com obstáculos.

CAPÍTULO

4

Navegação e Mapeamento de Ambientes Externos Utilizando Robôs Móveis

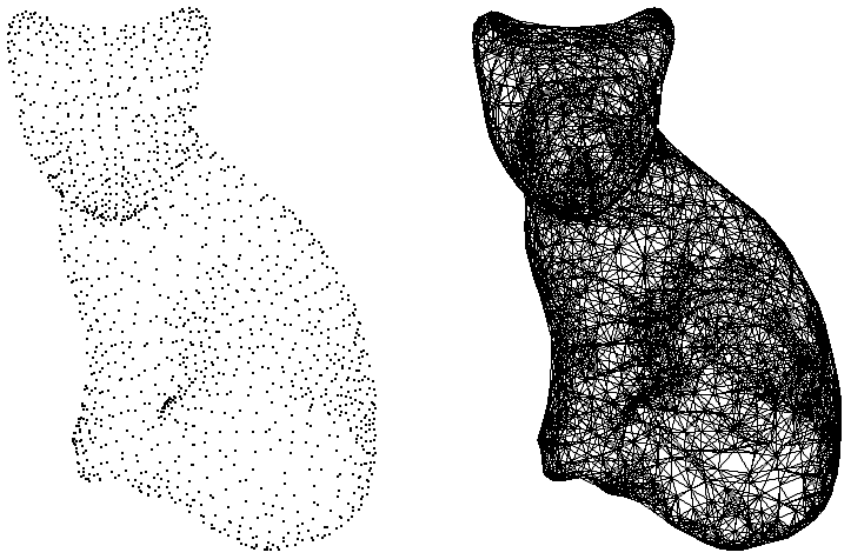
A abordagem realizada no Capítulo 3 teve como objetivo introduzir os elementos básicos presentes no software de um robô móvel autônomo. Entretanto, muitos dos algoritmos e métodos apresentados aplicam-se apenas aos robôs que trafegam em ambientes internos, como, por exemplo, dentro de um andar de edifício. Nesses casos, o ambiente possui apenas objetos estruturados (caixas, portas e paredes), não exigindo, portanto, análises muito complexas do cenário. Quando se trata de um ambiente externo, existem inúmeros parâmetros adicionais a serem ponderados. A saber, um robô que trafega em um cenário urbano deverá ter a capacidade de diferenciar entre regiões trafegáveis e não trafegáveis, deverá considerar inúmeros obstáculos imprevistos (pedestres e animais), levar em conta novos fatores de ruídos em seus sensores (iluminação, tipo de terreno), além de diversos outros aspectos. Por essa razão, muitas vezes os robôs que atuam em ambientes externos exigem algoritmos mais complexos e robustos comparados aos que atuam apenas em ambientes internos. Conseqüentemente, os algoritmos utilizados para operar robôs em ambientes estruturados necessitam de adaptações para atuarem em ambientes externos.

Neste capítulo serão descritos dois componentes fundamentais para a operação de robôs em ambientes externos: mapeamento e navegação. Serão também discutidas algumas aplicações robóticas que estão sendo pesquisadas atualmente na comunidade científica.

4.1 Representação Computacional de Ambientes Externos

Para permitir que o robô navegue em um ambiente desestruturado é essencial que o mesmo realize o mapeamento do cenário ao seu redor para identificar as regiões livres de obstáculos e seguir a sua rota com segurança. Normalmente, robôs que navegam em locais internos trabalham bem com mapas bidimensionais (ou mapas 2D), já que ambientes desse tipo são altamente estruturados e não apresentam alta complexidade. Em relação aos ambientes externos, métodos de representação como mapas topológicos e métricos (por exemplo, grades de ocupação) são insuficientes para retratar detalhadamente todo o cenário. Isto se justifica pelo fato de que os mapas topológicos são muito simples e os mapas métricos, no caso das grades de ocupação, exigiriam alta capacidade de armazenamento para representar todo o ambiente (Thrun *et al.*, 2004). Para resolver esse problema, são adotados métodos de representação tridimensionais utilizados na visualização computacional. Entre as diversas representações de mapas tridimensionais (ou mapas 3D) que podemos encontrar na literatura, podemos citar algumas mais utilizadas como, nuvem de pontos (ou *point clouds*) e malhas triangulares (ou *triangular meshes*).

Os mapas do tipo nuvem de pontos são representações do ambiente construídas a partir dos dados brutos obtidos por meio de sensores laser. Como tais dados consistem basicamente de um conjunto coordenadas de pontos, os mapas resultantes têm aspecto de uma malha de pontos espalhados no ambiente (Figura 4.1(a)). As imagens em nuvem de pontos podem ser convertidas em estruturas sólidas, para facilitar a sua visualização. Uma das transformações existentes é a triangulação de Delaunay, resultando em uma estrutura denominada malha triangular. Esses tipos de representações são construídos a partir da união dos pontos da nuvem de pontos, resultando em imagens formadas por triângulos de diversos tamanhos que ressaltam o aspecto tridimensional dos objetos (Figura 4.1(b)).



(a) (b)

Figura 4.1: Imagem de um gato em nuvem de pontos (a) e em malha triangular (b) (Hormann e Reimers, 2003).

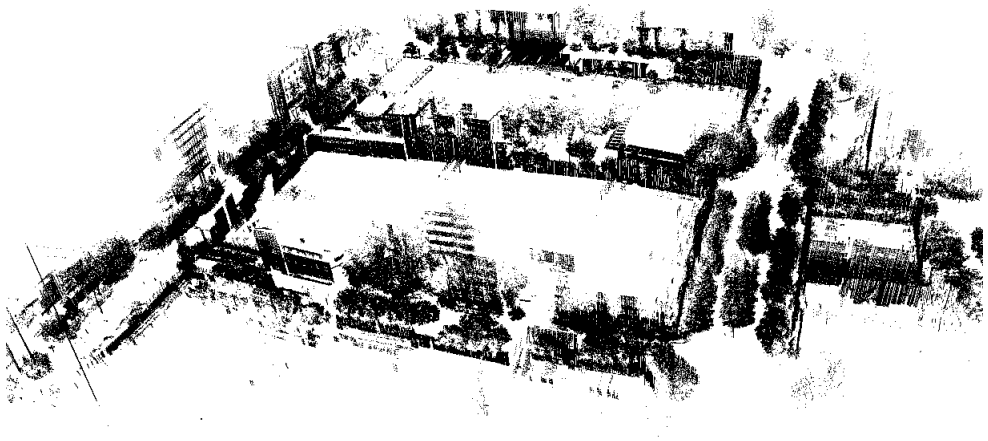
4.2 Mapeamento Tridimensional de Ambientes Externos

Para que o robô determine se um objeto à sua frente será tratado como um obstáculo ou não, é primordial que o mesmo seja capaz de mapear o ambiente. Como mencionado na Seção 4.1, os ambientes externos são melhor representados por meio de mapas tridimensionais, pois permitem representar os obstáculos com maior detalhamento em comparação com os mapas bidimensionais. Dentre os sensores mais utilizados para criar os mapas 3D encontram-se os sensores laser e as câmeras. Considerando-se que o sensor laser é o sensor adotado neste trabalho, serão tomados como foco os métodos que utilizam esse dispositivo para efetuar o mapeamento.

Existem vários métodos de mapeamento tridimensional que utilizam exclusivamente sensores laser 2D. Dentre os mais recentes, podemos citar (Thrun *et al.*, 2000) que utiliza dois sensores laser de alta resolução afixados ao robô. Um sensor laser direcionado para frente é utilizado para a tarefa de localização do robô no ambiente e o outro, direcionado para cima é utilizado para realizar o mapeamento 3D do ambiente. Com o sensor nesta configuração, é necessário que o robô esteja em movimento para realizar a varredura do ambiente, uma vez que estes sensores realizam apenas leituras planares. Para a localização do robô, são utilizados dados coletado por outros robôs que circulam no mesmo ambiente. Na Figura 4.2(a) é ilustrado um cenário virtual gerado por meio dessa técnica.

(a)

(b)



(c)

Figura 4.2: Exemplos de resultados de mapas 3D obtidos utilizando-se sensor laser. (a) Mapeamento de um corredor realizado por grupo de robôs (Thrun *et al.*, 2000). (b) Mapa de nuvem de pontos otimizado por meio da substituição de planos (Hähnel *et al.*, 2003). (c) Mapa de nuvem de pontos de um pátio (Wolf *et al.*, 2005a).

Em (Wolf *et al.*, 2005a), o grupo de robôs é substituído por um GPS para estimar a posição do robô. A partir disso, são mapeadas construções urbanas como prédios e casas em representação de nuvem de pontos (Figura 4.2(c)). Aproveitando-se a estrutura dessas construções, pode-se realizar também a identificação do conjunto de pontos que formam planos no ambiente para substituir tais pontos por simples planos. Isso permite reduzir drasticamente a quantidade de memória necessária para armazenar os mapas (Figura 4.2(b)).

Na abordagem dada por (Harrison e Newman, 2008), o sensor laser 2D é acoplado em uma plataforma móvel fixada ao robô. Para realizar o mapeamento do ambiente, a plataforma rotaciona o laser para cima e para baixo com um período de 1,2s, permitindo gerar as imagens tridimensionais mesmo com o robô parado. Porém, para gerar um mapa perfeito do ambiente, a cada leitura do laser é necessário conhecer precisamente a inclinação da plataforma. Isto acaba

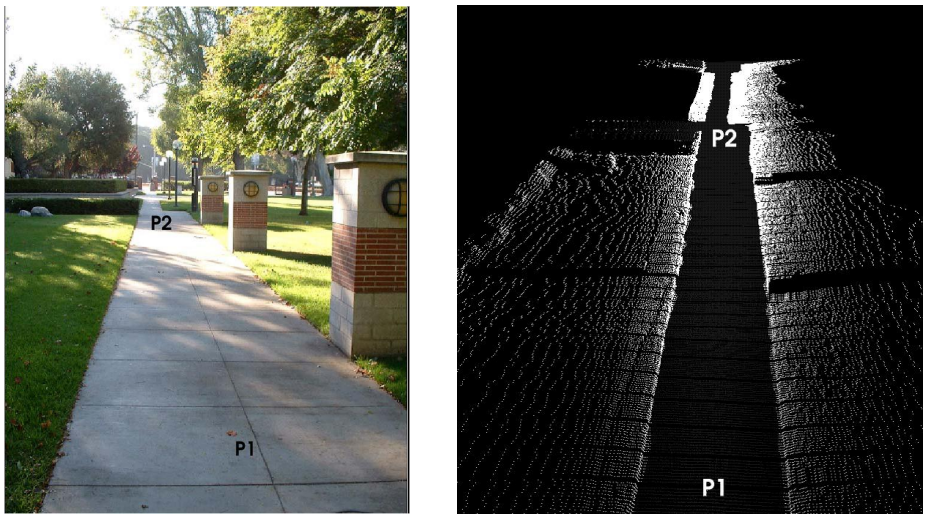
demandando um sincronismo perfeito entre o sensor laser e o mecanismo de inclinação do laser, o que nem sempre é possível. Uma metodologia similar é apresentada em (Blaer e Allen, 2007), porém, é utilizado um sensor laser 3D, descartando-se a presença de uma plataforma móvel. O problema desse método é o custo desses tipos de sensores que são muito mais caros em relação aos lasers 2D.

No trabalho proposto em (Pfaff *et al.*, 2007), é utilizado um carro autônomo para construir mapas 3D de regiões mais extensas, do tamanho de vilas ou pequenas cidades. O carro é equipado com um sensor GPS para a localização do robô e cinco sensores laser 2D para mapeamento. Três sensores laser estão direcionados para frente e os outros dois estão fixados verticalmente em um suporte rotativo no teto do carro. Os sensores que estão fixados na frente são utilizados para mapear o terreno, permitindo detectar as regiões navegáveis e os sensores que estão na parte superior possibilitam a varredura completa ao redor do carro.

Dentre os trabalhos que realizam o mapeamento de ambientes, existem aqueles que focam na construção de mapas 3D da superfície na qual o robô trafega. Esses mapas são necessários para definir se um determinado obstáculo poderá ser atravessado ou desviado (Ye e Borenstein, 2004). O modelo de representação utilizado para ilustrar os terrenos é um mapa comumente utilizado na geologia denominado modelo de elevação (ou *elevation map*). Nesses mapas, a geometria do terreno é denotada por meio de pontos dispostos em uma grade, os quais se elevam de acordo com a saliência do solo (Figura 4.3 (b)) (Kweon e Kanade, 1990).

Em (Wolf *et al.*, 2005b) são utilizados robôs móveis equipados com um sensor laser 2D inclinado para baixo. A partir do mapa gerado do ambiente (Figura 4.3(a)), o terreno é classificado em regiões navegáveis e não navegáveis (Figura 4.3(b)). Dessa forma, locais planos, tais como calçadas são rotulados como seguros para a passagem do robô e superfícies não uniformes como grama e cascalho são avaliadas como inseguras. No trabalho apresentado em (Schafer *et al.*, 2008), foi construído um robô dedicado ao mapeamento de coberturas vegetais. Para isso, o robô está munido além do sensor laser, uma câmera estéreo e um pára-choque sensível à pressão. Devido ao fato do terreno estar parcialmente coberto pela grama, o modelo da superfície é composto somente pelas leituras do sensor laser que não são interceptadas pela vegetação. Com isso, é calculado o ângulo de inclinação do terreno que, se for maior do que um limiar, o robô procurará outra região navegável.

Entre outras técnicas de mapeamento 3D, destaca-se o uso associado da câmera e do sensor laser para permitir a recriação do ambiente levando-se em consideração a textura dos objetos, resultando em mapas mais realísticos. Em (Bok *et al.*, 2007) é feita uma correlação entre os pixels das imagens capturadas pela câmera com as coordenadas dos pontos coletados pelo laser. Os dispositivos são fixados em um carro que realiza a coleta dos dados. Assim, ao invés de criar mapas de nuvem de pontos, são criados mapas contendo as próprias imagens do ambiente



(a) (b)

Figura 4.3: Modelo de elevação gerada por meio do mapeamento de uma via pública. O ambiente real é apresentado em (a) e o seu respectivo mapa em (b).

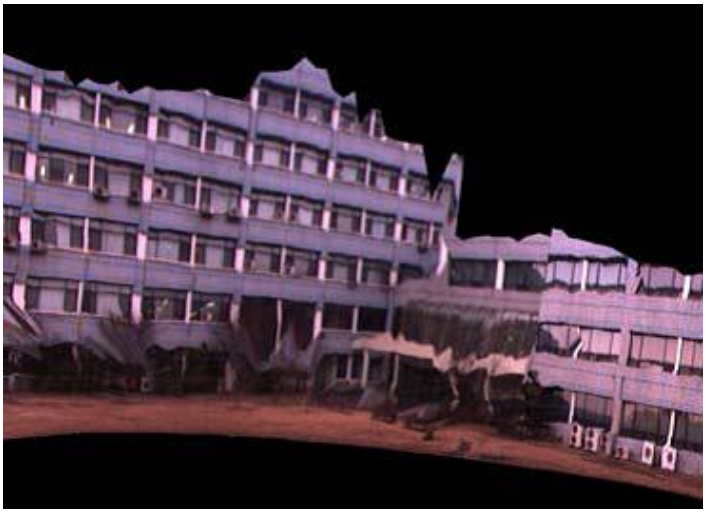
(Figura 4.4(a)). De forma similar ao método anterior, no trabalho proposto em (Banno e Ikeuchi, 2005), foi utilizado um balão com os mesmos sensores para mapear o ambiente. Porém, nesse caso, a câmera desempenhou o papel de texturizador do mapa e de correção dos dados da leitura do laser provocados pela trepidação do laser (Figura 4.4(b)).

4.3 Navegação Autônoma em Ambientes Externos

A navegação autônoma de robôs móveis em terrenos irregulares exige a capacidade de escolher apropriadamente os caminhos mais seguros à frente do robô (Langer *et al.*, 1994). Outra habilidade necessária é a geração correta dos comandos de direção e de velocidade para desviar dos obstáculos. Esses dois requisitos formam a negociação de obstáculo (ou *obstacle negotiation*) (Ye e Borenstein, 2003). A tarefa de negociação de obstáculos possui dois assuntos relacionados: mapeamento de terrenos (Seção 4.2) e planejamento de rotas.

Cang Ye e Johann Borenstein ainda subdividem o tópico planejamento de rotas em dois subproblemas, análise da navegabilidade do terreno e geração de rotas (Ye e Borenstein, 2004). A análise da navegabilidade do terreno tem como função identificar no plano do robô as regiões que possibilitam realizar as travessias mais seguras. Após essa verificação, a tarefa de geração de rotas se encarrega de traçar no mapa os caminhos de menor custo até o destino.

Na literatura podem ser encontradas diversas metodologias para a detecção das regiões navegáveis de um terreno e também para a geração das rotas. A técnica adotada em (Castejon *et al.*,



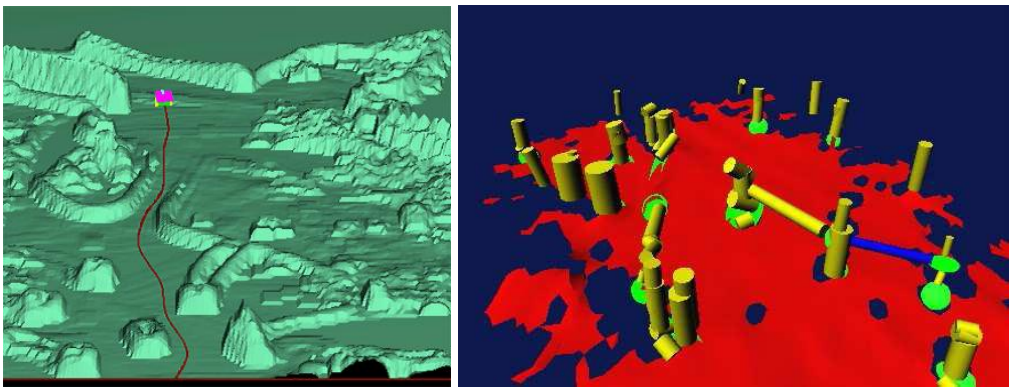
(a)

(b)

Figura 4.4: Resultados de reconstrução em 3D de estruturas combinando sensor laser e câmera. A figura (a) mostra um ambiente recriado com dados coletados por meio de um carro (Bok *et al.*, 2007) e a (b) por meio de um balão (Banno e Ikeuchi, 2005).

2008) identifica as áreas navegáveis através da análise estatística do terreno. Nesse processo, é calculada a variância esférica do chão para examinar o vetor normal dos pontos do mapa. Utilizando-se essa análise, ao contrário dos outros métodos, pode-se classificar a navegabilidade de estruturas inclinadas (pequenas pedras e rampas). As rotas do robô são construídas por meio do diagrama de Voronoi, no qual computa as trajetórias maximizando-se a distância entre o robô e os obstáculos.

O método proposto em (Ye, 2007) para a identificação das regiões navegáveis consiste em atribuir um índice de navegabilidade em cada célula do mapa do terreno, resultando em um mapa de navegabilidade do terreno. Esse processo é composto por duas etapas. Na primeira é estimada a inclinação e a irregularidade do terreno e na segunda, calculado o índice de navegabilidade para cada célula. Para o planejamento de rotas é computado para cada célula ao redor



(a) (b)

Figura 4.5: (a) Rota gerada para a navegação do robô em ambientes externos (Ye, 2007). (b) Método de classificação de terreno, sendo que objeto amarelo representa troncos de árvores, azul representa cabos, vermelho indica o chão e verde indica vegetação (Lalonde *et al.*, 2006).

do robô o índice polar de navegabilidade que permite buscar e agrupar os morros e vales mais seguros para a passagem do robô (Figura 4.5(a)).

Na abordagem proposta em (Lalonde *et al.*, 2006) é utilizado um classificador bayesiano para rotular porções do mapa em três categorias: "disperso"para indicar grama e folhagens; "linear", para representar fios e ramos de árvore e "superfície", para objetos sólidos, como pedras, chão e troncos de árvores (Figura 4.5(b)). Dessa forma, ao invés de classificar simplesmente o terreno em navegável ou não, permite-se um refinamento mais detalhado do cenário, possibilitando o uso do robô em ambientes compostos por vegetações ou em florestas.

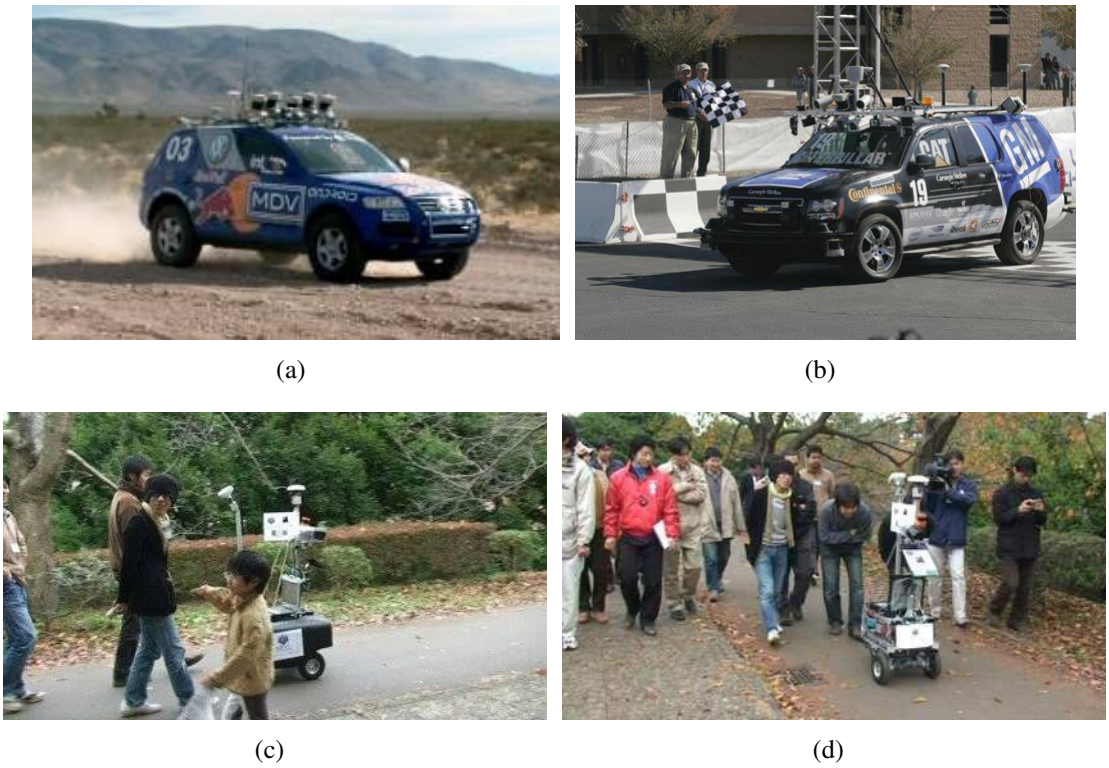
A estratégia tomada por pesquisadores da Universidade Carnegie Mellon (Hamner *et al.*, 2006) para planejar o controle do robô utiliza informações de um veículo controlado manualmente para treinar o sistema de desvio de obstáculos. Dessa maneira o robô simula os controles de um humano para navegar pelo ambiente. No carro autônomo desenvolvido pela universidade de Stanford (Stavens e Thrun, 2006), o processo de classificação de terrenos é feito pelo cálculo da diferença das alturas dos pontos com seus adjacentes. Assim de acordo com esse resultado, a célula é classificada em livre, ocupado ou desconhecido. De forma a corrigir erros de classificação causados por erros de leitura dos sensores, é utilizado um modelo markoviano que ajusta os parâmetros dos dados. As restrições tomadas para a geração das rotas levam em consideração o número de obstáculos sobre o caminho e rotas que estão o mais próximo possível do centro da via.

4.4 Aplicações

Quando se trata de robôs móveis autônomos que atuam em ambientes externos, pode-se pensar em inúmeras possibilidades de atuação desses mecanismos para auxiliar nas tarefas do homem. Isso permite, por exemplo, utilizar robôs para substituir o ser humano em trabalhos realizados em lugares perigosos ou de difícil acesso. Dessa forma, um trabalho de risco poderia ser trocado por uma simples tarefa de ajuste do robô.

Entre as diversas aplicações possíveis para esses robôs, certamente uma das mais promissoras são os veículos autônomos. Uma das primeiras tentativas para se automatizar a direção de um carro foi dada por uma iniciativa do governo europeu em um programa de redução de acidentes e aumento da eficiência do tráfego denominado PROMETHEUS (Sun *et al.*, 2004). Entretanto, devido a dificuldades financeiras, esse projeto não conseguiu ter continuidade. No ano de 2004, a agência militar americana DARPA organizou a primeira competição entre carros autônomos, denominada *Grand Challenge*, oferecendo um prêmio de um milhão de dólares para o primeiro colocado. A prova consistia em atravessar um trecho de 240 km do deserto norte-americano utilizando um carro autônomo não tripulado. Mesmo com várias equipes inscritas no evento, nenhuma conseguiu completar a prova, mostrando a grande dificuldade existente na construção de um carro autônomo. Este evento acabou por motivar toda a comunidade científica, gerando excelentes trabalhos (Bajracharya *et al.*, 2008). Em 2005, ocorreu a segunda edição do *Grand Challenge*, mantendo-se o objetivo do primeiro evento, mas agora com prêmio de dois milhões de dólares. Com os aperfeiçoamentos realizados nos carros, diversas equipes conseguiram completar a prova, indicando um considerável progresso nas pesquisas. O vencedor dessa edição foi a equipe da Universidade de Stanford, sendo que o carro autônomo (Figura 4.6(a)) contava com 5 sensores lasers para mapeamento de ambiente e uma câmera para detectar as regiões navegáveis (Thrun *et al.*, 2006). A última competição de carros autônomos promovida pelo DARPA ocorreu em 2007. Neste, focou-se a atuação dos carros em ambientes urbanos, assim, a edição ficou conhecida como *Urban Challenge*. Aqui o desafio era construir robôs (veículos) capazes circular dentro de cidades da Califórnia respeitando as leis de trânsito local (Patz *et al.*, 2008). Desta vez, a equipe vencedora foi da Universidade de Carnegie Mellon (Urmson *et al.*, 2008) que centralizou o sistema no planejamento dos movimentos do robô (Figura 4.6(b)).

Outro evento relacionado chamado Tsukuba *Challenge* com menores proporções em relação ao DARPA foi realizado em 2007 no Japão (Morales *et al.*, 2008). A idéia dessa competição era fazer um robô autônomo percorrer uma distância de 1 km em uma via pública, onde normalmente circulam pedestres e bicicletas. Dessa forma, os robôs deveriam cumprir a prova



(a) (b)

(c) (d)

Figura 4.6: (a) Stanley, carro vencedor da segunda edição do DARPA *Grand Challenge*. (b) Boss, carro vencedor do DARPA *Urban Challenge*. (c) e (d) Cenários do Tsukuba *Challenge*.

evitando trazer qualquer perturbação ao público e também não efetuar alterações no ambiente a sua volta (Figura 4.6(c) e (d)).

Além da automatização de veículos, os robôs autônomos podem ser aplicados na busca de vítimas de acidentes em locais de difícil acesso ao homem, como em túneis, áreas desabadas e locais incendiados. Em (Kurisu *et al.*, 2007) é proposto um robô capaz de mapear ambientes em busca de vítimas de acidentes utilizando um sensor laser. Já em (Zhang *et al.*, 2007) são utilizadas câmeras para o resgate das vítimas. Há também a proposta de um sistema para a localização de vítimas em avalanches (Pinies *et al.*, 2006) e também uso de mapas de elevação de terrenos para análise de terremotos (Muller e Harding, 2007).

Nota-se que existem diversas áreas nas quais os robôs móveis podem ser aplicados, sendo possível auxiliar nas tarefas realizadas pelo homem ou até mesmo substituir completamente os comandos do homem, como no caso dos carros autônomos. Porém muitas vezes, esses robôs estão preparados para agirem apenas em ambientes controlados, funcionando assim, de um modo restrito. Com isso, mesmo já tendo obtido excelentes resultados com robôs autônomos, ainda

é preciso muitas pesquisas para que os robôs funcionem perfeitamente em qualquer situação, sem oferecer risco ao usuário.

4.5 Considerações Finais

Os robôs que atuam em ambientes externos interagem com elementos que muitas vezes não estão presentes em um ambiente internos. Como esses elementos não possuem uma estrutura ou comportamento definido (terreno, árvores, pedestres), a análise do cenário ao redor do robô torna-se uma tarefa difícil. Sendo assim, existem muitos problemas que ainda não foram resolvidos pela comunidade científica. Apesar disso, pode-se constatar por meio dos trabalhos publicados sobre esse tema, uma grande evolução nas pesquisas. Pode-se dizer que os resultados mais promissores provêm dos veículos autônomos construídos para as competições organizadas pelo DARPA. Nela, foram apresentadas diversas soluções inovadoras para problemas como localização, mapeamento e navegação de robôs em ambientes externos.

Além desses, há também diversas propostas interessantes para resolver problemas fundamentais de robôs para trabalharem em ambientes externos. No caso de mapeamento, encontramos abordagens que utilizam associação de diversos sensores. Em relação à navegação, têm-se métodos estatísticos e de aprendizado de máquina para resolver esse problema.

Observa-se que dada à extensão do tema, diferentes propostas podem ser apresentadas, com cada uma focando uma parte específica do problema. Dessa forma, o presente trabalho pode ser visto como uma contribuição para a área da robótica, na qual apresenta novos métodos de mapeamento e navegação em ambientes externos.

CAPÍTULO

5

Ambiente de Simulação de Robôs Móveis Player/Stage

Muitas vezes, a programação dos controladores dos robôs móveis pode ser uma tarefa complicada e demorada, dado que o robô conta com diversos sensores e dispositivos que trabalham em conjunto (Gerkey *et al.*, 2001). Essa tarefa pode tornar-se ainda mais complexa, caso estiver trabalhando com muitos robôs ou com robôs distribuídos. Para facilitar a programação do controle do robô, foram criados programas de simulação que tornam o seu desenvolvimento mais rápido. Dentre os simuladores disponíveis estão: Microsoft Robot Studio, Webots, URBI e Player/Stage.

Player/Stage é um software em código aberto sob licença pública da GNU e foi criado pelo laboratório de robótica da Universidade do Sul da Califórnia. Essa ferramenta tem como objetivo permitir a programação em alto nível do sistema de controle dos robôs móveis, facilitando a manipulação dos diversos sensores equipados nos robôs. A ferramenta oferece também suporte à programação de sistemas multirobóticos, além de que todos os programas podem ser simulados em cenários que podem ser construídos pelo usuário. Como é um produto de código aberto, os usuários estão livres para modificá-lo, podendo inserir novos modelos de robôs ou controladores de sensores. Devido a essas e outras vantagens, neste projeto de pesquisa foi escolhido o uso do software de simulação Player/Stage para a simulação de ambientes e controle do robô em experimentos reais.

5.1 Estrutura do Software Player/Stage

O software Player/Stage é um sistema baseado no modelo cliente/servidor para efetuar a comunicação entre o hardware do robô e o usuário. Isto proporciona ao usuário uma ferramenta de alto nível para desenvolver o controlador dos dispositivos equipados no robô, facilitando a sua programação.

O componente Player desempenha o papel de servidor das informações dos periféricos do robô e o Stage a interface de simulação de robôs. Nas seções seguintes, será tratada em detalhes cada uma destas partes.

5.1.1 Servidor Player

De maneira análoga a um servidor de dispositivos soquetes, o software Player possibilita integrar os diversos dispositivos do robô (atuadores e sensores) com os clientes através do protocolo de comunicação TCP (do inglês, *Transfer Control Protocol*). Deste modo, os computadores que executam o servidor Player devem ter acesso aos periféricos do robô e os mesmos devem oferecer uma interface TCP (Gerkey *et al.*, 2001). Na Figura 5.1 está representado o modelo de cliente/servidor do Player.

O uso de um protocolo de rede para a comunicação traz algumas vantagens como:

- Neutralidade da linguagem de programação: Como a transferência de informações é feita por meio do protocolo TCP, qualquer linguagem que ofereça suporte a soquetes pode ser utilizada para o desenvolvimento dos programas clientes. Entre as linguagens que facilitam este trabalho estão, C, C++, Tcl, Phyton, Java e Common LISP.

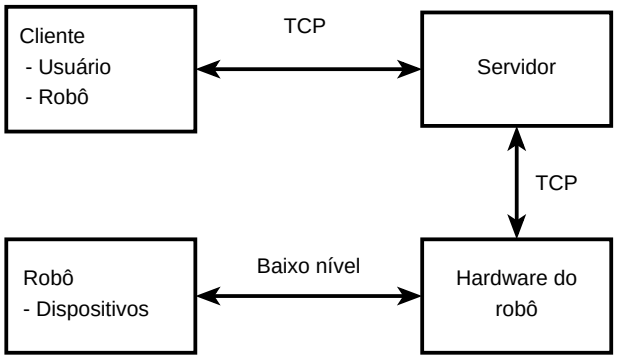


Figura 5.1: Modelo de cliente/servidor utilizado no Player/Stage.

- Neutralidade de plataforma: Os programas podem ser desenvolvidos para diversas plataformas, como Windows e Linux. Dessa forma, permite-se uma grande flexibilidade nas plataformas que podem ser aproveitadas.

- Neutralidade de localidade: O uso de soquetes permite ainda que o programa cliente possa ser executado em qualquer computador com conexão à rede. Isto torna possível a construção de sistemas de controle e de sensoriamento distribuídos.

5.2 Ambiente de Simulação Stage e Gazebo

Os programas desenvolvidos com o Player, antes de serem utilizados diretamente nos robôs físicos, podem ser testados no software de simulação Stage. Da mesma forma que o Player, o Stage atua como um servidor de dados utilizando o protocolo TCP, tornando possível o uso de dispositivos virtuais como se fossem hardwares reais. Dessa forma, para o cliente torna-se indiferente se está manipulando um dispositivo do Stage ou um dispositivo real, pois os comandos continuam iguais para os dois casos, sendo a única diferença, o endereço IP onde serão requisitados os dados.

Com a ferramenta Stage, pode-se testar um ou vários robôs simultaneamente, diversos tipos de sensores, e também a capacidade de criar cenários para realizar as simulações (Figura 5.2). O uso do Stage possui a princípio, três objetivos (Gerkey *et al.*, 2001):

- Possibilitar o desenvolvimento rápido de controladores que serão utilizados posteriormente em robôs reais.

- Possibilitar a realização de experimentos com robôs em ambientes aos quais não se tem o acesso físico.

- Realização de experimentos inovadores com dispositivos ainda não fabricados.

De acordo com os desenvolvedores, não há nenhuma garantia de que as simulações realizadas no ambiente virtual sejam iguais ao mundo real (Yan *et al.*, 2006). Entretanto, testes realizados apontam que o Stage funciona perfeitamente com pouca ou nenhuma modificação dos robôs reais.

Outro ambiente de simulação que pode ser utilizado juntamente com o Player é o Gazebo (Figura 5.3). Com recursos e funcionalidade semelhante ao Stage, o Gazebo se difere pelo fato de oferecer um ambiente de simulação tridimensional e também ter suporte à simulação da física de objetos. Os programas clientes implementados para o Stage funcionam normalmente

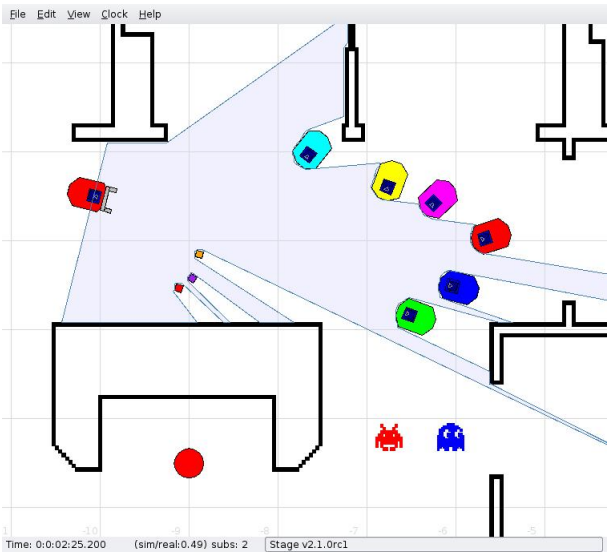


Figura 5.2: Interface gráfica do Stage exibindo alguns obstáculos, robôs e a atuação de um sensor laser.

no Gazebo, exigindo às vezes alguma ou nenhuma modificação. Uma vez que o Stage foi projetado para suportar muitos robôs com baixa fidelidade, o Gazebo permite a simulação de poucos robôs (muitas vezes devido à limitação do próprio hardware de simulação), mas com alta fidelidade.

5.3 Modelo dos Dispositivos

Da mesma forma que o sistema UNIX, os dispositivos do Player são tratados como arquivos para serem manipulados. Com isso, para se controlar os dispositivos, são utilizados os comandos de tratamento de arquivos. Para exemplificar, quando o usuário desejar obter os dados obtidos pelos sensores, será utilizado o comando **READ** pelo cliente. Seguindo a mesma idéia, para se controlar um atuador, o cliente acessará o sensor utilizando o comando **WRITE** (Gerkey *et al.*, 2001). No Apêndice B é apresentado um exemplo de código em C com comandos básicos do Player.

Além de trazer uma interface de alto nível para o controle dos dispositivos, outro grande benefício do Player é a interface padrão de comunicação. Essa característica permite que o mesmo programa utilizado para controlar um determinado robô seja capaz de funcionar em outro, sem a necessidade de efetuar modificações em seu código. Entre os dispositivos suportados estão variados sensores de distância, atuadores, identificadores de cor, GPS, câmeras e odômetros de robôs.

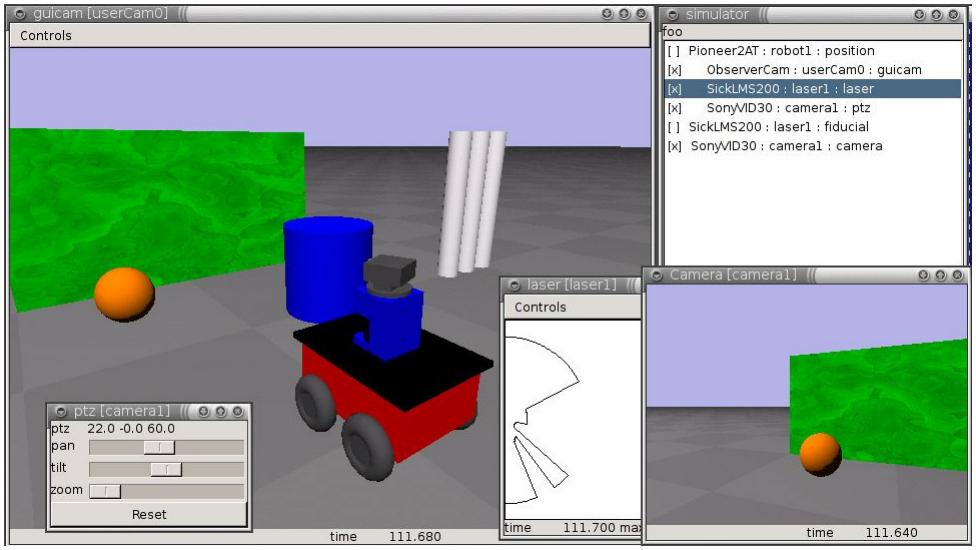


Figura 5.3: Ambiente de simulação de robôs Gazebo.

5.4 Considerações Finais

Este capítulo apresentou os principais recursos da ferramenta Player/Stage que foi fundamental no desenvolvimento da inteligência do robô. O software possibilita a programação em alto nível dos controladores do robô, logo, facilita e reduz o tempo de desenvolvimento de aplicativos robóticos. Além disso, oferece um simulador que permite validar os algoritmos elaborados antes de implantá-lo em um robô real.

CAPÍTULO

6

Métodos de Aprendizado Supervisionado

O Aprendizado de Máquina (AM) é uma área da Inteligência Artificial (IA) na qual são investigados métodos computacionais que automatizam a tarefa de aquisição de conhecimento, processo também denominado de aprendizagem (Baranauskas e Monard, 2003). Algoritmos de aprendizagem têm como objetivo simular comportamentos naturais ou artificiais. O AM divide os métodos de aprendizagem em três categorias: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço (Baranauskas e Monard, 2003) (Mitchell, 1997) (Carbonell *et al.*, 1984).

Em relação ao algoritmo de aprendizado supervisionado, é fornecido a um modelo um conjunto de pares de valores (entradas e saídas) rotulados que exemplificam o comportamento de um determinado ambiente. Para cada valor de entrada alimentado ao modelo, é verificado se o resultado produzido corresponde com o valor de saída fornecido. Caso isto não ocorra, o modelo é reestruturado de forma a produzir o resultado desejado (Mitchell, 1997). O processo de adapatação do modelo ao comportamento do ambiente é denominado treinamento. Exemplos de métodos que entram nessa categoria incluem redes neurais artificiais (*multi-layer* perceptron), máquinas de suporte vetorial, kNN (*k-Nearest Neighbour*) e árvores de decisão. Por outro lado, no aprendizado não-supervisionado, não é fornecido um conjunto de exemplos. Com isso, dado os valores de entrada, o algoritmo de aprendizado deverá ter a capacidade de separá-los

em grupos de acordo com critérios do problema (Ghahramani, 2003). Redes neurais artificiais (*self-organizing maps*) e Redes Bayesianas são exemplos de métodos de aprendizado não supervisionado. Por último, no algoritmo de aprendizado por reforço, caso o modelo gere ações positivas, o ambiente o retribui com recompensas. Assim, esse paradigma tem como objetivo criar modelos que recebam o maior número de recompensas possíveis (Kaelbling *et al.*, 1996). Métodos tradicionais desta classe incluem *Q Learning* e aprendizado por diferença temporal.

As seções seguintes apresentam os dois métodos de aprendizado supervisionado adotados neste projeto: redes neurais artificiais e máquinas de suporte vetorial.

6.1 Redes Neurais Artificiais

As redes neurais artificiais (RNAs), de uma forma simples, compreendem no paradigma da IA que busca a solução dos problemas por meio da simulação computacional dos mecanismos e estruturas do cérebro humano. Assim, como no modelo biológico, as RNAs são formadas por um conjunto de neurônios e suas interligações, os quais são responsáveis pelo processamento das informações. As RNAs determinam as soluções por meio da generalização de um conjunto de dados que modela o comportamento do problema. Devido a essa propriedade, as RNAs são normalmente utilizadas em problemas de classificação e de regressão (Rojas, 1996).

O elemento básico de uma RNA é o neurônio artificial que é composto de três elementos básicos:

1. Conexões: Permitem realizar ligações entre os neurônios de forma a propagar os dados através da rede. A cada conexão está associado um peso que possui a função de amplificar ou atenuar o sinal originado dessa ligação. Assim, um sinal x_j que provém de uma conexão j e que chega ao neurônio k, tem o seu valor multiplicado por um peso w_{kj}.

2. Somador: Efetua a somatória de todos os sinais multiplicados pelos respectivos pesos das conexões que chegam ao neurônio.

3. Função de ativação: Esta função determina o valor do sinal de saída do neurônio de acordo com o resultado atingido pelo somador. Em geral, os valores das saídas dos neurônios são normalizados para o intervalo $[0, 1]$ ou $[-1, 1]$.

Na Figura 6.1 é ilustrada a estrutura básica de um neurônio. As entradas são denotadas por $x_1 ... x_m$, sendo $w_{k1} ... w_{km}$ os seus respectivos pesos. Σ representa o somador da rede, υ_k o valor da somatória dos sinais, φ a função de ativação e y_k o sinal de saída do neurônio. O parâmetro

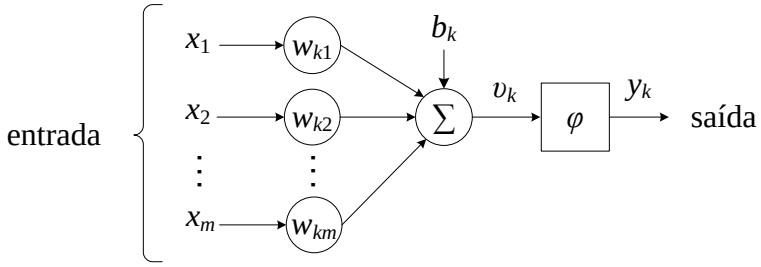


Figura 6.1: Componentes básicos de um neurônio artificial (adaptado de (Haykin, 1998)).

b_k corresponde ao bias do neurônio que tem a finalidade de aumentar ou diminuir o valor do sinal de entrada para a função de ativação (Haykin, 1998).

O primeiro modelo de neural artificial foi proposto por McCulloch e Pitts em 1943 (McCulloch e Pitts, 1943). Nesse modelo, a saída do neurônio assume um estado de ativação (valor 1) quando a somatória dos sinais das conexões que o atingem é maior que zero. Caso contrário, a saída do neurônio não é ativada (valor 0). A função de ativação do modelo de McCulloch-Pitts descrita anteriormente é denominada função *Heavyside* (Haykin, 1998). As Equações 6.1 e 6.2 descrevem a função de ativação do modelo McCulloch-Pitts. Além deste, existem outros tipos de função de ativação como a função *Piecewise-Linear* e a função Sigmoidal Simétrica (Haykin, 1998).

$$y_k = \varphi(v_k) = \begin{cases} 1 & \text{se } v_k \geq 0 \\ 0 & \text{se } v_k < 0 \end{cases} \tag{6.1}$$

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{6.2}$$

Uma rede neural consiste basicamente na interligação dos neurônios artificiais, formando uma malha composta de algumas ou várias unidades de processamento de sinais. A estrutura da RNA adotada implica diretamente no algoritmo de aprendizado a ser utilizado. É por meio do algoritmo de aprendizado que a RNA obtém os conhecimentos necessários para a resolução do problema. Consequentemente, este algoritmo acaba por determinar a capacidade de resolução de problemas pelas RNAs. Na Seção 6.1.1 serão descritas as principais arquiteturas de redes neurais e na Seção 6.1.2 os algoritmos de aprendizado de RNAs mais utilizados.

6.1.1 Arquiteturas de RNAs

As redes neurais podem ser organizadas em diversas configurações que são chamadas de arquiteturas ou topologias de RNAs. A arquitetura de uma RNA está diretamente ligada com o

algoritmo de aprendizado a ser utilizado (mais detalhes na Seção 6.2). Dessa forma, uma análise dos tipos de redes neurais deve ser feita para que o mesmo possa obter as melhores soluções para o problema (Braga *et al.*, 2000).

Tradicionalmente, nas RNAs os neurônios são dispostos em camadas, sendo que cada uma possui uma função específica para a rede. A primeira camada de uma rede, denominada camada de entrada, tem como finalidade única, fornecer dados para os neurônios da camada subsequente. Os neurônios presentes nessa camada, em geral, não realizam nenhum processamento. Por essa razão, não são contabilizados na contagem de camadas da rede neural. A camada intermediária da RNA possui o papel de computar os dados recebidos pela camada anterior. Em uma rede, pode-se ter nenhuma ou várias camadas ocultas. Por fim, a última camada da RNA, a de saída, efetua o último processamento dos dados e gera a resposta da rede. Os sinais da rede seguem um fluxo unidirecional e todos os sinais passam simultaneamente pela mesma camada. A arquitetura das RNAs é classificada em três grupos principais: perceptron, *muli-layer* perceptron e recorrente (Braga *et al.*, 2000).

Perceptron

A arquitetura perceptron (ou *single-layer feedforward*) consiste na configuração mais básica de RNA, sendo composta apenas de uma camada (ou seja, *single-layer*). Dessa forma, essa rede é constituída somente de uma camada de entrada e uma camada de saída, sendo esta última a única responsável pela computação dos dados. Outra característica importante é que essa rede não possui ciclos em sua estrutura (acíclico), ou seja, a saída de um neurônio não é realimentada para a própria rede. Devida a sua estrutura simplificada, essas redes estão limitadas à resolução de problemas lineares (Braga *et al.*, 2000).

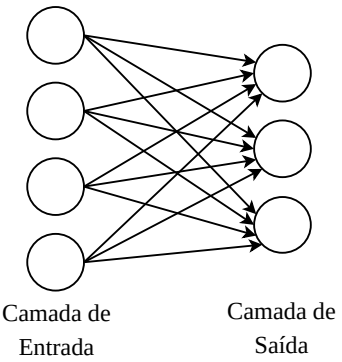


Figura 6.2: Rede neural perceptron com 4 neurônios na camada de entrada e 3 neurônios na camada de saída.

Em geral, as RNAs são representadas por meio de grafos direcionados (Haykin, 1998). Assim, os neurônios são denotados por meio de vértices do grafo e as suas interconexões por meio das arestas. Na Figura 6.2 é ilustrada uma rede neural perceptron.

Multi-Layer Perceptron

A arquitetura *multi-layer* perceptron (MLP ou *multi-layer feedforward*) adiciona à topologia perceptron as camadas intermediárias, formando uma rede neural que contém mais de uma camada (ou seja, *multi-layer*). A adição das camadas intermediárias permite que a rede neural represente funções contínuas, e dependendo do número de camadas ocultas utilizadas, pode representar funções não contínuas, tornando-o capaz de resolver problemas mais complexos (Russell e Norvig, 2003). Nesse sentido, existem pesquisas que investigam o número ideal de elementos intermediários a serem utilizados para cada problema (Haykin, 1998). Em geral, os métodos que determinam esse número, realizam testes exaustivos na rede neural (Braga *et al.*, 2000). Na Figura 6.3 pode-se verificar um exemplo de rede neural *multi-layer* perceptron.

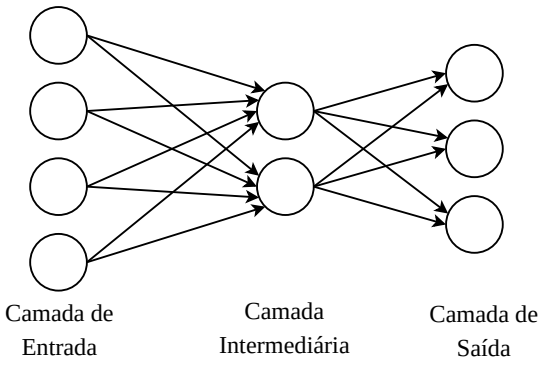


Figura 6.3: Rede neural multi-layer perceptron com 2 neurônios na camada intermediária.

Recorrente

Na arquitetura recorrente, há a presença de algum laço que conecta o neurônio de uma camada posterior com o neurônio de uma camada anterior, de forma a realimentar os sinais da RNA. Com essa estrutura, as redes recorrentes possuem a capacidade de simular uma memória de curto prazo. Estruturas recorrentes admitem também um número indefinido de camadas intermediárias (Rojas, 1996). A Figura 6.4 ilustra uma rede recorrente com uma camada intermediária.

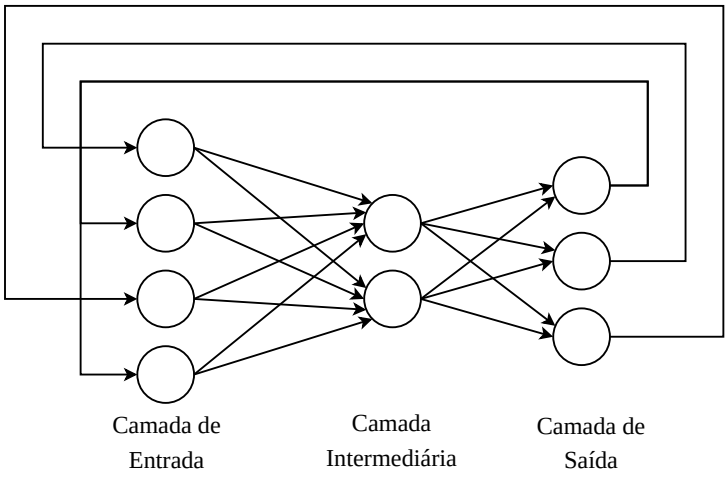


Figura 6.4: Uma rede recorrente com uma camada intermediária. Neste exemplo, as saídas da última camada alimentam a camada de entrada.

6.2 Algoritmo de Aprendizado de RNAs

O processo de aprendizagem é uma fase preliminar da RNA necessária para a resolução dos problemas. Nesse processo, os parâmetros da rede neural (bias e pesos das conexões) são ajustados de modo que o sistema se comporte o mais próximo possível do ambiente o qual se deseja simular (Russell e Norvig, 2003). Existem três paradigmas de aprendizado de RNAs: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço. Cada um deles utiliza uma técnica diferente para determinar os parâmetros da rede (pesos das conexões e bias). Pelo fato de que o atual projeto de pesquisa envolve apenas aprendizado supervisionado, somente este será abordado nesta seção (mais detalhes dos outros paradigmas podem ser conferidos na introdução deste capítulo).

No algoritmo de aprendizado supervisionado é utilizado um conjunto de exemplos rotulados contendo entradas e saídas esperadas. Para cada exemplo de entrada, é comparada a saída gerada com a saída esperada, sendo essa diferença refletida nos pesos das conexões. Os parâmetros da rede (peso das conexões e bias) são então modificados até que a saída gerada e a esperada fiquem próximas ou iguais (Braga *et al.*, 2000). Um dos algoritmos de aprendizado mais utilizados para treinamento de redes do tipo perceptron e *multi-layer* perceptron é o *backpropagation* e suas derivações (por exemplo, Rprop e Quickprop), os quais são descritos a seguir.

Backpropagation

O algoritmo de aprendizado *backpropgation* (ou *backprop*) (Rumelhart *et al.*, 1986) realiza o ajuste dos parâmetros da rede por meio da retropropagação do erro da saída para os pesos

das conexões. Esse processo é desempenhado em duas etapas distintas, uma de propagação (fase *forward*) e uma de retropropagação (fase *backward*). Na primeira etapa, os sinais de entrada propagam pela rede para gerar uma resposta da rede, não havendo alterações nos pesos das conexões. Na segunda, é computado o erro da saída gerado para determinar o fator de correção do peso das conexões, considerando que esse cálculo é feito de um neurônio por vez, indo em direção à camada de entrada (khalil Sari Alsmadi *et al.*, 2009). Para o cálculo desse erro, o *backpropagation* utiliza uma adaptação da Regra Delta (Rojas, 1996) que é utilizado no aprendizado de redes *perceptron*. Nessa adaptação, é levado em conta o sinal gerado pelo neurônio. A Equação 6.3 mostra o cálculo do gradiente do erro:

$$\frac{\partial E}{\partial w_{ij}} = y_i \frac{\partial E}{\partial y_i w_{ij}} \tag{6.3}$$

onde, E corresponde ao erro obtido pela rede neural, w_{ij} o peso da conexão que liga o neurônio i ao neurônio j e y_i o sinal gerado pelo neurônio i. Essa equação representa o valor do gradiente E em relação a um determinado sinal gerado por um neurônio da rede. A soma dos erros da fase *backward* até um determinado elemento é chamado de erro retropropagado. Se considerarmos o erro retropropagado até o neurônio j como δ_j, podemos reescrever a Equação 6.3:

$$\frac{\partial E}{\partial w_{ij}} = y_i \delta_j \tag{6.4}$$

Utilizando a derivada parcial da Equação 6.4, utilizamos o método do gradiente decrescente pela adição de cada w_{ij}, o valor de Δw_{ij} que é dado por:

$$\Delta w_{ij} = -\eta y_i \delta_j \tag{6.5}$$

onde η representa a taxa de aprendizado do *backpropagation*. A atualização dos pesos deve ser realizada para todas as conexões, a cada exemplo alimentado na rede. Esse processo termina quando o erro atinge um valor baixo ou quando o algoritmo alcança um determinado número de iterações.

Resilient Backpropagation

O algoritmo *resilient backpropagation* (Rprop) (Riedmiller e Braun, 1993) é uma adaptação do *backpropagation* com o intuito de suprimir suas desvantagens. O *backpropagation* torna-se lento quando utilizado para treinar redes com muitos nós, além disso, dependendo dos parâmetros escolhidos para o algoritmo, esse tempo pode-se tornar ainda maior. Sendo assim, o Rprop

realiza uma otimização não-linear sobre o *backpropagation*, tornando a atualização dos pesos dependente apenas da taxa de aprendizagem e do sinal da derivada parcial do erro em relação ao peso (Riedmiller e Braun, 1994). Isto permite que o algoritmo venha a convergir mais rápido quando está próximo do mínimo local. A taxa de aprendizado é do Rprop é mostrada na equação a seguir:

$$\eta_i(t+1) = \begin{cases} \min(\eta_i(t)u, \eta_{\max}) & \text{se } \frac{\partial E(t)}{\partial w_{ij}} \frac{\partial E(t-1)}{\partial w_{ij}} > 0 \\ \max(\eta_i(t)d, \eta_{\min}) & \text{se } \frac{\partial E(t)}{\partial w_{ij}} \frac{\partial E(t-1)}{\partial w_{ij}} < 0 \\ \eta_i(t) & \text{para outros casos} \end{cases} \tag{6.6}$$

onde, t é a iteração atual, $0 < u < 1 < d$, η_i é a taxa de aprendizado do neurônio i, $\eta_{\min}$ e $\eta_{\max}$ são respectivamente o valor mínimo e máximo estabelecidos para a taxa de aprendizagem.

Os valores η_{min} e η_{max} são necessários para impedir que a aprendizagem acelere ou desacelere excessivamente. De um modo simples, toda vez que a derivada parcial troca de sinal, ou seja, quando o algoritmo ultrapassou o mínimo local, o valor da taxa de aprendizado é reduzido. E se a derivada parcial mantém o sinal, a taxa de aprendizado é incrementada para acelerar a convergência (Rojas, 1996). Por fim, a atualização dos pesos é dada pela Equação 6.7.

$$\Delta w_{ij} = \begin{cases} -\eta_i(t) \cdot \text{sinal}\left(\frac{\partial E(t)}{\partial w_{ij}}\right) & \text{se } \frac{\partial E(t)}{\partial w_{ij}} \frac{\partial E(t-1)}{\partial w_{ij}} > 0 \\ 0 & \text{para outros casos} \end{cases} \tag{6.7}$$

Nessa equação pode-se perceber que os pesos serão modificados apenas se na iteração atual o sinal da derivada parcial for alterado.

Quickprop

Da mesma forma como o Rprop, o algoritmo quickprop (Fahlman, 1988) surgiu da modificação do *backpropagation*, despendendo menos iterações para a convergência da RNA. A ideia empregada no quickprop é a aproximação da função do erro da rede neural em uma função quadrática. Tal consideração é verificada pela expansão da série de Taylor da função do erro (Fahlman, 1988). No algoritmo assume-se também que a atualização de um peso não modifica outros pesos, isto é, os pesos são independentes. Na Equação 6.8 é mostra a equação para a atualização dos pesos:

$$\Delta w_{ij} = \eta_i(t) w_{ij} \left(\frac{\nabla E(t)}{\nabla E(t-1) - \nabla E(t)} \right) \tag{6.8}$$

onde, $\nabla E(t) = \frac{\partial E(t)}{\partial w_{ij}}$ e t representa a iteração atual.

A Equação 6.8 pode ser também vista como uma aproximação do método de Newton. Por meio desta técnica, o quickprop consegue utilizar passos maiores para encontrar mais facilmente o mínimo local.

6.3 Generalização em RNAs

A generalização é atingida em uma RNA quando ela consegue determinar corretamente uma saída, dada uma entrada nunca apresentada à rede (Braga *et al.*, 2000) (Rosin e Fierens, 1995). Os dados utilizados para a verificação da rede são chamados de dados de validação (ou teste). Existem dois conceitos associados à generalização. O primeiro é o *overfitting* que ocorre quando a rede neural é treinada demasiadamente, ocasionando a memorização dos dados utilizados no treinamento. Consequentemente, caso a rede seja alimentada com dados diferentes dos de treinamento, estes serão interpretados como ruídos do sistema. O segundo conceito é o *underfitting* que é ocasionado pela falta de treinamento da rede neural, tendo-se assim, pouca capacidade de generalização (Rosin e Fierens, 1995).

Para que uma rede possa generalizar adequadamente qualquer entrada fornecida, é necessário que o treinamento seja feito até atingir os níveis entre *underfitting* e *overfitting* (Haykin, 1998). A técnica denominada *early stopping* é utilizada para encontrar esse ponto. Nesse método, a cada treinamento com o conjunto de treinamento, a rede é alimentada com o conjunto de validação sem realizar a atualização dos pesos. Este ciclo é repetido até que não se observe mais a redução do erro gerado pelo conjunto de validação (Prechelt, 1998).

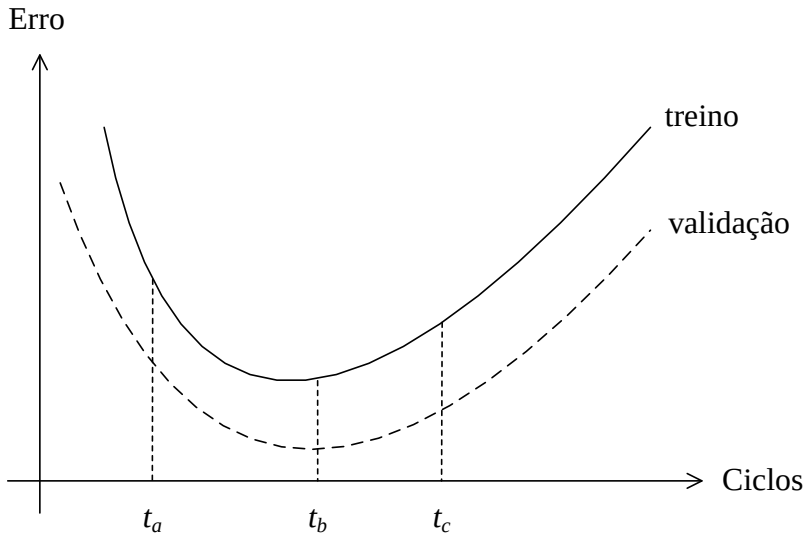


Figura 6.5: Curva do erro gerado conjunto de validação e de treinamento. t_a, t_b e t_c indicam respectivamente os pontos de *underfitting*, treinamento ideal e *overfitting*.

Na Figura 6.5 pode-se verificar um exemplo de treinamento com *early stopping*. Nota-se que no ciclo t_b a rede neural não apresenta mais melhorias para o conjunto de validação, indicando que a rede já é capaz de generalizar satisfatoriamente dados desconhecidos. Se o treinamento fosse interrompido antes de t_b (por exemplo, ciclo t_a), entraria no estado de *underfitting*, pois o erro ainda poderia ser reduzido. E caso suspenso após o ciclo t_b (ciclo t_c), o treinamento entraria no ponto de *overfitting*, pelo fato de que já teria ultrapassado o menor nível de erro obtido pela rede.

6.4 Máquinas de Suporte Vetorial

O algoritmo de aprendizado supervisionado das máquinas de suporte vetorial (SVM, do inglês *Support Vector Machines*) tem como fundamento, a determinação de um hiperplano que maximize a margem de separação de um conjunto de dados em classes distintas (Haykin, 1998) (Lorena e de Carvalho, 2007) (Wolf, 2006). Na fase de treinamento, exemplos constituídos por pares de entradas e saídas desejadas, são alimentados no SVM para determinar os melhores parâmetros desse hiperplano. Em virtude do seu modo de operação, o SVM é largamente utilizado em problemas de otimização e de regressão não-linear (Russell e Norvig, 2003). O SVM mais simples que consiste em um classificador linear (isto é, dados são separados em duas classes) é descrito na Seção 6.4.1, o SVM para classificação não-linear (isto é, dados podem ser separados em diversas classes) é detalhado na Seção 6.4.2 e o SVM para classificação multicalsse na Seção 6.4.3.

6.4.1 SVM para Classificação Linear

Os SVMs lineares separam os dados de entrada em duas classes, através de um hiperplano. Para detalhar o seu funcionamento, consideramos um conjunto de treinamento com N padrões que podem ser separados linearmente, $\{(x_i, y_i)\}_{i=1}^{N}$, onde x_i e y_i são respectivamente o i-ésimo padrão de entrada e saída desejado. Com $x_i \in X$ e $X \subset \Re^n$, $y_i \in Y$ e $Y = \{-1, 1\}$. A saída y_i pode assumir os valores -1 ou $+1$ para representar as duas classes. O hiperplano de separação das classes é representado na Equação 6.9.

$$w \cdot x + b = 0 \tag{6.9}$$

onde, x é o vetor de entrada, $w \in X$, é um vetor para o ajuste dos pesos de x_i, e b é o bias do hiperplano. O valor $b/\left\|w\right\|$ fornece a distância do hiperplano ao ponto de origem. A partir disso, podemos definir um sistema para dividir as duas classes:

$$\begin{cases} wx_i + b \geq +1 & \text{se } y_i = +1 \\ wx_i + b \leq -1 & \text{se } y_i = -1 \end{cases} \tag{6.10}$$

Este sistema pode ser simplificado como mostra a Equação 6.11.

$$y_i(wx_i + b) - 1 \geq 0 \tag{6.11}$$

Os pontos (x_i, y_i) que satisfazem a equação anterior, mas com o sinal de igualdade, são chamados de vetores de suporte. Os vetores de suporte determinam hiperplanos que limitam a região de cada classe, assim, denotamos H_1 para o hiperplano da classe $y_i = +1$ e H_2 para o hiperplano da classe $y_i = -1$. Sendo H_1 e H_2 paralelos, eles devem possuir a máxima distância d para otimizar a divisão das classes (vide Figura 6.6). Para isso, seja o ponto x_1 pertencente ao $H_1 : wx_i + b = 1$ e o ponto x_2 pertencente ao $H_2 : wx_i + b = -1$, a distância d é dada por:

$$d = (x_1 - x_2)\left(\frac{w}{\|w\|}\frac{x_1 - x_2}{\|x_1 - x_2\|}\right) \tag{6.12}$$

Sabendo que $wx_1b = +1$ e $wx_2 + b = -1$, podemos reformular a equação d como:

$$d = \frac{2}{\|w\|} \tag{6.13}$$

Com isso, a determinação do hiperplano pode ser resumida à minimização da Equação 6.14.

$$d = \min \frac{1}{2} \left\|w^2\right\| \tag{6.14}$$

Para a resolução desse problema, multiplicadores de Lagrange são utilizados. Obtendo-se o seguinte valor para w:

$$w = \sum_{i=1}^{N} \alpha_i x_i y_i \tag{6.15}$$

onde α_i é o multiplicador de Lagrange do i-ésimo padrão. Finalmente, o classificador linear do SVM é dado por:

$$\text{sinal}\left(\sum_{x_i} \alpha_i x_i y_i \cdot + b\right) \tag{6.16}$$

6.4.2 SVM para Classificação Não-Linear

O SVM linear apresentado anteriormente calcula um hiperplano para a separação das classes. Porém, em muitas situações, o conjunto de treinamento apresentado ao classificador não permite que sejam separados por esse mecanismo, como pode ser conferido no exemplo mostrado na Figura 6.7 (Cristianini e Shawe-Taylor, 2000) (Wolf, 2006). Por essa razão, os dados de treinamento devem ser transferidos para uma dimensão maior na qual possibilite efetuar a separação linear (Burges, 1998). Esta nova dimensão é denominada espaço de característica e a dimensão de origem, espaço de entrada (Lorena e de Carvalho, 2007). Esse mapeamento pode ser escrito como $\Phi : \Re^d \mapsto \Im$, sendo que $\Im$ representa o espaço de característica.

Para realizar as transformações de espaço, é necessário utilizar uma função chamada *kernel* que dado dois pontos do espaço de entrada, calcula o produto escalar entre eles no espaço de característica. Formalmente, a função *kernel* é definido como:

$$k(x_1, x_2) = \Phi(x_1) \cdot \Phi(x_2) \tag{6.17}$$

onde K é a função *kernel* e x_1 e x_2 pertencem ao espaço de entrada. Adaptando o *kernel* para o classificador linear, a Equação 6.13 é reestruturada na seguinte forma:

$$w = \sum_{i=1}^{N} \alpha_i y_i \Phi(s_i)\Phi(x_i) + b = \sum_{i=1}^{N} \alpha_i y_i K(s_i, x_i) + b \tag{6.18}$$

onde, s_i denota o vetor de suporte.

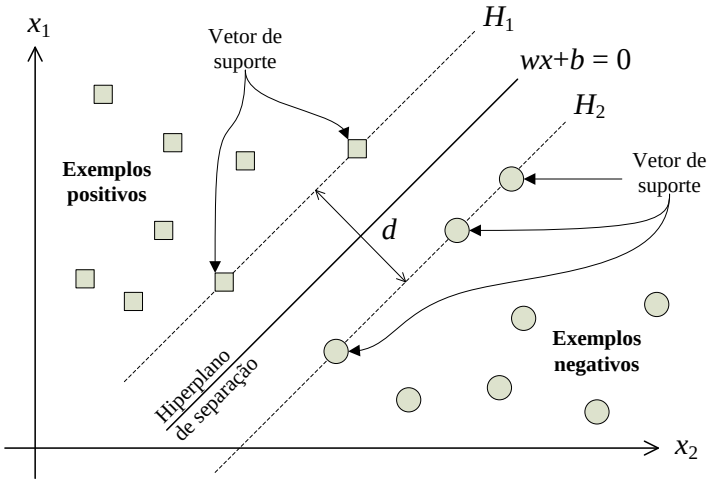


Figura 6.6: Ilustração do problema do SVM linear. Deve-se obter um hiperplano que maximize a margem (d) dos planos (H_1 e H_2) que separam as duas classes.

Na literatura, podemos encontrar diversas funções (Lorena e de Carvalho, 2007) para o *kernel*. Algumas delas estão listadas na Tabela 6.1:

Função	**Equação**
Polinomial	$K(s,x) = (as - x + b)^c$
Função de Base Radial(RBF)	$K(s,x) = e^{d\|sx\|^2}$
Sigmoid	$K(s,x) = \tanh(as - x + b)$

Tabela 6.1: Exemplos de funções *kernel*, onde a, b, c e d são parâmetros da função.

6.4.3 SVM para Classificação Multiclasse

Como o algoritmo padrão do SVM está restrito à classificação de dados em apenas dois grupos, não poderia ser aplicado em problemas de classificação de ordens superiores. Entretanto, essa desvantagem vêm sendo superada com uma categoria de SVM, chamada SVM multiclasse, que possibilita a classificação em várias classes. Um dos métodos mais utilizados é a combinação de vários classificadores SVM tradicionais para a resolução de problemas multiclasse (Crammer *et al.*, 2001). Dentre esses, destaca-se o método *one-versus-all* aplicando a estratégia *winner-takes-all* e o método *one-versus-one* utilizando *max-wins voting* (Duan e Keerthi, 2005). Na Figura 6.8 é ilustrado graficamente o problema do SVM multiclasse.

Para a descrição desses dois métodos, consideramos n o número de classes do problema, sendo que ω_i, $i = 1, ..., n$ representa cada classe e y_i a saída gerado pelo i-ésimo classificador.

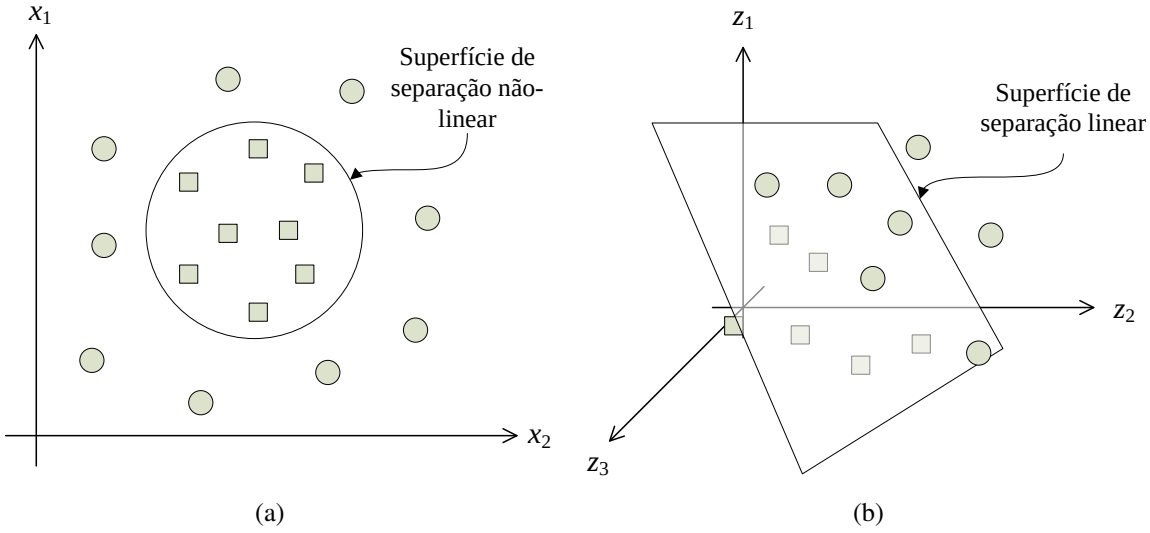


Figura 6.7: (a) Conjunto de dados que não podem ser separados linearmente. (b) Mapeamento dos dados para uma espaço que permite a separação linear (Burges, 1998).

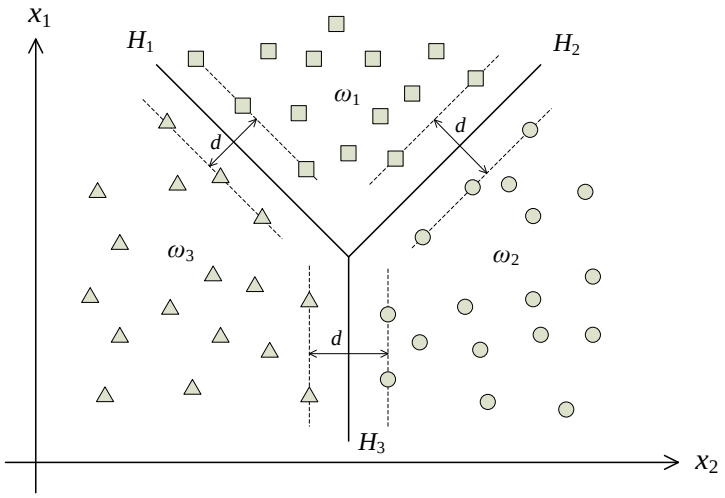


Figura 6.8: Separação de 3 classes (ω_1, ω_2 e ω_3) com SVM multiclasse.

One-Versus-All com Winner-Takes-All

Neste método são criados n classificadores SVM binários, sendo que o i-ésimo classificador é treinado com dados de ω_i, rotulados como exemplos positivos. Dados provindos de outras classes são considerados exemplos negativos. Um novo valor x, será atribuído à classe que resultar o maior valor de y_i.

One-Versus-One com Max-Wins Voting

Para este método são formados classificadores SVM binários para cada par distinto de classes, obtendo-se assim $n(n-1)/2$ classificadores. O classificador binário C_{ij} é treinado, tomando os exemplos de ω_i como positivos e ω_j como negativos. Na classificação de um valor x, se C_{ij} indicar que esse dado pertence à classe ω_i, então ω_i receberá um voto. Caso contrário, ω_j receberá o voto. Após a votação realizado por todos os classificadores, x será associado à classe que receber mais votos.

6.5 Considerações Finais

O aprendizado supervisionado é um dos paradigmas mais utilizados em problemas de classificação. Isto se deve principalmente pela facilidade de uso de seus métodos e aos seus resultados positivos. Neste projeto, redes neurais artificiais e máquinas de suporte vetorial foram escolhidos para desempenhar a tarefa de identificação das porções navegáveis de um terreno. Mesmo as duas técnicas possuindo finalidades iguais, o princípio de funcionamento é muito diferente, o

que reflete nos resultados obtidos pelos mesmos. A idéia é determinar qual abordagem permite desempenhar com maior precisão a tarefa de classificação de terrenos.

CAPÍTULO

7

Desenvolvimento do Sistema

Neste capítulo são descritos os procedimentos e os métodos utilizados para o desenvolvimento do sistema de mapeamento e de classificação de terrenos. Basicamente o sistema todo está dividido em duas partes: o algoritmo de mapeamento e o algoritmo de classificação de mapas. O processo de mapeamento tem como objetivo coletar dados do terreno por meio de um sensor e em seguida criar modelos computacionais desse ambiente. Dois métodos de mapeamento foram abordados: mapeamento 3D e mapas de navegabilidade. Para demonstrar o funcionamento dos algoritmos de mapeamento foram utilizados dados de ambientes reais coletados por meio de uma plataforma robótica equipada com um sensor laser 2D de alta resolução. Quanto ao método de classificação de mapas, este permite determinar quais regiões do terreno são apropriadas para a passagem do robô e quais não são. Sendo o aprendizado de máquina (AM), um dos paradigmas mais utilizados e bem sucedidos para problemas de classificação (Braga *et al.*, 2000), este foi escolhido como a base do nosso classificador. Com isso, foram desenvolvidos dois métodos de classificação, um fundamentado em redes neurais artificiais (RNAs) e o outro em máquinas de suporte vetorial (SVMs). As RNAs e SVMs são consideradas as técnicas mais importantes de AM supervisionado (Haykin, 1998). Posteriormente, os classificadores com RNAs e SVMs desenvolvidos foram comparados com o intuito de identificar qual técnica é a mais apropriada para a tarefa de classificação de terrenos.

7.1 Mapeamento de Terrenos

O mapeamento é uma das fases preliminares que possibilita a atuação do robô em um ambiente desconhecido (Thrun, 2003). Com os mapas, o robô torna a conhecer os obstáculos que estão a sua volta. Como a idéia desse sistema é a atuação do robô em um ambiente externo, será realizado o mapeamento do terreno para identificar as áreas seguras para a navegação. O mapeamento é realizado por meio de um sensor laser 2D disposto em um robô móvel.

Foram desenvolvidos dois métodos de mapeamento. O primeiro, mapeamento 3D, consiste no arranjamento de pontos dos objetos capturados pelo sensor em um cenário tridimensional (também chamado mapa de nuvem de pontos). O segundo, mapa de navegabilidade, consiste na simplificação do modelo criado anteriormente. Assim, ao invés dos pontos, são utilizados células que representam porções dos terrenos, reduzindo o tamanho dos mapas. Nas próximas subseções são detalhados os dois métodos criados.

7.1.1 Mapeamento 3D de Terrenos

Como descrito na Seção 3.1, o mapeamento é um componente essencial para auxiliar na navegação de robôs. Neste projeto, o mapeamento é utilizado para possibilitar a atuação de robôs em terrenos acidentados.

Basicamente, o algoritmo de mapeamento converte a coordenada do ponto coletado pelo sensor em coordenada do robô. Isto posteriormente, permitirá o robô ter o conhecimento da posição dos obstáculos a sua volta. Nos experimentos de mapeamento efetuados para a validação do algoritmo, utilizou-se conjunto de dados de ambientes previamente capturados pela navegação manual do robô.

O sistema de mapeamento foi desenvolvido para operar com um sensor laser 2D levemente inclinado com um ângulo de -10° em relação ao solo. Esse valor foi escolhido empiricamente, levando-se em consideração a altura da plataforma robótica na qual o sensor é afixado e a distância necessária para que o robô pudesse desviar de obstáculos. Nessa disposição, o sensor é capaz de detectar objetos a aproximadamente 2 metros à frente. Na Figura 7.2 é ilustrada a configuração do ambiente de experimentação. Para cada leitura realizada pelo laser é retornado um ponto $P_l(r, \alpha)$ no espaço de coordenadas polares, sendo que d e α são respectivamente a distância e o ângulo do obstáculo detectado (Figura 7.1).

O algoritmo de mapeamento 3D consiste basicamente em transformar o espaço de coordenada polar dos pontos (S) coletados pelo sensor para o espaço cartesiano ($\Re^3$) do robô. Essa função pode ser descrita como:

$$\Psi : S \mapsto \Re^3$$

onde, $S = \{(r, \alpha) | r, \alpha \in \Re\}$ e $\Re^3 = \{(x, y, z) | x, y, z \in \Re\}$. Na Figura 7.1 pode-se visualizar graficamente a transformação realizada pela função de mapeamento 3D.

Com isso, esse problema resume-se na obtenção do valor de $P_r(l_x, l_y, l_z) \in \Re^3$, dado o valor de entrada $P_l(r, \alpha) \in S$. Para simplificar, primeiramente analisamos o problema apenas no plano zx do robô, como pode ser visto na Figura 7.2. Examinando essa ilustração e sabendo que os valores de h e r são obtidos previamente, as incógnitas d e l_z podem ser determinadas por meio de cálculos trigonométricos. Assim, chegamos às seguintes expressões:

$$d = r \cdot \cos(10^\circ)$$

$$l_z = h - r \cdot \sin(10^\circ)$$

Posteriormente, para determinar as coordenadas l_x e l_y, analisamos o plano xy como mostra a Figura 7.3. Os valores p_x, p_y e θ podem ser obtidos pelo odômetro do robô, α é um dado indicado pelo sensor laser e d foi calculado anteriormente. Assim, o problema é reduzido ao cálculo dos catetos do triângulo retângulo ABC, o que leva às equações:

$$l_x = p_x + r \cdot \cos(\theta + \alpha)$$

$$l_y = p_y + r \cdot \cos(\theta + \alpha)$$

Com isso, a função de mapeamento 3D de terrenos é dada na Equação 7.1. O ponto $P_r(l_x, l_y, l_z)$ é referente a apenas um dado da leitura do laser. Dessa forma, são necessárias

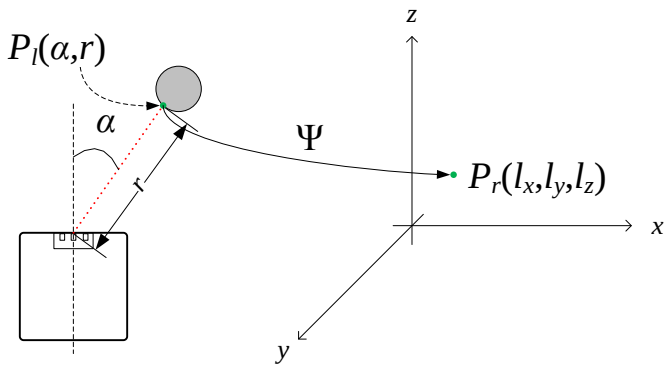
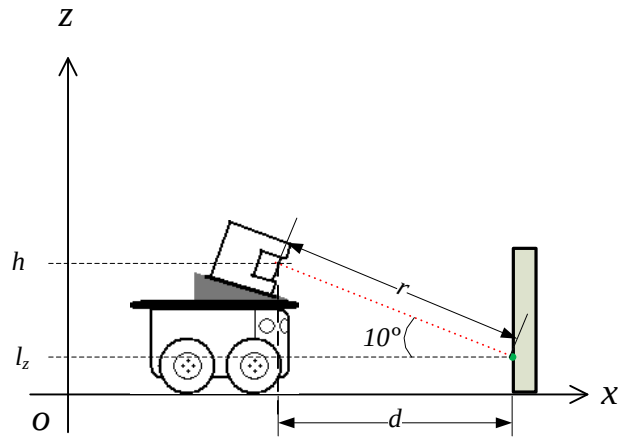


Figura 7.1: Representação gráfica da função (Ψ) de mapeamento 3D. Esta função transforma o ponto $P_l \in S$ coletado pelo laser em um ponto $P_r \in \Re^3$ do espaço do robô.



Onde:

- r é a distância entre o sensor laser e o obstáculo em relação ao plano do laser (valor dado diretamente pelo sensor)
- d é a distância entre o robô e o obstáculo em relação ao plano do robô
- l_z é a altura do obstáculo detectado pelo laser
- h é a altura do laser em relação ao plano do robô (o valor é uma constante obtida pela aferição manual)

Figura 7.2: Decomposição do feixe do laser no plano *zx*.

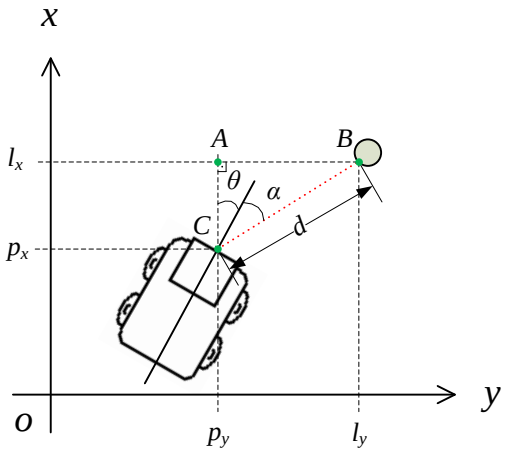
sucessivas transformações dos pontos coletados pelo sensor para se ter um mapa denso com os elementos do mundo real.

Para a representação dos mapas é utilizada uma linguagem para modelagem de cenários virtuais chamada VRML (do inglês, *Virtual Reality Modeling Language*). A linguagem VRML possibilita descrever formatos e comportamentos de objetos 3D em arquivos texto e o seu resultado visualizado através de um interpretador desse arquivo. No Apêndice C é apresentada a estrutura básica da linguagem VRML para a representação dos pontos do terreno.

$$P_r = \Psi(r, \theta) = \begin{cases} l_x = p_x + r \cdot \cos(\theta + \alpha) \\ l_y = p_y + r \cdot \cos(\theta + \alpha) \\ l_z = h - r \cdot \sin(10^\circ) \end{cases} \tag{7.1}$$

7.2 Mapa de Navegabilidade do Terreno

Como descrito na sessão anterior, os mapas criados pelo algoritmo de mapeamento descrevem os terrenos por meio de um conjunto de pontos dispostos em um espaço 3D. Esse tipo de mapa é capaz de representar cenários com um alto nível de detalhes, porém, a quantidade de pontos que esse arquivo possui pode tornar o processo de busca por obstáculos muito lento. Por essa razão, é interessante construir mapas mais compactos que possibilite ao algoritmo de classificação analisar porções do terreno, ao invés de pontos do terreno. Dessa forma, o algoritmo



Onde:

- θ é o ângulo de deslocamento do robô
- α é o ângulo do feixe do laser
- d é distância entre o robô e o obstáculo
- l_x e l_y são as coordenadas do obstáculo em relação ao plano xy
- p_x e p_y são as coordenadas do robô em relação ao plano xy

Figura 7.3: Decomposição do feixe do laser no plano xy.

de mapeamento foi remodelado para criar mapas reduzidos do terreno, chamados de mapas de navegabilidade.

O mapa de navegabilidade é representado através de uma grade bidimensional, sendo que cada porção, denominada célula, corresponde a uma região do terreno, o que reduz consideravelmente a quantidade de informações necessárias para descrever o terreno (Figura 7.4). A grade, bem como as suas células, possuem tamanhos ajustáveis. Cada célula armazena três valores que informam certas características da superfície:

1. Altura absoluta: Indica a altura da célula em relação ao plano do robô.
2. Altura relativa: Indica a altura da célula em relação às suas células vizinhas.
3. Navegabilidade: Indica se essa célula é navegável ou não.

No mapa de navegabilidade, o robô possui uma posição fixa no centro da grade. Dessa forma, a grade desloca-se conforme o movimento do robô. Em outras palavras, o mapa de navegabilidade funciona como uma janela que possui informações da área do terreno sendo explorada pelo robô. Na Figura 7.4 é ilustrado um mapa de navegabilidade de tamanho 9×7 cobrindo uma região de terreno.

O passo inicial para construir o mapa de navegabilidade é igual ao do mapa 3D do terreno. Assim, primeiramente, para cada dado de leitura do sensor laser, deve-se computar os valores l_x, l_y e l_z. Após isso, é verificado por meio do par l_x e l_y, em qual célula tal ponto se posiciona. Caso esta célula ainda não contenha nenhuma informação, armazena-se o valor da altura absoluta que é dada por l_z. Porém, se essa célula não esteje vazia, prevalece a maior altura absoluta. Toda vez que uma nova altura absoluta é armazenada deve-se atualizar a altura relativa da célula.

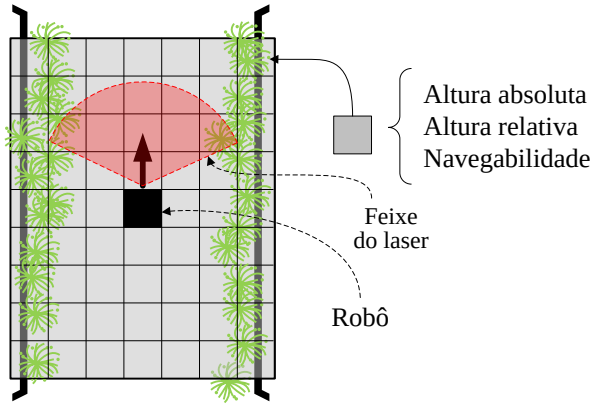


Figura 7.4: Representação de um mapa de navegabilidade de tamanho 9×7.

Para obter a altura relativa, calcula-se a diferença das alturas absolutas entre a célula atual e as vizinhas (apenas se não estiver vazia). A altura relativa será aquela que resultar na maior diferença. Outro item que deve ser computado é a navegabilidade do terreno, que é detalhado na próxima seção (Seção 7.3). Para a representação do mapa de navegabilidade, foi utilizada a biblioteca gráfica OpenCV (Bradski, 2000).

7.3 Classificação de Terrenos

Para que um robô possa navegar em um terreno desconhecido, é necessário saber os caminhos mais seguros a serem seguidos. Para determinar quais partes do terreno são trafegáveis, utiliza-se um classificador que com base nas informações contidas na célula do mapa de navegabilidade, retorna o nível de segurança. O classificador utilizado é baseado em método de aprendizado de máquina supervisionado. Cada célula é classificada em três categorias diferentes: navegável (ruas pavimentadas e calçadas), parcialmente navegável (grama e cascalho) e não navegável (paredes e postes).

O algoritmo de classificação utiliza os dados da altura relativa e absoluta da célula para verificar a navegabilidade. Sendo um método baseado em AM, a classificação é realizada através do modelo generalizado do conjunto de dados de treinamento. Dessa forma, para que o classificador funcione corretamente, é necessário que o mesmo seja treinado com exemplos que caracterizam adequadamente cada tipo de terreno. O classificador possui três saídas, uma para cada categoria de terreno, sendo que apenas uma pode ser ativada. Por fim, essa saída é armazenada na célula. A Figura 7.5 esquematiza o funcionamento da classificação de uma célula.

Para a classificação foram utilizados dois métodos: rede neural artificial (RNA) e máquina de suporte vetorial (SVM). Estas duas técnicas foram escolhidas pois são citadas na literatura como eficientes métodos para a resolução de problemas de classificação (Haykin, 1998) (Rojas, 1996) (Cristianini e Shawe-Taylor, 2000). Para a classificação com RNAs, contou-se com o auxílio da biblioteca SNNS (Zell *et al.*, 1993) e para a classificação com SVM, utilizou-se a biblioteca SVM *Light* (Joachims, 2002).

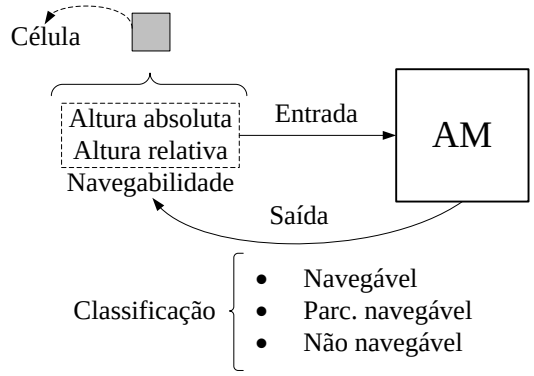


Figura 7.5: O algoritmo de classificação utiliza os valores da altura absoluta e relativa da célula para verificar a navegabilidade. Após isso, a categoria de terreno (navegável, parcialmente navegável e não navegável) retornado pelo classificador é armazenada na célula.

7.4 Experimentos e Resultados

Nesta seção serão apresentados resultados obtidos pelo algoritmo de mapeamento e de classificação de terrenos, utilizando dados de terrenos reais coletados por meio de uma plataforma robótica. Os resultados são também discutidos para demonstrar a validade dos métodos desenvolvidos.

7.4.1 Mapeamento de Terrenos

O primeiro experimento realizado consistiu na validação do algoritmo de mapeamento de terrenos. Inicialmente, foram coletados manualmente dados de terrenos utilizando um robô Pioneer 3-AT equipado com um sensor laser 2D SICK LMS 200, como mostrado na Figura 7.6. O sensor está configurado para coletar a cada varredura, 181 pontos do ambiente à 10Hz. Para controlar a plataforma robótica, a biblioteca Player foi empregada para intermediar a comunicação entre o servidor e o cliente. Todas as informações coletadas pelo robô (odometria e dados

Figura 7.6: Plataforma robótica utilizado nos experimentos.

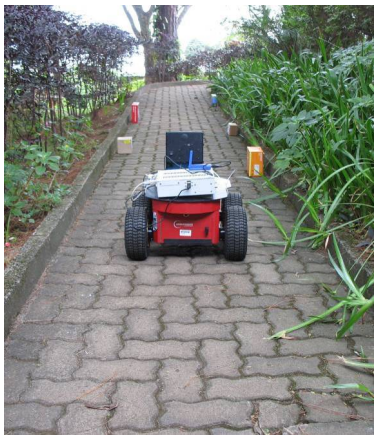
do sensor laser) foram armazenadas em arquivos de registros (*log files*) criados pela interface *Player*.

Foram escolhidos quatro cenários distintos (rotulados como Cenário I~IV) para a coleta dos dados, cada uma possuindo características específicas.

- Cenário I: Terreno inclinado com obstáculos e vegetação alta nas laterais (Figura 7.7(a)).
- Cenário II: Terreno plano com grama baixa na lateral (Figura 7.7(b)).
- Cenário III: Passagem com vegetação de tamanhos diversos nas laterais (Figura 7.7(c)).
- Cenário IV: Paralelepípedo com vegetação nas laterais (Figura 7.7(d)).

Após a fase de coleta de dados, os arquivos de registro foram alimentados no algoritmo de mapeamento 3D de terrenos. Para cada cenário foi gerado o mapa de nuvem de pontos no formato VRML, cujos resultados são apresentados na Figura 7.8. Nota-se que nos mapas gerados, pôde-se recriar detalhes precisos do terreno. Os mapas possuem na ordem de 150 mil pontos e arquivos com média de 5MB, para percursos de aproximadamente $10m$. Ou seja, em média, temos 15 mil pontos para descrever um trecho de $1m$.

Para a criação dos mapas de navegabilidade, foram utilizados os mesmos arquivos de registro coletados pela plataforma robótica apresentada inicialmente. A grade do mapa foi ajustada para tamanho 600×600 e células com dimensão de $5cm \times 5cm$. Sendo assim, a grade forma um janela que cobre uma área do terreno de $15m \times 15m$. Em um trecho de $1m$, são utilizadas 6 mil células, isto corresponde à menos da metade da quantidade de pontos utilizados no mapa 3D para descrever a mesma extensão. Com isso, podemos notar a redução obtida no mapa de navegabilidade. A Figura 7.9 mostra mapas de navegabilidade representando a altura absoluta e a



(a) Cenário I

(b) Cenário II

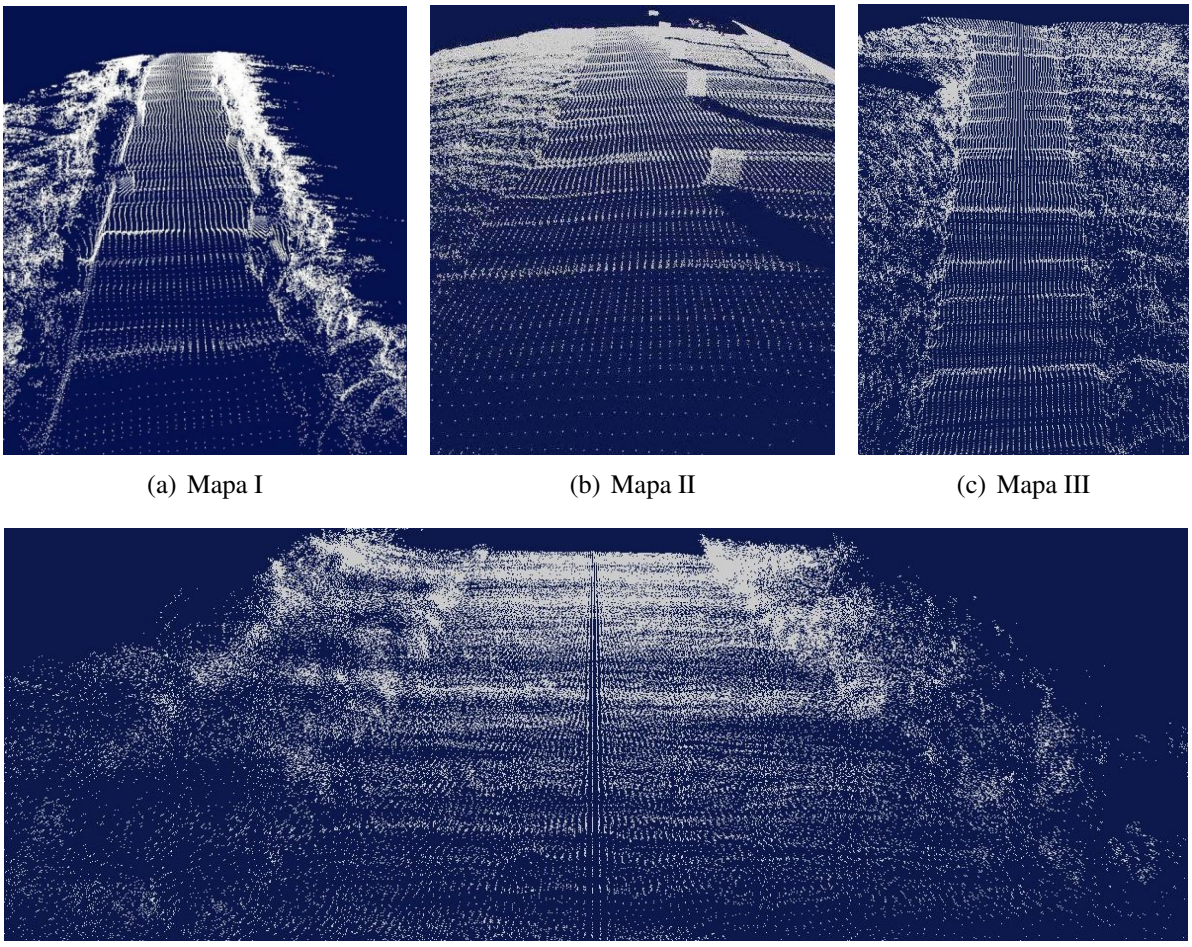
(c) Cenário III

(d) Cenário IV

Figura 7.7: Ambientes utilizados nos experimentos. (a) Via com grama na lateral. (b) Rampa. (c) Via com grama nas laterais. (d) Paralelepípedo.

altura relativa do Cenário I e do Cenário II. As alturas são representadas por meio de uma escala de cores, sendo que cores vermelhas denotam valores positivos, cores azuis, valores negativos e cores verdes, valores próximos a zero.

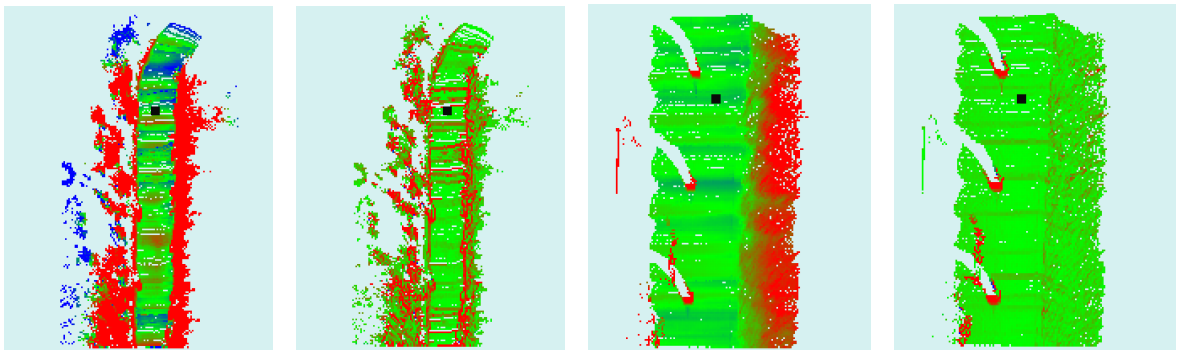
Analisando-se os mapas com as alturas absolutas, nota-se que a representação da altura do relevo do terreno é compatível com o cenário real. Porém, não poderíamos utilizar este valor como a única referência para a classificação de terrenos, pois o mesmo não detectaria declives e aclives, os quais possivelmente seriam rotulados como obstáculos (Figura 7.9(a) e Figura 7.9(c)). Em contrapartida, o uso exclusivo da altura relativa para a classificação do terreno, falharia na classificação de superfícies planas e altas como obstáculos (Figura 7.9(b) e Figura 7.9(d)). Sendo assim, esses dois valores foram utilizados em conjunto para avaliar a navegabilidade das porções do terreno.



(a) Mapa I (b) Mapa II (c) Mapa III

(d) Mapa IV

Figura 7.8: Mapas 3D gerados pelo algoritmo de mapeamento. Mapas I a IV correspondem respectivamente aos Cenários I a IV.



(a) Mapa I: Altura Absoluta (b) Mapa I: Altura Relativa (c) Mapa II: Altura Absoluta (d) Mapa II: Altura Relativa

Figura 7.9: Mapas de navegabilidade dos Cenários I e II.

7.4.2 Classificação de Terrenos

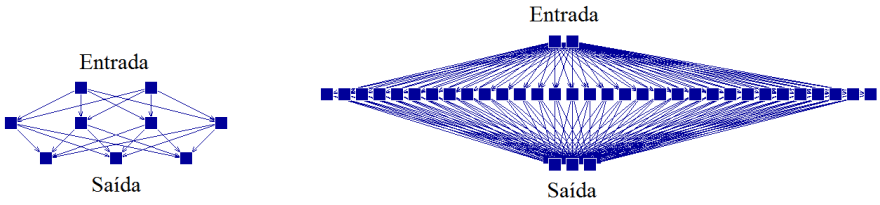
Para a classificação das células do mapa de navegabilidade gerados pelo processo de mapeamento, foram aplicados dois paradigmas de aprendizado supervisionado: RNA e SVM. Posteriormente, os resultados obtidos pelos dois métodos foram comparados de forma a determinar o classificador ideal para o conjunto de terrenos utilizados neste experimento (Cenário I~IV). No funcionamento do classificador de terrenos, a partir dos dados de entrada, os quais são a altura absoluta e relativa da célula, é determinada a classe de navegabilidade dessa porção de terreno, que se compreende em navegável (terrenos planos), parcialmente navegável (grama e cascalho) e não navegável (postes e paredes).

O passo inicial para a criação dos classificadores foi a geração de um conjunto de padrões de treinamento e um outro de validação. Na criação desse conjunto de dados, foram extraídas amostras de células do mapa de navegabilidade e rotulados manualmente. Para cada terreno apresentado na seção anterior (Cenário I~IV) foi criado um arquivo com padrão de treinamento e de validação (Padrão I~IV), utilizando amostras de células do mesmo cenário. Um arquivo adicional (Padrão V) contendo todos os padrões de treinamento do Cenário I a IV e um outro arquivo contendo todos os padrões de validação desses cenários foram gerados. Pelo fato de que o Cenário I visualmente não apresentava muitas porções de terreno consideradas parcialmente navegáveis, particularmente para este terreno, os arquivos de treinamento e de validação não continham padrões rotulados nessa categoria. Na Tabela 7.1 são listados os conjuntos de padrões criados e a quantidade de padrões de treinamento e de validação contidos em cada arquivo. Os modelos de redes neurais e SVM criados foram treinados e avaliados com os Padrões I~V.

No caso da RNA, foram utilizadas redes modeladas com a arquitetura *multilayer* perceptron. A topologia da rede consiste em 3 camadas, composta por 2 elementos de entrada (altura relativa e altura absoluta) e 3 elementos de saída (classes navegável, parcialmente navegável e não navegável). O número de elementos na camada oculta foi variado, sendo que foram criadas

		Número de padrões	
Conjunto de padrão	**Terreno**	**Treinamento**	**Validação**
Padrão I	Cenário I	628	363
Padrão II	Cenário II	783	262
Padrão III	Cenário III	2478	825
Padrão IV	Cenário IV	2529	841
Padrão V	Cenários I∼IV	6418	2291

Tabela 7.1: Lista do conjunto de padrões com informações sobre qual ambiente foi extraído os exemplos e o número de padrões de treinamento e de validação utilizados.



(a) RNA com 4 elementos na camada oculta

(b) RNA com 32 elementos na camada oculta

Figura 7.10: Ilustração das duas topologias de rede neural utilizadas nos experimentos.

redes com 4, 8, 16 e 32 elementos nessa camada. Para a manipulação das RNAs, contou-se com a biblioteca SNNS (Zell *et al.*, 1993). Na Figura 7.10 são ilustradas as topologias com 4 e 32 neurônios na camada oculta empregadas. Para o treinamento das redes neurais foi utilizado o algoritmo de aprendizado *resilient backpropagation*. Cada treinamento foi realizado com 10000 ciclos, sendo que foi constatado por métodos empíricos que este período é suficiente para a generalização das redes. Após o processo de treinamento, cada rede neural foi testada com os conjuntos de validações (Padrão I~IV) de forma a verificar a acurácia dos classificadores. A Tabela 7.2 mostra o erro obtido por cada classificador neural. Nota-se que as taxas de erro são menores nas redes testadas com padrões extraídos dos mesmos cenários que a do padrão de treinamento. Além disso, como mencionado anteriormente, como o Padrão I só tinha rótulos para duas classes (navegável e não navegável), a rede treinada com o mesmo não foi capaz de ser testada para outros padrões de validação, os quais possuiam três classes de saída. Na Figura 7.11 são ilustrados os gráficos de generalização das redes neurais que obtiveram os menores erros para cada padrão validado. Nota-se que em todos os casos, após 10000 ciclos de treinamento a rede perde a capacidade de generalização.

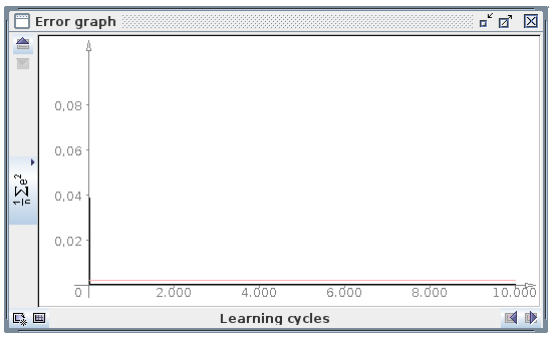
Para o treinamento do classificador SVM, foram utilizados quatro kernels diferentes: linear (K0), polinomial (K1), função base radial (RBF) (K2) e sigmoid (K3). A biblioteca SVM *Light* (Joachims, 2002) foi utilizada para a criação dos classificadores. Com o intuito de comparar o desempenho do SVM com a RNA, foram efetuados os mesmos experimentos no processo de treinamento e de validação dos classificadores SVM. Assim, os padrões de treinamento foram utilizados para a aprendizagem dos classificadores e em seguida testados com os padrões de validação. A Tabela 7.3 mostra as taxas de erro no processo de classificação dos padrões de validação. Note que o algoritmo de treinamento não convergiu para o *kernel* RBF para o Padrão IV e o Padrão V, consequentemente não puderam ser criados classificadores obtidos por meio destes padrões. Da mesma forma como nas RNAs, os modelos avaliados utilizando os mesmos cenários que os do treinamento obtiveram os menores erros de classificação.

Modelo de treinamento		**Conjunto de validação** (erro%)				
		Padrão I	**Padrão II**	**Padrão III**	**Padrão IV**	**Padrão V**
Padrão I	4 n.o.	**>0,00%**	-	-	-	-
	8 n.o.	>0,00%	-	-	-	-
	16 n.o.	>0,00%	-	-	-	-
	32 n.o.	>0,00%	-	-	-	-
Padrão II	4 n.o.	**2,29%**	1,10%	10,67%	21,05%	12,00%
	8 n.o.	5.73%	**0,28%**	6,91%	**14,15%**	**8,38%**
	16 n.o.	16,03%	4,68%	7,76%	42,93%	21,13%
	32 n.o.	11,83%	2,20%	**5,45%**	33,53%	15,98%
Padrão III	4 n.o.	13,36%	**1,10%**	5,45%	42,69%	19,34%
	8 n.o.	**8,02%**	3,03%	4,85%	**34,72%**	**15,89%**
	16 n.o.	11,07%	1,38%	**4,48%**	36,03%	16,32%
	32 n.o.	11,45%	2,75%	6,55%	37,46%	17,85%
Padrão IV	4 n.o.	11,45%	**1,65%**	22,67%	11,89%	14,10%
	8 n.o.	**10,31%**	5,23%	**19,52%**	**10,23%**	**12,79%**
	16 n.o.	11,45%	4,96%	20,12%	12,01%	13,75%
	32 n.o.	11,07%	6,34%	20,97%	11,77%	14,14%
Padrão V	4 n.o.	6,87%	**>0,00%**	6,67%	13,67%	7,99%
	8 n.o.	**4,96%**	>0,00%	6,91%	**11,41%**	**7,16%**
	16 n.o.	4,96%	0,28%	**5,94%**	14,39%	8,03%
	32 n.o.	3,82%	>0,00%	7,27%	13,91%	8,21%

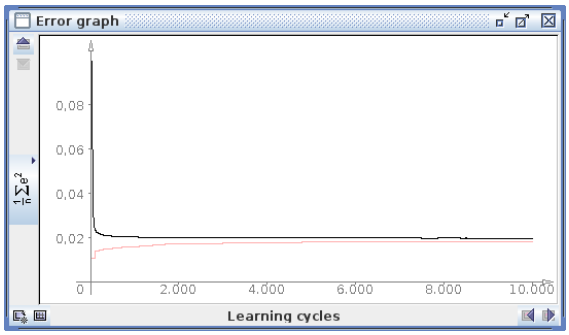
Tabela 7.2: Taxa de erro no processo de validação das redes neurais para cada padrão(n.o. denota o número de neurônios ocultos usados na rede). As menores taxas de erros de cada modelo avaliado estão destacadas em negrito.

Modelo de treinamento		**Conjunto de validação** (erro%)				
		Padrão I	**Padrão II**	**Padrão III**	**Padrão IV**	**Padrão V**
Padrão I	K0	34,35%	-	-	-	-
	K1	**0,38%**	-	-	-	-
	K2	0,38%	-	-	-	-
	K3	0,38%	-	-	-	-
Padrão II	K0	70,61%	52,62%	49,94%	41,62%	45,00%
	K1	38,55%	**0,28%**	**11,76%**	22,59%	13,92%
	K2	**38,17%**	2,20%	13,70%	22,59%	14,32%
	K3	38,55%	1,93%	12,24%	**21,05%**	**13,79%**
Padrão III	K0	91,60%	74,10%	51,52%	68,85%	62,55%
	K1	**46,56%**	1,10%	7,27%	38,41%	18,03%
	K2	48,09%	1,38%	**6,79%**	37,46%	17,37%
	K3	47,71%	**0,83%**	7,15%	**33,53%**	**15,93%**
Padrão IV	K0	79,77%	50,96%	54,30%	43,64%	47,58%
	K1	**32,06%**	11,02%	33,70%	17,72%	21,96%
	K2	32,44%	**10,74%**	**33,09%**	**17,48%**	**21,56%**
	K3	-	-	-	-	-
Padrão V	K0	79,39%	51,24%	54,06%	43,28%	47,45%
	K1	42,37%	**1,10%**	12,24%	**20,69%**	12,75%
	K2	**41,98%**	1,10%	**12,00%**	20,69%	**12,70%**
	K3	-	-	-	-	-

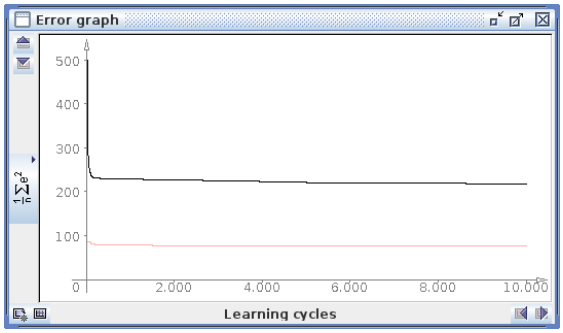
Tabela 7.3: Taxa de erro no processo de validação dos classificadores SVM utilizando quatro diferentes *kernels*. As menores taxas de erros de cada modelo avaliado estão destacadas em negrito.



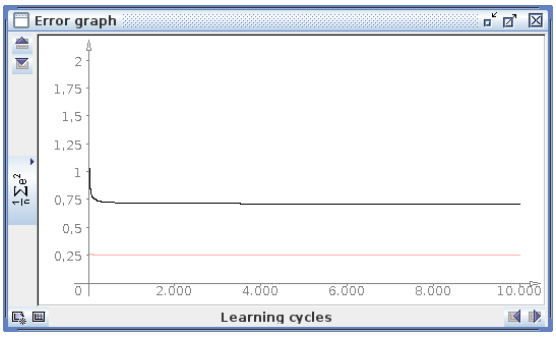
(a) RNA - Padrão I



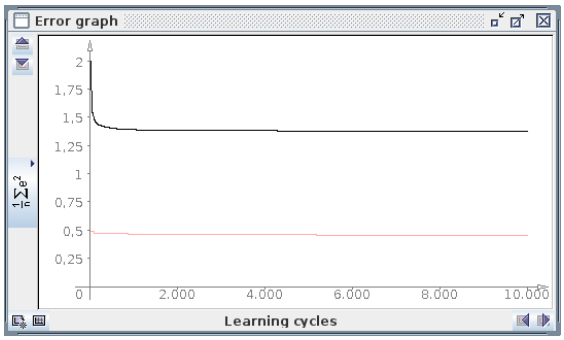
(b) RNA - Padrão II



(c) RNA - Padrão III



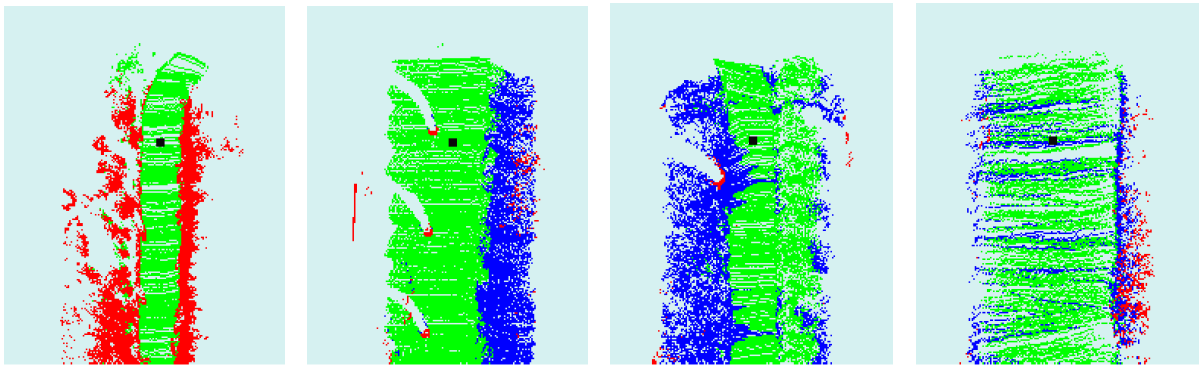
(d) RNA - Padrão IV



(e) RNA - Padrão V

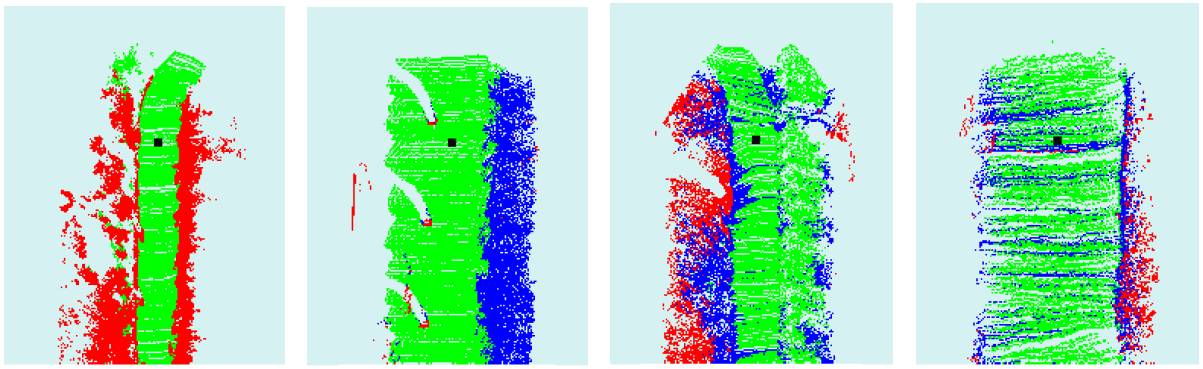
Figura 7.11: Gráficos da generalização de cada rede neural que obteve os melhores resultados para cada Padrão. A curva em preto representa o erro no processo de treinamento dos dados de treinamento e a curva em vervelho dos dados de validação. A função de erro utilizado é o erro quadrático médio.

Após o treinamento e teste dos classificadores de terrenos, utilizamo-os para classificar os arquivos de registros coletados inicialmente pela plataforma robótica (cada arquivo de registro contém dados de um terreno (cenário) capturado pelo sensor laser). Para esta tarefa utilizamos as redes e modelos SVM que obtiveram os menores erros no processo de validação de cada cenário. Na Figura 7.12 são apresentados os cenários classificados com RNA e na Figura 7.13 com SVM. Os pontos vermelhos se referem às porções não navegáveis, pontos azuis às porções



(a) Cenário I (b) Cenário II (c) Cenário III (d) Cenário IV

Figura 7.12: Resultado das melhores classificações utilizando RNA. Células vermelhas representam as regiões não navegáveis, azuis as regiões parcialmente navegáveis e verdes as regiões navegáveis.



(a) Cenário I (b) Cenário II (c) Cenário III (d) Cenário IV

Figura 7.13: Resultado das melhores classificações utilizando SVM. Células vermelhas representam as regiões não navegáveis, azuis as regiões parcialmente navegáveis e verdes as regiões navegáveis.

parcialmente navegáveis e pontos verdes às porções navegáveis. Nos Cenários I, II e IV, ambos os métodos obtiveram resultados bem similares, porém quanto ao Cenário III, através da análise visual, notamos que o método SVM classificou incorretamente algumas células do mapa que eram parcialmente navegáveis como não navegáveis.

Para determinar qual técnica de aprendizado de máquina alcançou os melhores resultados na tarefa de classificação de terrenos, foi comparado o número de vezes, para todos os modelos treinados, qual classificador obteve a menor taxa de erro para cada conjunto de validação. A Tabela 7.4 mostra os resultados da comparação. Nesta tabela podemos observar que exceto para o modelo treinado com o Padrão III, em todos os outros o classificador por RNA obteve melhores resultados. Em outras palavras, as redes neurais classificaram melhor 13 vezes contra

	Classificador	
Modelo treinado	**RNA**	**SVM**
Padrão I	1	0
Padrão II	3	0
Padrão III	2	2
Padrão IV	4	0
Padrão V	4	0
Total	**13**	**2**

Tabela 7.4: Número de vezes que os classificadores RNA e SVM obtiveram as menores taxas de erro para cada conjunto de validação.

duas para o SVM. Além de que nos resultados gráficos, pôde-se confirmar a superioridade das RNAs por gerar classificações mais precisas, como mencionado para o Cenário III. Outra desvantagem encontrada no SVM foi a incapacidade do algoritmo de convergir para o Padrão IV e Padrão V. Uma explicação para os resultados inferiores para o classificador SVM é o fato de que este método necessita do ajuste correto de diversos parâmetros de forma a funcionar corretamente.

7.5 Conclusão

Neste capítulo foi apresentado um método para mapeamento e classificação de ambientes externos. Na realização dos experimentos para a validação dos resultados, foram utilizados dados reais de ambientes coletados por meio de um sensor laser 2D equipado em um robô móvel.

Primeiramente, foi desenvolvido um método para criação de mapas 3D de ambientes. Este mapa é composto por um conjunto de pontos que descrevem objetos do ambiente. Nos mapas gerados, nota-se que detalhes minuciosos dos terrenos puderam ser reproduzidos, constatando a precisão do algoritmo. Entretanto, aplicar um mapa de nuvem de pontos para a classificação de terrenos não seria muito interessante, pois, seria necessário realizar uma análise ponto a ponto. Dessa forma, uma estrutura chamada mapas de navegabilidade foi desenvolvida para reduzir a quantidade de informação a ser processada para a identificação de obstáculo nos terrenos. A unidade básica dos mapas de navegabilidade são as células, as quais representam regiões quadriculares do terreno. Utilizando esta estrutura, pode-se reduzir mais que pela metade o volume de informações que seriam necessárias para a classificação de terrenos.

Em relação ao sistema de classificação de terrenos, foram abordados dois métodos baseados em aprendizado de máquina supervisionado, sendo estes, RNA e SVM. Para o treinamento dos classificadores, foram utilizados padrões de treinamento e de validação extraídos de ambientes

reais e rotulados manualmente. Verificando os resultados obtidos pela validação dos resultados, nota-se que os classificadores por RNAs geraram classificações mais precisas. Além disso, verificando-se os resultados gráficos das classificações, constata-se a superioridade das RNAs nestes experimentos.

CAPÍTULO

8

Conclusão

Esta dissertação apresentou um sistema de classificação da navegabilidade de terrenos acidentados utilizando técnicas de aprendizado supervisionado (RNA e SVM). Os resultados das classificações são apresentados em um mapa de navegabilidade, distinguindo as regiões navegáveis, parcialmente navegáveis e não navegáveis do terreno.

Para o desenvolvimento deste trabalho, inicialmente foram pesquisados os diversos tipos de sensores aplicados na robótica móvel. Foi verificado que o sensor laser possui uma alta freqüência para coleta de dados, além de uma boa faixa de alcance, dessa forma, este sensor foi escolhido neste projeto. Posteriormente, foram abordados os principais algoritmos utilizados na robótica móvel, divididos em mapeamento, localização e navegação. Por meio deste estudo, pôde-se constatar que o algoritmo de mapeamento exerce um papel fundamental para a navegação autônoma, pois, fornece ao robô informações sobre os possíveis obstáculos existentes no ambiente. Em seguida, foi realizado um levantamento bibliográfico com os pricipais trabalhos sobre mapeamento e navegação robótica am abientes externos dos últimos anos. Constatou-se que para a atuação de robôs nos ambientes externos, a análise do ambiente é muito mais minuscioso em relação aos ambientes internos, devido a presença de obstáculos complexos e imprevistos. Resultados promissores foram obtidos pela competição de carros autônomos organizado pelo DARPA, mostrando a evolução e a importância que este problema têm trazido pela comunidade científica. Foram também estudados os dois métodos de aprendizado de máquina supervisionado utilizados neste projeto: RNAs e SVMs. O funcionamento desses métodos é

baseado na generalização de modelos treinados com um conjunto de dados que retratam o comportamento de um determinado sistema. Sendo assim, os dados e o algoritmo de treinamento escolhidos influenciam na capacidade de resolução de problemas pelo modelo.

Quanto ao desenvolvimento do algoritmo de classificação de terrenos, este foi dividido em duas etapas: mapeamento e classificação de terrenos. O algoritmo de mapeamento consiste na criação de um modelo virtual do terreno por meio dos dados coletados por um sensor. Nos exeprimentos realizados, foi utilizado uma plataforma robótica Pioneer AT-3 equipado com um sensor laser SICK LMS 200. Com o mapa de nuvem de pontos gerados pôde-se observar o alto nível de detalhamento do cenário reproduzido. Considerando que o elevado número de pontos poderia se tornar um garagalo para o algoritmo de classificação de terrenos, este mapa foi convertido em um mapa de navegabilidade. Este mapa é composto por células quadradas que representam uma porção do terreno. Dessa forma, a quantidade de informações do mapa foi reduzida aproximadamente pela metade.

Os algoritmos para a classificação foram treinados e validados com dados reais extraídos de quatro terrenos coletados pela plataforma robótica. Comparando-se as taxas de acerto obtidos pelos dois métodos de aprendizado supervisionado, verificou-se que as RNAs resultaram em classificações mais precisas comparados com as SVMs. Analisando-se os mapas de navegabilidade classificados com as RNAs, nota-se que este sistema poderia ser eficientemente utilizado na navegação de robôs em terrenos desestruturados.

Este projeto demonstrou que apesar das diversas técnicas de mapeamento e classificação existentes na literatura, este é um assunto da robótica móvel que ainda pode ser explorado buscando-se algoritmos cada vez mais eficientes. Isto se justifica pelo fato de que os robôs que irão utilizar este sistema atuarão em ambientes externos, sendo assim, os mesmos deverão operar em tempo real, ser tolerante às falhas e ter baixo consumo de energia. Com isso, devemos apresentar as mais diversas soluções possíveis para este problema, de modo a determinar qual técnica se adapta melhor.

Além disso, este trabalho contribui para o projeto do Laboratório de Robótica Móvel do ICMC-USP em parceria com o INCT-SEC que tem como proposta o desenvolvimento de um carro autônomo. O veículo irá navegar autonomamente pelo ambiente, dispensando a presença do motorista. Dessa forma, o sistema de mapeamento e classificação de terrenos apresentado pode ser visto também como uma contribuição para o sistema de navegação deste veículo, sendo que pode-se utilizar o mapa de navegabilidade para determinar as áreas navegáveis da rua.

8.1 Trabalhos Futuros

Dentre os trabalhos futuros inclui o teste do sistema de classificação com outros métodos de aprendizado como algoritmos genéticos e lógica difusa. Além disso, visa-se a integração do sistema de classificação de terrenos com um algoritmo de navegação, possibilitando a atuação autônoma de robôs em ambientes externos. Podem ser realizados também testes experimentais com robôs em alta velocidade, de modo a verificar se o sistema permite operar em tempo real. Com isso, pode-se verificar a validade do método implementado para aplicações reais (veículos autônomos), analisando-se as melhorias que poderiam ser feitas ao sistema.

Referências Bibliográficas

BAJRACHARYA, M.; TANG, B.; HOWARD, A.; TURMON, M.; MATTHIES, L. Learning long-range terrain classification for autonomous navigation. In: *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 2008, p. 4018–4024.

BALAGUER, B.; CARPIN, S. Where am i? a simulated gps sensor for outdoor robotic applications. In: *SIMPAR '08: Proceedings of the 1st International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Berlin, Heidelberg: Springer-Verlag, 2008, p. 222–233.

BANNO, A.; IKEUCHI, K. Shape recovery of 3d data obtained from a moving range sensor by using image sequences. In: *Proc. Tenth IEEE International Conference on Computer Vision ICCV 2005*, 2005, p. 792–799.

BARANAUSKAS, J. A.; MONARD, M. C. Combining symbolic classifiers from multiple inducers. *Knowledge-Based Systems*, v. 16, n. 3, p. 129 – 136, 2003.
Disponível em `http://www.sciencedirect.com/science/article/B6V0P-45PK78W-3/2/097ca485beaba2de5390be29611fa7d6`

BATAVIA, P. H.; NOURBAKHSH, I. Path planning for the cye personal robot. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, 2000, p. 15–20.

BISHOP, R. Intelligent vehicle applications worldwide. *IEEE Intelligent Systems*, v. 15, n. 1, p. 78–81, 2000.

BLAER, P.; ALLEN, P. Data acquisition and view planning for 3-d modeling tasks. In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, p. 417–422.

BOK, Y.; HWANG, Y.; KWEON, I.-S. Accurate motion estimation and high-precision 3d reconstruction by sensor fusion. In: *ICRA*, 2007, p. 4721–4726.

BORENSTEIN, J.; EVERETT, H.; FENG, L. *"where am i?"sensors and methods for autonomous mobile robot positioning*. Relatório Técnico, University of Michigan, 1996.

BORENSTEIN, J.; EVERETT, H.; FENG, L.; WEHE, D. Mobile robot positioning sensors and techniques. *Journal of Robotic Systems, Special Issue on Mobile Robots, Vol. 14, No. 4*, 1997.

BORENSTEIN, J.; FENG, L. Measurement and correction of systematic odometry errors in mobile robots. *Robotics and Automation, IEEE Transactions on*, v. 12, n. 6, p. 869–880, 1996.

BORENSTEIN, J.; KOREN, Y. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In: *Proc. IEEE International Conference on Robotics and Automation*, 1990, p. 572–577.

BORENSTEIN, J.; KOREN, Y. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, v. 7, n. 3, p. 278–288, 1991.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

BRAGA, A. P.; DE CARVALHO, A. P. L.; LUDERMIR, T. B. *Redes neurais artificiais - teoria e aplicações*. 2 ed. Rio de Janeiro: LTC, 2000.

BURGARD, W.; FOX, D.; HENNIG, D.; SCHMIDT, T. Estimating the absolute position of a mobile robot using position probability grids. In: *In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Menlo Park*, AAAI, AAAI Press/MIT Press, 1996, p. 896–901.

BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining Knowl. Discov.*, v. 2, n. 2, p. 121–167, 1998.

CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. An overview of machine learning. In: MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M., eds. *Machine Learning: An Artificial Intelligence Approach*, Berlin, Heidelberg: Springer, p. 3–23, 1984.

CASTEJON, C.; BLANCO, D.; MORENO, L. Compact modeling technique for outdoor navigation. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, v. 38, n. 1, p. 9–24, 2008.

CONGDON, C.; HUBER, M.; KORTENKAMP, D.; BIDLACK, C.; COHEN, C.; HUFFMAN, S.; KOSS, F.; RASCHKE, U.; WEYMOUTH, T. Carmel vs flakey: A comparison of two robots. *AI Magazine*, 1993.

CRAMMER, K.; SINGER, Y.; CRISTIANINI, N.; SHAWE-TAYLOR, J.; WILLIAMSON, B. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, v. 2, p. 2001, 2001.

CRISTIANINI, N.; SHAWE-TAYLOR, J. *An introduction to support vector machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.
Disponível em `http://portal.acm.org/citation.cfm?id=345662`

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. Monte carlo localization for mobile robots. In: *IEEE International Conference on Robotics and Automation (ICRA99)*, 1999.

DUAN, K.-B.; KEERTHI, S. S. Which is the best multiclass svm method? an empirical study. p. 278–285, 2005.
Disponível em `http://dx.doi.org/10.1007/11494683_28`

FAHLMAN, S. E. Faster-learning variations on back-propagation: An empirical study. In: *Connectionist Models Summer School*, Los Altos CA: Morgan-Kaufmann, 1988, p. 11–20.

FOX, D. *Markov localization: A probabilistic framework for mobile robot localization and navigation*. Tese de Doutorado, University of Bonn, Germany, 1998.
Disponível em `citeseer.ist.psu.edu/fox98markov.html`

FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. *Robotics & Automation Magazine, IEEE*, v. 4, n. 1, p. 23–33, 1997.

GAT, E. Three-layer architectures. In: *Artificial intelligence and mobile robots: case studies of successful robot systems*, Cambridge, MA, USA: MIT Press, 1998, p. 195–210.

GERKEY, B. P.; VAUGHAN, R. T.; STOY, K.; HOWARD, A.; SUKHATME, G. S.; MATARIC, M. J. Most valuable player: a robot device server for distributed control. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, p. 1226–1231.

GHAHRAMANI, Z. Unsupervised learning. In: *Advanced Lectures on Machine Learning*, 2003, p. 72–112.

HÄHNEL, D.; BURGARD, W.; THRUN, S. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, v. 44, n. 1, p. 15–27, 2003.

HAMNER, B.; SCHERER, S.; SINGH, S. Learning to drive among obstacles. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, p. 2663–2669.

HARRISON, A.; NEWMAN, P. High quality 3d laser ranging under general vehicle motion. In: *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 2008, p. 7–12.

HAYKIN, S. *Neural networks: A comprehensive foundation (2nd edition)*. 2 ed. Prentice Hall, 1998.
Disponível em `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0132733501`

HEERO, K. *Path planning and learning strategies for mobile robots in dynamic partially unknown environments*. Tese de Doutorado, Faculty of Mathmatics and Computer Science, University of Tartu, Estonia, 2006.

HORMANN, K.; REIMERS, M. Triangulating point clouds with spherical topology. In: LYCHE, T.; MAZURE, M.-L.; SCHUMAKER, L. L., eds. *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Mathematics, Brentwood, TN: Nashboro Press, p. 215–224, 2003.

JOACHIMS, T. SVM light, http://svmlight.joachims.org. 2002.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, v. 4, p. 237–285, 1996.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, v. 5, n. 1, p. 90–98, 1986.

KORTENKAMP, D.; BONASSO, R. P.; MURPHY, R., eds. *Artificial intelligence and mobile robots: case studies of successful robot systems*. Cambridge, MA, USA: MIT Press, 1998.

KUIPERS, B.; TAI BYUN, Y. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, v. 8, p. 47–63, 1991.

KURISU, M.; MUROI, H.; YOKOKOHJI, Y.; KUWAHARA, H. Development of a laser range finder for 3d map-building in rubble; installation in a rescue robot. In: *Proc. International Conference on Mechatronics and Automation ICMA 2007*, 2007, p. 2054–2059.

KWEON, I. S.; KANADE, T. High resolution terrain map from multiple sensor data. In: *Proc. IEEE International Workshop on Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications' IROS '90*, 1990, p. 127–134.

LALONDE, J.-F.; VANDAPEL, N.; HUBER, D.; HEBERT, M. *Natural terrain classification using three-dimensional ladar data for ground robot mobility*, v. 23, n. 1, p. 839 – 861, 2006.

LANGER, D.; ROSENBLATT, J. K.; HEBERT, M. A behavior-based system for off-road navigation. v. 10, n. 6, p. 776–783, 1994.

LIN, W.; MU, C.; TAKASE, K. Path planning with topological map built with id tag and web camera. In: *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, 2006, p. 1739–1744.

LORENA, A. C.; DE CARVALHO, A. C. P. L. F. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada (RITA)*, v. XIV, n. 2, p. 43–67, 2007.

MATA, M.; ARMINGOL, J. M.; DE LA ESCALERA, A.; RODRIGUEZ, F. J. A deformable model-based visual system for mobile robot topologic navigation. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003, p. 3504–3509.

MATARIC, M. J. *A distributed model for mobile robot environment-learning and navigation*. Relatório Técnico, Cambridge, MA, USA, 1990.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–137, 1943.

MITCHELL, T. *Machine learning (mcgraw-hill international edit)*. 1st ed. McGraw-Hill Education (ISE Editions), 1997.
Disponível em `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0071154671`

MORALES, Y.; TAKEUCHI, E.; CARBALLO, A.; TOKUNAGA, W.; KUNIYOSHI, H.; ABURADANI, A.; HIROSAWA, A.; NAGASAKA, Y.; SUZUKI, Y.; TSUBOUCHI, T. 1km autonomous robot navigation on outdoor pedestrian paths &unning the tsukuba challenge 2007&. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2008*, 2008, p. 219–225.

MORAVEC, H.; ELFES, A. E. High resolution maps from wide angle sonar. In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985, p. 116 – 121.

MULLER, J. R.; HARDING, D. J. Using lidar surface deformation mapping to constrain earthquake magnitudes on the seattle fault in washington state, usa. In: *Proc. Urban Remote Sensing Joint Event*, 2007, p. 1–7.

MURPHY, R. R. *Introduction to ai robotics*. Cambridge, MA, USA: MIT Press, 2000.

PATZ, B. J.; PAPELIS, Y.; PILLAT, R.; STEIN, G.; HARPER, D. A practical approach to robotic design for the darpa urban challenge. *J. Field Robot.*, v. 25, n. 8, p. 528–566, 2008.

PFAFF, P.; TRIEBEL, R.; STACHNISS, C.; LAMON, P.; BURGARD, W.; SIEGWART, R. Towards mapping of cities. In: *Proc. IEEE International Conference on Robotics and Automation*, 2007, p. 4807–4813.

PIMENTA, L.; FONSECA, A.; PEREIRA, G.; MESQUITA, R.; SILVA, E.; CAMINHAS, W.; CAMPOS, M. Robot navigation based on electrostatic field computation. *Magnetics, IEEE Transactions on*, v. 42, n. 4, p. 1459–1462, 2006.

PINIES, P.; TARDOS, J. D.; NEIRA, J. Localization of avalanche victims using robocentric slam. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, p. 3074–3079.

PRECHELT, L. Early stopping-but when? In: *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, London, UK: Springer-Verlag, 1998, p. 55–69.

RASMUSSEN, C. Combining laser range, color, and texture cues for autonomous road following. *Proceedings of IEEE International Conference on Robotics & Automation*, 2002.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: *Neural Networks, 1993., IEEE International Conference on*, 1993, p. 586–591 vol.1.

RIEDMILLER, M.; BRAUN, H. *Rprop – description and implementation details*. Relatório Técnico, Universitat Karlsruhe, 1994.
Disponível em `citeseer.ist.psu.edu/riedmiller94rprop.html`

ROJAS, R. *Neural networks: a systematic introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1996.

ROSIN, P.; FIERENS, F. Improving neural network generalisation. In: *Geoscience and Remote Sensing Symposium, 1995. IGARSS '95. 'Quantitative Remote Sensing for Science and Applications', International*, 1995, p. 1255–1257 vol.2.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation, p. 318–362. 1986.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: A modern approach*. Pearson Education, 2003.
Disponível em `http://portal.acm.org/citation.cfm?id=773294`

SANDLER, B.-Z. *Robotics : designing the mechanisms for automated machinery*. London: Prentice Hall International, London, 1991.

KHALIL SARI ALSMADI, M.; OMAR, K. B.; NOAH, S. A. Back propagation algorithm: The best algorithm among the multilayer perceptron algorithm. *International Journal of Computer Science and Network Security (IJCSN)*, v. 9, n. 4, p. 378–383, 2009.

SCHAFER, H.; HACH, A.; PROETZSCH, M.; BERNS, K. 3d obstacle detection and avoidance in vegetated off-road terrain. In: *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 2008, p. 923–928.

SCHIELE, B.; CROWLEY, J. L. A comparison of position estimation techniques using occupancy grids. In: *In Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, p. 1628–1634.

SIEGWART, R.; NOURBAKHSH, I. R. *Introduction to autonomous mobile robots*. Bradford Book, 2004.

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics, p. 167–193. 1990.

STAVENS, D.; THRUN, S. A self-supervised terrain roughness estimator for off-road autonomous driving. In: *Proceedings of the Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, Arlington, Virginia: AUAI Press, 2006, p. 469–476.

SUN, Z.; BEBIS, G.; MILLER, R. On-road vehicle detection using optical sensors: a review. In: *Proc. 7th International IEEE Conference on Intelligent Transportation Systems*, 2004, p. 585–590.

THRUN, S. Bayesian landmark learning for mobile robot localization. *Mach. Learn.*, v. 33, n. 1, p. 41–76, 1998.

THRUN, S. Robotic mapping: a survey. In: *Exploring artificial intelligence in the new millennium*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, p. 1–35.

THRUN, S.; BÜCKEN, A. Integrating grid-based and topological maps for mobile robot navigation. *Proceedings of AAAI*, 1996.

THRUN, S.; BÜCKEN, A.; BURGARD, W.; FOX, D.; FRÖHLINGHAUS, T.; HENNIG, D.; HOFMANN, T.; KRELL, M.; SCHMIDT, T. Map learning and high-speed navigation in rhino, p. 21–52. 1998a.

THRUN, S.; BURGARD, W.; FOX, D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: *Proc. IEEE International Conference on Robotics and Automation ICRA '00*, 2000, p. 321–328.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. MIT Press, 2005.

THRUN, S.; GUTMANN, J.-S.; FOX, D.; BURGARD, W.; KUIPERS, B. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In: *AAAI/IAAI*, 1998b, p. 989–995.

THRUN, S.; MARTIN, C.; LIU, Y.; HAHNEL, D.; EMERY-MONTEMERLO, R.; CHAKRABARTI, D.; BURGARD, W. A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, v. 20, n. 3, p. 433–443, 2004.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; VAN NIEKERK, J.; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. *Stanley: The Robot that Won the DARPA Grand Challenge*, v. 23, n. 1, p. 661–692, 2006.

URMSON, C.; ANHALT, J.; BAE, H.; BAGNELL, J. A. D.; BAKER, C.; BITTNER, R. E.; BROWN, T.; CLARK, M. N.; DARMS, M.; DEMITRISH, D.; DOLAN, J.; DUGGINS, D.; FERGUSON, D.; GALATALI, T.; GEYER, C. M.; GITTLEMAN, M.; HARBAUGH, S.; HEBERT, M.; HOWARD, T.; KOLSKI, S.; LIKHACHEV, M.; LITKOUHI, B.; KELLY, A.; MCNAUGHTON, M.; MILLER, N.; NICKOLAOU, J.; PETERSON, K.; PILNICK, B.; RAJKUMAR, R.; RYBSKI, P.; SADEKAR, V.; SALESKY, B.; SEO, Y.-W.; SINGH, S.; SNIDER, J. M.; STRUBLE, J. C.; STENTZ, A. T.; TAYLOR, M.; WHITTAKER, W. R. L.; WOLKOWICKI, Z.; ZHANG, W.; ZIGLAR, J. *Autonomous driving in urban environments: Boss and the Urban Challenge*, v. 25, n. 1, p. 425–466, 2008.

WOLF, D.; HOWARD, A.; SUKHATME, G. S. Towards geometric 3d mapping of outdoor environments using mobile robots. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005a, p. 1507–1512.

WOLF, D. F. *Semantic mapping using mobile robots*. Tese de Doutorado, University of Southern California, 2006.

WOLF, D. F.; SUKHATME, G. S.; FOX, D.; BURGARD, W. Autonomous terrain mapping and classification using hidden markov models. In: *Proc. IEEE International Conference on Robotics and Automation ICRA 2005*, 2005b, p. 2026–2031.

YAN, X.; FENG LI, W.; FANG CHEN, D. A new mechanism for robots control based on player/stage. In: *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO '06*, 2006, p. 750–754.

YE, C. Polar traversability index: A measure of terrain traversal property for mobile robot navigation in urban environments. In: *Proc. ISIC Systems, Man and Cybernetics IEEE International Conference on*, 2007, p. 2342–2347.

YE, C.; BORENSTEIN, J. A new terrain mapping method for mobile robots obstacle negotiation. *Unmanned Ground Vehicle Technology V*, v. 5083, n. 1, p. 52–62, 2003. Disponível em `http://link.aip.org/link/?PSI/5083/52/1`

YE, C.; BORENSTEIN, J. A method for mobile robot navigation on rough terrain. In: *Proc. IEEE International Conference on Robotics and Automation ICRA '04*, 2004, p. 3863–3869.

ZELL, A.; MACHE, N.; HÜBNER, R.; MAMIER, G.; VOGT, M.; UWE HERRMANN, K.; SCHMALZL, M.; SOMMER, T.; HATZIGEORGIOU, A.; DÖRING, S.; POSSELT, D.; MARTIN, M. R. Snns - stuttgart neural network simulator, http://www.ra.cs.uni-tuebingen.de/snns/. 1993.

ZHANG, Z.; GUO, H.; NEJAT, G.; HUANG, P. Finding disaster victims: A sensory system for robot-assisted 3d mapping of urban search and rescue environments. In: *Proc. IEEE International Conference on Robotics and Automation*, 2007, p. 3889–3894.

APÊNDICE

A

Mapeamento 3D de Ambientes

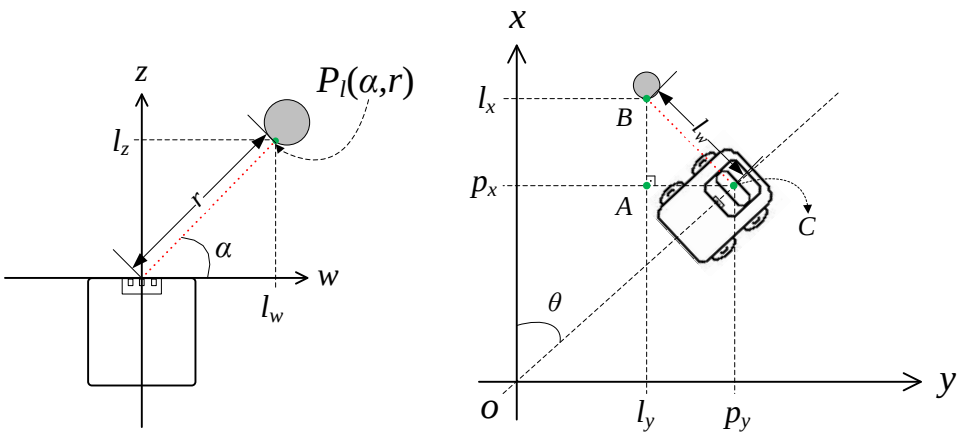
O sistema de mapeamento abordado neste projeto teve como objetivo recriar modelos de terrenos. Porém, muitas vezes é interessante gerar mapas não apenas do terreno, mas também contendo todos os elementos existentes no ambiente externo. Isto acontece para casos em que é utilizado utiliza-se robôs aéreos para a navegação, manutenção de estruturas de construções ou simplesmente para o reconhecimento de regiões de difícil acesso.

A.1 Desenvolvimento do Método de Mapeamento 3D de Ambientes

Para realizar o mapeamento 3D de ambientes, foi utilizada a mesma plataforma robótica do mapeamento de terrenos. Entretanto, o sensor laser 2D é direcinado para cima (inclinação de 90° em relação ao solo), de forma a capturar dados dos objetos presentes ao redor do robô.

O processo de mapeamento consiste basicamente transformar o sistema coordenadas dos pontos coletados pelo sensor laser no sistema de coordenadas do odômetro do robô. Assim, dado o ponto P_l coletado pelo sensor, deve-se determinar as suas coordenadas l_x, l_y e l_z em relação a posição do robô.

Assim, primeiramente foi decomposto os componentes do laser obtendo-se os valores l_z e l_w (Figura A.1(a)):



(a) (b)

Figura A.1: (a) Elementos do feixe de laser apontado para cima. (b) Elementos do feixe do laser e da posição do robô.

$$l_z = r \cdot \sin(\alpha) \tag{A.1}$$

$$l_w = r \cdot \cos(\alpha) \tag{A.2}$$

Em seguida, foi analisado o robô no plano xy para se determinar as coordenadas l_x e l_y do ponto lido (Figura A.1(b)). Foram obtidas as seguintes expressões para l_x e l_z:

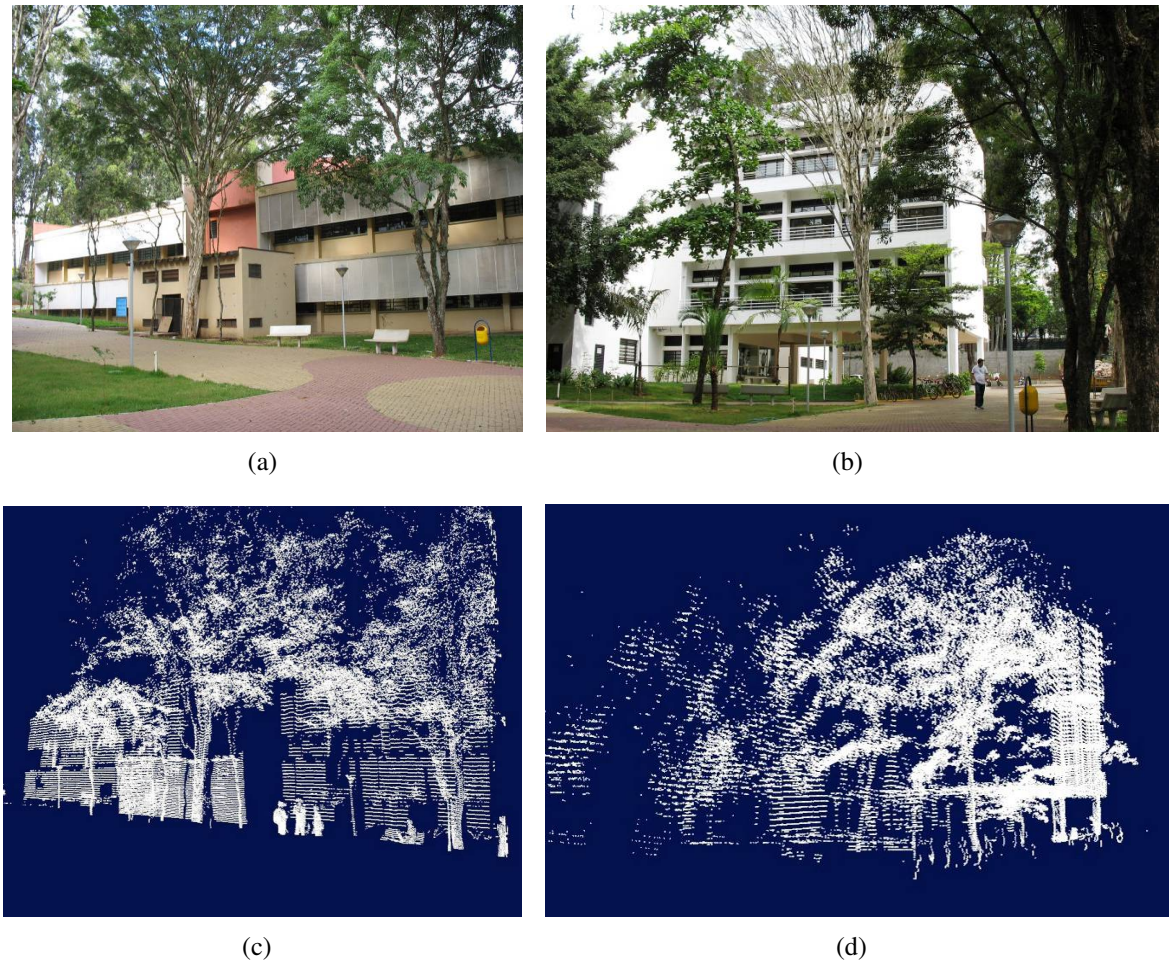
$$l_x = p_x + l_w \cdot \sin(\theta) \tag{A.3}$$

$$l_y = p_y - l_w \cdot \cos(\theta) \tag{A.4}$$

Dessa forma, o algoritmo de mapeamento cosiste em obter sucessivamente os valores de l_x, l_y e l_z para cada ponto lido pelo sensor laser, formando um mapa tridimensional do cenário.

A.2 Resultados

Para avaliar o algoritmo de mapeamento de ambientes foram realizados experimentos com o robô tanto em locais com elementos estrurados como prédios e desestruturados como árvores. Na Figura A.2 são apresentados os resultados obtidos. Nota-se que o sensor pôde capturar deta-



(a) (b)

(c) (d)

Figura A.2: Mapeamento 3D de ambientes realizado com o robô.

lhes de todos os elementos presentes no ambiente, comparando-se com as imagens do ambiente real.

APÊNDICE

B

Exemplo de Código em C Utilizando a Biblioteca Player/Stage

```
1  //Exemplo - Manipulação do robô
2
3  #include <stdio.h>
4
5  #include <libplayerc/playerc.h>
6
7  int main(int argc, const char **argv)
8  {
9      int i;
10     playerc_client_t *client;
11     playerc_position2d_t *position2d;
12
13     // Criação de um objeto cliente e conecta-o ao servidor.
14     // Caso o Stage for utilizado, a própria máquina local será
15     // o servidor (localhost).
16     client = playerc_client_create(NULL, "localhost", 6665);
17     if (playerc_client_connect(client) != 0)
```

```
18      {
19          fprintf(stderr, "error: %s\n", playerc_error_str());
20          return -1;
21      }
22
23      // Cria um proxy position2d (odômetro) e associa-o no modo
24      // leitura/escrita
25      position2d = playerc_position2d_create(client, 0);
26      if (playerc_position2d_subscribe(position2d,
27          PLAYERC_OPEN_MODE) != 0)
28      {
29          fprintf(stderr, "error: %s\n", playerc_error_str());
30          return -1;
31      }
32
33      // Ativação dos motores do robô
34      playerc_position2d_enable(position2d, 1);
35
36      // Faz com que o robô se desloque à 0.4 m/s e
37      // rotacionando 0.1 rad/s
38      playerc_position2d_set_cmd_vel(position2d, 0.4, 0, 0.1, 1);
39
40      for (i = 0; i < 200; i++)
41      {
42          // Leitura dos dados do servidor
43          playerc_client_read(client);
44          //Exibe a posição atual do robô (x,y,azimute)
45          printf("position : %f %f %f\n",
46              position2d->px, position2d->py, position2d->pa);
47      }
48
49      // Desconecta o odômetro e o cliente do servidor
50      playerc_position2d_unsubscribe(position2d);
51      playerc_position2d_destroy(position2d);
52      playerc_client_disconnect(client);
53      playerc_client_destroy(client);
```

```
54
55      return 0;
56 }
```

APÊNDICE

C

Exemplo de Código VRML para representação de pontos

```
1 # VRML V2.0 utf8
2 # Representação das coordenadas:
3 #      (1.5, 1.2, 5.5), (10.2, −3.12, 0.01) e (−2, −9, 1000)
4
5 Shape{
6       geometry PointSet{
7             coord Coordinate{
8                   point [
9                         1.5 1.2 5.5
10                        10.2 −3.12 0.01
11                        −2 −9 1000
12                  ]
13            }
14      }
15 }
```

APÊNDICE

D

Publicações Decorrentes Deste Trabalho

D.1 Artigos Publicados

1. HATA, A. Y.; WOLF, D. F. Mapeamento de ambientes externos utilizando robôs móveis. *Workshop Robocontrol : Applied Robotics and Collaborative System Engineering*, 2008.

2. HATA, A. Y.; WOLF, D. F. Mapeamento de terrenos utilizando robôs móveis e sensor laser. In: *Latin American Informatics Conference (CLEI)*, Pelotas - RS: Universidade Federal de Pelotas (UFPel), 2009, p. 49.

3. HATA, A. Y.; WOLF, D. F. Outdoor mapping using mobile robots and laser range finders. *IEEE Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, p. 209-214, 2009.

4. HATA, A. Y.; WOLF, D. F. Terrain mapping and classification using support vector machines. *IEEE Latin American Robotic Symposium, LARS*, 2009.

5. HATA, A. Y.; WOLF, D. F.; PESSIN, G.; OSÓRIO, F. S. Terrain mapping and classification using neural networks. In: *Proceedings of the 2009 ACM International Conference on Hybrid Information Technology, ICHIT 2009*, Daejeon, Korea, p. 438-442.

6. HATA, A. Y.; WOLF, D. F.; PESSIN, G.; OSÓRIO, F. S. Terrain mapping and classification in outdoor environments using neural networks. *International Journal of u- and e-Service, Science and Technology (IJUNEST)*, p. 51-61, 2009.

D.2 Artigos Submetidos

1. HATA, A. Y.; WOLF, D. F. Comparing support vector machines and artificial neural networks to terrain classification task. In: *International Conference on Intelligent Autonomous Vehicles (IAV)*, 2010.

2. HATA, A. Y.; WOLF, D. F. Mapeamento e Classificação de Terrenos Utilizando Aprendizado Supervisionado. In: *XVII Congresso Brasileiro de Automática (CBA)*, 2010.