Capítulo 1

Robótica Inteligente: Da Simulação às Aplicações no Mundo Real

Denis Fernando Wolf, Eduardo do Valle Simões,
Fernando S. Osório, Onofre Trindade Junior

Abstract

This tutorial aims to present and organize the main concepts related to the design of intelligent and autonomous mobile robotic systems, as well as, to describe some important techniques from the state-of-the-art in this field. The design of some interesting robotic applications is presented, describing the main challenges of mobile robots development focused on real world applications. The use of modeling and virtual simulation tools will be also discussed in the tutorial, illustrating different methodologies and projects of intelligent robotic systems.

Resumo

Este tutorial foi elaborado com a intenção de abordar e organizar os principais conceitos envolvidos no projeto de sistemas robóticos autônomos e inteligentes, bem como apresentar a aplicação das principais técnicas do estado-da-arte no projeto de soluções robóticas para os principais desafios do mundo real. Serão abordadas as principais ferramentas de modelagem e simulação de robôs e serão apresentados vários exemplos de projetos desenvolvidos com a participação dos autores para ilustrar diferentes metodologias de implementação de sistemas robóticos práticos.

1.1. Introdução

Este tutorial visa apresentar conceitos, técnicas e aplicações de robôs móveis autônomos inteligentes. Serão englobados desde o projeto do sistema robótico, utilizando-se de ferramentas de modelagem e simulação de robôs, até a implementação e o uso destes sistemas em aplicações no mundo real. Neste tutorial serão abordados aspectos envolvendo: i) a apresentação de fundamentos e conceitos de robótica móvel e de controle de robôs; ii) o desenvolvimento de modelos realísticos de simulação de robôs (simulação de sensores, atuadores, comportamento físico, incluindo ruído e imprecisão nos dados); iii) a apresentação de ferramentas de simulação de robôs móveis e sua aplicação prática; iv) o projetos e a validação de sistemas robóticos inteligentes, incluindo técnicas de controle, localização, mapeamento e navegação robótica; v) a apresentação de exemplos de aplicações de robótica autônoma e inteligente, incluindo sistemas robóticos indoor (aplicações na área de segurança e auxílio a deficientes), veículos outdoor (carros inteligentes), esquadrões robóticos inteligentes e robôs articulados (humanóides). vi) apresentação do novos desafios das pesquisas atuais na área de robótica inteligente e as tendências futuras nesta área.

1.2. Fundamentos de Robótica Móvel

A evolução dos robôs, e em especial dos robôs móveis, tem recebido nos últimos anos um amplo destaque junto à mídia e à sociedade de um modo geral. Onde no passado se falava muito em robôs industriais e braços mecânicos robóticos, atualmente as atenções se voltaram para robôs móveis, capazes de se deslocar no ambiente em que se encontram, e principalmente vem se focando nos ***Robôs Móveis Autônomos*** – RMAs e Veículos Autônomos Inteligentes [Jung 2005]. Os RMAs possuem, como características fundamentais, as capacidades de locomoção e de operação de modo semi ou completamente autônomo. Também deve ser considerado que maiores níveis de autonomia serão alcançados somente à medida que o robô passe a integrar outros aspectos considerados da maior importância, como: *capacidade de percepção* (sensores que conseguem "ler" o ambiente onde ele atua), *capacidade de agir* (atuadores e motores capazes de produzir ações, tais como o deslocamento do robô no ambiente), *robustez e inteligência* (capacidade de lidar com as mais diversas situações, de modo a resolver e executar tarefas por mais complexas que sejam). O uso de técnicas de planejamento e controle robusto para a navegação e operação autônoma dos robôs é conhecido pelo termo "Controle Robótico Inteligente", no qual este controle inteligente permite dotar os RMAs da capacidade de executar as mais diversas e complexas tarefas.

Pode-se citar aqui alguns exemplos famosos de Robôs Móveis (ver Fig. 1.1), resultantes da pesquisa e desenvolvimento que vem ocorrendo nesta área: os robôs de exploração espacial como o *Mars Pathfinder's Sojourner*, *Spirit* e *Opportunity Rovers* [Bajracharya 2008]; robôs domésticos usados para limpar a casa como o *Roomba* e *Scooba* [iRobot 2009], e para cortar grama como o *AutoMower* [Huskvarna 2009, Sahin 2007]; os robôs com pernas capazes de caminhar como o cachorro *Aibo* [Sony 2009], o humanóide *Asimo* [Honda 2009] e o *BigDog* [Boston Dymanics 2009]; veículos terrestres não tripulados como o Stanley de Stanford [Thrun 2006, Gibbs 2006, Jung 2005], que competiu e venceu o Darpa Challenge em 2005; e veículos aéreos não tripulados (UAVs) como os VANTs brasileiros do Projeto Arara [Neris 2001] e

AGplane [AGX 2009]. Estes exemplos demonstram claramente os progressos e resultados da pesquisa e desenvolvimento em robótica móvel desta última década.

Figura 1.1. Exemplos de Robôs Móveis: a) Roomba [iRobot 2009], b) Automower [Huskvarna 2009], c) Aibo [Honda 2009] e d) Asimo [Sony 2009]

Tabela 1.1. Sensores para Robôs Móveis

Sensor	Principal Função	Exemplos
De Posição e Orientação	Determinar a posição absoluta	GPS (Sistema de Posicionamento Global)
	ou direção de orientação do robô	Bússola [Compass]
		Inclinômetro
		Triangulação usando marcas (Beacons)
De Obstáculos	Determinar a distância até um objeto	Sensor Infra-Vermelho (IR - Infrared)
	ou obstáculo	Ultrassom (Sonar)
		Radar
		Sensor Laser (Laser rangefinder)
		Sistemas de Visão Estéreo (Stereo Vision)
De Contato	Determinar o contato com um objeto	Sensores de Contato (Bumbers, Switches)
	ou posição de contato com marcação	Antenas e "bigodes" (Animal whiskers)
		Marcações (barreiras óticas e magnéticas)
De Deslocamento	Medir o deslocamento do robô	Inercial (Giroscópio, Acelerômetros)
e Velocidade	Medidas relativas da posição e	Odômetro (Encoders: Optical, Brush)
	orientação do robô	Potenciômetros (Angular)
		Sensores baseados em Visão
Para Comunicação	Envio e recepção de dados e sinais	Sistemas de Visão e Sensores Óticos
	externos (troca de informação)	Sistemas de Comunicação (RF)
Outros tipos	Sensores magnéticos, indutivos, capacitivos, reflexivos	
	Sensores de temperatura, carga (bateria), pressão e força, etc.	
	Detectores: detector de movimento, de marcações, de gás/odores	

Os Robôs Móveis Autônomos, como se pode constatar pelos exemplos acima, possuem diferentes configurações de dispositivos de Hardware (HW) embarcados, de acordo com a função e as tarefas para as quais são projetados. Os principais dispositivos de HW de um robô são seus sensores e atuadores. A Tabela 1.1 apresenta uma lista dos sensores mais usados em robótica [Dudek 2000]. A Tabela 1.2 apresenta uma descrição dos principais sistemas atuadores adotados em robôs móveis [Dudek 2000, Bekey 2005]. Uma descrição mais detalhada dos diferentes tipos de mecanismos de locomoção em robôs móveis pode ser encontrada em [Siegwart 2004].

As Tabelas 1.1 e 1.2 demonstram o quão complexo pode ser o projeto de um sistema robótico, que envolve a especificação e seleção de diferentes componentes, sensores e atuadores (cada um com suas especificidades), e a combinação destes em um sistema autônomo. Este sistema deve ser projetado de modo a ser dotado de dispositivos capazes de prover os dados necessários (obtidos através dos seus sensores), para que o sistema de controle robótico inteligente possa planejar e realizar o acionamento dos seus dispositivos de modo a executar a ação desejada.

Tabela 1.2. Atuadores para Robôs Móveis

Atuador	Principal Tipo/Função	Exemplos
Base Fixa	Braço robótico com base fixa	Robôs industriais PUMA
Base Móvel: Rodas	2 Rodas independentes (diferencial)	Robôs Khepera e Pioneer P3-DX
	3 Rodas (triciclo, omni-directionais)	Robô BrainStem PPRK
	4 Rodas (veículos robóticos - ackermann)	Stanley - Stanford (Darpa Challenge)
Base Móvel: Esteira	Esteira (Slip/Skid locomotion - tracks)	Tanques e veículos militares
Base Móvel: Juntas e	Bípedes	Robôs Humanóides
Articulações	4 Patas (quadpods)	Robôs Sony Aibo, BigDog
	6 Patas (hexapods)	Robôs Inseto (Lynxmotion Hexapods)
Base Móvel: Propulsão	Veículos aéreos com hélices	Aviões, Helicópteros e Dirigíveis
Hélices ou Turbinas	Veículos aquáticos com hélices	Barcos autônomos
	Veículos sub-aquáticos	Submarinos autônomos
Outros tipos	Braços manipuladores com base móvel	Garras (Grippers) embarcadas
	Garras com ou sem feed-back sensorial	Mão robótica
	Mecanismos de disparo	Disparo do chute (futebol de robôs)

Um dos grandes desafios da robótica é justamente como integrar as informações vindas de todos estes sensores, de modo a gerar comandos e controlar os diferentes dispositivos de atuação do robô, garantindo que a tarefa seja executada de modo correto e sem colocar em risco tanto o robô quanto aqueles que o cercam. O robô deve preservar a sua integridade bem como dos elementos presentes no ambientes (e.g. seres humanos, objetos como utensílios e mobiliário, outros robôs). Um robô só deve agir de modo ativo sobre um determinado objeto-alvo, se realmente for programada (prevista) a sua ação sobre este objeto. Esta questão sempre foi alvo de muitas discussões (em propostas como as 3 Leis da Robótica de Asimov [Asimov 1968]), pois um robô móvel pode causar danos tanto a pessoas como a objetos que o cercam, de modo deliberado ou não, constituindo-se assim de um ***Sistema Embarcado Crítico***, onde seu desenvolvimento deve imperativamente envolver um cuidadoso projeto tanto de Hardware (HW) como de Software (SW).

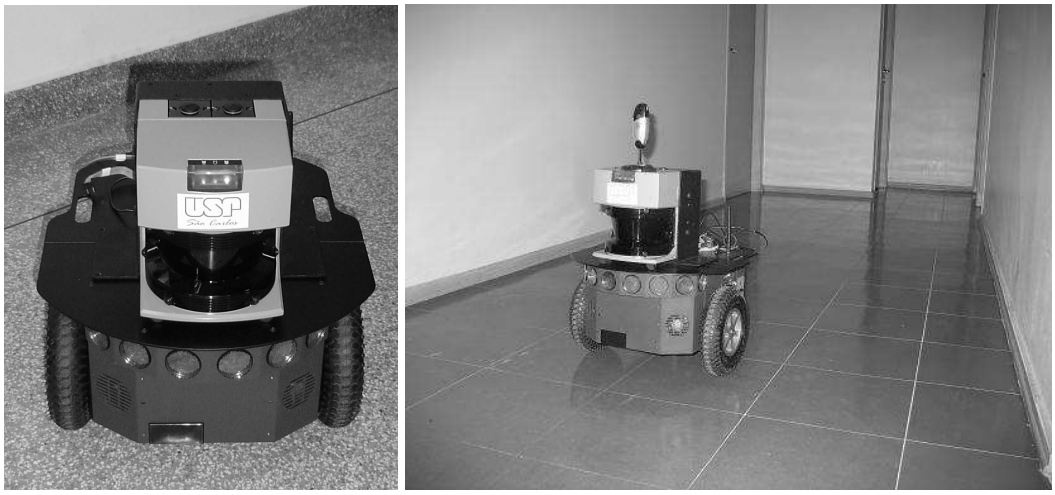


Figura 1.2. Robô Pioneer 3DX com sensores de sonar e laser rangefinder

É importante destacar que o projeto do sistema de controle de um robô como o da Figura 1.2 deve levar em conta uma série de quesitos e componentes, como por exemplo:

- *Tipo de tarefa do robô:* patrulhamento e vigilância de uma instalação (indoor);
- *Tipo e precisão dos sensores embarcados:* Laser Sick LMS200 (Fig 1.3), com capacidade de detectar obstáculos em um *range* máximo de 25m a 50m de distância (fecho laser estreito e direcional) com precisão de até 10mm (erro estatístico de 5mm), ângulo de abertura de 100 a 180 graus e tempo de resposta de entre 10 e 50ms; Sonar (*ultrasonic transducer - range-finding sonar*) composto por um arranjo de 8 sensores dispostos de modo a ter uma cobertura frontal de 180 graus, com alcance variando de 15cm até aproximadamente 7m (detecta a reflexão do sinal do sonar propagado em forma de cone); câmera de vídeo com resolução de 640x480 pixels e varredura de até 20fps; *encoders* com precisão de 500-ticks usado para medidas de deslocamento (odometria);
- *Tipo e precisão dos atuadores:* Acionamento diferencial (2 motores independentes), robô de características holonômicas (sem restrições de graus de liberdade, podendo girar em qualquer direção), e podendo atingir velocidade de até 1.6 m/seg;

Os sensores fornecem apenas uma "visão de mundo", parcial, incompleta e sujeita a erros, sendo papel do sistema de controle adquirir e tratar estas informações de forma inteligente. Além disto, os comandos de atuação também não são precisos, sendo que muitas vezes uma instrução de avançar em linha reta ou de girar em certa direção também pode não ser executada de forma correta e precisa, pois está sujeita a erros de posicionamento do robô, de acionamento dos motores, bem como está sujeita também a forças e componentes externos (e.g. fricção, gravidade, aceleração, desaceleração, inércia, colisão com obstáculos e derrapagem das rodas).

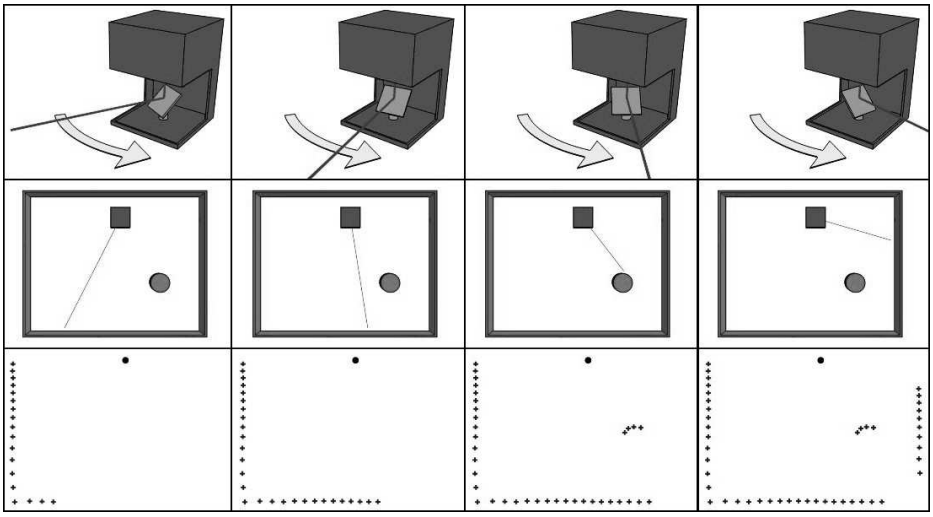


Figura 1.3. LIDAR Sensor (Light Detection and Ranging), Laser Sick [Lidar 2009]

O projeto do sistema de controle de um robô como o apresentado acima (Fig. 1.2) deve usualmente, através da adoção de uma *arquitetura específica de controle*, ser capaz de realizar algumas (ou todas) das seguintes tarefas de:

- *Fusão de Sensores:* adquirir e integrar os diversos dados recebidos a partir dos sensores, compensando as limitações de alcance e precisão de cada sensor, através da fusão dos dados;
- *Desviar de obstáculos:* detectar obstáculos e poder assim evitar a colisão com os mesmos, preservando a integridade do robô e dos elementos externos. O desvio de obstáculos deve evitar tanto obstáculos estáticos (e.g. mobiliário), como obstáculos dinâmicos (e.g. elementos móveis como pessoas e outros robôs);
- *Auto-Localização:* determinar a localização do robô no ambiente (posição e orientação), com ou sem o uso de um mapa do ambiente, de modo a poder planejar e executar o deslocamento seguindo uma determinada trajetória;
- *Mapeamento do ambiente:* exploração e construção de um mapa do ambiente, usando técnicas como SLAM (*Simultaneous Localization And Mapping*) [Thrun 2005];
- *Planejamento de trajetórias:* através do uso de um mapa é possível planejar ações de alto-nível, como definir previamente uma trajetória a ser executada pelo robô, especificando as ações elementares a serem realizadas de modo a se deslocar de uma posição-origem até uma posição-alvo;
- *Planejamento de ações:* uma vez definida a tarefa a ser realizada pelo robô móvel, é possível estabelecer um plano de ações, que pode ser composto da execução de sub-tarefas mais elementares, como por exemplo: explorar o ambiente (*wander*); seguir uma parede ou corredor até o seu final, se deslocar em uma direção-alvo desviando de obstáculos, recolher objetos, seguir um comboio, patrulhar uma área, etc;
- *Navegação robótica:* capacidade de executar as ações planejadas de modo robusto, como por exemplo, executar o deslocamento de uma posição-origem até

uma posição-alvo, realizando os devidos ajustes durante o deslocamento para evitar a colisão com obstáculos. A navegação robótica é a execução do plano de trajetórias e ações previamente definido;

- *Interação e Comunicação:* capacidade de interagir e se comunicar com outros agentes presentes no ambiente. Por exemplo, um robô pode ter a capacidade de agir de modo colaborativo, interagindo e trocando informações com outros robôs de modo a executar uma tarefa de forma cooperativa e compor assim um sistema multirrobótico. Um exemplo clássico deste tipo de sistema é o futebol de robôs, no qual múltiplos robôs devem coordenar suas ações a fim de atingir objetivos comuns.

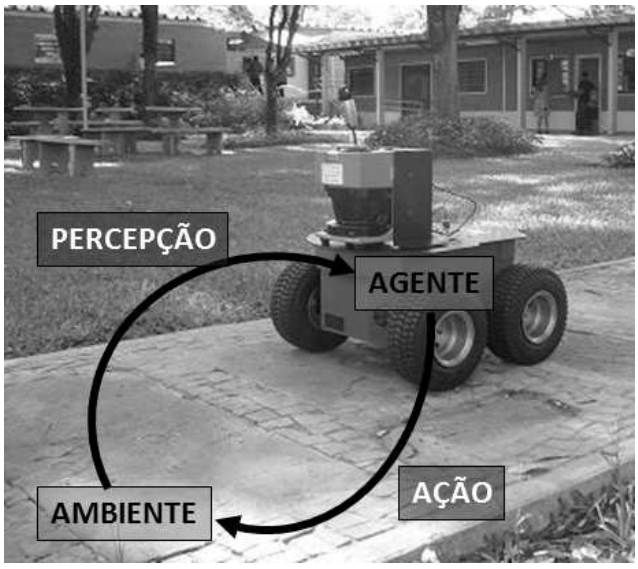


Figura 1.4. Robô Móvel Autônomo: Percepção-Decisão-Ação

As arquiteturas computacionais de controle de robôs móveis autônomos [Medeiros 1998] permitem que tarefas como as listadas acima sejam planejadas e executadas, através do gerenciamento dos diferentes dispositivos embarcados no robô. Os principais aspetos destas arquiteturas são relativos a: Percepção (incluindo a comunicação), Raciocínio/Decisão e Ação. A Figura 1.4 apresenta este esquema representando este modelo de percepção-decisão-ação, no qual o robô "sente" o ambiente, decide e age sobre ele, sendo que suas ações irão afetar o ambiente e, por conseqüência, suas percepções no instante de tempo seguinte. As arquiteturas computacionais de controle de veículos autônomos são bastante diversas e mesmo na literatura são encontrados diferentes enfoques e abordagens [Dudek 2000, Medeiros 1998, Bekey 2005], entretanto, podem ser citadas algumas das arquiteturas de controle que se tornaram as mais conhecidas e reconhecidas pelas suas características e potencialidades: controle reativo, controle deliberativo, controle hierárquico e controle híbrido. Serão detalhadas a seguir estas principais arquiteturas de controle [Jung 2005].

1.2.1. Controle Reativo

O controle reativo consiste de um sistema de reação sensorial-motora. Este tipo de controle normalmente é o mais simples de ser implementado (reativo puro), não necessitando de muitos recursos computacionais para sua implementação. No controle

reativo existe um laço sensório-motor de: *(i)* leitura dos sensores; *(ii)* processamento imediato destas informações; *(iii)* geração de um comando de resposta para os atuadores. Usualmente um esquema de controle reativo considera apenas as leituras sensoriais realizadas no presente para fins de tomada de decisão e geração de comandos de ação. Um sistema reativo é bastante útil para implementar comportamentos elementares como: desviar de obstáculos (*avoid collision behaviour*: reage a presença de um obstáculo), e seguir um objeto (*wall-following behaviour*: acompanhar um elemento guia).

1.2.2. Controle Deliberativo

O controle deliberativo (ou cognitivo) consiste na aplicação de um mecanismo de planejamento das ações, podendo ser estabelecido um plano prévio de execução de uma seqüência de ações, baseado nos conhecimentos que o sistema possui sobre o problema a ser resolvido (e.g. mapa do ambiente, rotas disponíveis). No controle deliberativo é assumida a existência de um processo de alto nível de raciocínio e tomada de decisões, usualmente mais complexo de ser implementado do que o controle reativo. Este processo permite que sejam planejadas ações de modo a tratar e executar tarefas que exigem um nível de controle mais sofisticado, como por exemplo, definir (traçar uma rota) e executar a tarefa de se deslocar de um ponto a outro do ambiente, considerando um mapa do mesmo.

Entretanto, o controle "deliberativo puro" possui limitações quando colocado frente a eventos imprevistos, como por exemplo, um obstáculo que se moveu obstruindo a sua rota. Neste caso, o controle deliberativo puro terá dificuldades de reagir a uma nova configuração do ambiente, que não havia sido prevista em seu planejamento inicial. Pode-se dizer que o ideal em um sistema de controle seria que este tivesse a capacidade de reação de um sistema reativo, com a capacidade de planejamento e execução de tarefas complexas de um sistema deliberativo. Desta combinação entre reativo e deliberativo surgem os sistemas hierárquicos/híbridos, abordados no próximo item.

1.2.3. Controle Hierárquico e Híbrido

O controle hierárquico/híbrido consiste da combinação de múltiplos módulos de controle reativo e/ou deliberativo em camadas dispostas de modo que estes possam operar de modo hierárquico ou em paralelo. A combinação dos diferentes módulos de controle leva a adoção de um esquema de prioridades em relação às múltiplas camadas do sistema, onde é comum encontrar estes sistemas de controle classificados como: sistemas hierárquicos com decomposição vertical e decomposição horizontal [Dudek 2000, Heinen 2002a]. Os sistemas hierárquicos/híbridos apresentam a vantagem de poderem combinar os comportamentos obtidos de seus diferentes módulos a fim de obter um comportamento mais robusto e uma execução de tarefas mais complexas. Por exemplo, é possível implementar um módulo de controle deliberativo que realiza o planejamento e execução de uma trajetória com o uso de um mapa, ao mesmo tempo em que um módulo reativo de maior prioridade pode intervir para realizar o desvio de obstáculos, e finalmente um módulo de auto-localização permite que sejam realizados ajustes no plano de trajetória através de um constante monitoramento da posição atual do robô. A Arquitetura COHBRA (Controle Híbrido de Robôs Autônomos) [Heinen 2002a], implementa este tipo de arquitetura, sendo um exemplo de arquitetura híbrida.

1.2.4. Controle Robótico Robusto e Inteligente

O sistema computacional de controle de um robô móvel autônomo deve ser projetado de modo a ser robusto e permitir a correta execução das tarefas que lhe são atribuídas. Em algumas situações, um sistema de controle reativo pode ser adotado, demonstrando ser robusto o suficiente para a execução de uma determinada tarefa.

Por exemplo, alguns sistemas simples de controle reativo permitem a um robô móvel executar tarefas como: seguir uma marcação no chão (e.g. AGVs – *Automated Guided Vehicles*), seguir um comboio, se dirigir em direção a um foco de luz ou calor, realizar tarefas como a limpeza de um ambiente (alternando entre comportamentos de vagar livremente e se dirigir para uma base de recarga), e até mesmo apresentar comportamentos coletivos (Inteligência de Enxames - *SwarmBots*) ou implementar o controle de sistemas multirrobóticos baseados em ACO (*Ant Colony Algorithms*). Também é possível implementar com sucesso sistemas de controle deliberativo puro, que executam tarefas em ambientes bem estruturados, onde não há ocorrência de imprevistos ou erros na execução das tarefas.

Porém, o desenvolvimento de uma arquitetura de controle robótico robusto com diferentes módulos, capazes de tratar questões como a fusão de sensores, a auto-localização, o mapeamento do ambiente, o desvio de obstáculos estáticos ou dinâmicos, o planejamento de trajetórias (com ou sem uso de mapas), a navegação robótica, a execução de tarefas de alto nível e a comunicação e cooperação em sistemas multirrobóticos é uma tarefa bastante complexa e que demanda um grande esforço em termos do projeto e desenvolvimento de um sistema computacional de controle robótico com estas funcionalidades.

A fim de viabilizar o desenvolvimento de sistemas mais complexos, que buscam uma maior autonomia, robustez e adaptabilidade a diferentes situações e ambientes, usualmente se faz necessário o uso de técnicas mais avançadas de Robótica e de Inteligência Artificial. No desenvolvimento destes Sistemas de Controle Robótico Robusto e Inteligente, em um grande número de situações, se faz necessário o ***uso de sistemas simulados***, para que se possa projetar e ajustar o sistema até o ponto que este possa ser implementado em robôs reais e posto em funcionamento na prática. Na seção seguinte será discutida a importância do uso da simulação no projeto e aperfeiçoamento dos sistemas inteligentes de controle robótico.

1.3. O Papel da Simulação Virtual

Nesta seção será discutida a aplicação da simulação virtual como ferramenta de auxílio ao projeto dos robôs móveis, partindo do projeto de sua configuração de hardware até o projeto do software através da implementação do sistema inteligente de controle dos robôs.

O uso de um ambiente virtual de simulação de robôs tem se demonstrado uma ferramenta poderosa, apresentando uma série de vantagens: i) *economia de recursos financeiros*, pois diversos testes podem ser realizados antes de ser implementado fisicamente o robô; ii) *economia de tempo*, pois podemos realizar um maior número de experimentos através de simulação, nos quais a realização de um experimento requer um menor tempo para configurar o experimento (não há necessidade de recarregar

baterias e posicionar equipamentos e objetos), além de se poder "acelerar o tempo" do relógio virtual na execução das simulações; iii) *evitar danos ao robô*, pois através das simulações pode-se verificar previamente as situações que podem provocar danos ao robô, devido por exemplo a fortes colisões, acionamento indevido dos motores, ou exposição do robô à ambientes perigosos para testes; iv) *evitar acidentes e aumentar a segurança*, pois através da simulação pode-se realizar diversos testes visando garantir uma maior segurança e robustez do sistema robótico, evitando assim a incidência de acidentes com pessoas e com elementos presentes no ambiente de atuação do robô, permitindo inclusive uma melhor análise de como adicionar novos dispositivos de salva-guarda em HW e SW que permitam aumentar a segurança e confiabilidade dos robôs; v) *aperfeiçoamento do HW e SW*, uma vez que se pode, através de simulações, testar diferentes configurações de HW, bem como testar e avaliar novas implementações e ajustes nos parâmetros do SW de controle, permitindo assim uma melhoria do sistema como um todo e a otimização do uso dos recursos disponíveis a fim de obter uma maior eficiência do sistema robótico.

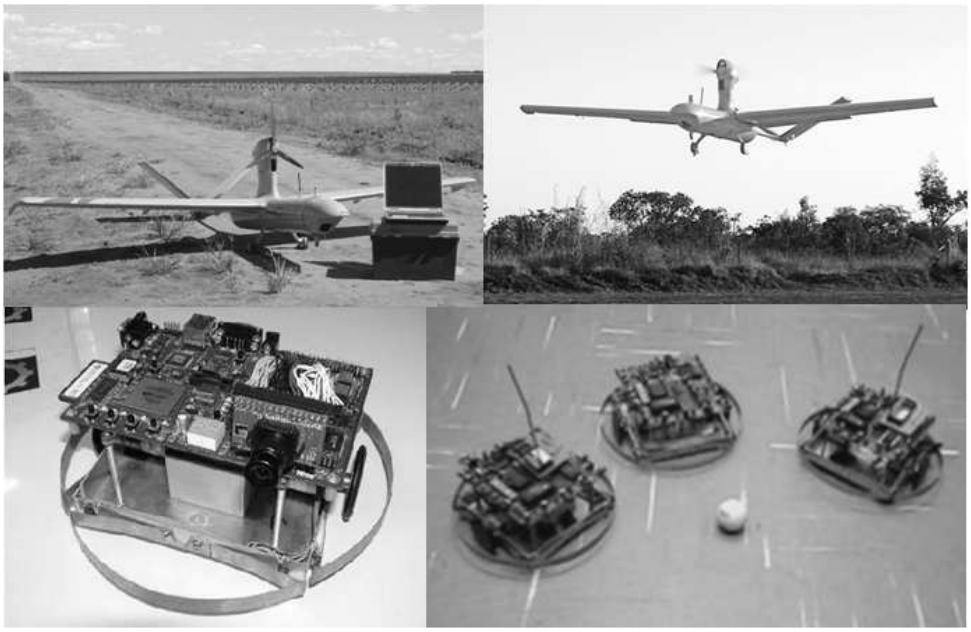


Figura 1.5. Robô Autônomo *x* Ambiente Físico: VANT AGPlane2 com GPS embarcado e Robôs Móveis construídos no ICMC-USP com sensores IR, rádio-modem de RF, Câmera e pára-choques com sensores de contato

O projeto físico de um robô é uma etapa bastante trabalhosa, que usualmente tem início na definição do tipo de ambiente onde este robô irá atuar e na especificação do conjunto de tarefas que este robô deverá executar. A partir destas informações, tem início uma etapa de configuração dos sensores, atuadores e HW de controle que serão embarcados no sistema robótico. O ambiente de trabalho do robô irá condicionar a sua forma de atuação (terrestre, aérea ou aquática), bem como as dimensões que este robô deverá ter, e por fim também condicionará a sensibilidade e precisão dos sensores embarcados no robô (ver Fig.1.5). Por exemplo, um robô de inspeção de dutos deve ter uma configuração adequada de atuadores e sensores, bem como de sua dimensão, de modo a poder melhor explorar os dutos e executar sua tarefa de identificação de rachaduras e danos nestes dutos.

Os atuadores dependem diretamente do ambiente em que o robô irá atuar, no qual deverão ser feitas escolhas primeiramente em relação ao meio: terrestre (indoor, outdoor estruturado ou não), aquático (superfície ou submarino) ou aéreo (avião, helicóptero ou dirigível). No caso de um robô terrestre, também é importante definir melhor o tipo de ambiente para que se possa configurar o método de locomoção adotado: usando rodas, esteiras ou pernas. Para cada tipo de método de locomoção e de configuração dos atuadores do robô, tem-se um tipo de comportamento cinemático diferenciado, sendo este um dos elementos muito importantes para que se possa depois implementar um modelo adequado de simulação. É importante destacar também que tarefas de detecção e desvio de obstáculos, de auto-localização e navegação irão depender fortemente do ambiente em que o robô está inserido, condicionando inclusive a opção por diferentes tipos de sensores (e.g. uso de radar e GPS em dispositivos aéreos, de sensores laser em ambientes terrestres indoor e outdoor, de sonar e infravermelho em ambientes fechados).

Os sensores devem ser definidos levando em consideração, além do ambiente externo, questões como a capacidade sensorial embarcada necessária para a execução da tarefa (tipo e quantidade de sensores) e a relação de custo *x* benefício em função da escolha dos diferentes tipos de dispositivos disponíveis no mercado (avaliando a precisão, erro, qualidade da informação medida pelo sensor, faixa de atuação, consumo de energia, e inclusive o peso da carga adicional a ser carregada pelo robô). Por exemplo, um robô de segurança e vigilância para prédios deve ter sensores que permitam se localizar e se deslocar em um ambiente indoor, detectar e desviar de obstáculos estáticos, mas também deve possuir câmeras para a detecção de movimentos e de intrusos, onde possivelmente um sensor de calor (e.g. câmera infra-vermelho) e capacidade de comunicação seriam requisitos necessários.

Além dos sensores e atuadores o robô deve possuir um HW com poder de processamento suficiente para poder coletar e processar todas as informações dos sensores e realizar o envio de comandos aos atuadores. O projeto de Software, assim como o projeto físico, também envolve decisões sobre o tipo de módulos que devem compor a arquitetura do sistema de controle robótico, e sobre quais componentes e técnicas (algoritmos) devem ser implementados junto ao sistema de controle. Estas decisões de projeto do SW, assim como os ajustes dos parâmetros de controle dos diferentes algoritmos, terão um impacto direto sobre a capacidade do sistema robótico em realizar suas tarefas de modo autônomo e com robustez.

Como se pode então projetar um sistema tão complexo? Como projetar um sistema que envolve a seleção de dispositivos de HW, sua integração e posicionamento junto ao robô, a obtenção e processamento das informações provenientes dos sensores embarcados, a tomada de decisões quanto às ações e o controle dos atuadores de modo a executar as tarefas planejadas? A resposta é simples, o projeto de um RMA deve começar pela elaboração de um modelo inicial, passível de simulação, para que se possa testar e validar o funcionamento do robô, considerando seus diversos módulos de HW e SW.

1.3.1. Modelo de Simulação

Para se criar um modelo para simulação virtual é necessário criar um modelo simplificado da realidade, implementando este em um ambiente computacional [Osorio 2006]. O modelo simplificado deve desconsiderar o que não for necessário e/ou relevante para efeitos da simulação dos componentes reais do robô. Entretanto, este modelo deve integrar os elementos e características que são mais relevantes para o funcionamento do sistema robótico, o que usualmente na área de robótica significa modelar:

- *Sensores*: os diversos sensores apresentados na Tabela 1.1 devem ser passíveis de simulação, onde, por exemplo, no caso dos sensores de medida de distância, o modelo criado deve levar em consideração o alcance (menor/maior distância medida), a precisão, e questões relativas ao direcionamento no processo de detecção de obstáculos. Também deve ser modelado o erro/ruído típico do sensor em questão, sendo que sabidamente não existem "sensores perfeitos";
- *Atuadores*: os atuadores devem ser modelados, de forma a simularem as ações realizadas pelo robô. Os principais atuadores a serem modelados são os responsáveis pela locomoção do robô móvel, nos quais também é importante que o modelo implementado considere a precisão/erros associados às ações. Por exemplo, ao se comandar um giro do robô de N graus, muito provavelmente haverá um erro associado a este comando, resultando em uma nova orientação próxima a desejada, na qual este não deverá atingir exatamente o ângulo de orientação previsto.
- *Comportamento Físico do Robô*: usualmente não é necessária uma modelagem individual de cada atuador do robô, podendo ser integrado um modelo de comportamento físico deste, através de um modelo cinemático do mesmo. Deste modo será possível prever a trajetória que será executada pelo dispositivo móvel a partir do modelo físico/comportamental. Em alguns casos, como por exemplo, em robôs dotados de pernas, é necessária a implementação de um modelo mais completo, que deve levar em consideração questões referentes à dinâmica do robô: equilíbrio, fricção, gravidade, reação a colisões, etc.

Existem na literatura diversos modelos já previamente estudados e testados de sensores e atuadores [Dudek 2000, Siegwart 2004]. Diferentes dispositivos sensores como pára-choques com sensores de contato (*Bumbers*), sensores Infra-Vermelho (IR), Sonares, Sensores Laser (*Laser Rangefinder*), acelerômetros, bússolas (Electronic *Compass*), GPS, entre outros, possuem modelos de simulação bastante confiáveis que podem ser usados para fins de testes em ferramentas de simulação virtual de robôs. Além destes modelos de sensores, diferentes modelos cinemáticos também foram amplamente estudados, nos quais é possível simular robôs com diferentes tipos de controle de locomoção: robôs omni-direcionais com cinemática diferencial de duas rodas (ou com uma terceira roda livre), robôs com cinemática *Ackermann* (simulam veículos com tração e barra de direção), robôs com cinemática do tipo *Slip/Skid* (simulação de tração por esteiras), robôs aéreos, robôs aquáticos, entre outros modelos. Estes modelos são bastante fiéis ao comportamento real do robô, inclusive no que diz respeito ao erro de resposta aos comandos de atuação.

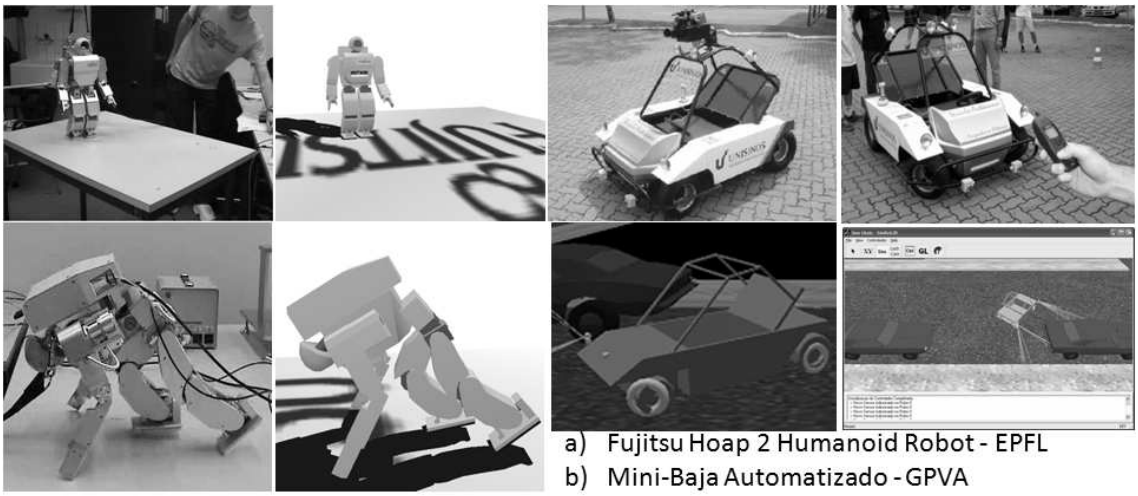


Figura 1.5. Modelos Reais e Modelos de Simulação Virtual de um a) Robô Humanóide do tipo Fujitsu-Hoap2 - EPFL[Cominoli 2005] e de um b) Veículo Autônomo tipo Mini-Baja do GPVA [Jung 2005]

1.3.2 Ferramentas de Simulação Virtual

A implementação de um simulador virtual de robôs móveis autônomos envolve algumas decisões quanto ao ambiente simulado: representação em 2D, 3D plano (*indoor*), 3D plano (*outdoor*), 3D com terrenos irregulares (*outdoor off-road*) ou 3D com movimentação espacial (aviões e submarinos). Usualmente deve-se dar preferência pelo uso de simuladores em ambientes virtuais 3D, pois estes reproduzem de forma mais adequada os ambientes em que estarão inseridos os robôs. Simuladores 2D permitem avaliar uma série de questões relativas ao projeto e controle robótico, mas no entanto possuem limitações quanto a representar o retorno sensorial quando o ambiente possui elementos de diferentes formas e tamanhos. Por exemplo, a implementação de um sonar simulado deve levar em consideração a propagação do som no espaço (propagação na forma de um cone), na qual a medida de distância é estimada pela reflexão do som que permite identificar a presença de um objeto localizado no espaço tridimensional dentro de seu raio de ação.

A tarefa de implementação de um simulador virtual para robôs autônomos também não pode ser considerada uma tarefa simples, por isso, na maioria dos casos é feito o uso de ferramentas de simulação virtual que ajudam no projeto destes simuladores. As ferramentas básicas usadas na criação de um simulador virtual são bibliotecas (APIs, dlls e Libs) e *engines* gráficas, capazes de fornecer uma visualização 2D ou 3D do ambiente e da simulação do robô, bem como outros componentes que permitem simular funções mais específicas. Estes componentes específicos incluem bibliotecas de: Simulação Física (cinemática e dinâmica dos corpos), Inteligência Artificial e Aprendizado de Máquina, Simulação de Terrenos 3D e de Elementos Naturais, Simulação Dinâmica de Eventos (industrias, incêndios, inundações), e Simulação Multiagentes (e.g. grupos, multidões, enxames – *swarms*, *flocking*). A Tabela 1.3 apresenta um resumo das principais ferramentas usadas na implementação de ambientes virtuais de simulação de robôs.

Tabela 1.3. Ferramentas usadas para implementar Simulações Virtuais

Ferramenta	Aplicação	Referência
SDL	Biblioteca gráfica usada para visualização 2D (C/C++)	http://www.libsdl.org/
OpenGL	Biblioteca gráfica usada para visualização 3D	OpenGL - http://www.opengl.org/
	OpenGL é usada por diversas outras ferramentas	Ogre3D - http://www.ogre3d.org/
	e engines gráficas, como por exemplo:	Java3D - http://www.java3d.org/
	OSG, Ogre3D, Java3D, GLScene (C++, Java, Delphi)	GLScene - http://glscene.sourceforge.net/
OSG	Open Scene Graph (C++)	http://www.openscenegraph.org/
	Ferramenta de visualização de ambientes virtuais 3D	
ODE	Open Dynamics Engine (C/C++)	http://www.ode.org/
	Simulação física da dinâmica de corpos rígidos	
	Cinemática e dinâmica de corpos rígidos articulados	
Demeter	Visualização de terrenos 3D usando OpenGL e C++	http://demeter.sourceforge.net/
Terrain Engine		
A.I.	Machine Learning Tools - Weka (Java)	http://www.cs.waikato.ac.nz/ml/weka/
	Redes Neurais Artificiais - SNNS (C e Java)	http://www.ra.cs.uni-tuebingen.de/SNNS/
	Algoritmos Genéticos - GALib (C++)	http://lancet.mit.edu/ga/
	A* (A Star) PathFinding (C++)	http://www.generation5.org/content/2002/ase.asp
	Flocking Simulation - OpenSteer (C++)	http://opensteer.sourceforge.net/
	Swarm Framework (C e Java)	http://www.swarm.org/index.php/Swarm_main_page
Simulação	Player/Stage/Gazebo (C/C++)	http://playerstage.sourceforge.net/
Robótica	Cyberbotics Webots	http://www.cyberbotics.com/products/webots/
	MRS - Microsoft Robotics Studio	http://msdn.microsoft.com/en-us/robotics/
	Mission Simulation Facility (MSF) - NASA	http://ti.arc.nasa.gov/projects/msf/
Simulação	Matlab - Simulink	http://www.mathworks.com/products/simulink/
Científica	Microsoft ESP - Visual simulation platform	http://www.microsoft.com/esp/
2D e 3D	FlexSim Simulation Software	http://www.flexsim.com/
	Quest3D - VR Simulation	http://www.quest3d.com/

1.3.2.1 Exemplos de modelagem e simulação de robôs e veículos autônomos

O simulador SEVA3D – Sistema de Estacionamento de Veículos Autônomos [Heinen 2006] foi desenvolvido de forma a ajudar no projeto, implementação e testes de um veículo capaz de estacionar de modo autônomo em uma vaga. A modelagem de um ambiente virtual de simulação 3D permitiu que fossem definidos e avaliados detalhes como o do posicionamento e direcionamento de cada um dos sensores (sonares), bem com foi feita a modelagem tridimensional dos demais objetos distribuídos no espaço de atuação do veículo (e.g. carros, cordão da calçada). É importante salientar que os sonares precisaram ser dispostos em posições chaves do carro, e no caso da detecção do meio-fio foi necessário um ajuste adequado de direcionamento do sonar, ficando voltado para o chão de modo a detectar o meio-fio da calçada (ver Fig. 1.6). Além dos sensores, também foi simulado o comportamento cinemático e dinâmico do veículo (modelo cinemático de Ackermann), permitindo assim que fosse desenvolvido um sistema de controle inteligente do veículo. O sistema de controle foi implementado com o uso de uma Rede Neural Artificial que recebe os dados dos sensores e gera os comandos para os atuadores do veículo (velocidade e giro da direção).

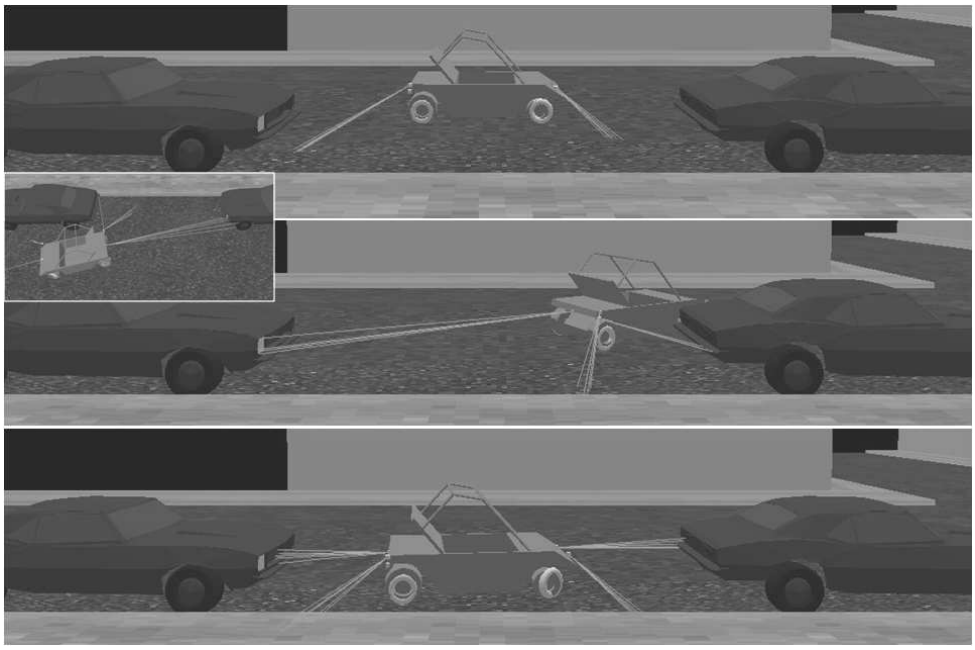


Figura 1.6. Simulador SEVA3D

Outro exemplo de simulação virtual de robôs autônomos é o sistema LEGGEN [Heinen 2006a] que foi desenvolvido para ajudar no projeto de robôs dotados de pernas/patas. O simulador LEGGEN foi implementado fazendo uso da biblioteca de simulação física de corpos rígidos articulados ODE. Este simulador permitiu o estudo e aperfeiçoamento da morfologia dos robôs articulados, assim como a validação e testes de diferentes sistemas de controle do caminhar dos mesmos. É importante destacar que a biblioteca ODE foi fundamental para permitir a realização dos experimentos, pois através de seu uso foi possível simular a gravidade, fricção, articulações, colisões com elementos do ambiente, equilíbrio em pé, e as conseqüentes quedas e tropeços do robô. Esta ferramenta tornou possível validar e avaliar os sistemas de controle inteligentes adotados, onde foi implementado um sistema evolutivo, baseado em Algoritmos Genéticos com o uso da biblioteca GALib, que aprendia a caminhar sozinho.

As duas ferramentas de simulação virtual de sistemas autônomos descritas acima foram implementadas a partir do uso de bibliotecas genéricas, como o OpenGL, ODE, GALib, e SNNS. É importante destacar que também existem diversas ferramentas de simulação que foram desenvolvidas especificamente para o projeto, desenvolvimento e experimentação através de simulações de sistemas robóticos, tais como o *Player/Stage*, *Cyberbotics Webots* e o *Microsoft Robotic Studio* (MRS). Estas ferramentas serão abordadas na próxima seção.

1.4. Simuladores de Robótica Móvel

Existem diversos simuladores de robôs móveis atualmente disponíveis na Internet. Alguns desses simuladores exigem uma licença para sua utilização, podendo ser inclusive comercializados juntamente com alguns modelos de robôs comerciais, enquanto outros podem ser utilizados livremente. Outra característica importante presente em alguns simuladores é a integração com programas que permitem controlar diretamente os robôs reais. Dessa forma, os programas de controle testados no

simulador podem ser facilmente portados para uso com robôs reais. Nessa seção são apresentados alguns dos simuladores mais utilizados e bibliotecas de controle de robôs móveis.

1.4.1 Player/Stage/Gazebo

O *Player* é um sistema para controle de robôs móveis amplamente utilizado por universidades e institutos de pesquisa [Collett 2005]. Seu desenvolvimento foi iniciado em 2000 por pesquisadores da *University of Southern Califórnia*, e atualmente conta com a colaboração de pesquisadores das mais diversas instituições. Por ser um sistema de código aberto e de livre distribuição (compatível com o Linux), o Player está em constante desenvolvimento para se adequar e disponibilizar implementações de um número cada vez maior de plataformas robóticas e sensores comerciais. A estrutura do Player é baseada no modelo cliente/servidor. O servidor faz a interface com o robô e com outros sensores, obtendo dados, enviando-os para o cliente e recebendo instruções do cliente para o controle do robô e dos sensores. O cliente é o programa que controla efetivamente o robô (aplicação – sistema de controle). O cliente é responsável por obter os dados do servidor, interpretá-los e enviar instruções para o servidor (robô) para a execução de determinada tarefa. A arquitetura cliente/servidor permite grande versatilidade no controle robôs e sensores, de forma que um cliente (programa de controle) pode controlar diversos servidores e diferentes clientes podem controlar diferentes sensores de um mesmo robô. O cliente Player foi projetado para ser compatível com diversas linguagens. Atualmente, existem bibliotecas disponíveis para clientes em C, C++, Java, Python, Tcl, entre outras linguagens. Além de prover funções para o controle de robôs e sensores, o Player também disponibiliza algoritmos para navegação com desvio de obstáculos, planejamento de trajetória, localização e mapeamento de ambientes. Atualmente, o Player suporta mais de 10 plataformas robóticas comerciais e diversos tipos de sensores como lasers, sonares, IMUs, câmeras de vídeo e GPS. O Player vem sendo desenvolvido juntamente com dois outros módulos de simulação: o *Stage* e o *Gazebo*.

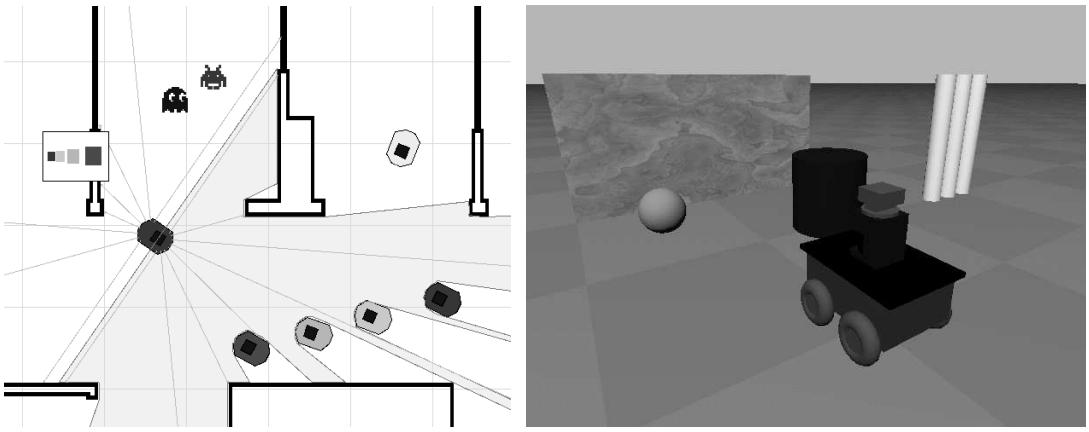
1.4.1.1 Stage

O *Stage* é um simulador de múltiplos robôs voltado para ambientes bidimensionais (sensores e visualização 2D), sendo compatível com o Player. Múltiplos robôs e sensores podem ser simulados simultaneamente, controlados por um ou mais clientes. Devido a sua alta eficiência, o Stage é capaz de simular dezenas de robôs operando simultaneamente em um único PC. Os robôs simulados pelo Stage são baseados no *Pioneer* (robô fabricado pela empresa *MobileRobots*) [Pioneer 2009], uma vez que essa plataforma é amplamente utilizada em laboratórios de pesquisa e ensino. O Stage simula o deslocamento dos robôs e também os sensores, como odômetros, lasers, sonares, câmeras e garras.

1.4.1.2 Gazebo

Além do Stage, o Player também é compatível com o simulador *Gazebo*. O Gazebo é um simulador de robôs móveis em 3D que permite uma simulação e visualização mais realista de ambientes complexos. Através do uso de bibliotecas gráficas (OGRE) e de modelagem física (ODE), o Gazebo permite uma simulação fiel do comportamento e da interação física de robôs e objetos do ambiente. Por ser computacionalmente mais

complexo que o Stage, o Gazebo requer mais recursos computacionais, o que limita o número de robôs que podem ser simulados simultaneamente. O Gazebo normalmente é utilizado em situações nas quais a simulação bidimensional do Stage não é fiel o suficiente para a realização dos experimentos virtuais. Por exemplo, a simulação em ambientes externos, onde o solo é irregular, pode resultar em uma inclinação dos robôs e de seus sensores durante a operação, ou também quando em situações nas quais a altura dos objetos e sensores deve ser levada em consideração para a execução de um experimento. Atualmente o Gazebo simula robôs do tipo Pioneer DX, Pioneer AT e Segway, mas é também possível modelar outros tipos de robôs. Quanto aos sensores, além daqueles simulados pelo Stage, o Gazebo simula também GPS, unidades de medida inercial e câmeras estéreo.



(a) Stage **(b) Gazebo**

Figura 1.7. Simulador Player/Stage (2D) e Gazebo (3D)

1.4.2 CARMEN

O *Carnegie Mellon Robot Navigation Toolkit* (CARMEN) é uma biblioteca modular e de código aberto para o controle de plataformas robóticas e sensores comerciais. Seu desenvolvimento teve inicio na *Carnegie Mellon University*, e hoje conta com a colaboração de pesquisadores de diversas instituições. Da mesma forma que o Player, o CARMEN disponibiliza funções de acesso ao hardware de robôs e sensores, bem como algoritmos normalmente utilizados em aplicações robóticas como navegação, mapeamento e localização. O CARMEN foi desenvolvido para plataformas Linux, utilizando-se a linguagem C. Atualmente o mesmo oferece também suporte a linguagem JAVA. A lista do hardware suportado pelo CARMEN inclui algumas plataformas robóticas comerciais e alguns poucos sensores. O CARMEN também possui uma ferramenta gráfica que permite a visualização da simulação de robôs móveis. Através dessa ferramenta é possível simular robôs móveis e sensores lasers, além de ler e utilizar mapas previamente criados por robôs, bem como realizar tarefas simples de navegação.

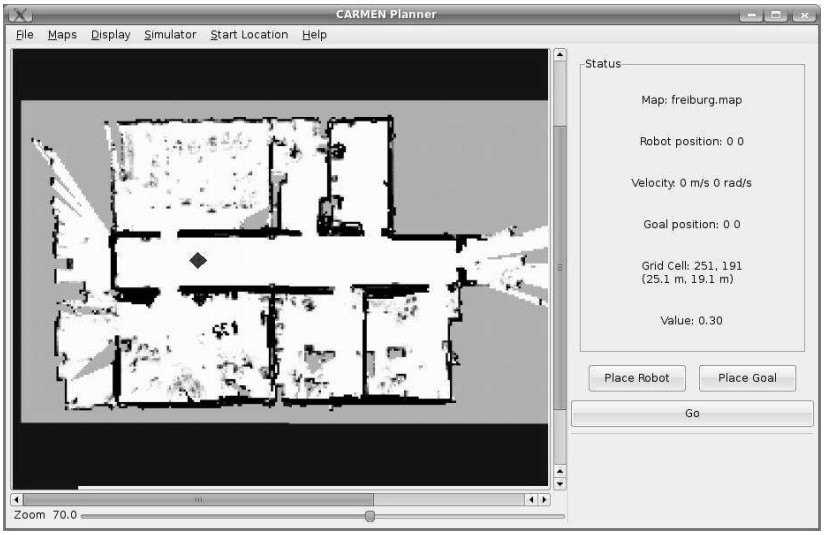


Figura 1.8. Planejamento de trajetória do simulador CARMEN

1.4.3 ARIA/MobileSim

O ARIA (*Advanced Robotics Interface for Applications*) é um sistema desenvolvido para o controle dos robôs fabricados pela empresa MobileRobots. Dentre os robôs fabricados pela MobileRobots, destaca-se o Pioneer, que é amplamente utilizado para testes experimentais em diversas universidades e institutos de pesquisa. Devido à popularidade dessa plataforma, o ARIA, bem como seu antecessor Saphira, também vêm sendo adotados como padrão de controle de robôs móveis em diversas instituições. O ARIA utiliza a linguagem C++ e é compatível com as plataformas Windows e Linux.

O MobileSim é um simulador de ambientes bidimensionais para robôs móveis compatível com o controlador ARIA. Esse simulador é baseado no código do Stage, com algumas modificações feitas pela empresa MobileRobots, dentre elas o suporte para ambientes Windows.

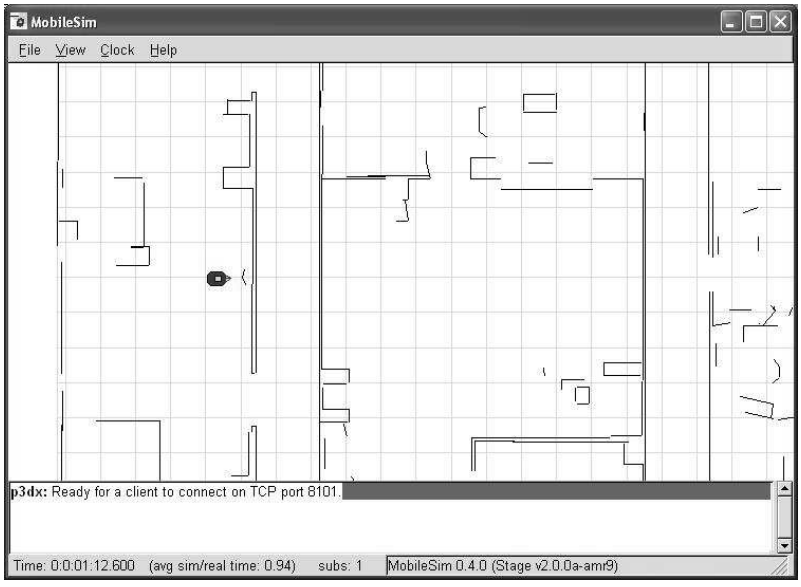


Figura 1.9. Simulador MobileSim

1.4.4 Microsoft Robotics Studio

O *Microsoft Robotics Studio* (MRS) é uma ferramenta distribuída sob licenças *professional*, *academic* e *express*, podendo exigir o pagamento da licença ou sob certas condições podem ser obtidas gratuitamente, porém esta não é uma ferramenta de código aberto (a versão *express* é gratuita mas de uso mais restrito). O controlador/simulador de robôs desenvolvidos pela Microsoft é compatível o ambiente Windows e com suas ferramentas visuais de programação .NET. O hardware suportado atualmente incluiu alguns tipos de robôs comerciais, entre eles o *Lego Mindstorms* e alguns sensores. O MRS suporta a simulação de futebol de robôs, luta de robôs, robôs humanóides, braços mecânicos, entre outros. O simulador MRS possibilita a simulação virtual em ambientes 3D e implementa simulação física baseada no PhysX (NVidia/AGEIA).

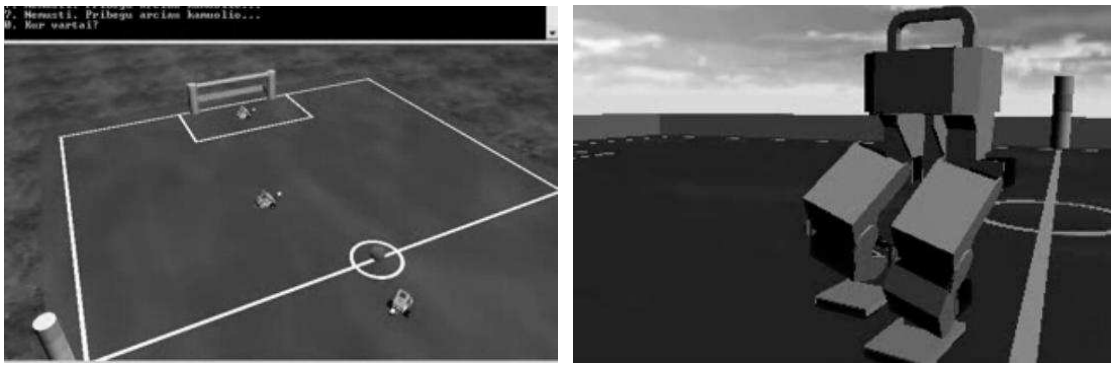


Figura 1.10. Simulações realizadas no Microsoft MRS

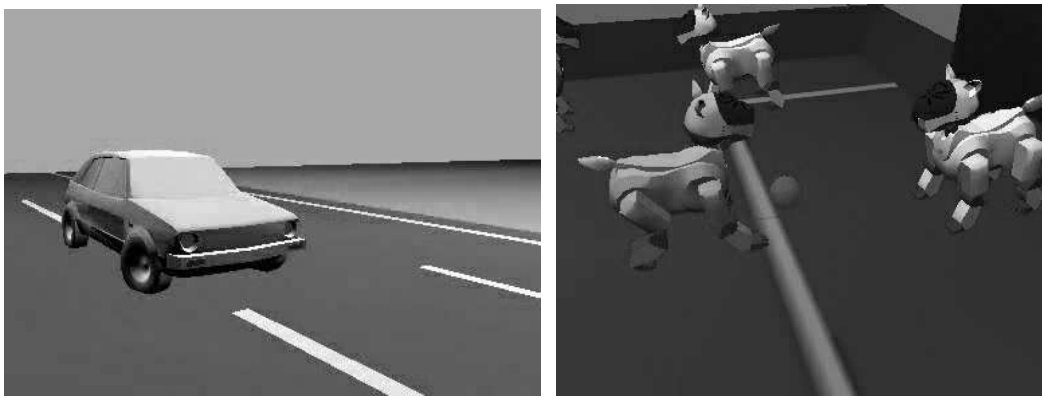


Figura 1.11. Simulações realizadas no Webots da Cyberbotics

1.4.5 Webots

O Webots é um controlador/simulador, desenvolvido pela empresa Cyberbotics em conjunto com pesquisadores do *Swiss Federal Institute of Technology*. O sistema vem sendo utilizado em mais de 450 universidades e institutos de pesquisa e é compatível com plataformas Linux, Windows e Macintosh e liguagens C, C++ e Java. Apesar da versão completa do software exigir o pagamento de licença, uma versão para teste pode ser baixada gratuitamente. O Webots suporta uma grande quantidade de plataformas robóticas e sensores, dentre eles, os robôs AIBO e Lego Mindstorms. O simulador do Webots permite que robôs e ambientes sejam modelados em 3D, suportando a simulação física de sistemas complexos que incluem articulações mecânicas, sistemas

dinâmicos e detecção de colisões, baseado na biblioteca ODE. O Webots pode simular uma grande variedade de ambientes e sistemas como futebol de robôs, veículos robóticos, robôs humanóides, cachorros robóticos e etc.

1.5. Projeto de Sistemas Robóticos Inteligentes

O projeto de um Sistema Robótico Inteligente, conforme apresentado nas seções anteriores, envolve o projeto de Hardware e o projeto de Software. O HW é responsável pela implementação física de sensores, atuadores, interfaces e do processamento dos dados através do uso de um processador embarcado: microprocessadores, microcontroladores, hardware dedicado, hardware reconfigurável (FPGAs, Soft-Cores, SoPC, PSoC – *Programmable System-on-Chip)*. O SW implementa o sistema de controle robótico, responsável pela análise dos sinais dos sensores, planejamento e tomada de decisões, e controle dos atuadores responsáveis pelo deslocamento e ações do robô. Esta seção começará com a apresentação dos componentes de SW usualmente adotados em sistemas de controle robótico inteligente, concluindo com uma discussão sobre o co-projeto de HW e SW de sistemas robóticos inteligentes.

Grande parte da dificuldade em se programar robôs móveis deve-se ao fato de que os elementos que compõe o robô, como sensores e atuadores, apresentam um certo grau de incerteza nos dados obtidos ou nas ações desempenhadas. Além disso, o ambiente em que esses robôs operam, normalmente, é dinâmico e imprevisível. A simples tarefa de se mover de uma extremidade de um corredor até a outra pode se tornar consideravelmente mais difícil se existirem pessoas andando nos corredores, se objetos que possam bloquear a passagem do robô forem colocados em seu caminho ou se os sensores e atuadores do robô não funcionarem com precisão. Dentre os algoritmos usados na área de robôs móveis, a maioria é baseada em estimação estatística. A teoria das probabilidades vem sendo usada com sucesso para acomodar a incerteza existente, tanto internamente no robô, como no ambiente em que ele opera. Hoje em dia, a maior parte dos algoritmos de localização e mapeamento utilizam teoria probabilística de alguma forma [Thrun 2005]. A seguir serão apresentadas técnicas de localização, mapeamento e navegação robótica, de grande importância para a implementação de sistemas autônomos robustos e inteligentes, onde muitas destas técnicas fazem uso de algoritmos como os referidos acima.

1.5.1 Localização

Localização consiste em estimar a posição de um robô no ambiente. Determinar sua própria posição é uma capacidade básica para que qualquer tarefa de navegação seja executada. É fundamental que um robô saiba exatamente a posição em que se encontra dentro de um ambiente para que o mesmo possa planejar a uma trajetória até o seu destino e cumprir as tarefas que lhe forem alocadas [Fox 1999].

A grande maioria dos robôs móveis possui um sistema de odometria que permite estimar o deslocamento dos mesmos e, conseqüentemente, calcular a posição dos robôs no ambiente. Normalmente, as informações odométricas são obtidas através de sensores acoplados às rodas dos robôs, de forma a calcular o deslocamento dos mesmos a partir do movimento destas. Na prática, existem diversos fatores que causam erros nesse tipo de dispositivo como a imprecisão mecânica do sistema e irregularidades do terreno em

que o robô atua. Outro fator que inviabiliza a localização do robô a partir da odometria é o fato de que os pequenos erros causados durante o cálculo da posição do robô vão se acumulando ao longo do tempo. Uma pequena variação na pressão dos pneus de um robô pode facilmente gerar um erro de muitos metros na estimativa de localização após algum tempo de navegação do robô. Devido a sua grande importância dentro da área de robótica móvel, o problema da localização vem sendo amplamente estudado pelos pesquisadores da área e, atualmente, existem algoritmos bastante eficientes na solução do mesmo. Normalmente, é fornecido um mapa do ambiente e a localização é estimada com base nas informações fornecidas pelos sensores. O grande desafio na solução do problema de localização está no fato de que tanto as informações sobre o ambiente como os dados fornecidos pelos sensores são normalmente limitados e imprecisos. Nesse contexto, técnicas probabilísticas têm sido amplamente utilizadas na solução desse problema.

O problema da localização de robôs móveis pode ser dividido nas categorias local e global [Thrun 2005]. No problema de localização local, a posição inicial do robô é conhecida. A solução para esse problema consiste em lidar com a imprecisão dos sensores e manter uma estimativa consistente da posição do robô no ambiente. No caso da localização global, o robô não tem informações prévias de sua posição no mapa, o que torna o processo de localização consideravelmente mais complexo. Dentre os algoritmos de localização para robôs móveis, destaca-se o método de Markov. O algoritmo de localização de Markov é baseado no filtro de Bayes, uma poderosa técnica de estimação estatística, aplicado ao contexto da localização robótica [Thrun 2005]. Como se trata de uma formulação geral para a solução do problema, existem diversas formas de implementar esse algoritmo e de representar os dados do ambiente e da posição dos robôs. Dentre as quais destacam-se o filtro de Kalman, o método de grid e o método de Monte Carlo.

1.5.1.1 Localização de Kalman

O método de localização baseado no filtro de Kalman utiliza o algoritmo homônimo para estimar a posição do robô através de informações imprecisas obtidas por sensores. Esse algoritmo é amplamente utilizado em diversas áreas da engenharia elétrica em sistemas nos quais as informações são imprecisas e nem sempre disponíveis. Existe uma robusta formulação matemática por trás do mesmo, provando sua convergência quando determinados requisitos são atendidos. No caso da localização de robôs móveis, a formulação tradicional do filtro de Kalman utiliza determinados pontos no ambiente que servem como referência para que a posição do robô seja estimada [Leonard 1991]. O filtro de Kalman utiliza funções Gaussianas para representar cada uma das coordenadas da posição do robô no ambiente. Uma vez que a função Gaussiana é unimodal, esse método é indicado apenas na solução do problema de localização local.

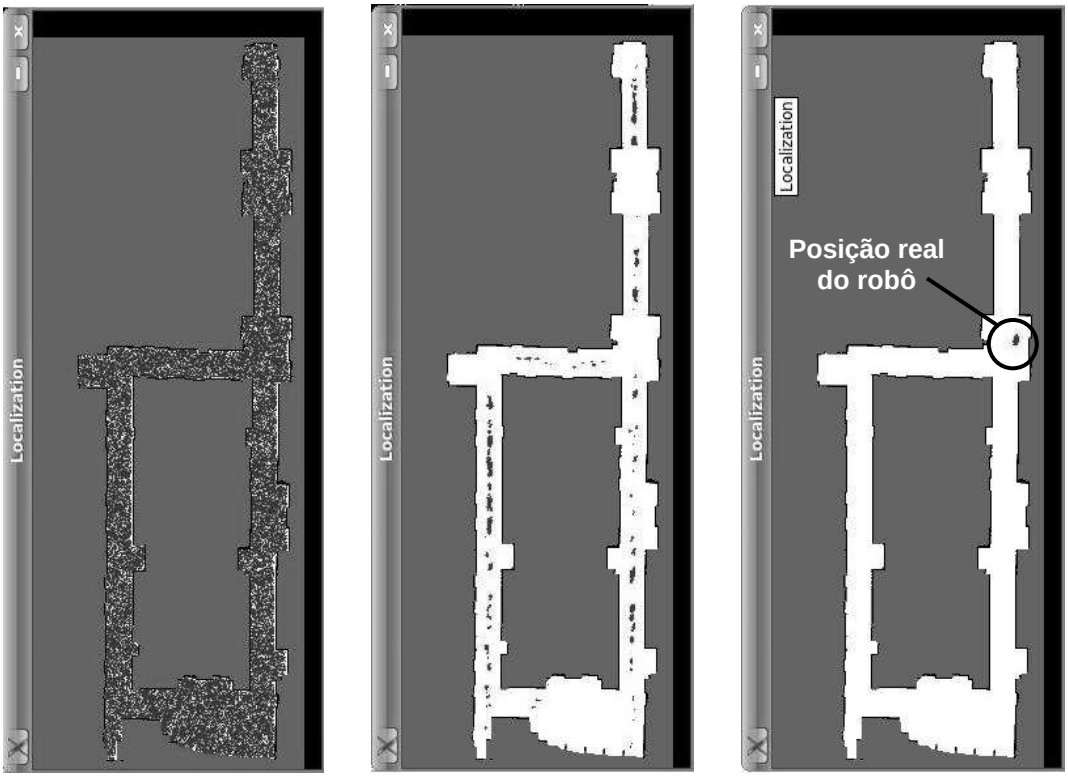
1.5.1.2 Localização em Grid

O algoritmo de localização de Markov consiste em dividir o ambiente em pequenas unidades de espaço chamadas células. Para cada uma dessas células é calculada a probabilidade de o robô estar ocupando a posição correspondente àquela célula [Fox 1999]. Este método, portanto, permite que múltiplas hipóteses sobre a posição do robô sejam mantidas, sendo adequado para a solução dos problemas de localização local e

global. Conforme o robô se desloca e processa os dados obtidos dos sensores, algumas hipóteses de localização são descartadas e outras reforçadas até que a localização correta do robô seja determinada. O ambiente também é representado através das células, no qual cada uma delas é marcada como ocupada ou livre As células ocupadas representam a presença de obstáculos naquela posição do ambiente, como paredes, mesas e cadeiras. As células livres representam espaços onde não há presença de obstáculos. A representação de ambientes através de grids de ocupação é amplamente utilizada na área da robótica e será discutida com mais detalhes na seção 1.5.2.

1.5.1.3 Localização de Monte-Carlo

O algoritmo de localização de Monte Carlo consiste em manter várias hipóteses de localização no ambiente, mas cada hipótese é representada por uma unidade chamada partícula (daí esse algoritmo também ser conhecido como filtro de partículas) [Fox 1999b]. Ao ser iniciado, o algoritmo distribui partículas por todo o mapa do ambiente. Comparando os dados obtidos dos sensores com as informações disponíveis no mapa, é associado um peso para cada partícula. Este peso representa a chance dessa partícula representar a real localização do robô. Durante a execução do algoritmo, o peso de cada partícula é calculado. Partículas com peso baixo são eliminadas enquanto as partículas com peso alto são replicadas. Com o tempo todas as partículas tendem a se concentrar no local do mapa que representa a localização correta do robô. Essa técnica também utiliza grids de ocupação para representar o ambiente.



(a) **(b)** **(c)**

Figura 1.12. Localização de Monte Carlo

A Figura 1.12 mostra o mapa de um ambiente com diversos corredores interligados e a evolução da execução do algoritmo de Monte Carlo. As áreas brancas na Figura representam o espaço livre, as áreas pretas representam as regiões ocupadas (paredes e portas) e as áreas cinza representam regiões com a ocupação desconhecida pelo robô. Os pontos cinzas (partículas) que aparecem na área branca são possibilidades do robô ocupar local naquele momento. Na medida em que o robô se desloca e obtém mais informações dos sensores, o algoritmo descarta mantém apenas as partículas com chance real de representar a posição correta do robô. No início da execução do algoritmo (Figura 1.12(a)), não existe nenhuma informação sobre a posição do robô, portanto praticamente todo espaço branco é ocupado por partículas. Na Figura 1.12(b) pode-se notar que diversas partículas foram eliminadas, porém a posição do robô ainda não pode ser identificada. Por fim, a Figura 1.12(c) mostra o resultado da localização após a convergência do algoritmo.

1.5.2 Mapeamento

A aquisição desses mapas é um problema de estimação que consiste em uma tarefa fundamental para o desenvolvimento de robôs móveis autônomos. Normalmente, tarefas básicas executadas por um robô necessitam de uma representação do ambiente para serem executadas. Supondo, por exemplo, que um robô deve planejar uma trajetória para se locomover de sua posição atual até uma posição de destino. Nesse caso, um mapa do ambiente é fundamental para que o caminho mais eficiente seja encontrado. Através do mapa, o robô verifica os possíveis caminhos que levam até a posição desejada e os obstáculos que devem ser evitados [Thrun 2002]. Outra aplicação robótica que é normalmente necessita de um mapa é a localização, que consiste em estimar a posição de um robô em um ambiente previamente conhecido. Dentro do contexto da navegação, é necessário que o robô saiba com precisão sua localização dentro do ambiente para que o mesmo possa determinar uma trajetória até o ponto de destino. Uma vez que os sensores odométricos dos robôs comerciais normalmente apresentam algum grau de imprecisão, são utilizadas informações dos sensores, juntamente com um mapa do ambiente, para se estimar a posição do robô. Existem também técnicas que permitem que o mapa seja criado ao mesmo tempo em que o robô se localiza no ambiente. Essas técnicas são chamadas de localização e mapeamento simultâneos (SLAM), e são consideravelmente mais complexas do que técnicas de mapeamento e localização em separado.

Existem basicamente dois tipos de mapas criados por robôs móveis: topológicos e métricos. Os mapas topológicos representam o ambiente por grafos, no qual os vértices representam regiões de interesse no ambiente e os arcos representam as vias que interligam essas regiões [Mataric 1990][Kuipers 1991]. Os mapas topológicos são bastante eficientes para se estimar trajetórias em um ambiente, mas apresentam uma representação muito pobre do ambiente físico. Os mapas métricos representam os ambientes físicos em detalhes e também podem ser utilizados para se estimar trajetórias. Entre os mapas métricos, destaca-se a representação do ambiente utilizando-se grids de ocupação (*occupancy grids*) [Elfes 1989].

O método de malha de ocupação consiste em criar uma representação bidimensional, na qual o ambiente é dividido em pequenas unidades de espaço ou

células (semelhantes às unidades usadas no método de localização de Markov), e para cada uma dessas células é calculada a possibilidade daquele espaço estar ocupado ou não. Este cálculo é realizado com base nas informações obtidas dos sensores. Normalmente, são utilizados sensores que detectam a distância entre o robô e os obstáculos (sonares e lasers) para o mapeamento de ambientes.

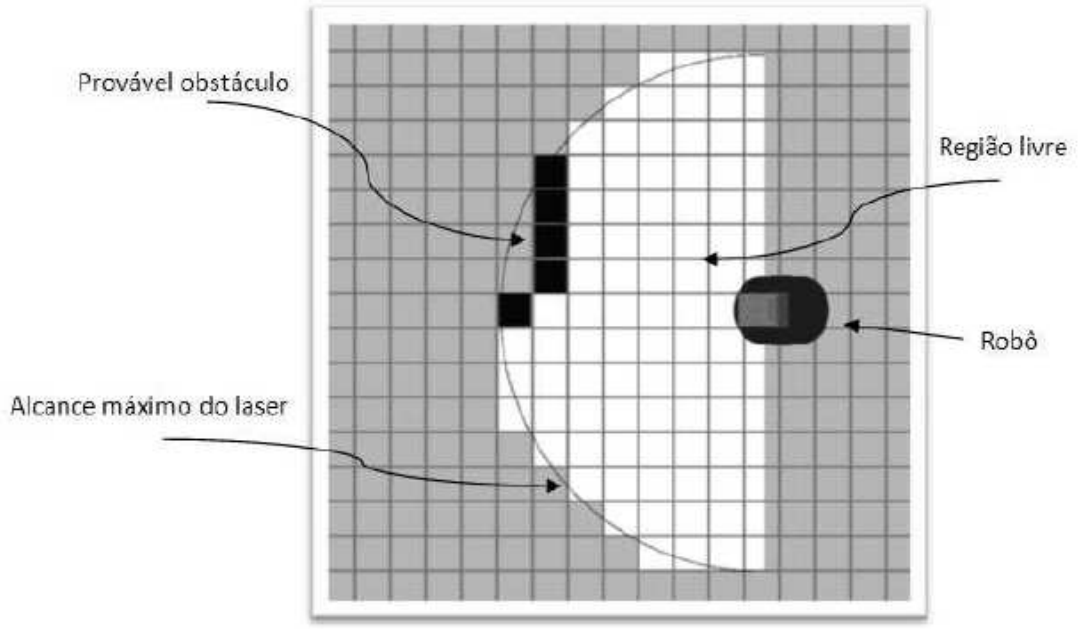


Figura 1.13. Grid de ocupação

Como o robô inicialmente não tem informações sobre o ambiente, todas as células são classificadas como desconhecidas. À medida que novas informações são obtidas pelos sensores, a classificação de cada célula é atualizada. As células onde o sensor detecta algum tipo de obstáculos têm sua probabilidade de estar ocupadas aumentadas à medida que os sensores confirmam a presença de desses obstáculos (Figura 1.13). As células que se encontram entre o robô e os obstáculos são consideradas livres e têm sua probabilidade de estarem ocupadas reduzidas.

A Figura 1.14 mostra uma planta de um andar de um prédio e o respectivo mapa criado por um robô móvel utilizando sensor laser. Áreas brancas correspondem ao espaço livre, áreas pretas correspondem à área ocupada (paredes e portas) e a área cinza representa as áreas desconhecidas.

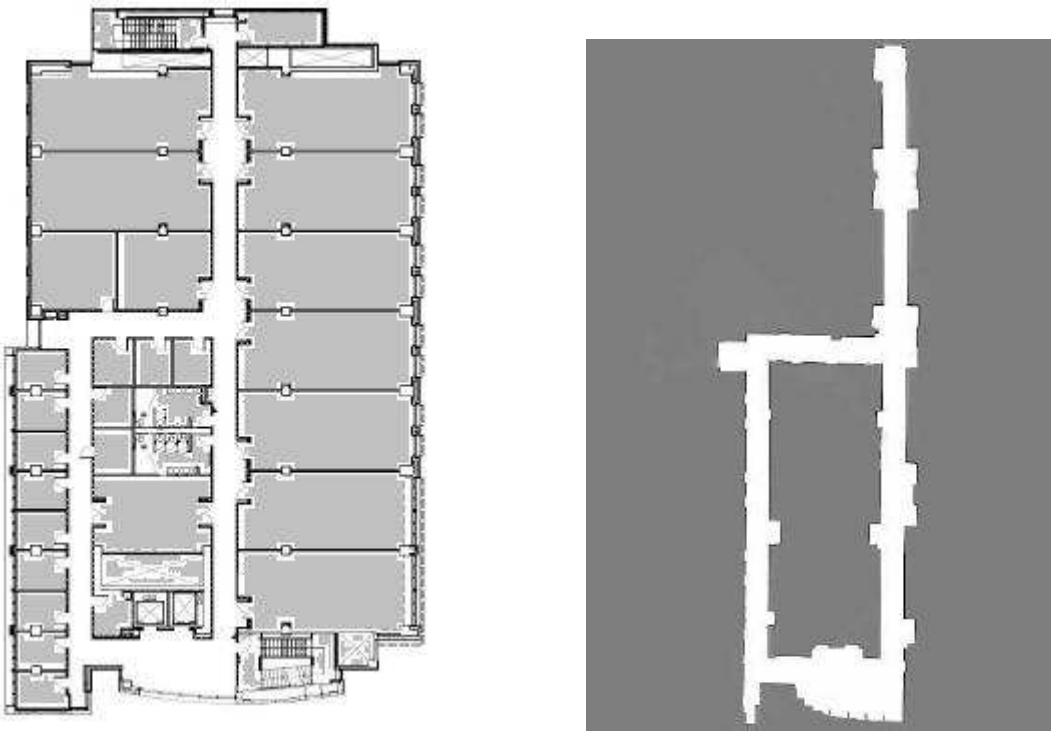


Figura 1.14. Mapa métrico criado por um robô móvel

1.5.3 SLAM

Como visto nas seções anteriores, a localização e o mapeamento são fundamentais para que robôs móveis possam executar a maioria das tarefas que envolvem navegação. A maioria dos algoritmos de localização utilizam um mapa ou alguma informação do ambiente para estimar a posição do robô. Da mesma forma, a maior parte dos algoritmos de mapeamento precisa de boa localização para construir mapas que representem fielmente o ambiente. Existem situações nas quais não se tem nem informações prévias sobre o ambiente nem uma boa localização do robô. Para solucionar esse problema, existem algoritmos de mapeamento de localização simultâneos ou SLAM (*simultaneous localization and mapping*). Esses algoritmos são consideravelmente mais complexos do que os algoritmos de localização e mapeamento vistos anteriormente. Dentre os algoritmos desenvolvidos para a solução do problema de SLAM destacam-se o filtro de Kalman e o FastSLAM.

Da mesma forma que o algoritmo que utiliza o filtro de Kalman para a localização de robôs móveis, a solução para o problema de SLAM que utiliza a mesma técnica também utiliza mapas formados por pontos de referência no ambiente (*landmarks*) [Smith 1990] [Dissanayake 2001]. Como, nesse caso, o robô não tem nenhuma informação prévia do ambiente, tanto a posição do robô como a posição desses pontos de referência devem ser estimados simultaneamente. Conseqüentemente, o número de variáveis que devem ser estimadas cresce proporcionalmente ao tamanho do mapa. O algoritmo armazena essas variáveis em vetores e matrizes proporcionais à quantidade de pontos de referencia no espaço. Durante o processo de estimação são realizadas diversas operações utilizando essas matrizes e vetores. Dessa forma, o tempo de execução desse algoritmo depende do número de variáveis que deve ser estimado,

invibializando sua utilização em situações onde o número de pontos de referência no ambiente é muito grande. O algoritmo original é capaz de processar mapas com algumas centenas de *landmarks* em tempo real, porém, existem técnicas na literatura para tornar esse algoritmo mais eficiente [Montemerlo 2003].

O FastSLAM consiste em outra alternativa para a solução do problema de localização e mapeamento simultâneos. Esse algoritmo combina um filtro de partículas para estimar a trajetória do robô e diversos filtros de Kalman, sendo que, cada um deles estima a posição de um *landmark* no mapa [Montemerlo 2002] [Montemerlo 2003]. O princípio do FastSLAM baseia-se no fato de que, supondo-se que a localização do robô é conhecida, a posição de cada *landmark* pode ser estimada de maneira independente. Essa estimativa da posição de cada *landmark* no mapa é realizada com um filtro de Kalman. Como cada um desses filtros estima apenas as 2 coordenadas (x, y) de cada ponto de referência no ambiente, a atualização dos mesmos é bastante eficiente, permitindo que centenas de milhares de *landmarks* possam ser estimados simultaneamente. No entanto, de acordo com a própria definição do problema, a posição do robô não é conhecida. Desse modo, o FastSLAM utiliza um filtro de partículas, sendo que cada partícula representa uma possível trajetória para o robô. A propagação das partículas é baseada nas informações odométricas com a adição de um erro aleatório, que funciona como uma possível compensação da imprecisão do odômetro. Como o erro adicionado a cada uma das partículas é aleatório, existe uma pequena diferença no trajeto de cada uma das partículas. Quando essas partículas visitam regiões previamente mapeadas, as informações sobre o mapa obtido até então são comparadas com os dados dos landmarks detectados naquele momento. Quanto melhor a correspondência entre a esses landmarks e aqueles que foram registrados previamente no mapa, maior é a chance de que essa partícula represente a trajetória real do robô. Desse modo, partículas com boa correspondência continuam ativas enquanto as partículas nas quais não houve correspondência são substituídas. Quanto maior o número de partículas, maior a chance se conseguir uma estimativa fiel da trajetória do robô e do mapa do ambiente.

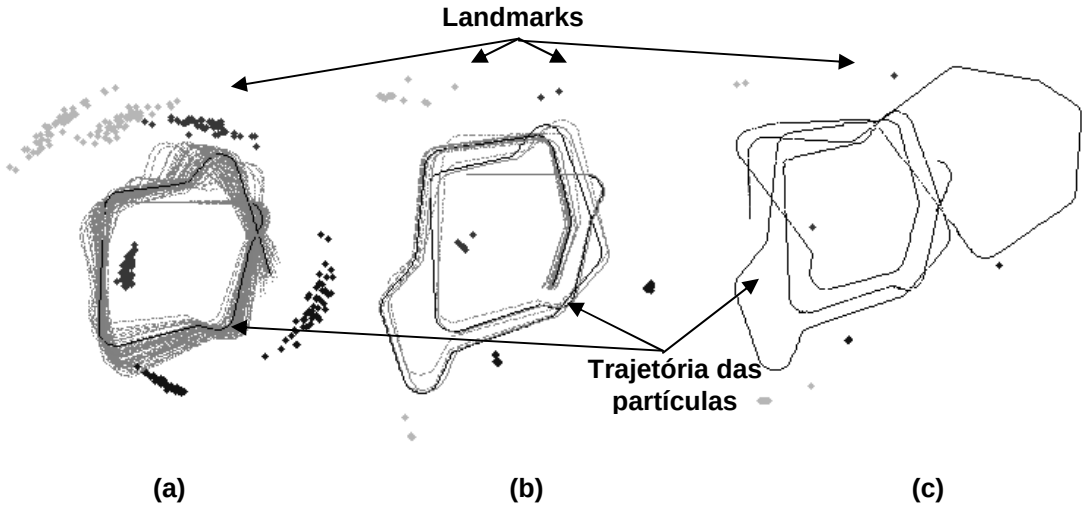


Figura 1.15. Trajetória e mapa estimados utilizando-se o FastSLAM

A Figura 1.15 mostra o processo de localização e mapeamento simultâneos através do FastSLAM. Inicialmente, conforme mostra a Figura 1.15(a), as partículas se dispersam e criam diversas possíveis trajetórias para o robô, juntamente com diversas estimativas de posição para os landmarks. Após o robô visitar parte do ambiente mapeado, diversas partículas são eliminadas, juntamente com as respectivas estimativas de posição dos landmark (Figura 1.15(b)). Por fim, após o robô passar diversas vezes pelo ambiente mapeado, o algoritmo converge para uma única trajetória, que teoricamente, representa a melhor estimativa da trajetória real do robô e da posição correta dos landmarks no mapa, conforme mostra a Figura 1.15(c).

O problema de SLAM consiste em uma área de pesquisa bastante ativa da robótica móvel. Dessa forma, pesquisadores de diversas instituições continuam a procurar soluções mais eficientes na solução desse problema. Além das técnicas descritas nessa seção, existem outros algoritmos que estimam a posição do robô e o mapa do ambiente simultaneamente. Dentre eles, pode-se citar: [Thrun 1998], [Wolf 2005], [Eliazar 2004], [Haehnel 2003] e [Gutmann 1999].

1.5.4 Navegação Robótica

A Navegação robótica consiste em controlar o deslocamento do robô de uma posição inicial até uma posição de destino. Este tipo de tarefa envolve diferentes técnicas, como por exemplo, o planejamento de uma trajetória e a execução da navegação utilizando um mapa previamente disponibilizado do ambiente, ou até mesmo a navegação baseada em uma experiência prévia de "navegação supervisionada", usando técnicas como a navegação visual. A tarefa de navegação usualmente é implementada através do uso de uma arquitetura de controle do tipo reativo, deliberativo ou hierárquico/híbrido, devendo incluir mecanismos para evitar colisões, incluindo o desvio de obstáculos e o tratamento de situações imprevistas.

(a) (b)

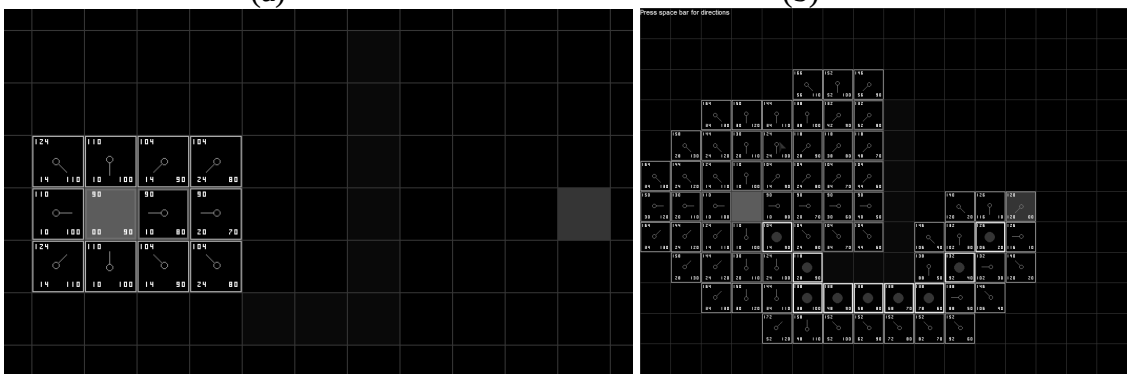


Figura 1.16. Planejamento de Trajetória A* - Mapa de Ocupação [Lester 2005]
(a) Início da exploração de caminhos possíveis, (b) Caminho encontrado

1.5.4.1 Navegação baseada em Mapas

Inicialmente o sistema de controle deve estabelecer uma rota, ou seja, planejar uma trajetória a ser executada, baseando-se no uso de um mapa do ambiente (mapa topológico, mapa métrico ou mapa de ocupação tipo "grid"). Alguns dos métodos mais usados de navegação baseada em mapas são a busca de um caminho "ótimo" através do

uso de algoritmos como o A* (*AStar*) ou o algoritmo de Djikstra [Osorio 2007, Jung 2005, Latombe 1991]. O algoritmo A* usualmente é aplicado em mapas do tipo mapa de ocupação (*grid*), podendo também ser usado com mapas métricos, nos quais este algoritmo se tornou muito conhecido também devido a sua aplicação na implementação de agentes autônomos para jogos [Osorio 2007]. O A* usa uma heurística de busca de caminhos no espaço de configurações, sendo um algoritmo muito eficiente. A Figura 1.16 apresenta um exemplo de trajetória com o uso do A*.

O planejamento de trajetórias também pode ser feito baseado em um mapa métrico, no qual usualmente é feita a busca usando grafos que representam as possíveis soluções (caminhos), assim como a solução ótima (caminho mais curto). A busca em grafos é baseada em um Grafo de Visibilidade, ou em Diagramas de Voronoi, sendo aplicado um método de seleção do melhor caminho entre os diversos caminhos que podem ter sido encontrados [Heinen 2000, Dudek 2000]. O Algoritmo de Djikstra permite encontrar o menor caminho em um grafo, sendo muito usado no planejamento de trajetórias com grafos, apesar de ter um elevado custo computacional. A Figura 1.17 apresenta um exemplo de trajetória baseada no grafo de visibilidade e seleção do menor caminho.

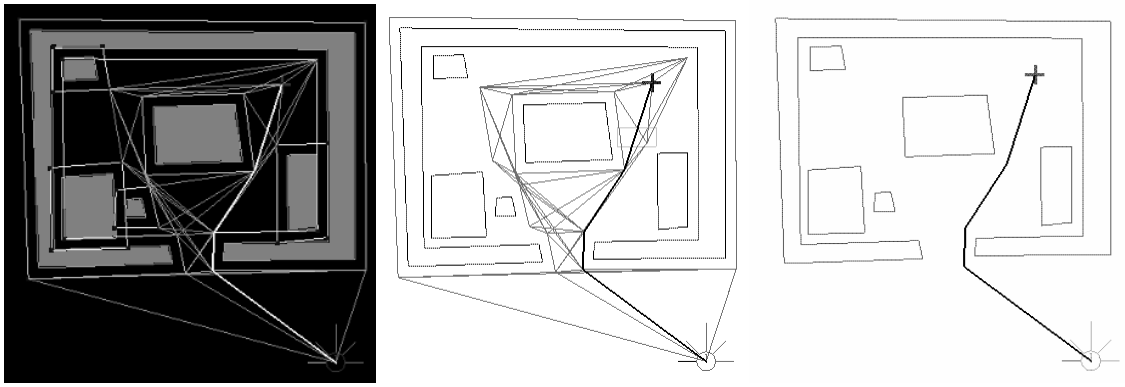


Figura 1.17. Planejamento de Trajetória com Grafo de Visibilidade [Heinen 2000]

É importante destacar que a navegação baseada em mapas funciona bem somente quando existe um mecanismo confiável de localização associado ao processo de deslocamento junto à trajetória planejada. Por exemplo, tomando-se o exemplo da Figura 1.17, se o robô não estiver exatamente onde ele “pensa” estar (devido a erros de deslocamento e posicionamento acumulados durante a sua navegação), dificilmente ele irá conseguir passar por uma porta ou passagem estreita, pois as ações que foram planejadas previamente (comandos a serem enviados aos atuadores) não serão adequadas em função deste possível deslocamento do robô em relação à trajetória inicialmente planejada.

1.5.4.1 Navegação sem o uso de mapas

A navegação em um ambiente pode não requerer um conhecimento muito estruturado e completo como o de um mapa métrico ou de ocupação, onde existem algumas técnicas bastante adotadas de navegação baseada em campos potenciais, navegação visual (baseada em informações visuais) e navegação por composição de comandos.

A navegação por campos potenciais e campos vetoriais [Dudek 2000, Heinen 2002a, Pio 2003] utiliza conceitos de forças de atração e repulsão, nos quais o robô é atraído pelo seu ponto de destino (podendo usar uma bússola ou GPS para saber a direção para onde deve se deslocar) e é repelido pelos obstáculos presentes em seu caminho, ajustando constantemente sua rota, para desviar dos obstáculos que se encontram próximos a ele (detectados pelos seus sensores). Quanto mais próximo de um obstáculo, maior deverá ser a força de repulsão que este obstáculo exerce sobre o robô, forçando-o a se desviar um pouco de sua rota original, mas no entanto evitando a colisão. Este tipo de algoritmo pode ser usado em ambientes com uma concentração não muito grande de obstáculos, sendo que o robô pode em certas configurações específicas do ambiente acabar ficando "preso" (bloqueado). É possível também integrar a técnica de campos potenciais com o uso de mapas, o que permite resolver algumas de suas limitações.

A navegação visual [Jung 2005, Righes 2004] consiste em criar uma "memória fotográfica" do caminho a ser percorrido pelo robô, ou seja, primeiramente o robô é conduzido pelo caminho que ele deverá percorrer, registrando uma memória visual do caminho por onde passou. Em um momento posterior, quando o robô tiver que refazer o caminho ele irá usar os registros visuais do caminho como uma referência que permite manter (e ajustar) sua rota em relação a uma trajetória que foi previamente aprendida. É importante destacar que este tipo de algoritmo não necessita de uma descrição sob a forma de mapas do ambiente, sendo um método bastante fácil e rápido de ser posto em funcionamento, pois basta mostrar ao robô o caminho que ele deve seguir. A Figura 1.18 apresenta um exemplo de navegação visual.

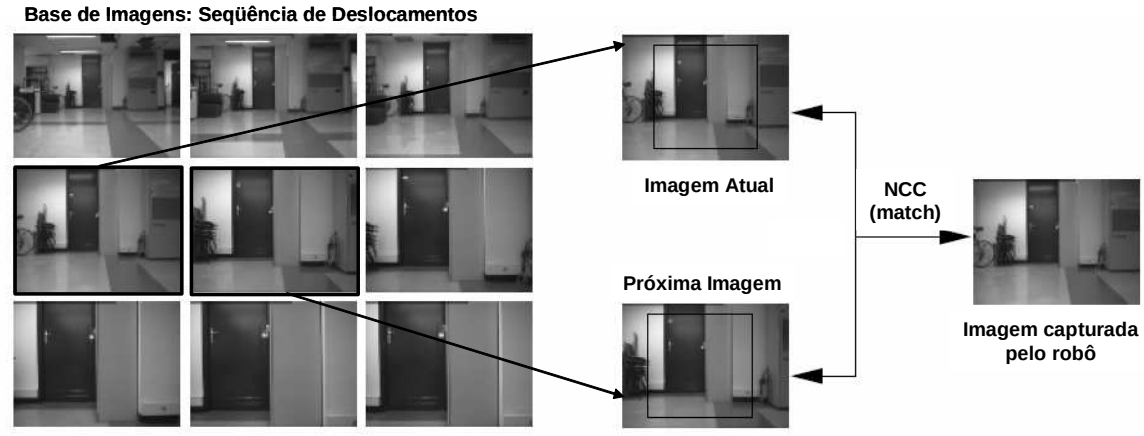


Figura 1.17. Navegação baseada em imagens [Jones 1997]

A navegação por composição de comandos permite que sejam compostos comandos para executar a navegação em determinados trechos da trajetória, de forma pré-determinada, como por exemplo: seguir uma parede, avançar por um corredor, passar por uma porta, deslocar 10 metros em frente e depois dobrar a esquerda, etc. Este tipo de navegação pode ser baseado em comandos reativos (acompanhar uma parede ou marcação no chão), ou também pode ser baseado em comandos deliberativos (avançar uma certa distância, dobrar com um determinado ângulo). Este tipo de navegação também pode usar um método similar ao anterior, no qual o robô registra previamente todos os comandos de navegação e depois volta a reproduzí-los. O problema deste tipo

de estratégia é que o tratamento de erros de posicionamento e de desvio de obstáculos não previstos pode se tornar bastante problemático. Este tipo de técnica é normalmente combinada em sistemas híbridos de controle que realizam em paralelo tarefas de localização e de ajuste de trajetórias (ou simplesmente param quando encontram um obstáculo não previsto em sua trajetória).

Portanto, o planejamento e navegação robótica são tarefas que devem ser projetadas de modo a realizar um controle robótico inteligente e robusto, muitas vezes através da implementação e integração de diferentes técnicas, como as citadas acima, que irão dotar o robô de uma maior capacidade de navegação autônoma e desvio de obstáculos.

1.5.5 Co-Projeto de Hardware e Software de Sistemas Robóticos

O projeto de um sistema robótico, conforme apresentado nas seções anteriores, é uma tarefa complexa, que envolve o projeto de hardware e de software, onde ambos estão diretamente relacionados: é preciso projetar o hardware para com ele se poder posteriormente desenvolver um software adequado e que explore suas potencialidades. Além disto, durante o projeto e desenvolvimento do sistema de controle de um sistema robótico, são necessários diversos ciclos de ajuste do hardware do robô, seja pela adição de mais sensores, ajuste e posicionamento destes sensores, adaptações nos atuadores, e inclusive precisando realizar constantes aperfeiçoamentos no sistema de controle embarcado dos robôs. Diante deste ciclo de projeto, com revisões constantes de hardware e software, fica bastante clara a importância de se adotar metodologias modernas de projeto de sistemas como o Co-Projeto de Hardware e Software baseado em ferramentas de simulação. Através do uso de ferramentas de simulação é possível realizar uma melhor avaliação do projeto de hardware e de software, sendo inclusive possível realizar a implementação em paralelo do hardware, ao mesmo tempo em que se desenvolve o software usando ferramentas se simulação. Com o uso das ferramentas de simulação da plataforma robótica, é possível desenvolver, testar e validar o sistema de controle antes mesmo de se ter pronta e disponível a plataforma de hardware.

É importante destacar também que atualmente os novos paradigmas do projeto de sistemas embarcados vêm de aplicações que exigem um grande paralelismo no processamento de informações de diferentes fontes, com é o caso de sistemas automotivos ou de sistemas robóticos de controle e navegação. Para tanto, existem diversas alternativas de implementação disponíveis, que vão desde a utilização de processadores de propósito geral ou processadores com um conjunto de instruções dedicado a determinadas aplicações (como os microcontroladores ou *Digital Signal Processors* – DSPs) até a utilização de hardware programável (como os FPGAs – *Field-Programmable Gate Arrays*) ou ASICs – *Application-Specific Integrated Circuits*. Destas, as duas últimas vem ganhando terreno principalmente devido às novas necessidades dos sistemas embarcados atuais, pois permitem o projeto concorrente de hardware e software em um único chip.

Para implementar sistemas de controle em robôs reais uma solução atraente é a utilização do processador (geralmente um microcontrolador) e memória, comumente encontrados nos robôs comerciais (como os Khepera e os Pioneer), para acomodar versões em software dos controladores e algoritmos de navegação. Uma alternativa para isso é a utilização de FPGAs para implementar os módulos de software e hardware do

sistema de controle. Algumas das vantagens dos FPGAs sobre microcontroladores são a maior velocidade de trabalho, menor consumo de energia e paralelismo. Existem diversos exemplos de aplicações de robótica com sistemas embarcados desenvolvidos usando hardware reconfigurável (FPGAs), como por exemplo, sistemas de visão [Assunção 2007, Bonato 2006 e 2008] e sistemas de localização e mapeamento baseados também em FPGAs [Wolf 2005, 20007][Bonato 2007].

Dentro deste contexto, as diversas alternativas de FPGAs do mercado apresentam recursos atraentes como os chamados SOPCs (System on a Programable Chip). SOPCs são sistemas que integram toda a funcionalidade de processadores, periféricos e memória, entre outras, num único chip. Os FPGAs possuem grande capacidade e diversidade de elementos lógicos programáveis e conseguem acessar dispositivos externos com grande velocidade. O uso de sistemas reconfiguráveis (FPGAs) e a adoção de uma metodologia de co-projeto de hardware e software através do uso de ferramentas de simulação permitem assim dar uma maior flexibilidade, economia, rapidez, e obter melhores resultados no projeto de sistemas robóticos.

1.6. Aplicações práticas de Robótica Móvel

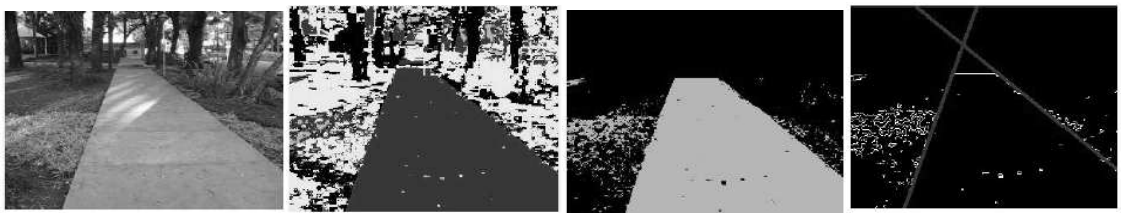
Nesta seção serão apresentados diversos projetos de robótica móvel autônoma desenvolvidos com a participação dos membros do LRM (Laboratório de Robótica Móvel) do ICMC USP [LRM 2009]. Estes projetos permitem demonstrar de modo prático a aplicação de diversos conceitos e técnicas descritos neste artigo, relacionados com desenvolvimento de aplicações de robótica móvel e de sistemas de controle robótico inteligente. Esta seção será concluída com a apresentação de outro conjunto de aplicações de sucesso da robótica móvel, destacando-se o caso do *DARPA Grand Challenge* e do *DARPA Urban Challenge* [Thrun 2006, Urmson 2008], competições focadas no desenvolvimento de veículos autônomos realizadas nos Estados Unidos em 2005 e 2007 respectivamente.

1.6.1. Navegação autônoma em ambientes externos

A navegação é um dos problemas fundamentais da robótica móvel, sendo um pré-requisito para que outras tarefas possam ser executadas. A grande maioria dos algoritmos de navegação é desenvolvida para atuar em ambientes internos estruturados, onde o grande desafio da navegação é o desvio de obstáculos. A navegação em ambientes externos consiste em um problema muito mais interessante e complexo. Além do desvio de obstáculos, é necessário que o robô identifique o terreno em que pode navegar. Uma aplicação direta dessa tecnologia é o desenvolvimento de sistemas de direção autônoma para veículos. Neste caso, é necessário identificar os limites da rua (área em que o carro pode navegar), e possíveis obstáculos e depressões em seu caminho.

Dentro desse contexto, foi desenvolvido um sistema de navegação em ambientes externos baseado em visão computacional. Baseado nas informações obtidas por uma câmera, o sistema deve encontrar a via navegável (rua ou calçada). A solução para esse problema proposta em [Wolf 2008] utiliza um conjunto de técnicas de processamento de imagens descrito abaixo.

A primeira tarefa para se identificar a região navegável na imagem é a utilização de algoritmos de segmentação de imagens. A segmentação classifica os pixels da imagem em classes distintas de acordo com sua cor. Desse modo, é possível isolar a região de interesse na imagem. Após a segmentação, é necessário identificar os limites da área desejada (área navegável) para que os controles do robô sejam controlados com o objetivo de manter o mesmo dentro da área de interesse. Essa tarefa é realizada através de filtros de imagem que detectam as bordas entre as regiões segmentadas e permitem que o robô se atenha à região delimitada e classificada como segura.



(a) **(b)** **(c)** **(d)**

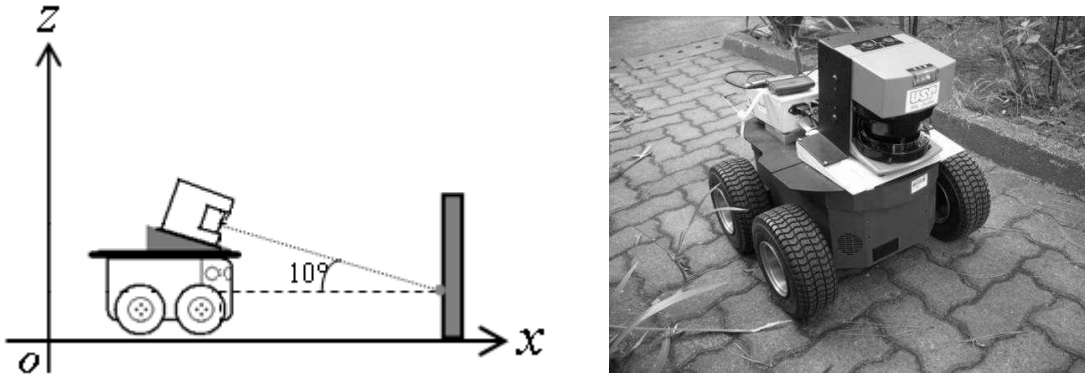
Figura 1.18. Identificação da via navegável utilizando processamento de imagens

A Figura 1.18(a) mostra um exemplo de ambiente utilizado durante os experimentos de classificação visual do terreno. As Figuras 1.18(b) e 1.18(c) mostram os resultados da segmentação e da seleção da região de interesse, respectivamente. Por fim, a Figura 1.18(d) apresenta as linhas que delimitam a região navegável.

1.6.2. Mapeamento tridimensional de ambientes externos

A construção de modelos do ambiente é uma tarefa fundamental para o desenvolvimento de robôs móveis autônomos. Dentre as aplicações desses modelos (mapas) pode-se citar: planejamento de trajetória, localização e monitoramento. Grande parte dos mapas criados por robôs móveis são modelos bidimensionais, que são adequadas para representar ambientes razoavelmente estruturados, como interiores de prédios. No caso de ambientes externos, parcialmente estruturados, mapas 2D representam o ambiente de maneira pobre, uma vez que não é possível capturar detalhes de árvores, veículos e outros objetos utilizando esse tipo de representação. Nesse caso, a utilização de técnicas que permitam a criação de mapas tridimensionais é mais adequada, permitindo que os detalhes dos objetos possam ser capturados com maior precisão.

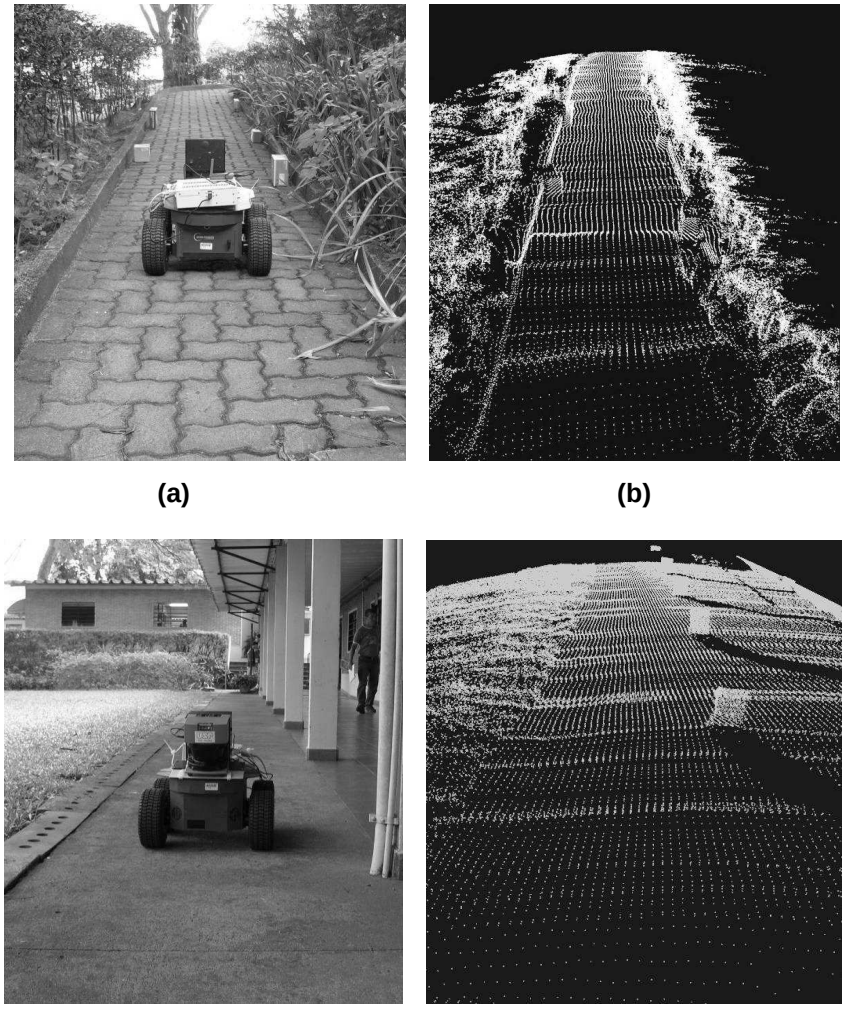
Uma das técnicas utilizadas na criação de mapas 3D é a nuvem de pontos. Basicamente, este método consiste em representar as estruturas e objetos do ambiente através de conjuntos de pontos no espaço cartesiano. Normalmente são utilizados sensores lasers na aquisição dos dados, devido a sua boa precisão e longo alcance. Dentre as possibilidades de mapeamento 3D, o mapeamento do terreno em que o robô atua é particularmente útil para tarefas que envolvem navegação.



(a) **(b)**

Figura 1.19. Robô móvel utilizando sensor laser para mapeamento de terreno

A Figura 1.19 mostra um robô móvel utilizando um sensor laser para o mapeamento de terreno. Pode-se notar que é utilizado um suporte para que o laser seja direcionado para baixo, de forma a obter informações sobre o terreno.



(a) **(b)**

(c) **(d)**

Figura 1.20. Robô móvel utilizando sensor laser para mapeamento de terreno.

As Figuras 1.20(a) e (c) mostram os ambientes reais que foram utilizados nos experimentos de mapeamento de terreno, enquanto as Figuras 1.20(b) e (d) apresentam os mapas 3D criados pelo robô. A mesma técnica foi utilizada montando-se o sensor laser apontado para cima. Dessa forma, é possível obter informações sobre prédios e outros objetos no ambiente, conforme mostra a Figura 1.21.

Figura 1.21. Mapa de um dos prédios do ICMC-USP construído por um robô móvel

1.6.3. Veículos Inteligentes

Os sistemas de auxílio à condução de veículos são uma importante aplicação de robótica móvel, seja de modo completamente autônomo ou de modo semi-autônomo atuando para aumentar a segurança do condutor e das vias em geral [Jung 2005, Kelber 2003]. Existem diversas aplicações que vem sendo desenvolvidas neste sentido, nas quais pode-se citar alguns exemplos: detecção de pedestres, detecção de placas de sinalização (com alerta ao motorista), estacionamento autônomo de veículos, condução autônoma de veículos de passageiros e de carga, comboios automatizados, sistema de controle para evitar saída de pista, sistema de alerta e prevenção contra colisões, entre outras aplicações.

O projeto SEVA – Sistema de Estacionamento de Veículos Autônomos é um exemplo de uma aplicação que foi desenvolvida a partir da simulação virtual [Heinen 2007]. Este projeto consiste no desenvolvimento de um Sistema de Controle Inteligente baseado no uso de Redes Neurais Artificiais (RNA), capazes de aprender a controlar um veículo autônomo de forma a estacionar em uma vaga paralela. O sistema de controle permite que sejam adquiridos dados dos sensores (sonares) e atuadores (acelerador e barra da direção), quando um motorista realiza o estacionamento do veículo de forma manual, e, de posse destes dados, é aplicado um algoritmo de aprendizado de máquina capaz de “aprender” a estacionar o veículo de forma autônoma. A Figura 1.22 apresenta o processo de estacionamento autônomo do veículo, controlado pela RNA.

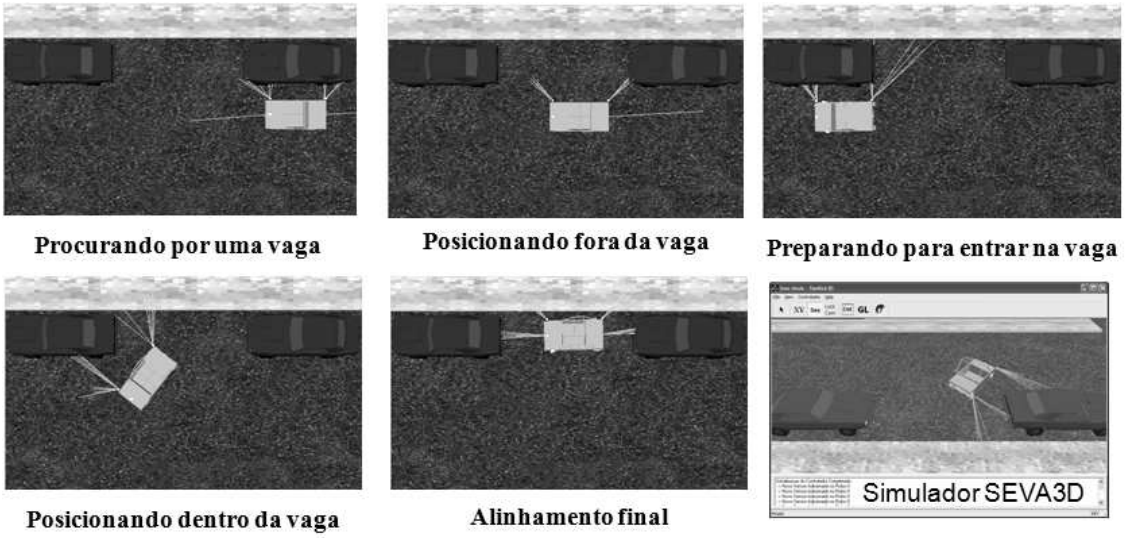
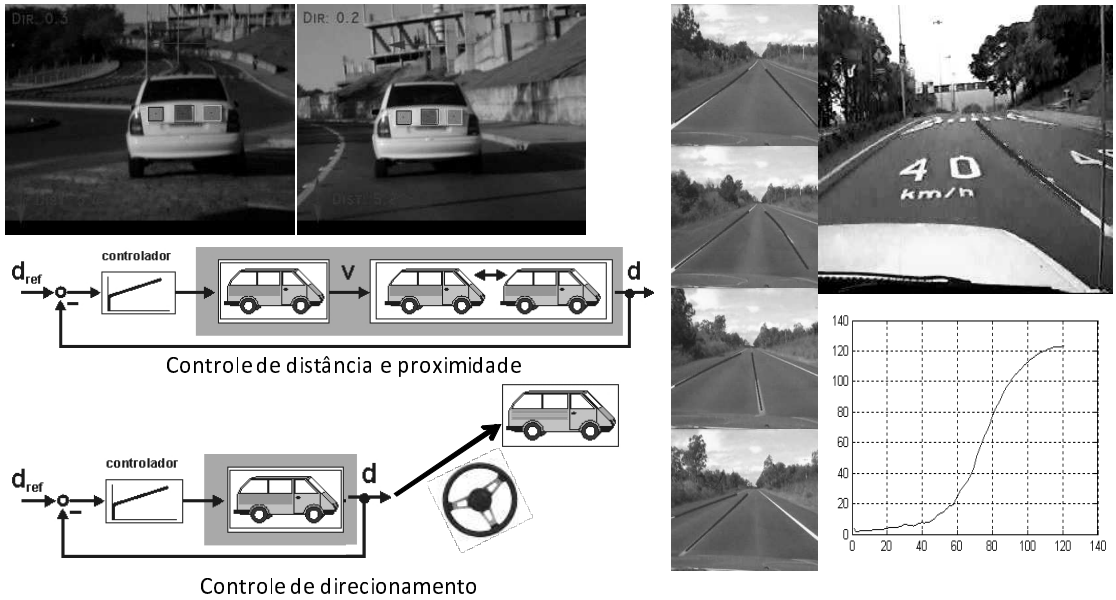


Figura 1.22. SEVA – Sistema de Estacionamento Autônomo de Veículos

A Rede Neural, após o aprendizado, implementa um autômato finito (FSA) que irá controlar o veículo, no qual em cada instante de tempo a RNA é responsável por ler o estado dos sensores (sendo robusta aos ruído e as imprecisões dos sonares) e decidir qual ação deve efetuar (controle de direção e aceleração), quando então se deve passar para o estado seguinte do autômato, de modo a ir executando a seqüência de ações referentes ao estacionamento do veículo. O simulador SEVA3D foi muito importante para que este projeto fosse desenvolvido, permitindo uma adequada configuração e ajustes do posicionamento dos sensores, assim como, tornou possível o aperfeiçoamento do sistema de controle inteligente do veículo.



a) Comboio: acompanhamento visual b) Acompanhamento da marcação e saída da pista

Figura 1.23. Navegação baseada em Sistemas de Visão

Atualmente existe em andamento um projeto em parceria entre a EESC e o ICMC da USP de São Carlos, o projeto SENA [SENA 2009], que visa trabalhar com um veículo em escala real, doado por uma montadora brasileira, realizando testes de controle e navegação autônoma. Este veículo está em fase de adaptação para o uso de sensores embarcados, baseado em sensor laser e sistemas de visão. O uso de algoritmos de Navegação Visual [Heinen 2004, Righes 2004, Jung 2007, Bonato 2008] também tem sido pesquisados de modo a serem utilizados junto ao sistema de controle inteligente deste veículo autônomo (Figura 1.23).

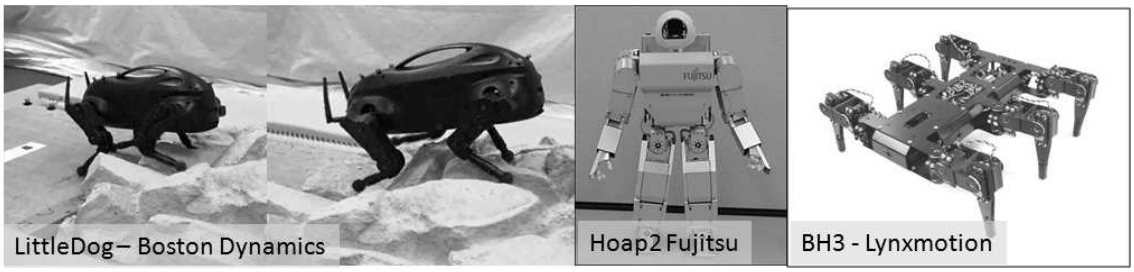


Figura 1.24. Robôs móveis articulados que caminham

1.6.4. Robôs Articulados com Pernas

O desenvolvimento de robôs com pernas (humanóides) ou com patas (imitando animais) é um tema de pesquisa que tem atraído a atenção de diversos grupos ao redor do mundo (ver Figura 1.24), uma vez que a locomoção através de dispositivos articulados permite que estes tipo de robôs móveis sejam melhor adaptados a certos ambientes, podendo por exemplo, subir e descer escadas e desníveis, e se locomover em terrenos bastante irregulares. Entretanto o projeto de robôs capazes de caminhar é uma tarefa bastante complexa, uma vez que é necessário levar em conta o equilíbrio do robô e a interação entre diversas forças que podem levar este a cair. O projeto de um robô articulado com patas deve levar em conta diferentes sensores, incluindo sensores para o *feed-back* tátil que indicam quando (e com que força) uma pata encosta no chão, e sistemas capazes de estimar a posição e o deslocamento do robô no espaço (e.g. acelerômetros, inclinômetros), capazes de estimar se o robô ameaça tombar. Além disto, o robô deve contar com um sistema de juntas e articulações, controladas de forma coordenada, que lhe permitam fazer os movimentos de suas pernas/patas e assim caminhar.

Para que fosse possível realizar um estudo referente ao projeto da configuração de HW e SW de um robô articulado com pernas, foi criado um sistema denominado de LEGGEN [Heinen 2006a]. Este sistema permite a simulação em um ambiente virtual 3D de robôs, no qual suas juntas e atuadores são definidos e controlados através do uso da biblioteca de simulação física ODE. Com isto é possível simular os robôs, considerando a sua colisão com outros objetos do ambiente, o contato das patas com o chão, o seu estado de equilíbrio e desequilíbrio, sendo sujeitos à gravidade e a outras forças presentes no ambiente, gerando assim uma simulação adequada para os fins de validação do projeto destes robôs móveis autônomos articulados. O sistema de controle inteligente foi implementado com o uso de Algoritmos Genéticos – AGs (uso da GALib), que possibilitam a otimização do conjunto de parâmetros usado no controle do caminhar. A Figura 1.25 apresenta exemplos de robôs quadrúpedes com diferentes graus de evolução do sistema de controle de seu caminhar.

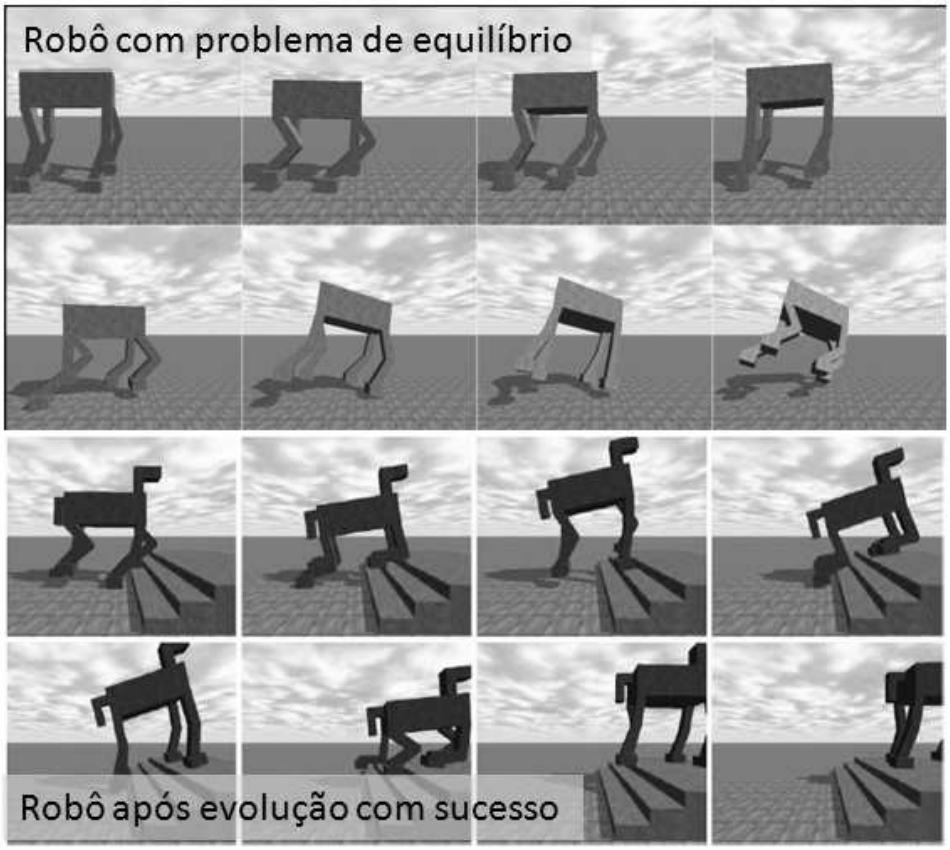


Figura 1.25. LEGGEN - Robôs Articulados que aprendem a caminhar

1.6.5. Esquadrões de Robôs

A realização de experimentos com um robô móvel usualmente já requer um tempo considerável para preparar e realizar os experimentos, e quando se trata de realizar experimentos com sistemas multirrobóticos, isto pode requerer uma complexidade ainda bem maior. Por isso, para economizar principalmente tempo, recursos humanos e recursos materiais, o uso de simulações tem se mostrado uma solução bastante viável no estudo e desenvolvimento de sistemas de controle inteligentes para sistemas multirrobóticos.

Um sistema multirrobótico tem um grande apelo, seja em aplicações de entretenimento como o futebol de robôs, ou em aplicações “sérias” de segurança e intervenção em situações de emergência (e.g. combate a incêndios, busca e resgate, ou aplicações militares). O sistema RoBombeiros [Pessin 2008] é um exemplo de uma aplicação de um sistema multirrobótico para uso no combate a incêndios florestais. Esta aplicação foi desenvolvida com a finalidade de criar um esquadrão de robôs autônomos capazes de definir e executar uma estratégia de ataque e extinção de um incêndio florestal. Os robôs atuam através do deslocamento até a zona afetada pelo foco do incêndio, realizando uma atuação coletiva a fim de criar um aceiro (retirada da vegetação criando um quebra-fogo) em torno da zona de propagação do incêndio.

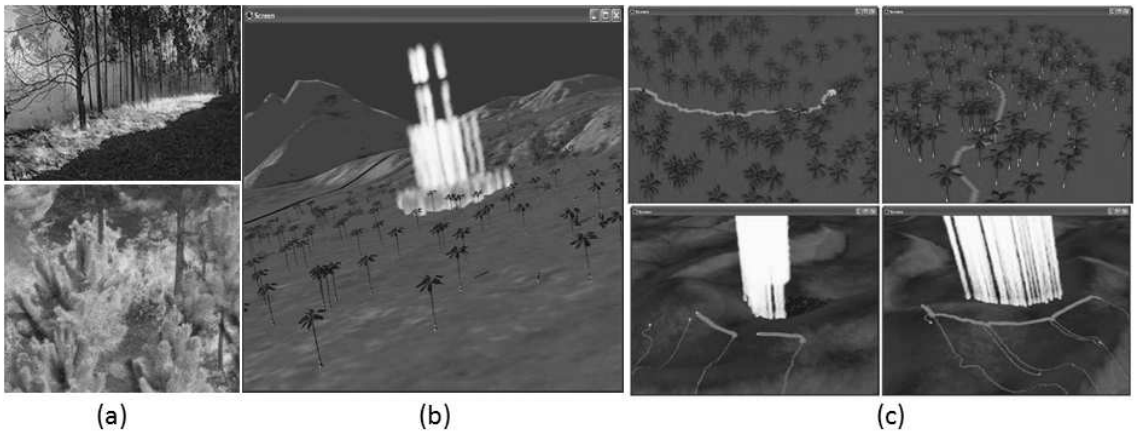


Figura 1.26. Esquadrão Inteligente de Robôs de Combate a Incêndios: a) Incêndio Real, b) Simulação do Incêndio, c) Deslocamento dos Robôs

O Projeto RoBombeiros envolveu a criação de um ambiente virtual 3D com simulação de terreno e vegetação, simulação da propagação do incêndio, e simulação física de robôs móveis autônomos. Para a criação do ambiente virtual foi desenvolvido um simulador usando as seguintes ferramentas: OSG (visualização 3D), Demeter (simulação de terrenos), ODE (simulação física), RNA (aprendizado de máquina para navegação autônoma) e AGs (otimização da estratégia de ação dos robôs-bombeiros). A Figura 1.26 apresenta alguns elementos do simulador do projeto RoBombeiros.

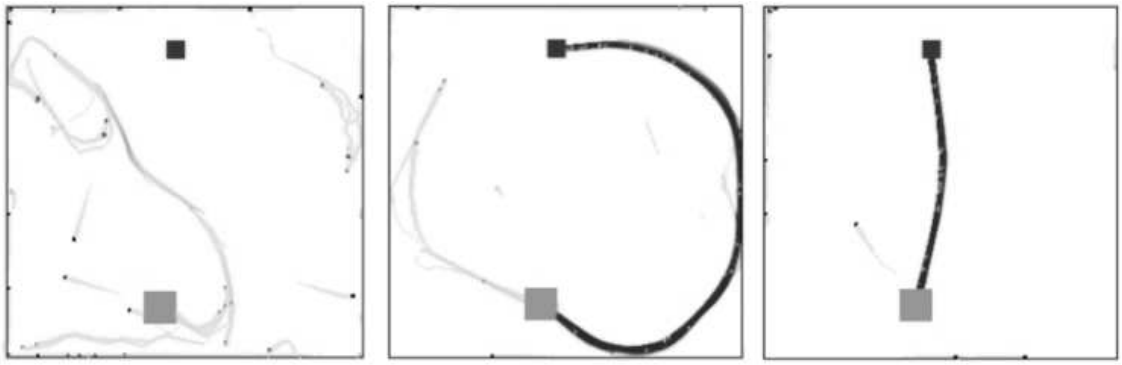


Figura 1.27. Robôs-Formiga: Busca e Otimização de Caminhos através da Inteligência de Enxames

Um outro projeto de sistemas multirrobóticos que explorou o comportamento coletivo dos robôs foi desenvolvido através do uso de um simulador de robôs-formiga [Costa 2007]. Este projeto demonstrou que é possível desenvolver comportamentos de inteligência de enxames baseados em colônias de formigas (ACO), que apresentam propriedades interessantes de cooperação e otimização na solução de problemas (exemplo: achar um caminho otimizado até um alvo ou “comida”). A Figura 1.27 apresenta um exemplo de otimização do caminho percorrido pelos robôs-formiga.

1.6.6. VANT – Veículo Aéreo Não Tripulado

O projeto ARARA (Aeronaves de Reconhecimento Assistidas por Rádio e Autônomas) [Neris 2001], está focado no desenvolvimento e uso de UAVs ou VANTs (Veículos Aéreos Não Tripulados), de escala reduzida, para monitoramento aéreo. Seu principal objetivo é o desenvolvimento de aeronaves a serem utilizadas na obtenção de fotografias aéreas, para monitoramento de áreas agrícolas e áreas sujeitas a problemas ambientais, bem como para aplicações de militares ou civis de segurança e defesa. O projeto ARARA foi criado a fim de desenvolver UAVs de pequeno porte que podem voar através de um controle manual remoto, ou também podem realizar missões pré-estabelecidas pelos usuários, através de um controle e navegação autônoma. O projeto ARARA está sendo desenvolvido junto ao Instituto de Ciências Matemáticas e de Computação (ICMC) da USP São Carlos em cooperação com a EMBRAPA Instrumentação Agropecuária de São Carlos.

Figura 1.28. VANT AGPlane – Decolagem e Estação de Controle

Através de parcerias com empresas, como a AGX, foi desenvolvido um VANT denominado de AGPlane [AGX 2009], capaz de navegar seguindo rotas pré-especificadas de modo autônomo, com o uso de um GPS, e realizando missões de imageamento aéreo. O projeto ARARA e do VANT AGPlane envolveram um estudo que vem sendo desenvolvido há diversos anos, englobando o projeto: estrutural e aerodinâmico da aeronave, do HW de controle embarcado, do sistema de imageamento aéreo, do sistema de comunicação com a base de controle em solo, do sistema de navegação autônoma por GPS. Este projeto vem sendo aperfeiçoado com o apoio do INCT-SEC (Instituto de Sistemas Embarcado Críticos), sendo um projeto estratégico que envolve questões de navegação autônoma inteligente, sistemas embarcados, de comunicação e de aplicações críticas (com riscos importantes a serem considerados em caso de acidentes e falhas).

A Figura 1.28 apresenta o VANT AGPlane da AGX e na Figura 1.29 podemos ver exemplos de imagens capturadas pelo VANT e seu processamento a fim de extrair informações relevantes das zonas de cultivo fotografadas.

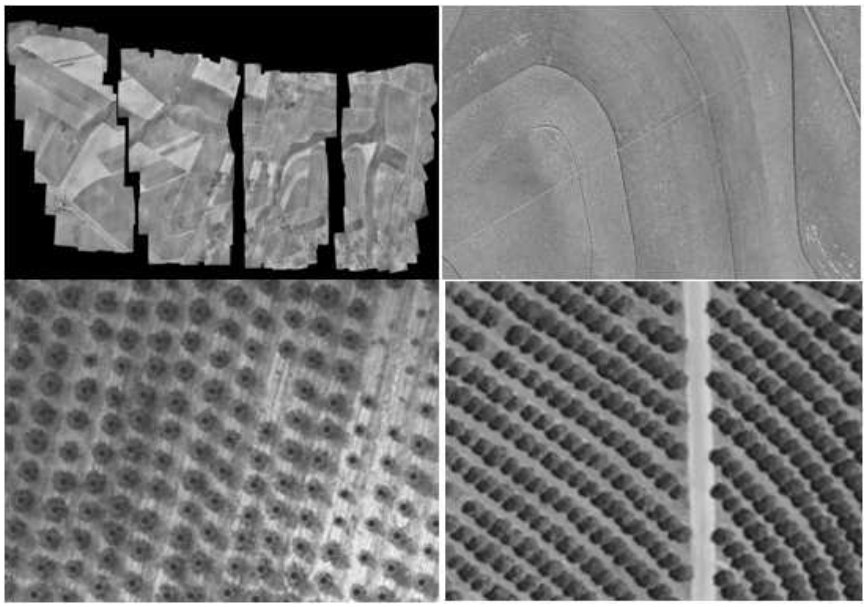


Figura 1.29. Imageamento Aéreo e Contagem de Árvores Plantadas

1.6.7. Cadeira de Rodas Robótica

O projeto de uma Cadeira de Rodas Robótica integra sensores de proximidade de obstáculos e um controlador inteligente em um robô móvel autônomo, configurando uma cadeira de rodas motorizada capaz de conduzir o deficiente autonomamente pelas dependências de um ambiente residencial. Isso é realizado sem necessidade de controle por parte do mesmo ou de terceiros. Este projeto vem viabilizar a independência de locomoção de deficientes com pouca coordenação motora, pois o robô cadeira é capaz de corrigir as manobras indicadas através de um joystick para desviar automaticamente de obstáculos e centralizar o veículo em corredores e espaços estreitos. As rotas percorridas poderão ser armazenadas em memória para serem refeitas posteriormente de maneira automática pelo robô cadeira.

Este projeto visa desenvolver um sistema de auxílio à mobilidade para deficientes físicos na forma de uma cadeira de rodas motorizada com capacidade de mapear rotas em um ambiente residencial para depois segui-las de maneira autônoma (ver Figura 1.30). O resultado pretendido visa atender às necessidades não somente dos portadores de deficiências física, mental e múltipla, mas também de pessoas idosas, através de uma interface amigável que procura corrigir automaticamente dificuldades de controle em tempo real. Para isso, pretende-se integrar dispositivos *ZigBee* [Gislaso 2006], que são componentes de comunicação via rádio similares aos *Bluetooth*, para a auto-localização da cadeira em um ambiente específico via triangulação da energia recebida. Isso permite que o usuário defina uma rota ao conduzir sua cadeira a um local específico, pois cada ponto do trajeto será armazenado na memória, para que a cadeira possa repetir a rota de maneira autônoma sempre que o usuário precisar. No caso de deficientes que não possam conduzir a cadeira através do joystick, outra pessoa pode conduzi-lo pela residência uma primeira vez, gerando uma rota que poderá ser acionada posteriormente pelo deficiente, contribuindo para sua independência. Com os sensores embarcados, é possível detectar o risco de colisão com obstáculos e pessoas e a cadeira é capaz de manobrar sozinha para evitar o choque.

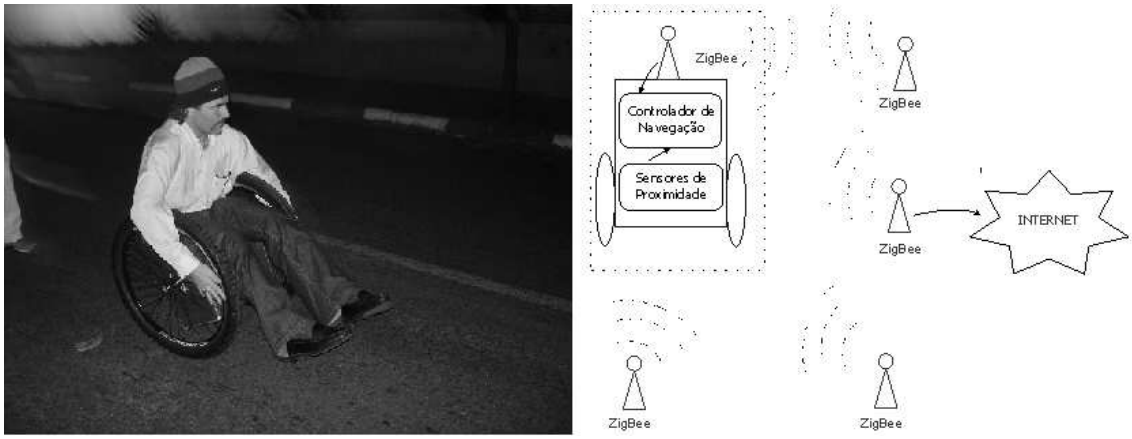


Figura 1.30. Test-drive do Sistema de Mobilidade Autônomo Pessoal, constituído de uma cadeira de rodas motorizada, controlador de navegação embarcado e bases fixas na residência para auto-localização.

No presente projeto, dispositivos dotados de ZigBee [Gislaso 2006] serão utilizados para a estimativa de sua posição relativa. A razão para utilizar uma rede ZigBee, também conhecida como IEEE 802.15.4, é que a mesma foi especialmente desenvolvida para a transmissão de dados entre sensores em uma solução simples e de baixo custo. ZigBee opera na faixa ISM, que não requer licença da Anatel, com capacidade de transmissão de 250 kbps. Para a implementação da rede ZigBee, é utilizado o kit de desenvolvimento CC2431 da Chipcon-Texas Instruments [Texas 2006], que possui módulos para a implementação de uma rede ZigBee e ainda hardware especial para a estimação da posição relativa do dispositivo, reduzindo o tempo e custos da implementação.

O usuário pode interagir com o sistema através de um manete de controle (joystick). Um módulo de mapeamento de rotas recebe informações da posição da cadeira do dispositivo ZigBee e armazena os pontos com as coordenadas da trajetória em um banco de rotas, para que o módulo de planejamento de trajetória possa repeti-la posteriormente a partir da informação de posição recebida do ZigBee. O usuário poderá controlar o sistema proposto da seguinte maneira:

- Com o uso da manopla de controle será possível dirigir a cadeira através do ambiente e salvar a sua trajetória que poderá ser utilizada posteriormente como rota;
- Através da utilização de indicadores de posicionamento zigbee pode-se configurar a cadeira para seguir um conjunto de rotas previamente armazenado de forma automática;
- O usuário pode executar um conjunto de movimentos que serão gravados para posterior execução automática;
- Seleção de Destino - o usuário identifica um destino e o sistema se encarrega de gerar uma rota para alcançá-lo;
- Após a configuração de um conjunto de movimentos, estes poderão ser executados como instruções de alto nível.

1.6.8. Outras Aplicações de Sucesso de Robôs Móveis

Os robôs móveis receberam uma grande atenção da mídia e do público em geral a partir de aplicações de grande sucesso, como foi o caso dos *Rovers* enviados a Marte (*Sojourner, Spirit e Opportunity rovers*) [Bajracharya 2008] e dos desafios do *DARPA, o Grand Challenge* de 2005 e o *Urban Challenge* de 2007 [Thrun 2006, Urmson 2008, Gibbs 2006] (ver Figura 1.31). Estas aplicações possuem em comum o fato de implementarem sistemas de controle robótico inteligente e robusto que permitiram o sucesso da operação destes. A missão dos robôs Spirit e Opportunity completou cinco anos no solo de Marte em Janeiro de 2009, demonstrando o grande avanço alcançado pela robótica móvel, que possibilitou esta "sobrevivência" em um ambiente hostil por um período bastante longo.

Figura 1.31. Vencedores do DARPA Grand Challenge e Darpa Urban Challenge

No que diz respeito ao Darpa Grand Challenge de 2005, o desafio proposto era de realizar uma "grande corrida" pelo deserto americano, operando de modo completamente autônomo e atravessando grandes distâncias (132 milhas) de um percurso pré-definido em menos de 10 horas. Cinco dos veículos autônomos que participaram da competição completaram o percurso, sendo quatro deles em menos de 10 horas. O vencedor deste rally de 2005 foi o Stanley da equipe de Stanford [Thrun 2006]. Em 2007 o desafio incluiu a "interação" entre múltiplos veículos em um ambiente urbano, sujeito a restrições (regras de trânsito). A competição de 2007 exigiu um grande aperfeiçoamento dos sistemas de controle e navegação inteligente dos robôs, na qual a equipe Boss do CMU se classificou em primeiro lugar, tendo alcançado com sucesso as principais metas da competição: navegação autônoma em um ambiente urbano bem estruturado com múltiplos veículos e regras de condução.

1.7. Perspectivas Futuras na área de Robótica Móvel Inteligente

Este tutorial apresentou diversos conceitos, técnicas e exemplos de aplicações em robótica móvel. Podemos constatar através dos diversos exemplos apresentados, que existe uma tendência de se continuar a aperfeiçoar os sistemas de controle inteligente de robôs móveis, onde o foco atual das pesquisas tem sido no aumento da autonomia e da robustez destes sistemas de modo a poderem atuar em uma gama cada vez maior de tarefas e aplicações. O projeto integrado com a simulação virtual e a validação em robôs reais é uma tendência atual nesta área, na qual os sistemas de simulação vêm sendo bastante aperfeiçoados de forma a se aproximar cada vez mais os modelos reais e simulados dos robôs.

As perspectivas futuras nesta área são muito amplas, incluindo pesquisa sobre robôs humanóides, robôs aéreos (UAVs – Unmanned Aerial Vehicles), sistemas nano-robóticos (ex. simulador do projeto Claytronics - http://www.cs.cmu.edu/~claytronics/) e sistemas de exploração espacial. Também é preciso destacar a importância que tem tomado o desenvolvimento de sistemas multirrobóticos cooperativos, sendo esta também uma importante área de pesquisas para o futuro, onde temos algumas palavras-chave que devem ser destacadas: comunicação, interação, colaboração e inteligência coletiva. Podemos prever que a robótica móvel autônoma será responsável por uma nova revolução no mundo moderno, trazendo inúmeros benefícios e grandes desafios para um futuro próximo.

Agradecimentos

Os autores gostariam de agradecer pelas contribuições para este trabalho por parte dos membros do GPVA – Grupo de Pesquisa em Veículos Autônomos (Farlei Heinen, Milton Heinen, Christian Kelber, Gustavo Pessin e Cláudio Jung) e do Projeto SENA - Sistema Embarcado de Navegação Autônoma (Marcelo Becker, Glauco Caurin e Valdir Grassi Jr.). Também gostaríamos de agradecer pelo apoio e suporte financeiro dos órgãos de fomento a pesquisa, CNPq e FAPESP, bem como ao INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos) e ao LRM-ICMC-USP (Laboratório de Robótica Móvel do ICMC USP).

Referências Bibliográficas

AGX (2009). AGX - AGplane. http://www.agx.com.br/ (Acesso em 08/09/2009).

Asimov, I. (1968) “I, Robot” (a collection of short stories originally published between 1940 and 1950), Voyager - HarperCollinsPublishers, 249p.

Assumpção Jr., Jecel; Wolf, Denis F.; Marques, Eduardo (2007). "Towards a Hardware Accelerated Obstacle Avoidance System for Mobile Robots using Monocular Vision", In Proceedings of IEEE International Symposium on Industrial Embedded Systems - SIES, 2007.

Bajracharya, M.; Maimone, M.W.; Helmick, D.; (2008) “Autonomy for Mars Rovers: Past, Present, and Future”. IEEE Computer, Vol. 41, Issue 12, Dec. 2008 p.44-50.

Bekey, George A. (2005) “Autonomous Robots: From Biological Inspiration to Implementation and Control”. The MIT Press: Cambrigde, London. 563p.

Bonato, Vanderlei (2008) “Proposta de uma arquitetura de hardware em FPGA implementada para SLAM com multicâmeras aplicada à robótica móvel”. Tese de Doutorado – USP ICMC. São Carlos. Disponível on-line em http://www.teses.usp.br/

Bonato, Vanderlei ; Holanda, José Arnaldo de ; Marques, E. (2006). “An Embedded Multi-Camera System for Simultaneous Localization and Mapping”. In: International Workshop on Applied Reconfigurable Computing (ARC 2006), Delft.

Bonato, Vanderlei; Peron, Rafael; Wolf, Denis F.; Holanda, Arnaldo M.; Marques, Eduardo; Cardoso, João M. P. (2007). "An FPGA implementation for a Kalman filter with application to mobile robotics". In Proceedings of IEEE International Symposium on Industrial Embedded Systems - SIES, 2007.

Boston Dynamics (2009). Robotic Products (BigDog and RHex) Acesso: 08/02/2009. http://www.bostondynamics.com/content/sec.php?section=robotics

Collett, Toby H.J.; MacDonald, Bruce A. and Gerkey, Brian P. (2005). "Player 2.0: Toward a Practical Robot Programming Framework". In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005), Sydney, Australia, December 2005.

Cominoli, Pascal (2005). “Development of a physical simulation of a real humanoid robot”. Diploma Work. BIRG, Logic Systems Laboratory (LSL) – EPFL, Swiss Federal Institute of Technology, Lausanne. http://birg.epfl.ch/page54249.html

Costa, Danilo. (2007) “Implementação de uma Colônia Artificial de Robôs-Formiga”. Dissertação de Mestrado em Ciências da Computação – ICMC. São Carlos, SP. [Disponível on-line em http://www.teses.usp.br/].

Dissanayake, G., Newman, P. Durrant-Whyte, H. F., Clark, S. and Csobra, M. (2001) ‘A solution to the simultaneous localisation and map building (SLAM) problem’. IEEE Trans. on Robotics and Automation, vol.17(3), pp.229-241.

Dudek, G.; Jenkin, M. (2000). “Computational Principles of Mobile Robotics”. Cambridge, London, UK: The MIT Press, 280 p.

Elfes, A. (1989) “Using occupancy grids for mobile robot perception and navigation”, Computer, vol. 22(6), pp.46-57.

Eliazar, A.I. and Parr, R. (2004), "DP-SLAM 2.0", In IEEE International Conference on Robotics and Automation, pp.1314- 1320.

Fox, D., Burgard, W., and Thrun, S. (1999) “Markov localization for reliable robot navigation and people detection”, In Modeling and Planning for Sensor-Based Intelligent Robot Systems. Springer Verlag, Berlin.

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

Fox, D., Burgard, W., Dellaert, F. and Thrun, S. (1999b) “Monte carlo localization: Efficient position estimation for mobile robots”, In Proceedings of the National Conference on Artificial Intelligence (AAAI), Orlando, FL, 1999. AAAI.

Gibbs, W. (2006) “Innovations from a robot rally”, Scientific American. vol.294, January 2006, p.64-71.

Gislaso, D. (2006). “ZigBee Wireless Networking”, CMP Books, 2006, 256 p.

Gutmann, J. S. and Konolige, K (1999), "Incremental mapping of large cyclic environments", IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp.318-325.

Haehnel, D. and Burgard, W. and Fox, D. and Thrun, S. (2003), "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.206-211.

Heinen, F. J. (2000). Robótica Autônoma: Integração entre Planificação e Comportamento Reativo. Série Produção Discente, Editora Unisinos. São Leopoldo, RS. Web: http://ncg.unisinos.br/robotica/ (acesso: fev. de 2009)

Heinen, F.; Kelber, C.; Jung, C.R.; Osório, F.S. (2004). "Navegação de Veículos de Carga Autônomos utilizando Visão Computacional com Algoritmo de Segmentação por Cores". In: IV Congresso Internacional de Automação, Sistemas e Instrumentação, 2004, São Paulo: ISA South America, 2004. v. 1. p. 1-10.

Heinen, F.; Osório, F. (2002). "HyCAR - A Robust Hybrid Control Architecture for Autonomous Robots". In: HIS 2002 – Proc. Of Hybrid Intelligent Systems, Santiago, Chile. Soft-Computing Systems - Amsterdam: IOS Press. V. 87, p. 830-840.

Heinen, Farlei (2002a). "Sistema de Controle Híbrido para Robôs Móveis Autônomos". Unisinos – PIPCA – Dissertação de Mestrado em Computação Aplicada. São Leopoldo, RS. Disponível on-line em: http://bdtd.unisinos.br/ Acesso: 08/02/2009.

Heinen, M. R.; Osório, F. S. (2006a). "Applying Genetic Algorithms to Control Gait of Physically Based Simulated Robots". In: IEEE Congress on Evolutionary Computation,. Proceedings of the WCCI - CEC. Vancouver: IEEE Press. v. 1. p. 6714-6721.

Heinen, M. R.; Osório, F. S.; Heinen, F.; Kelber, C. (2006) "SEVA3D: Using Artificial Neural Networks to Autonomous Vehicle Parking Control". In: IJCNN - IEEE Intenational Joint Conference on Neural Networks, 2006, Vancouver. Proceeding of the WCCI - IJCNN. Vancouver, Canadá : IEEE Press, v. 1. p. 9454-9461.

Heinen, M.R.; Osório, F.S.; Heinen, F.; Kelber, C. (2007). "SEVA3D: Autonomous Vehicles Parking Simulator in a three-dimensional environment". INFOCOMP. Journal of Computer Science, Lavras: UFLA. v. 6, p. 63-70, 2007.

Honda (2009). Humanoid robot ASIMO. http://world.honda.com/ASIMO/ ou http://world.honda.com/ASIMO/ (Acesso em 08/02/2009).

Huskvarna (2009). Automower (Robotic lawn mower). http://www.automower.com/ (Acesso em 08/02/2009).

iRobot (2009). Cleaning Robots (Roomba, Scooba, Dirt Dog, Verro) http://www.irobot.com/ ou http://en.wikipedia.org/wiki/Roomba (Acesso: 08/02/09).

Jones, S. D., Andersen, C. S., and Crowley, J. L. (1997). Appearance based processes for visual navigation. In: *IROS'97 – Proceedings of the IEEE Intelligent Robots and Systems Conference*, Vol. 2, pp. 551-557.

Jung, C. R.; Osório, F. S.; Kelber, C.; Heinen, F. (2005) "Computação embarcada: Projeto e implementação de veículos autônomos inteligentes", In: Anais do CSBC'05 XXIV Jornada de Atualização em Informática (JAI). São Leopoldo, RS: SBC, v. 1, p. 1358–1406. http://osorio.wait4.org/palestras/jai2005.html (Acesso em 08/02/2009).

Kelber, C.R.; Osório, F.S.; Jung, C.R.; Heinen, F.J.; Dreger, R.S.; Gules, R.; Mello Jr., C.D.; Silveira, M.A.; Schumacher, W.; (2003) "Tecnologias para Automação Veicular - Soluções em Mecatrônica e Sistemas de Apoio ao Motorista"; Revista Estudos Tecnológicos; São Leopoldo: Unisinos. Vol. XXIII, No. 24, p.37-47.

Kuipers, B and Byun, Y. T. (1991), “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations.”, Journal of Robotics and Autonomous Systems, vol.8 pp.47–63.

Latombe, J. (1991). Robot Motion Planning. Kluwer Academic Publisger, Boston, MA.

Leonard, J. J. and Durrant-White, H. F. (1991), “Mobile Robot Localization by tracking Geometric Beacons”, In IEEE Transactions on robotics and Automation, vol.7, pp-376-382.

Lester, Patrick (2005). “A* PathFinding for Beginners”. [online]. GameDev WebSite. http://www.gamedev.net/reference/articles/article2003.asp (Acesso em 08/02/2009).

Lidar – Wikipedia (2009). LIDAR Sensor (Light Detection and Ranging) - Laser Sick http://en.wikipedia.org/wiki/LIDAR (Acesso em 08/02/2009).

LRM – Laboratório de Robótica Móvel do ICMC USP. http://www.icmc.usp.br/~lrm/ (Acesso em 08/02/2009).

Mataric, M. J. (1990), “A distributed model for mobile robot environment-learning and navigation” Master’s thesis, MIT, Cambridge, MA.

Medeiros, Adelardo A.D. (1998). “A Survey of Control Architectures for Autonomous Mobile Robots”. JBCS - Journal of the Brazilian Computer Society, special issue on Robotics, vol. 4, n. 3.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002) “FastSLAM: A factored solution to the simultaneous localization and mapping problem”, In Proceedings of the AAAI National Conference on Artificial Intelligence.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003), “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”, In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI).

Neris, Luciano de Oliveira (2001). “Um piloto automático para as aeronaves do projeto ARARA”, USP – ICMC. Dissertação de Mestrado do PGCCMC. Orientador: Onofre Trindade Junior [disponível on-line em http://www.teses.usp.br/]. Dez. 2001.

Osório, F. S.; Musse, S. R.; Vieira, R.; Heinen, M. R.; Paiva, D. C. (2006). “Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation”. In: X. Fischer. (Org.). Proceedings of the Virtual Concept Conference 2006. 1 ed. Berlim: ESTIA - Virtual Concept - Springer Verlag, v. 1, p. 1-45.

Osório, F. S.; Pessin, G.; Ferreira, S.; Nonnenmacher, V. (2007). “Inteligência Artificial para Jogos: Agentes especiais com permissão para matar... e raciocinar”. (Tutorial). SBGAMES 2007 - SBC. São Leopoldo, RS.

Pessin, Gustavo (2008). “Evolução de Estratégias e Controle Inteligente em Sistemas Multi-Robóticos Robustos”. Dissertação de Mestrado – PPG em Computação Aplicada da Unisinos: São Leopoldo, RS. Disponível on-line: http://bdtd.unisinos.br/ ou http://pessin.googlepages.com/ (Acesso em 08/02/2009).

Pio, J. L. de Souza e Campos, M. F. M. (2003). “Navegação Robótica”. XXII Congresso da SBC – Anais JAI’03. Campinas, SP.

Pionneer (2009). MobileRobots Inc. http://www.mobilerobots.com/ (Acesso em 08/02/2009).

Righes, E.M.; (2004) "Processamento de Imagens para Navegação de Robôs Autônomos"; Relatório TCC - Informática; UNISINOS, São Leopoldo, RS. http://osorio.wait4.org/oldsite/alunos/tcc/righes-tc.pdf (Acesso em 08/02/2009).

Sahin, H.; Guvenc, L. (2007) "Household robotics: autonomous devices for vacuuming and lawn mowing", IEEE Control Systems Magazine, Vol.27, N.2, Apr 2007 p.20-96

SENA (2009). Projeto Sistema Embarcado de Navegação Autônoma (SENA) – Desenvolvido na USP de São Carlos pela EESC com colaboração do ICMC. http://www.eesc.usp.br/sena/ (Acesso em 08/02/2009).

Siegwart, Roland and Nourbakhsh, Illah R. (2004). "Introduction to Autonomous Mobile Robots". A Bradford Book, The MIT Press: Cambridge, London. 317p.

Smith, R., Self, M., and Cheeseman, P. (1990), "Estimating uncertain spatial relationships in robotics. In Autonomous Robot Vehicles, pp. 167-193.

Sony (2009). AIBO Entertainment Robots. http://support.sony-europe.com/aibo/ (Acesso em 08/02/2009).

Texas Instruments, CC2431DK – Development Kit User Manual – Revision 1.1, http://www.chipcon.com/files/CC2431DK%20Development%20Kit%20User%20Manual_1_11.pdf, acessado em 31/07/2006, 2006, 35 p.

Thrun, S. et al. (2006) "Stanley: The Robot that Won the DARPA Grand Challenge," Journal of Field Robotics, Vol. 23, No. 9, June 2006, p.661-692. http://robots.stanford.edu/papers.html (Acesso em 08/02/2009).

Thrun, S., Fox, D., and Burgard, W. (1998), "A probabilistic approach to concurrent mapping and localization for mobile robots", Machine Learning, vol. 31, pp.29-53.

Thrun, S.; Burgard, W.; Fox, D. (2005). Probabilistic Robotics. Cambridge: The MIT Press. 667p.

Urmson, Chris et al. (2008). "Autonomous driving in urban environments: Boss and the Urban Challenge". In: Journal of Field Robotics. Vol. 25 , Issue 8 (August 2008). Special Issue on the 2007 DARPA Urban Challenge, Part I. Pages 425-466.

Wolf, D. F. and Sukhatme, G. S. (2005) "Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments," In Autonomous Robots, v.19, n.1, pp. 53-65.

Wolf, Denis; Holanda, F. José A. de; Bonato, Vanderlei; Peron, Rafael; Marques, Eduardo (2007). "A FPGA-based mobile robot controller", In proceedings of IEEE Southern Conference on Programmable Logic, 2007.