Anna Helena Reali Costa

Robótica Móvel Inteligente: progressos e desafios

Tese apresentada à Escola Politécnica da Universidade de São Paulo como requisito para obtenção do Título de Professor Livre Docente, junto ao Departamento de Engenharia de Computação e Sistemas Digitais.

Área:
Inteligência Artificial

São Paulo
2003

Anna Helena Reali Costa

Robótica Móvel Inteligente: progressos e desafios

Tese apresentada à Escola Politécnica da Universidade de São Paulo como requisito para obtenção do Título de Professor Livre Docente, junto ao Departamento de Engenharia de Computação e Sistemas Digitais.

Área:
Inteligência Artificial

São Paulo
2003

Aos meus pais e
às minhas filhas,
Regina e Marina.

Agradecimentos

Agradeço a todos que colaboraram, direta ou indiretamente, na execução deste trabalho.

Aos colegas, professores e funcionários da Escola Politécnica da Universidade de São Paulo e, em especial, do Departamento de Engenharia de Computação e Sistemas Digitais – PCS / EPUSP, meus sinceros agradecimentos pelo apoio e incentivo.

Esta tese é o resultado de anos de frutíferas e estimulantes cooperações com meus colegas e estudantes do Laboratório de Técnicas Inteligentes do PCS / EPUSP. Muitos, mais que colegas e estudantes, tornaram-se meus amigos e eu não poderia deixar de agradecê-los, principalmente aos inestimáveis Bianchi, Jaime, Rafael, Valguima, Márcio, Júlio, Alex e Jomi.

Gostaria ainda de externar o meu mais profundo agradecimento aos meus familiares, que sempre me acompanharam nesta jornada, fornecendo carinho, incentivo e apoio.

Em especial, agradeço a minhas queridas filhas pela compreensão por meus momentos de ausência.

Aos meus pais, minha gratidão e amor.

Resumo

O presente trabalho objetiva sistematizar a pesquisa e obra da autora na área de Robótica Móvel Inteligente, adotando uma visão integradora de paradigmas e tecnologias através do caminho do desenvolvimento teórico, simulação e experimentação. As contribuições da autora nesta área apóiam-se no seu esforço continuado de pesquisa e desenvolvimento ao longo dos últimos dez anos.

Em seu trabalho, a autora aborda um espectro concreto de temas relevantes, visando a pesquisa e desenvolvimento de um ou múltiplos robôs móveis completos atuando em um ambiente dinâmico e com exigências de atuação em tempo real. O interesse consiste em investigar e desenvolver agentes robóticos que sejam:

- Eficientes: capazes de atingir metas específicas sob restrições de tempo e recursos, através da integração de planejamento reativo e deliberativo de atividades;
- Autônomos: capazes de decidir e adaptar-se a mudanças, de aprender conforme adquiram mais experiência, através do refinamento de suas preferências de seleção de ações individuais e colaborativas;
- Sociais: capazes de interagir com outros agentes presentes no ambiente e, em particular, colaborar com outros agentes para resolver problemas e realizar tarefas muito complexas em relação às suas capacidades individuais.

Este texto não se destina somente a sistematizar idéias e realizações científicas, mas visa também servir como incubador de novas pesquisas e desenvolvimentos, capacitando a autora, colaboradores e orientados para a atuação nacional e internacional continuada e também para a transferência de novas tecnologias para a indústria e os serviços.

Abstract

This thesis aims at systemizing the research and work of the author in the area of Intelligent Mobile Robotics, addopting an integrated vision of paradigms and technologies by following a path of theoretical development, simulation and experimentation. The author's contributions in this area are supported by her continued reasearch and development efforts along the last ten years.

In her work, the author tackles a concrete spectrum of relevant themes, targetting research and development of complete mobile robots acting in dynamic environments and subject to real time actuation requirements. The interest is in investigating and developing robotic agents that are:

- Efficient: able to reach specific goals under resource and time constraints, through integration of reactive and deliberative planning of activities;
- Autonomous: capable of deciding and adapting to changes, learning with experience via refinement of individual and colective action selection preferences;
- Social: able to interact with other agents, particularly regarding collaboration for problem solving and task accomplishment that are too complex for individual capabilities.

This text is not destinated solely for systemization of ideas and scientific realizations. In addition, it also aims at serving as incubator of new research and developments, qualifying the author, collaborators and supervised students for continued national and international action, and also for transferring new technologies for the industrial and service sectors.

Sumário

Lista de Figuras

1 Introdução

Robótica Móvel Inteligente é uma fascinante área de pesquisa, por diversas razões. Primeiro, porque a transformação de um robô móvel, de um simples "computador sobre rodas" capaz de sensoriar algumas propriedades físicas do ambiente, em um agente inteligente apto a identificar atributos e propriedades, detectar padrões e regularidades, aprender com sua experiência, localizar, construir mapas e navegar por ambientes desconhecidos e não-estruturados, é uma tarefa que requer a aplicação simultânea e coordenada de diversas disciplinas. Engenharia e Ciência da Computação são disciplinas nucleares em robótica móvel, porém, quando questões sobre comportamento inteligente são formuladas, Inteligência Artificial, Ciências Cognitivas, Biologia, Psicologia, Sociologia e Filosofia oferecem valiosas hipóteses e respostas. Do mesmo modo, a Matemática e a Física também contribuem de forma substancial, em diversos itens. Assim, a Robótica Móvel Inteligente ocupa um lugar especial pela possibilidade de conjugar esforços em diversas disciplinas, caracterizando-se como uma área multidisciplinar de pesquisa.

Uma segunda razão para seu fascínio decorre das aplicações comerciais da Robótica Móvel Inteligente, que são diversas e têm atraído crescente interesse, uma vez que possibilita a atuação não-supervisionada de máquinas em tarefas complexas que requerem interação com o meio físico. Busca e salvamento de sobreviventes em situações de catástrofe, detecção de fogos em florestas, transporte de objetos, vigilância e limpeza de grandes áreas (zona econômica exclusiva, tráfego de automóvel, estacionamentos, praias, etc), exploração subaquática ou planetária, navegação de veículos autônomos, movimentação coordenada de um conjunto de veículos autônomos, aplicações em agricultura (colheita autônoma, tratamento da terra, semeadura, etc), constituem alguns exemplos de aplicação de robôs móveis. A adoção comercial de robôs móveis autônomos tem ganhado recentemente um novo ímpeto através do aumento da sua taxa de crescimento e

de desenvolvimento, graças à disponibilidade de simuladores e sistemas de hardware a preços razoáveis. Acompanhando este desenvolvimento, ficam patentes os desafios científicos e tecnológicos associados à pesquisa e desenvolvimentos na área da Robótica Móvel Inteligente.

Finalmente, uma terceira razão de fascínio consiste no fato de robôs móveis serem, atualmente, a melhor aproximação de máquinas que imitam e emulam seres vivos. Assim, a Robótica Móvel Inteligente contribui grandemente em pesquisas em vida artificial, motivadas pela questão de o que seria vida e como entendê-la. Percepção e ação são fortemente acopladas em seres vivos. Para ver, animais executam movimentos específicos de olhos e cabeça; para interagir com o ambiente, antecipam o resultado de suas ações e prevêem o comportamento de outros seres e objetos; para comunicar, alteram o ambiente, deixando marcas e rastros. Deste modo, robôs móveis autônomos constituem uma plataforma bastante adequada para investigar comportamentos inteligentes.

1.1 Objetivo

O presente trabalho objetiva sistematizar a pesquisa e obra da autora na área de Robótica Móvel Inteligente, adotando uma visão integradora de paradigmas e tecnologias através do caminho do desenvolvimento teórico, simulação e experimentação. As contribuições da autora nesta área apóiam-se no seu esforço continuado de pesquisa e desenvolvimento ao longo dos últimos dez anos.

Em seu trabalho, a autora aborda um espectro concreto de temas relevantes, visando a pesquisa e desenvolvimento de um ou múltiplos robôs móveis completos atuando em um ambiente dinâmico e com exigências de atuação em tempo real. O interesse consiste em investigar e desenvolver agentes robóticos que sejam:

- Eficientes: capazes de atingir metas específicas sob restrições de tempo e recursos, através da integração de planejamento reativo e deliberativo de atividades;
- Autônomos: capazes de decidir e adaptar-se a mudanças, de aprender conforme adquiram mais experiência, através do refinamento de suas preferências de seleção de ações individuais e colaborativas;

- Sociais: capazes de interagir com outros agentes presentes no ambiente e, em particular, colaborar com outros agentes para resolver problemas e realizar tarefas muito complexas em relação às suas capacidades individuais.

Este texto não se destina somente a sistematizar idéias e realizações científicas, mas visa também servir como incubador de novas pesquisas e desenvolvimentos, capacitando a autora, colaboradores e orientados para a atuação nacional e internacional continuada e também para a transferência de novas tecnologias para a indústria e os serviços.

1.2 Histórico

O trabalho de pesquisa da autora resulta da interação entre importantes tópicos de pesquisas: Visão Computacional, Aprendizado Autônomo de Máquina e Robótica Inteligente.

Visão Computacional tem por objetivo estudar os princípios que norteiam a automação da percepção visual, assim como desenvolver algoritmos e métodos para dotar máquinas da capacidade de interpretação visual de imagens adquiridas por meio de câmeras. Esta linha aborda principalmente os conceitos fundamentais envolvidos na determinação da estrutura e propriedades do mundo tridimensional, possivelmente dinâmico, a partir de suas imagens bidimensionais coloridas e monocromáticas. Muitos desenvolvimentos e resultados são aplicados à Visão Robótica.

A linha de pesquisa Aprendizado Autônomo de Máquina objetiva dotar agentes da capacidade de adaptação às mudanças no ambiente, nas tarefas e na sua organização, visando melhorar seu desempenho através da aquisição de novos conhecimentos e da generalização ou especialização de conceitos e comportamentos pré-existentes.

Finalmente, a Robótica Inteligente busca integrar de modo consistente (teoria e prática) a percepção, raciocínio e ação de robôs, de modo a dotá-los com funções cognitivas de alto nível para que possam sensoriar, raciocinar e agir em ambientes dinâmicos, desconhecidos e imprevisíveis. Especial atenção é dada à Robótica Móvel.

A atuação da autora em visão computacional começou em 1988, no reconhecimento de objetos bidimensionais (2D), inicialmente tratando somente objetos rígidos isolados em imagens binárias de intensidade luminosa. Foi desenvolvido um sistema de visão binária, posteriormente integrado à Célula Flexível de Montagem da Escola Politécnica (RILLO; COSTA; COSTA, 1992a; RILLO; COSTA; COSTA, 1992b), composta por dois manipuladores, controladores programáveis (RILLO, 1983), câmeras e um sistema de controle e planejamento de atividades (veja Apêndice I para uma descrição da composição da célula). Esta célula executa tarefas simples de montagem e as peças são alimentadas por meio de uma esteira (peças isoladas e contrastantes com a esteira). O reconhecimento no sistema de visão binária se dá por meio da modelagem, extração e análise de atributos globais dos objetos.

Para contornar a restrição de isolamento entre as peças, nova abordagem foi proposta e implementada para permitir não somente o reconhecimento de objetos isolados numa cena, mas também de peças que se tocam ou superposicionam, desde que partes consideráveis se mantivessem visíveis (COSTA, 1990a; COSTA, 1990b). Uma vez que modelos globais não têm utilidade quando comparados a vistas parciais dos objetos, modelos contendo atributos locais foram propostos, utilizando segmentos do contorno das vistas dos objetos como atributo para efetuar a correspondência. Numa fase de treinamento onde os objetos que compõem a base de modelos do sistema são apresentados, os modelos são automaticamente construídos e atributos definidos. A abordagem integrada, permitindo reconhecimento de peças isoladas e parcialmente oclusas, foi apresentada como dissertação de mestrado (COSTA, 1989).

No entanto, visando ainda aumentar o domínio de aplicação do sistema de visão, novo passo foi dado no sentido de tratar também peças tridimensionais (3D), permitindo tarefas de inspeção da montagem. O sistema RECTRI foi desenvolvido (COSTA, 1992a; COSTA, 1992b; COSTA, 1992c; COSTA, 1992d; COSTA, 1993; COSTA, 1994a; COSTA, 1995), onde o reconhecimento se dá através da busca por uma correspondência espacial entre modelos e atributos 2D extraídos e perceptualmente agrupados na imagem. A definição destes atributos é feita de modo autônomo pelo sistema numa fase de treinamento, onde modelos CAD dos objetos 3D são fornecidos e uma estratégia de busca por reconhecimento é criada. Este trabalho resultou no doutorado da autora (COSTA, 1994b).

Trabalhos posteriores da autora já se preocuparam com a integração de sistemas de percepção visual a outros módulos da cognição, buscando uma abordagem integrada, necessária para o projeto de robôs móveis e agentes artificiais. Paradigmas para visão computacional foram estudados (ALOIMONOS, 1994; JOLION, 1994; MARR, 1982; TARR; BLACK, 1994a; TARR; BLACK, 1994b), resultando numa proposta de integração do paradigma propositado (*purposive vision*) a planejamentos reativos e deliberativos através de arquiteturas multi-agentes (COSTA, 1996; BIANCHI; COSTA, 1996). A arquitetura VIBRA – *Vision Based Reactive Architecture* foi um resultado deste trabalho (BIANCHI; COSTA, 1997a; BIANCHI; COSTA, 1997b; BIANCHI; COSTA, 1997c; COSTA; BARROS; BIANCHI, 1998).

VIBRA pode ser vista como uma sociedade de agentes autônomos, cada um responsável por um comportamento específico, que cooperam entre si para conduzir a sociedade no processo de atingir suas metas. VIBRA tem a capacidade de processar metas definidas assincronamente, ditadas por dados sensoriais advindos da visão, definir prioridades, suspender temporariamente ações de menor prioridade e intercalar comportamentos compatíveis. VIBRA foi aplicada ao ambiente robótico da referida célula de montagem, utilizando um dos manipuladores e um sistema de visão computacional, onde tarefas de montagem e de remoção de objetos indesejados na área de montagem são executadas, ao mesmo tempo em que devem ser evitadas colisões entre o manipulador e humanos e/ou outros manipuladores (COSTA; BARROS; BIANCHI, 1998).

No entanto, a arquitetura VIBRA propõe uma estrutura fixa, pré-definida, de autoridade entre os agentes, resultando num desempenho ineficiente, pois um agente com maior prioridade de ação irá sempre realizar suas operações antes das atuações de um outro agente – inferior – mesmo que as condições do domínio no momento, dadas pela percepção, mostrassem ser desnecessárias estas operações. Buscando melhorar esta solução, as arquiteturas híbridas L-VIBRA (COSTA; BIANCHI, 2000) e ANT-VIBRA (BIANCHI; COSTA, 2002a; BIANCHI; COSTA, 2002b; BIANCHI; COSTA, 2002c) foram propostas e implementadas. Estas arquiteturas utilizam aprendizagem por reforço – seqüencial, na L-VIBRA e distribuída, na ANT-VIBRA – para determinar a melhor ordem de execução da tarefa pelos agentes, de forma dinâmica, em função das informações adquiridas pelo sistema de visão computacional da célula de montagem.

Na busca por um domínio de aplicação mais adequado ao estudo das interações entre agentes, a autora iniciou investigações na área de futebol de micro-robôs, onde coordenou o desenvolvimento do time FUTEPOLI (BIANCHI; COSTA, 2000) e do time GUARANÁ (COSTA, 1999a; COSTA, 1999b; COSTA; PEGORARO, 2000), sendo que este último conquistou o título de vice-campeão mundial na categoria de micro-robôs do *Micro-Robot Soccer Tournament* – MIROSOT, nos jogos da *Federation of International Robot-Soccer Association* – FIRA, em Paris, França, julho de 1998.

Futebol de micro-robôs tem se estabelecido ultimamente como um excelente ambiente de desenvolvimento e teste de pesquisa em inteligência artificial distribuída (IAD – Sistemas Multi Agentes), onde jogadores de um time precisam interagir e cooperar entre si e com o ambiente de modo a competir com um time adversário, de forma totalmente autônoma. A natureza do jogo exige que os jogadores tenham rápida percepção (visual) do ambiente (BIANCHI; COSTA, 2002d), rápido planejamento da tática e estratégia a serem empregadas, rápidas ações, controle preciso e apropriada cooperação entre jogadores.

Em oposição à Robótica Industrial (ROMANO, 2002), constituída basicamente por manipuladores (estacionários) ou AGVs (*Automated Guided Vehicles*), que atuam em ambientes fortemente controlados e estruturados, a Robótica Móvel Inteligente (RIBEIRO; COSTA; ROMERO, 2001) objetiva investigar e construir robôs móveis que possam atuar em ambientes do dia-a-dia – hospitais, escritórios, escolas, parques, museus, supermercados, etc – com todo dinamismo, variações e incertezas a que estes ambientes estão expostos.

Para que um robô móvel possa lidar com incertezas, ambigüidades, dinamismo, contradições e informações ruidosas, ele deve possuir um alto grau de autonomia, inteligência, capacidade de aprender e de se adaptar a mudanças. Dois paradigmas básicos sobressaem para organizar a inteligência em robôs móveis: funcional e comportamental. No paradigma funcional os dados sensoriais são filtrados e processados seqüencialmente através de uma série de módulos, arranjados segundo uma hierarquia, até que um comando motor surja no final da seqüência. No paradigma comportamental, por outro lado, propõe-se uma decomposição em diversos comportamentos, simples e independentes, cada um deles com acesso aos sensores e atuadores do robô, estabelecendo uma forte ligação entre percepção e

ação. A interação entre estes diversos comportamentos conduz aos chamados efeitos sinérgicos e inteligência emergente.

Para tarefas como seguir trajetórias e evitar colisões, o paradigma comportamental se mostra bastante eficiente (BIANCHI; SIMÕES; COSTA, 2001), possibilitando que arquiteturas apropriadas sejam usadas com sucesso no projeto e desenvolvimento de robôs móveis. A autora tem explorado este tema de pesquisa, desenvolvendo robôs que usam tanto arquiteturas com coordenação competitiva dos comportamentos (BRAMBILA; COSTA, 2002) quanto arquiteturas com coordenação cooperativa dos comportamentos (PACHECO; COSTA, 2002a; PACHECO; COSTA, 2002b).

Uma tarefa fundamental da Robótica Móvel consiste na navegação. Navegação envolve localização, planejamento de trajetórias e construção e uso de mapas do ambiente no qual o robô se movimenta e atua. Nesta área, pesquisas e desenvolvimentos têm sido conduzidos pela autora e seus orientados na (i) construção automática de mapas de ambientes desconhecidos (internos) usando robôs Pioneer 2DX e sonares (BARRA; DOMENECCI; COSTA, 2002), (ii) numa abordagem probabilística para determinar a localização do robô em um ambiente interno (KAWAI; COSTA, 2002) e (iii) na navegação baseada em marcos visuais de referência do mundo real, aprendidos autonomamente (MARTINEZ; COSTA, 2002).

Um robô móvel de fato autônomo precisa ter a competência de lidar com um mundo incerto, dinâmico, ambíguo. Ele precisa ser capaz de aprender através de suas interações com o mundo e com outros agentes, avaliando eventos com relação às suas metas e alterando seu comportamento sempre que necessário. Esta capacidade de aprender tem sido investigada pela autora e seu grupo em diversos aspectos.

Na percepção visual, uma tarefa extremamente importante e difícil é a segmentação de imagens, que depende de uma boa classificação de seus elementos constituintes em unidades de interesse. Em aplicações de robótica móvel, a cor de objetos é uma propriedade muito útil e bastante explorada. Desta forma, a autora tem conduzido pesquisas e desenvolvimentos no uso de técnicas de aprendizagem e adaptação para classificação de cores de interesse no mundo sensoriado por sistemas de visão computacional dos robôs. Nesta linha, tem explorado o uso de redes

neurais artificiais (SIMÕES; COSTA, 2000a; SIMÕES; COSTA, 2000b; SIMÕES; COSTA, 2000c; SIMÕES; COSTA, 2001; SIMÕES; COSTA, 2002; SIMÕES, 2002; SIMÕES; COSTA, 2003) e técnicas de classificação nebulosa (*fuzzy*) não supervisionada (BONVENTI Jr.; COSTA, 2000b; BONVENTI Jr.; COSTA, 2000a; BONVENTI Jr.; COSTA, 2002) na tarefa de segmentação de imagens baseada em cores.

O uso de aprendizado por reforço é bastante conveniente em Robótica Móvel, uma vez que o robô interage repetidas vezes com o ambiente, possibilitando o uso de um sinal de reforço (captado pelos sensores do robô) para indicar a conveniência ou não de suas atuações. Este tema tem sido bastante explorado pela autora, em diversas aplicações, tais como: (i) na determinação de políticas adequadas de navegação (PEGORARO, 2001; PEGORARO; COSTA, 2000), (ii) na definição de atuações em jogos de futebol (SCÁRDUA; COSTA; CRUZ, 1999a; SCÁRDUA; COSTA; CRUZ, 1999b; SCÁRDUA; COSTA; CRUZ, 2000; PEGORARO; COSTA; RIBEIRO, 2001a; PEGORARO; COSTA; RIBEIRO, 2001b), (iii) na interação entre robôs, determinando momentos adequados para comunicação explícita entre eles (COSTA; VELOSO, 2000), (iv) na coordenação de robôs (BIANCHI; COSTA, 2002a; BIANCHI; COSTA, 2002b; BIANCHI; COSTA, 2002c) visando políticas adequadas de cooperação entre eles para atingir metas comuns – ganhar jogos de futebol (PEGORARO, 2001; PEGORARO; COSTA, 2000; PEGORARO; COSTA; RIBEIRO, 2001a; PEGORARO; COSTA; RIBEIRO, 2001b; RIBEIRO; PEGORARO; COSTA, 2002), e (iv) na combinação de redes neurais e aprendizado por reforço na determinação de uma política precisa de controle – no controle de um descarregador de navios (SCÁRDUA; CRUZ; COSTA, 2000; SCÁRDUA; CRUZ; COSTA, 2003b; SCÁRDUA; CRUZ; COSTA, 2003a).

Decorrente deste trabalho em aprendizado autônomo, foi proposto um novo algoritmo de aprendizado por reforço para uso em jogos de dois jogadores, com soma zero, denominado Minimax-QS (PEGORARO, 2001; PEGORARO; COSTA, 2000; PEGORARO; COSTA; RIBEIRO, 2001a; PEGORARO; COSTA; RIBEIRO, 2001b; RIBEIRO; PEGORARO; COSTA, 2002), com prova de convergência para valores ótimos, sob condições adequadas (RIBEIRO; PEGORARO; COSTA, 2002).

Atualmente, esforços estão sendo feitos na aplicação destas técnicas em robôs reais, testadas em ambientes simulados, na proposta e desenvolvimento de algoritmos adequados (e adaptáveis) para a percepção visual (MAIA Jr.; COSTA,

2002; SEIXAS; COSTA, 2002), e na formulação de algoritmos que visem acelerar o aprendizado autônomo de agentes por meio de abstração estrutural, onde decomposições são consideradas como forma de especificar representações multi-resoluções de espaço.

Atualmente, as investigações da autora e seu grupo estão sendo conduzidas no âmbito dos seguintes projetos de pesquisa:

MAPPEL *Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning.* Projeto binacional de pesquisa coordenado pela autora, com financiamento do National Science Foundation - NSF e do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq MAPPEL (processo ProTeM CC No. 680033/99-8), envolvendo o LTI - PCS/EPUSP (Brasil), Instituto Tecnológico de Aeronáutica - ITA (Brasil), Instituto de Matemática e Estatísticas da USP (Brasil), Instituto de Computação da UNICAMP (Brasil) e CS/CMU (Estados Unidos).

MultiBot Modelos e Técnicas para Robôs Móveis Inteligentes Aplicados a Tarefas Individuais e Cooperativas. Projeto binacional de pesquisa coordenado pela autora, com financiamento da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES e do Gabinete de Relações Internacionais da Ciência e do Ensino Superior - GRICES (CAPES/GRICES Proc. No. 099/03), envolvendo as seguintes instituições: LTI- PCS/EPUSP (Brasil), ITA (Brasil), e o Instituto de Sistemas e Robótica do Instituto Superior Técnico da Universidade Técnica de Lisboa - ISR / IST (Portugal).

MANET *Manufacturing Automation Network.* Projeto vinculado ao Ministério de Ciência e Tecnologia, onde a autora pertence ao Grupo de Robótica e Automação, com financiamento da FINEP, dentro do programa RECOPE / PRODENGE (FINEP / RECOPE Proc. No. 77.97.0937.00).

AACROM Aprendizagem Autônoma da Coordenação de Comportamentos para Robôs Móveis. Projeto com financiamento da FAPESP (FAPESP Proc. N. 2001/14588-2), envolvendo a Divisão da Ciência da Computação (coordenação do projeto) e a Divisão de Engenharia Eletrônica, ambas do ITA, e o LTI - PCS / EPUSP.

1.3 Conteúdo e Organização

Este texto está dividido em duas partes:

Parte I - Conceitos Fundamentais onde os fundamentos básicos da Robótica Móvel Inteligente são descritos e formalizados;

Parte II - Pesquisas e Desenvolvimentos onde as contribuições desenvolvidas pela autora e seu grupo são descritas.

A Parte I é composta por sete capítulos. O Capítulo 2 define robôs móveis inteligentes e fornece, de modo resumido, as noções teóricas básicas associadas ao projeto e funcionamento de robôs móveis.

Os robôs móveis inteligentes interagem com o ambiente através de sensores e atuadores. Assim, o Capítulo 3 apresenta os sensores e atuadores mais comumentemente usados em Robótica Móvel.

No Capítulo 4 são descritos dois paradigmas da robótica móvel: funcional e comportamental. Este Capítulo também fornece os princípios básicos da interação do robô com o ambiente e conclui apresentando uma formalização para comportamentos dos robôs móveis e uma descrição das funções principais de coordenação de comportamentos.

No Capítulo 5 são apresentadas as arquiteturas de comportamentos: arquiteturas reativas, arquiteturas deliberativas e arquiteturas híbridas reativa/deliberativa. O estudo de arquiteturas mostra modos diferentes de uso de componentes e ferramentas associados a um paradigma, na construção de robôs móveis inteligentes.

No Capítulo 6 é apresentada uma habilidade essencial em Robótica Móvel: navegar, que envolve a capacidade de se movimentar pelo ambiente, visando atingir metas, desviando de obstáculos e preservando sua integridade física.

No Capítulo 7 é discutido como times de robôs móveis podem ser utilizados, descrevendo-se aplicações típicas para múltiplos robôs. São também definidas características dos times e descritas algumas arquiteturas representativas na área e suas aplicações.

Finalmente, o Capítulo 8 encerra a Parte I deste texto, abordando adaptação do robô através do aprendizado, de modo a capacitá-lo a melhorar seu desempenho para executar uma determinada tarefa. São descritas técnicas de aprendizado supervisionado, não supervisionado e de aprendizado por reforço, todos com potenciais aplicações em robótica móvel.

A Parte II é composta por cinco capítulos. No Capítulo 9 é apresentado de forma resumida as contribuições da autora em diversos tópicos da Robótica Móvel Inteligente e de outros domínios.

O Capítulo 10 descreve as pesquisas e desenvolvimentos realizados na área de percepção visual, muitos deles incorporando técnicas de aprendizado autônomo, e conclui descrevendo aplicações desenvolvidas.

As contribuições da autora na área de arquiteturas para Robótica Móvel encontram-se no Capítulo 11, onde arquiteturas reativas e híbridas foram investigadas.

O Capítulo 12 descreve as pesquisas e desenvolvimentos na área de navegação robótica, incluindo construção de mapas, localização e navegação baseada em marcos visuais, assim como no uso de aprendizado por reforço para definir o controle de atuação de robôs.

A descrição dos times de futebol de robôs FUTEPOLI e GUARANÁ, este último vice-campeão mundial na categoria MIROSOT nas competições da FIRA, em Paris, França, 1998, encontram-se no Capítulo 13. Finalizando este Capítulo, um algoritmo de aprendizado multi-agentes é apresentado, para domínios modelados como jogos de dois jogadores e soma zero.

Finalmente, as conclusões apresentadas no Capítulo 14 encerram este texto.

Parte I

Conceitos Fundamentais

2 Fundamentos da Robótica Móvel Inteligente

Robótica Móvel Inteligente é um campo distinto, tanto historicamente quanto em escopo, da Robótica Industrial. A Robótica Industrial concentra-se em questões de teoria de controle, particularmente aquelas relativas à cinemática e dinâmica de um robô. A Robótica Móvel Inteligente, por sua vez, concentra-se em como um robô móvel deve tratar de eventos imprevistos, em um mundo não-estruturado. O projeto de um robô móvel, inteligente e autônomo, deve considerar como o robô irá representar seu conhecimento sobre o mundo, quais mecanismos serão usados para perceber, sensoriar o mundo real, se entenderá linguagem natural, se ele poderá aprender tarefas e habilidades, que tipo de planejamento e resolução de problemas terá, quanto de inferência será necessária.

Neste texto, define-se um Robô Móvel Inteligente como um agente inteligente, artificial, autônomo, com capacidade de locomoção, imerso no mundo físico real. *Agente inteligente* por decidir de forma racional; *artificial* por ser máquina e não uma entidade criada pela natureza; *autônomo* por ser capaz de decidir por si só, de se auto-governar, de atuar no ambiente de forma propositada, não passiva, de se adaptar a mudanças ocorridas, no ambiente ou em si próprio, e continuar a atingir suas metas; *com capacidade de locomoção* por poderem se mover no ambiente (RIBEIRO; COSTA; ROMERO, 2001; MURPHY, 2000).

2.1 Agente, Tarefa e Ambiente

O comportamento de um robô não pode ser avaliado independentemente da tarefa que está executando e do ambiente no qual está imerso (veja Figura 2.1).

O funcionamento e operação de um robô são definidos pelo próprio compor-

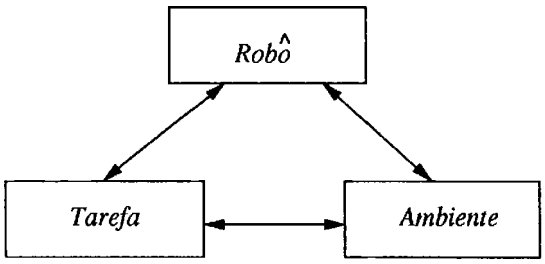


Figura 2.1: Interdependência entre robô, tarefa e ambiente.

tamento do robô, em um ambiente específico, considerando uma tarefa específica. Isto significa, conseqüentemente, que um robô de propósito geral (no sentido amplo) não pode existir. Somente a descrição simultânea do agente, tarefa e ambiente descreve um agente - ser vivo ou artificial - completamente.

2.2 Interação do Robô com o Ambiente

Um robô móvel inteligente, ou simplesmente robô móvel, pode extrair informações do ambiente no qual está imerso e usar seu conhecimento sobre um determinado domínio para realizar tarefas através da execução de ciclos de interação com o ambiente, da seguinte maneira (veja a Figura 2.2):

1. *percepção* das situações e condições dinâmicas no ambiente, por meio de sensores;

2. *raciocínio* para interpretar percepções, resolver problemas, realizar inferências e planejar as ações a executar.

3. *atuação* para executar uma ação, através dos seus atuadores, afetando as condições no ambiente, produzindo novas situações.

O projeto de robôs móveis que possam interagir com o mundo real, de maneira robusta e versátil, é, na verdade, muito mais difícil do que inicialmente pensado (BROOKS, 1991), uma vez que o mundo real apresenta uma natureza ruidosa, imprevisível e dinâmica. No entanto, alguns problemas considerados intratáveis tornaram-se consideravelmente simplificados através da participação ativa do robô no ambiente, através de seu ciclo de percepção-ação.

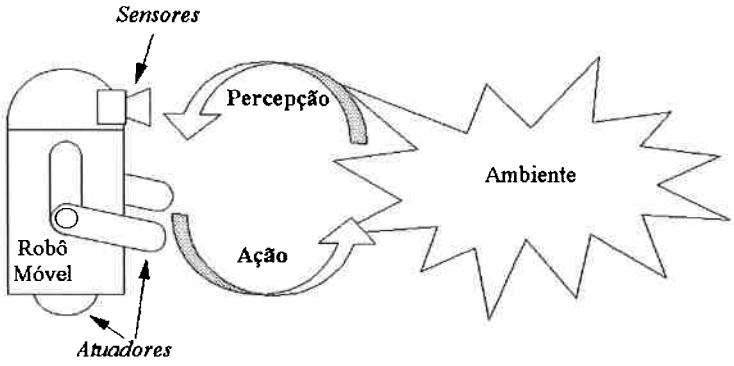


Figura 2.2: Interação do robô com o ambiente através de percepção (realizada pelos sensores) e ação (executada pelos atuadores).

O problema central de qualquer agente autônomo é a geração de comportamento apropriado, no tempo apropriado, uma vez que tanto seu estado interno quanto as situações externas (ambiente) mudam continuamente. Para um agente autônomo, ações devem sempre ter precedência sobre outras atividades. Isto não significa que um agente autônomo deva ser puramente reativo, mas sim que ele deve ter a habilidade de flexivelmente combinar as circunstâncias imediatas com suas metas de longa duração, de tal forma que ajuste continuamente seu comportamento de modo apropriado para ambas.

O fato de ser inteligente indica que realiza uma decisão de forma racional, isto é, que seleciona as ações que maximizam a probabilidade de alcançar o sucesso desejado na execução da tarefa a ele designada, dado o que foi por ele percebido do ambiente. O grau de sucesso de sua atuação pode ser definido por uma medida de desempenho, que o robô visa maximizar. Assim, considere que seja designada a um robô móvel a tarefa de percorrer, sem colisões, o trajeto de um ponto a outro de uma fábrica. Se o robô conseguir executar esta tarefa com sucesso, conduzindo-se até o local desejado e desviando dos eventuais obstáculos, diz-se que ele agiu racionalmente e que, portanto, é inteligente. Se o tempo de percurso for utilizado como medida de desempenho, o grau de sucesso na execução da tarefa será tanto maior quanto menor for o tempo que ele levar para realizar o percurso, evitando colisões.

2.3 Paradigmas

Um paradigma é uma filosofia, um conjunto de hipóteses ou técnicas que caracterizam uma abordagem para uma classe de problemas (MURPHY, 2000). Existem dois paradigmas básicos para organizar a inteligência em robôs: funcional e comportamental. Estes paradigmas são descritos de duas formas:

- Pela relação entre três primitivas: PERCEPÇÃO, PLANEJAMENTO, ATUAÇÃO. A primitiva PERCEPÇÃO possui entrada de dados por meio dos sensores e transforma estes em informação para outra função; o PLANEJAMENTO utiliza informação (dos sensores ou conhecimento interno) para produzir uma ou mais tarefas a serem executadas, envolve deliberação e conhecimento sobre o mundo; a ATUAÇÃO traduz as tarefas a serem executadas pelo robô em termos de comandos de baixo nível. Os dois paradigmas básicos são definidos em termos das três primitivas na Figura 2.3. O paradigma funcional processa os dados dos sensores de forma hierárquica, seqüencial, até determinar a ação motora a ser executada pelos atuadores, através de ciclos de PERCEPÇÃO - PLANEJAMENTO - ATUAÇÃO; o paradigma comportamental define comportamentos como mapeamentos de PERCEPÇÃO - ATUAÇÃO, muitas vezes reativos, e a inteligência emerge por meio da interação entre os comportamentos.

- Pelo modo com que o dado sensorial é processado e distribuído. No paradigma comportamental, a informação sensorial é processada de forma específica e dedicada para algumas funções do robô – os comportamentos – sendo, assim, de processamento local, em cada função. No paradigma funcional, a informação sensorial é processada e armazenada num único modelo global do mundo, para então suas partes serem distribuídas para outras funções.

Devido à combinação da natureza imprevisível do mundo real com as restrições de tempo freqüentemente impostas pela tarefa a ser executada, grande parte das pesquisas realizadas em robôs móveis tem sido restrita a desenvolvimentos de sistemas onde as ações são completamente determinadas pelas situações imediatas do ambiente, detectadas pelos sensores do robô, seguindo o paradigma comportamental, de forma reativa (BROOKS, 1991; KROGH, 1984).

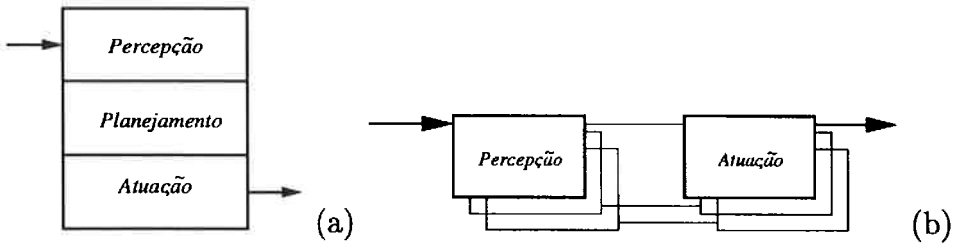


Figura 2.3: Dois paradigmas básicos em Robótica Móvel: a) Funcional e b) Comportamental.

Por outro lado, um robô móvel que raciocina mais profundamente para decidir sobre suas ações pode antecipar as conseqüências das suas ações, iniciar alguma atividade independentemente das situações imediatas detectadas pelos seus sensores, e organizar a seqüência de ações a ser executada considerando possíveis futuras configurações do ambiente. Esta habilidade reside no fato de que, apesar de ser verdade que o mundo real é complicado e com um certo grau de imprevisibilidade, ele também exibe uma grande quantidade de estrutura, a qual pode ser explorada pelo robô, seguindo o paradigma funcional, hierárquico (MEYSTEL, 1990; ALBUS, 1997).

Um grande desafio no projeto de robôs móveis consiste em definir compromissos adequados de tal modo que o robô exiba um comportamento apropriado sempre que necessário, seja em não continuar raciocinando sobre um evento futuro enquanto eventos imprevistos ocorridos no ambiente exigem reação rápida e imediata, seja em evitar que a percepção interrompa constantemente seu raciocínio, definindo arquiteturas apropriadas para um paradigma híbrido funcional e comportamental, deliberativo e reativo (ARKIN, 1986; GAT, 1991; LYONS; HENDRIKS, 1992).

2.4 Sensores e Atuadores

Robôs móveis interagem com o ambiente através de sensores, que efetuam a percepção, e atuadores, que permitem que o robô se movimente, manipule objetos ou direcione seus sensores para regiões de interesse.

As características físicas dos sensores e atuadores são importantes, pois delas dependem fortemente as capacidades de um robô móvel. Nos últimos anos, este

fato tem se evidenciado na importância dada ao projeto e análise de desempenho realizados em robôs reais, mesmo quando resultados em simulação se mostram suficientemente bons.

2.5 Navegação

Uma das habilidades mais importante em robôs móveis é sua capacidade de navegar, que consiste em atingir um alvo por meio de locomoção, ao mesmo tempo em que é capaz de evitar colisões com obstáculos e de se manter em condições seguras de operação.

Navegação pode ser definida como a combinação de três competências fundamentais:

1. Localização: denota a competência do robô em determinar sua própria posição no ambiente;
2. Planejamento de trajetória: consiste em determinar um caminho, velocidades e acelerações, para atingir sua meta, a partir de sua localização atual.
3. Construção e interpretação de mapas: consiste em determinar o mapa do ambiente – representado internamente em qualquer notação, métrica ou não – assim como em definir a habilidade do robô em interpretar (e utilizar) o mapa.

Robôs apresentam grandes dificuldades quando navegam longas distâncias, em ambientes que apresentam informação conflitantes e inconsistentes; eles funcionam melhor em ambientes que contenham marcos ou pistas, seguindo rotas fixas identificadas por meio de tais marcos ou pistas. Quanto maior o grau exigido de flexibilidade, mais difícil é para o robô móvel navegar de forma confiável e precisa. Assim, os estudo de técnicas, modelos e arquiteturas apropriadas para a navegação é um tópico muito importante de pesquisa em Robótica Móvel Inteligente.

2.6 Times de Robôs

Algumas vezes, a tarefa a ser executada por um robô móvel é tão complexa que fica muito difícil atribuir a um único robô a incumbência de realizá-la; neste caso, pode ser apropriado o uso de times de robôs.

Tarefas adequadas para atribuir a times de robôs são aquelas que possuem características inerentemente distribuídas, tanto em espaço, quanto em tempo ou funcionalidade, como, por exemplo, exploração (subaquática, planetária, etc), busca e resgate (de pessoas, plantas, animais, objetos), remoção de minas terrestres, inspeção de tubulações (ar condicionado, de água, esgoto, etc), combate a incêndios em campos e florestas, vigilância e limpeza de grandes áreas (estacionamentos, praias), etc.

Seria bastante razoável o uso de um time de robôs simples e baratos para executar estes tipos de tarefas, oferecendo vantagens tanto em termos de tempo de execução (um time de robôs cobre uma área num tempo muito menor que o dispendido por um único robô), quanto em termos de custo (um único robô para executar uma tarefa complexa exige maior capacidade de processamento, maior autonomia e maior robustez do que seria necessário para cada robô de um time).

Além disso, outra grande vantagem no uso de múltiplos robôs consiste na redundância oferecida: caso um robô falhe ou seja destruído, o restante do time pode continuar a execução da tarefa, podendo inclusive haver uma redistribuição ou redefinição das sub-tarefas alocadas aos robôs disponíveis. Assim, o uso de múltiplos robôs pode ser bastante vantajoso em uma série de aplicações (MATARIC, 1992; PARKER, 1998).

2.7 Aprendizado Autônomo

É de grande interesse, em várias situações, que o robô possua a capacidade de adaptar-se a mudanças que ocorram no ambiente, na tarefa ou em si próprios (falhas, etc).

A capacidade de aprender incute habilidade adaptativa aos robôs, a qual pode se manifestar em diversas dimensões: na geração e aprimoramento de políticas apropriadas de comportamentos, no aumento da eficiência da percepção

e atuação, na melhoria da coordenação da atuação conjunta de um time de robôs, entre outras (MITCHELL, 1997; ARKIN, 1999; MURPHY, 2000).

São três os paradigmas principais de aprendizado: supervisionado, não supervisionado e por reforço.

No aprendizado supervisionado, o conhecimento a respeito do domínio é representado por um conjunto disponível de exemplos de entrada/saída desejáveis e uma medida de erro entre a saída atual e a desejada pode ser computada e usada para atualizar a função de mapeamento entrada-saída do aprendiz.

O aprendizado não supervisionado classifica informações recebidas, sem o auxílio de pares entrada-saída de treinamento, mas através da exploração das estruturas e regularidades estatísticas embutidas nos dados de entrada.

Finalmente, no aprendizado por reforço, o agente aprende a resolver uma tarefa através de repetidas interações com o ambiente, por tentativa e erro, recebendo recompensas ou punições como retorno (MITCHELL, 1997; RIBEIRO, 2002).

3 Sensores e Atuadores

3.1 Sensores

Sensores são dispositivos que podem sensoriar e medir propriedades físicas de um ambiente, como temperatura, luminância, peso, etc. Estes dados e medidas são ruidosas, imprecisas, freqüentemente contraditórias e ambíguas.

Os sensores mais usados em robôs móveis se dividem em duas categorias: ativos e passivos.

3.1.1 Sensores Ativos

Sensores ativos são aqueles que gastam energia de forma propositada para captar a informação de interesse. Tipicamente, esta energia é dispendida na forma de um pulso (luz, som, etc), que é refletido por algum elemento do ambiente e captado de volta pelo próprio sensor. Os mais comuns são:

Sonares Um sonar é um dispositivo que emite e detecta pulsos em uma freqüência de som apropriada. Com base no tempo necessário para que o pulso refletido seja detectado, pode-se estimar a distância do objeto que produziu a reflexão relativamente ao sonar, em um processo similar ao usado por morcegos e golfinhos. São sensores baratos, capazes de medir com precisão razoável em uma faixa relativamente grande de distâncias.

Sensores de Infravermelho São similares aos sonares, porém operam em freqüências muito mais altas (radiação na faixa de infravermelho). Os mais simples são LEDs emissores de luz acoplados a receptores, baratos e pequenos, porém muito ruidosos.

Sensores Laser Emitem e detectam pulsos de radiação laser de baixa potência.

Têm maior acurácia do que outros sensores ativos. Um feixe de radiação laser refletido a partir de um obstáculo, por exemplo, pode ser usado para construir uma imagem razoavelmente precisa que identifica as distâncias de cada ponto do obstáculo relativamente ao robô. Sensores laser, porém, ainda são muito caros para a maioria das aplicações.

3.1.2 Sensores Passivos

Sensores passivos são aqueles onde nenhuma energia é emitida pelo sensor com o objetivo explícito de obter uma reflexão detectável. Os mais comuns são os seguintes:

Odômetros Dispositivos usualmente acoplados aos motores ou eixo das rodas, que estimam a distância linear percorrida pelo robô através da contagem do número de giros das rodas, em um processo conhecido como *dead reckoning*. A medida fornecida por esses sensores é muito imprecisa, devido a imprecisões de manufatura de rodas (variações de raio nominal), deslizamento em superfícies lisas e irregularidades do terreno.

Sensores de Choque Estes sensores normalmente consistem em um arranjo formado por dois fios condutores (um externo e um interno), separados por uma pequena distância e colocados no perímetro do robô (geralmente próximo à base). Ao se chocar com um obstáculo, o fio mais externo é empurrado na direção do fio interno, fechando um circuito e produzindo um sinal elétrico interpretado como uma colisão. A sensibilidade a choques pode ser ajustada para diferentes pressões de contato entre os fios.

Bússolas Medem a componente horizontal do campo magnético natural da Terra. O princípio de bússola mais usado em robôs móveis é aquele que mede a força do campo magnético através de mudanças nas propriedades de um elemento eletromagnético. Distorções da medida ocorridas por proximidade a objetos metálicos ou campos magnéticos artificiais (linhas de potência) são inevitáveis. Assim, as bússolas não fornecem uma referência absoluta verdadeira, mas sim importantes referências locais.

GPS Um sistema GPS (*Global Positioning System*) acoplado a um robô móvel funciona recebendo sinais de satélites e combinando-os para estimar a po-

sição do robô relativamente a um sistema de coordenadas locais. Por ser uma técnica de estimação local, GPS equivale a um processo de odometria, só que muito mais preciso. Alguns sistemas mais sofisticados, no entanto, permitem estimação de posição relativa a um sistema global de coordenadas. O preço comercial de sistemas GPS tem caído muito nos últimos anos, tornando-o um tipo de sensor a ser considerado em aplicações de baixo custo.

Câmeras Assim como em seres humanos, a capacidade de visão é extremamente útil em robôs móveis. A maioria das câmeras em uso são baseadas na tecnologia CCD (*charged coupled device*), em que a luz incide sobre uma matriz de capacitores fotossensíveis produzindo uma imagem, normalmente disponibilizada na forma analógica. Circuitos de conversão A/D dedicados de baixo custo (*framegrabbers*) são utilizados para transformar essas imagens para o formato digital, adequado para processamento computacional. Câmeras digitais, cujo custo tem caído sensivelmente nos últimos anos, utilizam CCDs acoplados a conversores A/D que produzem diretamente uma saída digital, dispensando o uso de *framegrabbers*. É comum o uso de duas ou mais câmeras montadas sobre um robô móvel, de modo a permitir visão com informação de profundidade (visão estereoscópica).

3.1.3 Visão Computacional

Visão Computacional refere-se ao processamento de dados advindos de qualquer sensor que use o espectro eletromagnético para produzir imagens, que são representadas por meio de uma matriz de *pixels* (contração de *picture element*) – eventualmente, utilizando um processo de conversão analógico-digital – matriz esta denominada *imagem digital*.

Os sensores usados para produzir imagens podem ser câmeras, sensores térmicos, raio-X, sensores a laser, radares, etc. O tipo de sensor usado define qual informação é armazenada nos pixels.

Graças ao baixo custo e disponibilidade atuais dos sensores e dispositivos, aliados aos progressos científicos e algoritmos da área, o uso de Visão Computacional em robôs está se difundindo muito nos últimos anos. Tarefas importantes

e bastante exploradas dos sistemas de visão de robôs móveis baseiam-se na cor como importante propriedade a ser identificada e classificada nas imagens. A determinação de regiões na imagem e respectivas correspondências a objetos no ambiente – baseadas em cor, forma, textura, etc – são de grande interesse. Visão também é usada para determinar medidas de distância – por estereoscopia, por exemplo – e para determinar movimento na cena – analisando o fluxo ótico em uma seqüência de imagens, por exemplo. São, portanto, crescentes os esforços e interesses em Visão Computacional na Robótica Móvel Inteligente.

3.1.4 Fusão Sensorial

Mesmo para tarefas de pouca complexidade, geralmente um único sensor não é suficiente, necessitando a percepção de um robô móvel de uma fusão sensorial. Este termo refere-se a, usando informação captada por diversos sensores, formar um único modelo do ambiente.

Devido às diferentes características dos diversos sensores, suas vantagens e desvantagens, a tarefa de fusão sensorial é extremamente difícil. A informação advinda de diferentes sensores nem sempre é consistente; hipóteses devem ser feitas para formar um modelo global do mundo e resultados devem ser retificados sob a luz de novas evidências.

3.2 Atuadores

3.2.1 Forma de locomoção

Embora atuadores tais como braços mecânicos sejam úteis em tarefas que envolvam manipulação ou exame de objetos no ambiente, aqui serão considerados apenas os mais importantes em um robô móvel, ou seja, aqueles responsáveis por sua locomoção. Uma descrição detalhada a respeito de atuadores não relacionados à locomoção pode ser encontrada em (ANDEEN, 1988).

Rodas São os atuadores mais usados em robôs móveis. A configuração mais comum é a baseada em *drive* diferencial, na qual duas rodas são montadas em um eixo comum e comandadas por dois motores independentes. Normalmente, este arranjo requer uma terceira roda de apoio para manter o

equilíbrio do robô. Uma configuração mais estável e similar àquela utilizada em automóveis consiste em uma ou duas rodas ligadas a um eixo móvel (para realização de curvas), e um segundo par de rodas em um eixo fixo, comandadas por um motor para controle de velocidade linear. Esta configuração, porém, tem a desvantagem de não permitir que o robô gire em torno do prórpio centro. Em geral, a locomoção baseada em rodas tem fraco desempenho em terrenos nos quais as irregularidades têm altura não desprezível quando comparadas ao raio das rodas.

Hastes A locomoção baseada em hastes que se assemelham a pernas ou patas de insetos pode ser interessante em robôs que se deslocam em ambientes específicos, tais como aqueles com escadas, regiões muito íngremes, etc. Em geral, o projeto de sistemas para controlar o movimento destes atuadores (que têm no mínimo dois graus de liberdade) é bastante complexo. O custo desses sistemas também é relativamente alto, já que pelo menos dois motores são necessários em cada "perna".

Lagartas A locomoção baseada em lagartas é interessante nos casos em que o robô deve se mover em terrenos acidentados ou pouco estáveis. Em geral, porém, um robô movendo-se sobre lagartas é muito pouco eficiente: a fricção das rodas dentro das lagartas e o deslizamento das esteiras durante os movimentos de giro são grandes dissipadores de energia.

3.2.2 Acionamento

Quanto ao tipo de acionamento, pode-se ter atuadores elétricos, pneumáticos ou hidráulicos.

O tipo mais comum usado em robôs móveis é o motor elétrico – motor DC ou motor de passo. O motor elétrico DC é simples e fácil de operar, oferece torque moderado e controle adequado, enquanto o motor de passo é usado para realizar movimentos finos, precisos.

Os atuadores pneumáticos, tipicamente, operam entre posições fixas – controle tipo *bang-bang* – por bombeamento de ar comprimido entre câmaras e, assim, movendo um pistão. São simples e baratos, mas não oferecem alta precisão de movimento.

Os atuadores hidráulicos usam óleo pressurizado para gerar movimento. Oferecem potência e precisão, mas são pesados e caros para serem usados em robôs móveis.

3.2.3 Tipo de movimento

O movimento de um robô móvel pode ser caracterizado pelas restrições a ele impostas.

Sistemas não-holonômicos estão sujeitos a restrições que envolvem a velocidade, enquanto isso não ocorre em sistemas holonômicos. Como exemplo, considere o movimento de uma roda num eixo: a velocidade do ponto de contato da roda com o chão, na direção do eixo, é zero; desta forma, o movimento da roda fica sujeito a restrições não-holonômicas.

4 Comportamentos em Robôs móveis

Comportamento é uma função que relaciona de modo genérico os estímulos sensoriais a respostas produzidas. Neste capítulo, um enfoque baseado em comportamentos é proposto e justificado como uma base adequada para o estudo de robôs móveis inteligentes, e alguns métodos para a implementação de comportamentos são discutidos.

Existem dois comportamentos básicos na Robótica Móvel: comportamento reativo e comportamento deliberativo.

4.1 Reatividade

Reatividade corresponde a mapeamentos simples, ligando estímulos sensoriais a respostas (ações).

Considere o que ocorre ao tocarmos com a mão um objeto muito quente: imediatamente, uma ação reflexa de afastamento da mão é executada, sem que tenhamos a chance de raciocinar sobre o que aconteceu, e sobre que medida seria a mais apropriada a tomar na situação. O nosso cérebro não é chamado a atuar neste caso porque não há um problema a ser resolvido que exija raciocínio: a integridade do nosso corpo está em jogo, e a evolução dotou o homem e outros animais de uma capacidade de reação rápida a situações que a exijam.

Uma capacidade deste tipo também pode ser implementada em robôs móveis. Assim como no homem, tal capacidade não requer intermediação do "cérebro" do robô: ao ser apresentado o estímulo (por exemplo, calor excessivo medido por um sensor de temperatura), o robô deve imediatamente executar uma ação pré-definida. No exemplo, uma ação reativa típica seria um comando aos motores

das rodas para que afastem o robô da direção da fonte de calor. Considere por exemplo um robô móvel de base circular, com um sensor de temperatura localizado à frente. Uma implementação típica de capacidade reativa seria:

```
SE (medida de sensor de temperatura à frente > 50 graus)
  ENTAO (reverta o movimento das rodas)
```

que faz com que o robô se afaste da fonte de calor. Observe que a reação é implementada de forma muito simples: corresponde a uma reação (reverter movimento das rodas) a um estímulo observado pelo robô (calor "sentido" pelo sensor de temperatura).

4.2 Deliberação

Um robô inteligente pode tentar construir um modelo interno representando o ambiente no qual está inserido. Se um problema fosse então designado ao robô, este poderia tentar primeiro explorar o espaço de soluções obtidas a partir de seu modelo interno, gerando um plano de atividades e, então, realizar a ação física propriamente dita, através de seus atuadores. A geração deste plano de atividades é conhecida como *deliberação* ou planejamento, e corresponde portanto a um mecanismo que define como uma sequência de ações deve ser realizada, considerando o estágio de execução da tarefa e tendo em vista o objetivo a ser alcançado.

Diferentemente de uma ação reativa, uma ação escolhida via deliberação envolve uma análise do modelo interno, não correspondendo portanto a um simples mapeamento entre sensores e atuadores. Embora seja teoricamente possível a geração de um único plano de ações a partir de um modelo interno fiel ao domínio (ou seja, escolha de uma sequência de ações a partir de uma única deliberação), na prática novos planos devem ser criados à medida que o robô interage com o seu ambiente, pois aspectos não modelados se apresentam com frequência em problemas reais. A geração de um novo plano para compensar deficiências de um plano original é conhecida como *replanejamento*.

4.3 Definição de Comportamento

Um comportamento relaciona estímulos sensoriais s a ações a produzidas sobre os atuadores do robô, de acordo com um plano p realizado a partir de um modelo interno do ambiente:

$$a = c(s, p) \tag{4.1}$$

Cada comportamento implementado em um robô corresponde a uma função deste tipo. De acordo com a complexidade do plano, é possível definir uma escala crescente de complexidade do comportamento: no extremo mais simples desta escala estão os comportamentos reativos, que não utilizam planos; no extremo oposto estão os comportamentos deliberativos mais complexos, ou seja, baseados em planos que utilizam um modelo interno de alto grau de detalhamento (Figura 4.1).

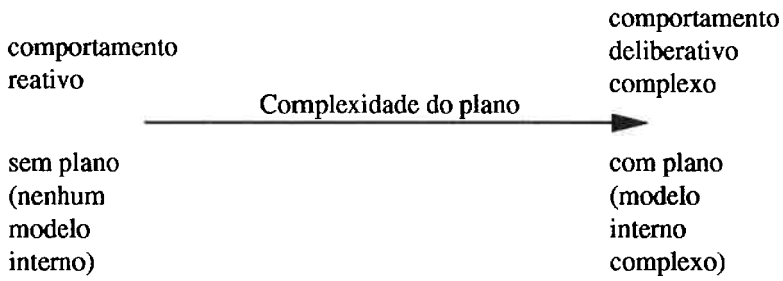


Figura 4.1: Escala de comportamentos para robô móveis, de acordo com a complexidade do plano.

É interessante notar que a definição dada pela Equação 4.1 não faz referência explícita ao processamento da informação fornecida pelos sensores. Desde que esta informação não seja usada para a criação de um modelo global (a partir do qual seria gerado um plano), admite-se que comportamentos reativos possam processar a informação sensorial usando algoritmos de razoável grau de sofisticação. O processamento da informação sensorial neste caso é dito *propositado*, ou seja, é feito com o propósito de satisfazer unicamente as exigências relativas ao comportamento, para o estímulo em questão. Uma característica importante deste tipo de processamento de informação é ser centrado no robô: como não se produz um modelo global, todas as referências de posição, velocidade, etc. são relativas a um sistema de coordenadas local ao robô.

Pode-se utilizar fusão sensorial para processar a informação sensorial advinda

de vários sensores distintos, de modo a se adquirir uma informação de melhor qualidade que combine as melhores características dos sensores envolvidos.

4.3.1 Exemplos

Esta Seção apresenta dois exemplos simples de comportamentos, um reativo e um deliberativo.

Um Comportamento Reativo para Evitar Obstáculos

Considere um robô móvel de base circular e duas rodas, acionadas por motores independentes e com controle de velocidade. Considere ainda que este robô tem um conjunto de seis sensores de infra-vermelho, sendo três destes posicionados no hemisfério direito do robô e três no hemisfério esquerdo. Sensores de infra-vermelho saturam quando existe um objeto (obstáculo, parede, etc.) posicionado a uma distância pequena do sensor. Seja portanto L o valor de saturação para estes sensores. Caso os sensores do lado direito saturem, é natural desviar o robô para o lado esquerdo. Analogamente, caso os sensores do lado esquerdo saturem, o robô deve girar para o lado direito. Assim, o conjunto de regras:

```
SE (leitura(algum sensor do lado direito) > L)
ENTAO (aplique vel.< 0 na roda esquerda, vel.> 0 na roda direita)
SENAO( SE (leitura(algum sensor do lado esquerdo) > L)
       ENTAO (aplique vel.> 0 na roda esquerda,
                      vel.< 0 na roda direita)
       SENAO (aplique vel.> 0 nas duas rodas) )
```

faz com que o robô desvie de obstáculos na presença destes e siga em frente no caso contrário. Observe que o comportamento é completamente reativo: o mapeamento entre sensores e atuadores é direto, implementado por regras de produção e sem necessidade de um estágio intermediário de deliberação. Comportamentos reativos como este e outros mais complexos foram analisados por Braitenberg (BRAITENBERG, 1984), em um livro clássico da literatura de Robótica Móvel.

Um Comportamento Deliberativo para Aproximação de Alvo

Considere um robô em um ambiente formado por salas, corredores e portas, cuja tarefa é aproximar-se de um objeto (alvo) posicionado em algum ponto de uma das salas. Segundo o enfoque deliberativo, este problema é resolvido por meio de técnicas de planejamento de trajetórias, caracterizadas por:

- Disponibilidade para uso pelo robô de a) um mapa do ambiente que inclui as posições do alvo e do próprio robô e b) um modelo dos efeitos das ações do robô no mapa. Este conhecimento constitui o modelo interno do domínio.
- Um algoritmo para cálculo de trajetórias sobre o mapa, cuja aplicação corresponde ao processo de deliberação.
- Um algoritmo de replanejamento que, a partir das observações sensoriais, possibilite ao robô atualizar seu modelo interno e reaplicar o algoritmo de planejamento, caso seja necessário.

Para cada um destes estágios, existem várias técnicas descritas na literatura especializada. O aspecto mais crítico, porém, é a obtenção e atualização do modelo interno.

4.4 Métodos para Implementação de Comportamentos

Do ponto de vista de Engenharia, é essencial que o projeto de um robô móvel inteligente seja realizado de acordo com um conjunto de princípios e regras, correspondentes a um método ou modo de proceder. São apresentados a seguir alguns dos métodos mais comuns para auxiliar o projeto de robôs móveis inteligentes.

4.4.1 Métodos Baseados em Etologia

A Etologia estuda os hábitos dos animais e sua acomodação às condições do ambiente. Como frequentemente ocorre em Engenharia, modelos comportamentais obtidos a partir de estudos nas Ciências Biológicas têm sido traduzidos (após

grande simplificação) em modelos aplicáveis a entidades artificiais (no nosso caso, robôs móveis). As conclusões fundamentais que os estudos em Etologia têm demonstrado são:

1. Comportamentos complexos podem ser obtidos a partir de comportamentos mais simples (frequentemente puramente reativos).
2. Informação sensorial só é utilizada se for necessária para a execução do comportamento em questão.
3. Existem mecanismos de coordenação atuantes quando vários comportamentos estão ativos.
4. Os indivíduos têm comportamentos que são adequados para o ambiente (nicho) em que vivem.

A adoção da Etologia como uma base para a implementação de comportamentos não exige que se siga uma linha de projeto puramente reativa. A Figura 4.2 ilustra um método de projeto baseado em Etologia para a implementação de comportamentos em robôs móveis.

4.4.2 Métodos Baseados em Atividade Situada

Estes métodos para implementação de comportamentos baseiam-se na idéia de que cada comportamento deve ser definido e projetado para situações específicas que o robô pode encontrar. O projeto do robô é iniciado com a identificação detalhada de todas estas situações. A seguir, são criadas as respostas adequadas para cada situação, considerando-se as limitações físicas do robô e as características do ambiente. Os comportamentos produtores de tais respostas são então projetados e implementados. Uma avaliação é feita e realiza-se um processo de ajuste em que os comportamentos são aperfeiçoados várias vezes. Diferentemente do enfoque baseado em Etologia, métodos baseados em atividade situada não procuram uma analogia entre os comportamentos desenvolvidos para as situações e seus possíveis similares biológicos. A Figura 4.3 ilustra os passos do desenvolvimento de projeto para métodos baseados em atividade situada.

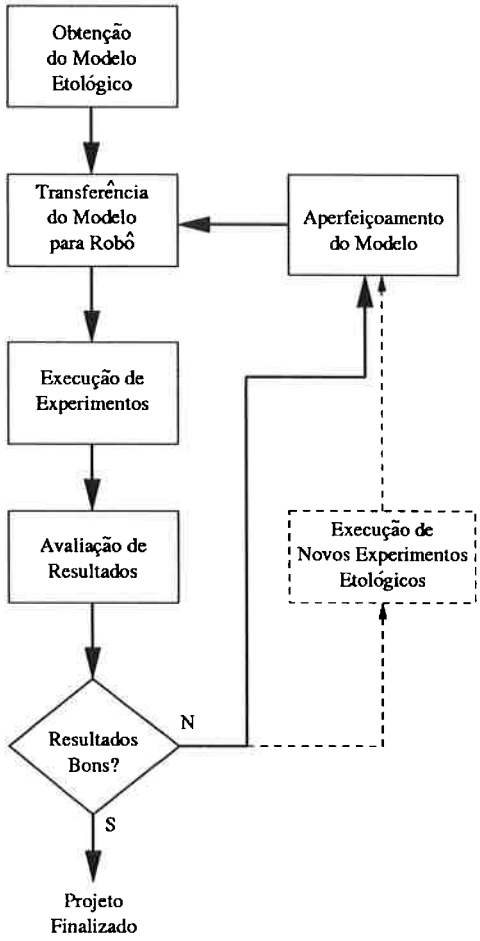


Figura 4.2: Método de projeto baseado em etologia.

4.4.3 Métodos Guiados por Experimentação

Métodos guiados por experimentação definem inicialmente um conjunto mínimo de comportamentos, desenvolvidos a partir de uma análise prévia de situações, similar àquela realizada em métodos baseados em atividade situada. Este conjunto mínimo é então avaliado por intermédio de experimentos, e novos comportamentos são adicionados para compensar os defeitos daqueles presentes, até que se obtenha um desempenho satisfatório (Figura 4.4).

4.5 Codificação de Comportamentos

No início deste capítulo, definiu-se um comportamento como uma função que mapeia estímulos sensoriais em ações, possivelmente de acordo com um plano preparado a partir de um modelo interno do domínio.

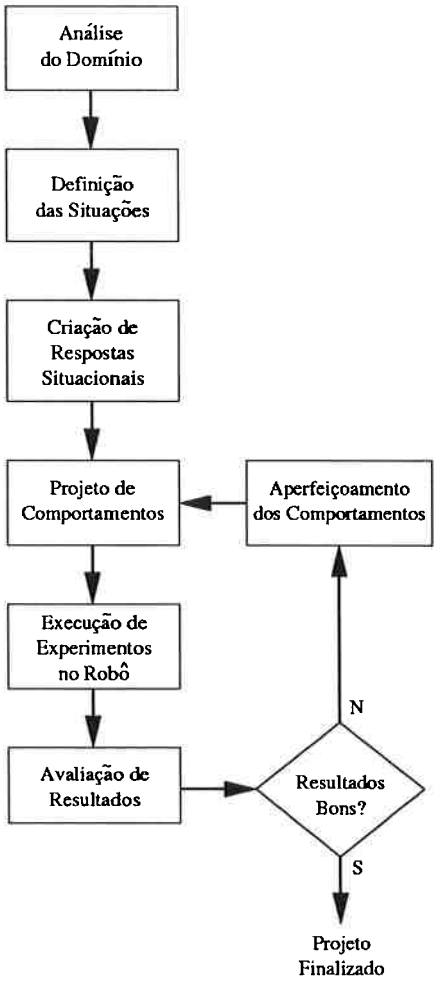


Figura 4.3: Método de projeto baseado em atividade situada.

Nesta Seção, identificam-se de forma mais precisa os parâmetros necessários para definir as ações do robô e os estímulos sensoriais. A seguir, será apresentado um formalismo apropriado para a definição e análise de comportamentos e sua coordenação em robôs móveis. Finalmente, será apresentada uma classificação para as funções mais comuns de coordenação de comportamentos.

4.5.1 Parametrização de Ações

Do ponto de vista cinemático, uma ação em Robótica Móvel objetiva transladar ou rotacionar o robô. Do ponto de vista dinâmico, uma ação corresponde a uma força ou momento aplicado ao robô, de modo a produzir o movimento desejado. O estudo de robôs móveis segundo o enfoque de Inteligência Artificial não considera os aspectos de possibilidade dos movimentos (tais como não-holonomicidade), que são normalmente estudados em Teoria de Controle. Para todos os efeitos,

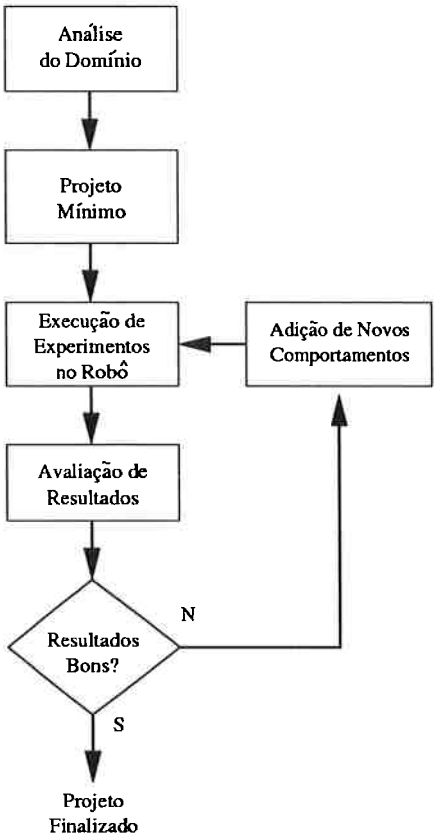


Figura 4.4: Método de projeto guiado por experimentação.

considera-se aqui que uma dada ação corresponde a uma realização de alto nível, que pode corresponder a um ou mais controladores de baixo nível de projeto bastante complexo. Exemplos de ações de interesse são: girar 90 graus para a esquerda, avançar 20 cm, manter velocidade constante de 12 cm./seg., cada uma das quais sendo implementada por um controlador projetado de acordo com técnicas de Controle, considerando-se o sinal de controle disponível (por exemplo, a tensão aplicada em motores que controlam a velocidade de giro das rodas).

De modo geral, os parâmetros que definem uma ação a de movimento aplicada a um robô móvel são a sua *magnitude* M e sua *direção* D. Assim,

$$a = a(M, D) \tag{4.2}$$

Para os exemplos acima, seria, respectivamente: $M = 90$, $D =$ "Esquerda", $M = 20$, $D =$ "Para frente", $M = 12$, $D =$ "Para frente".

Naturalmente, cada uma destas ações é projetada a partir de um ou mais controladores de baixo nível.

4.5.2 Parametrização de Estímulos

A parametrização dos estímulos depende muito do tipo de sensor considerado. Por exemplo, um sensor de infra-vermelho para detecção de obstáculos pode ter como único parâmetro a intensidade instantânea da radiação de infra-vermelho captada, enquanto a informação fornecida por uma câmera CCD pode ser parametrizada pela intensidade de brilho para cada pixel em uma matriz bidimensional.

Normalmente, existe um valor limiar L_{es} a partir do qual um estímulo sensorial é detectável pelo sensor correspondente. Este limiar é uma característica física do sensor. Um outro valor limiar L_{sr}, a partir do qual o estímulo, tal como captado pelo sensor, pode disparar (evocar) uma resposta por um dado comportamento. Este limiar é um parâmetro de projeto.

O limiar L_{sr} introduz portanto uma segunda função para uma percepção sensorial: além de *guiar*, ou seja, servir como elemento do domínio para a função $c(s,p)$, existe também uma função de disparo (*releasing*), que pode não envolver os mesmo sensores usados para a guiagem. Considere por exemplo o disparo de um alarme de incêndio em um cinema: a audição deste alarme (um estímulo auditivo) acima do seu limiar disparará um comportamento de fuga, guiado basicamente por estímulos visuais (evitar obstáculos, procurar a saída, etc.), olfativos (evitar regiões com cheiro de fumaça) e táteis (evitar calor).

4.5.3 Uma Formalização para Comportamentos e sua Coordenação

Formalização para Comportamentos

Em geral, cada comportamento c_i mapeia (possivelmente de modo não-determinístico) um conjunto de estímulos $\{s_i{}^1, s_i{}^2, \ldots, s_i{}^{k_i}\}$ e um plano p_i em uma ação a_i. Assim, para um robô móvel baseado em um conjunto de n comportamentos, tem-se:

$$\begin{aligned} a_1 &= c_1([s_1{}^1\ s_1{}^2\ \ldots\ s_1{}^{k_1}],\ p_1,\ w_1) \\ a_2 &= c_2([s_2{}^1\ s_2{}^2\ \ldots\ s_2{}^{k_2}],\ p_2,\ w_2) \\ &\vdots \quad\ \vdots \\ a_n &= c_n([s_n{}^1\ s_n{}^2\ \ldots\ s_n{}^{k_n}],\ p_n,\ w_n) \end{aligned} \tag{4.3}$$

onde w_i é um parâmetro aleatório cuja distribuição de probabilidade $P(w_i = w)$ é um parâmetro de projeto. Cada comportamento c_i é portanto uma função $c_i : \Re^{k_i} \times \mathcal{P} \times \Re \mapsto \mathcal{A} \cup \lambda$, cujos argumentos são um vetor real $\mathbf{s}_i$ de tamanho k_i formado pelos estímulos ($\mathbf{s}_i = [s_i{}^1\ s_i{}^2\ \ldots\ s_i{}^{k_i}]$), um plano $p_i \in \mathcal{P}$ convenientemente codificado ($\mathcal{P}$ representa o espaço de possíveis planos), e uma perturbação aleatória $w_i \in \Re$, presente apenas quando o comportamento considerado é *definido* como não-determinístico. Ao conjunto de ações possíveis $\mathcal{A}$ é acrescido uma ação inócua λ, que atua apenas como uma marca para indicar a inatividade do comportamento correspondente. Uma notação vetorial equivalente a 4.3 é:

$$\mathbf{A} = \mathbf{C}(\mathbf{S}, \mathbf{P}, \mathbf{W}) \tag{4.4}$$

onde

$\mathbf{S} = [\mathbf{s}_1\ \mathbf{s}_2\ \ldots \mathbf{s}_n]^T$ é a matriz de estímulos;

$\mathbf{P} = [p_1\ p_2\ \ldots\ p_n]^T$ é o vetor de planos;

$\mathbf{W} = [w_1\ w_2\ \ldots\ w_n]^T$ é o vetor de perturbações;

$\mathbf{C}(\mathbf{S}, \mathbf{P}, \mathbf{W}) = [c_1(\mathbf{s}_1, p_1, w_1)\ c_2(\mathbf{s}_2, p_2, w_2)\ \ldots c_n(\mathbf{s}_n, p_n, w_n)]^T$ é vetor de comportamentos do robô;

$\mathbf{A} = [a_1\ a_2\ \ldots\ a_n]$ é o vetor de ações definidas pelos comportamentos.

Mapeamentos Discretos e Contínuos

É importante observar que a formalização acima admite tanto codificações discretas como contínuas para o mapeamento de estímulos sensoriais e planos em ações.

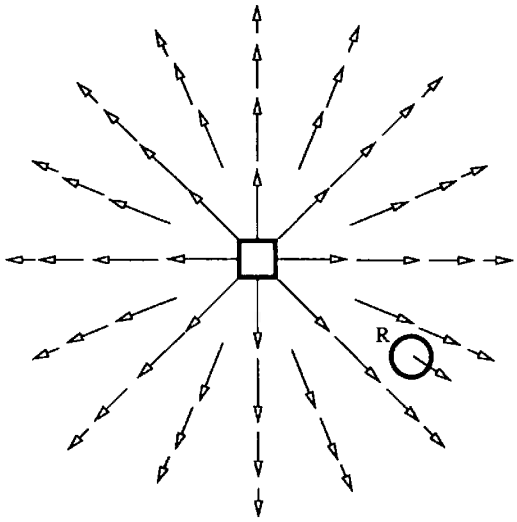
Codificações discretas referem-se àquelas em que o comportamento em questão corresponde a um conjunto finito de pares (*situação,ação*), onde o primeiro termo identifica um subconjunto de estímulos sensoriais e planos para o qual a *ação* correspondente é produzida pelo comportamento. Codificações discretas admitem uma realização em termos de regras de produção, tal como normalmente consideradas em sistemas de produção (regras do tipo SE-ENTÃO-SENÃO) (RUSSELL; NORVIG, 1995). Um exemplo de codificação discreta para um comportamento reativo foi apresentado na Seção 4.3.1.

Codificações contínuas são aquelas em que o comportamento é representado por uma relação funcional como a da equação 4.4, em que o mapeamento en-

tre estímulos sensoriais e ações correspondentes é contínuo. Uma implementação típica de codificação contínua para comportamentos reativos é a técnica de campos potenciais (ARKIN, 1999), que associa um vetor de ação a cada estado em um espaço contínuo (possivelmente estimado por um modelo interno). Um exemplo típico é um comportamento reativo repulsivo a obstáculos. A ação calculada tem a direção da linha imaginária ligando o robô à posição do obstáculo, o sentido é o de afastamento relativo ao obstáculo, e a magnitude é:

$$a = \frac{K}{d^2} \tag{4.5}$$

onde K é uma constante convenientemente escolhida e d é a distância Euclidiana entre o robô e o obstáculo. Um comportamento deste tipo gera um campo potencial como ilustrado na Figura 4.5. Note porém que este campo não precisa ser calculado durante o projeto do comportamento: quando ativado, o comportamento calcula o vetor de ação apenas para a situação correspondente.



Campo potencial associado a um comportamento contínuo de rejeição do obstáculo representado pelo pequeno quadrado. Apenas alguns vetores estão representados, pois o campo é contínuo. O robô R deverá produzir a ação indicada pela magnitude e direção do vetor do campo cuja origem coincide com a posição do seu centro.

Figura 4.5: Campo potencial associado a um comportamento contínuo de rejeição a um obstáculo.

Formalização para a Coordenação de Comportamentos

A coordenação dos vários comportamentos c_i é feita por uma *função de coordenação* Cc tal que:

$$a = Cc(\mathbf{C}(\mathbf{S}, \mathbf{P}, \mathbf{W}), \mathbf{S}, \mathbf{P}) = Cc(\mathbf{A}, \mathbf{S}, \mathbf{P}) \tag{4.6}$$

onde a denota a ação resultante da coordenação dos comportamentos. Observe que a coordenação dos comportamentos pode depender explicitamente de informações sensoriais e de planos. Sem perda de generalidade, considere que estes são também utilizados por algum ou vários dos comportamentos definidos. Caso a informação sensorial ou um plano sejam exclusivamente usados pela função de coordenação, pode-se incorporar ao modelo um comportamento *dummy* com saída λ (ação inócua indicando comportamento inativo), cujos argumentos são precisamente as leituras dos sensores correspondentes e o plano em questão.

Como um exemplo simples, considere um robô móvel equipado apenas com sensores de proximidade, atuando em um ambiente formado por paredes e obstáculos e realizando uma tarefa de exploração. Este robô deve combinar dois comportamentos básicos:

Comportamento c_1: Evitar Obstáculos , que pode ser implementado conforme o exemplo mencionado na Seção 4.3.1.

Comportamento c_2: Explorar o Ambiente , que pode ser implementado por comandos aleatórios de avanço e rotação.

Tem-se, portanto:

$$\mathbf{C}(\mathbf{S}, \mathbf{P}, \mathbf{W}) = [c_1(\mathbf{s}_1)\ c_2(w_2)]^T \tag{4.7}$$

Observe que nenhum dos comportamentos requer um plano para definir as respectivas ações (comportamentos puramente reativos). O comportamento c_1 é determinístico (ações bem definidas para situações de colisão iminente com obstáculos) e o comportamento c_2 é não-determinístico e independente de informação sensorial. Uma possível implementação de c_2 seria:

$$c_2(w_2) = \begin{cases} \text{avançar} & \text{se } w_2 \geq 1 \\ \text{girar à direita} & \text{se } 0.5 \leq w_2 < 1 \\ \text{girar à esquerda} & \text{se } w_2 < 0.5 \end{cases}$$

onde, de modo a manter uma maior probabilidade de avanço para o robô, a distribuição de w_2 poderia ser definida como $P(w_2 \geq 1) = 0.5$, $P(0.5 \leq w_2 < 1) = 0.25$ e $P(w_2 < 0.5) = 0.25$. Para uma função de coordenação Cc, tem-se:

$$\begin{aligned} a &= Cc(\mathbf{C}(\mathbf{S}, \mathbf{P}, \mathbf{W}), \mathbf{S}, \mathbf{P}) \\ &= Cc([c_1(\mathbf{s}_1)\ c_2(w_2)]^T, \mathbf{s}_1) \end{aligned} \tag{4.8}$$

É natural que, quando da iminência de colisão com um obstáculo ou parede, o comportamento c_1 seja definidor da ação a, de modo a preservar a integridade física do robô. Nas demais situações, o comportamento c_2 deve ser o único responsável pela definição da ação a. Uma função Cc adequada deve portanto selecionar o comportamento c_1 sempre que uma colisão (indicada pelos valores das leituras dos sensores $\mathbf{s}_1$) for iminente, e deve selecionar c_2 nas demais situações:

$$a = Cc([c_1(\mathbf{s}_1)\ c_2(w_2)]^T, \mathbf{s}_1) = \begin{cases} c_1(\mathbf{s}_1) & \text{se } s_1^i \geq L_{sr} \text{ para algum } i = 1, 2, \ldots, k_i \\ c_2(w_2) & \text{caso contrário} \end{cases}$$

onde L_{sr} é um valor limiar definido para os sensores de proximidade, para o disparo do comportamento de desvio de obstáculos.

Observe que, no caso acima, a função de coordenação reutiliza a informação sensorial $\mathbf{s}_1$ usada pelo comportamento c_1. Uma alternativa seria considerar c_1 ativo apenas quando $s_1^i \geq L_{sr}$, ou seja, o próprio comportamento se desativaria quando não fosse necessário. Nesse caso, seria:

$$a = Cc([c_1(\mathbf{s}_1)\ c_2(w_2)]^T) = \begin{cases} c_1(\mathbf{s}_1) & \text{se } c_1(\mathbf{s}_1) \neq \lambda \\ c_2(w_2) & \text{caso contrário} \end{cases}$$

Estes exemplos correspondem a funções de coordenação *competitivas*. Na Seção seguinte, são formalmente definidas as funções de coordenação mais comuns.

4.5.4 Funções Importantes de Coordenação de Comportamentos

Coordenação Competitiva As funções de coordenação apresentadas nos exemplos anteriores são típicos exemplos de coordenação competitiva: a cada instante de tempo, uma única ação associada a um ou mais comportamentos

"vencedores" é escolhida. A ação é sempre selecionada a partir dos comportamentos ativos (ou seja, comportamentos que produzem ações diferentes de λ no momento da coordenação). A competição entre os comportamentos pode ocorrer de três maneiras diferentes:

Competição com hierarquia pré-definida Neste caso, a função de coordenação seleciona o comportamento de acordo com uma hierarquia pré-estabelecida, definida por um plano p_{prior} projetado *a priori* e sem necessidade de sensoreamento (ou seja, a função de coordenação tem a forma $Cc(\mathbf{C}(\mathbf{S},\mathbf{P},\mathbf{W}),p_{prior})$. Este caso corresponde ao segundo exemplo da última Seção: o comportamento vencedor é escolhido de acordo com uma hierarquia simples (c_1 tem prioridade sobre c_2), em função da atividade ou inatividade do comportamento de desvio de obstáculos, independentemente da informação sensorial.

Competição com definição dinâmica da hierarquia A função de coordenação escolhe a ação vencedora com base na definição dinâmica da hierarquia, que é estabelecida de acordo com a informação sensorial e o conhecimento a respeito do estágio de execução da tarefa (incluído em um plano), ou seja, tem-se $Cc(\mathbf{C}(\mathbf{S},\mathbf{P},\mathbf{W}),\mathbf{S},\mathbf{P})$.

Competição Baseada em Votação A ação é selecionada de acordo com uma espécie de votação realizada pelos comportamentos ativos: aquela ação sugerida pela maioria é escolhida. A função de coordenação neste caso apenas conta os "votos" e aplica a ação vencedora correspondente. Caso haja empate entre ações vencedoras distintas, uma competição com hierarquia pré-definida ou definida dinamicamente é realizada.

Coordenação Cooperativa Na coordenação cooperativa, a função de coordenação produz uma ação resultante para a qual contribuem (em maior ou menor grau) todos os comportamentos ativos. É essencial, portanto, que os comportamentos sejam representáveis como produtores de ações que admitam operações de soma, multiplicação ou similar. Um exemplo são as representações em termos de campos potenciais (Seção 4.5.3), que representam ações associadas a cada comportamento como vetores. A cooperação entre os comportamentos é portanto representada por uma operação (frequentemente uma soma vetorial ou escalar) envolvendo as ações produzidas pelos

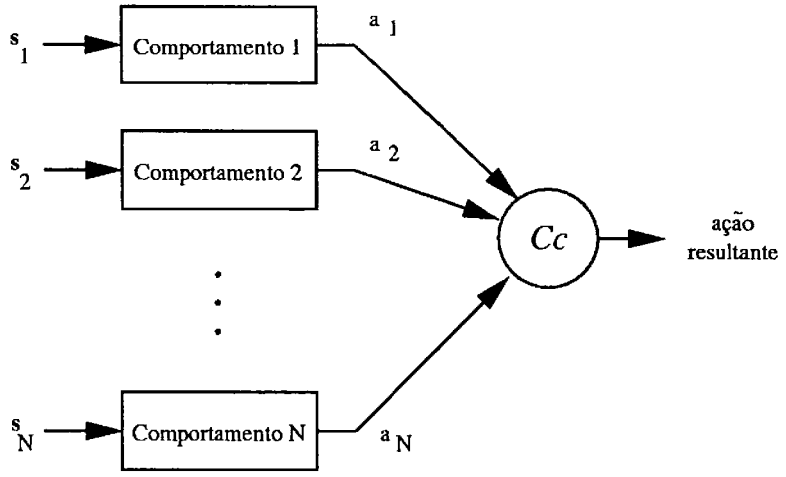


Figura 4.6: Coordenação cooperativa entre comportamentos.

comportamentos ativos, conforme indicado na Figura 4.6.

5 Arquiteturas para Comportamentos

Neste Capítulo serão discutidas três arquiteturas de comportamento para robôs móveis: reativa, deliberativa e híbrida. Estas arquiteturas definem como o robô é projetado, a partir de uma coleção de blocos fundamentais, correspondentes a comportamentos, módulos para inibição ou ativação destes, estruturas especializadas em planejamento de tarefas, etc.

5.1 Arquiteturas Reativas

As arquiteturas reativas são aquelas formadas pela coordenação simples de comportamentos independentes. Em geral, estas arquiteturas produzem robôs adequados para operação em tempo real, já que a simplicidade da coordenação dos comportamentos favorece uma alta velocidade de processamento computacional. Arquiteturas reativas também permitem prototipação rápida: a implementação de alguns poucos comportamentos em um robô real é relativamente simples. Entretanto, implementar um projeto de interesse real, que implique definir um conjunto grande de comportamentos e mecanismos de coordenação entre estes, é uma tarefa difícil. A seguir, são discutidas as duas arquiteturas reativas mais conhecidas.

5.1.1 Arquitetura *Subsumption*

A arquitetura *subsumption* (BROOKS, 1986) organiza os comportamentos em *camadas de competência*: comportamentos em níveis mais altos correspondem àqueles direcionados ao objetivo da tarefa especificada, enquanto que aqueles em níveis mais baixos correspondem a ações mais simples, menos propositadas

(reativas). Além das características intrínsecas do paradigma reativo, tais como simplicidade dos comportamentos (mapeamentos estímulo-ação) e processamento local da informação sensorial por cada comportamento, a arquitetura *subsumption* estabelece um mecanismo de prioridade de comportamentos das camadas em nível mais alto sobre aqueles em nível mais baixo, configurando uma coordenação competitiva de comportamentos com hierarquia pré-definida. Este mecanismo pode assumir duas formas:

Supressão Neste caso, a saída produzida pelo comportamento de nível mais alto substitui aquela produzida pelo comportamento de nível mais baixo. Este último permanece ativo, mas sua ação não produz nenhum efeito, por ter sido suprimida por aquela correspondente ao comportamento prioritário.

Inibição No mecanismo de inibição, o comportamento em nível mais baixo é desativado por aquele de nível mais alto. Nesse caso, não ocorre uma substituição da ação de nível mais baixo, mas uma inibição do comportamento em si.

É importante observar que a hierarquia na arquitetura *subsumption* é estabelecida de forma específica entre os comportamentos. A prioridade de um comportamento de nível mais alto ocorre apenas sobre uma coleção específica de comportamentos de nível mais baixo, e não sobre todos esses, indiscriminadamente. Assim, comportamentos básicos mas não necessários para preservar a integridade do robô, como por exemplo um comportamento de exploração não-direcionada do ambiente, pode ser inibido ou suprimido por um comportamento de nível mais alto, como um comportamento de aproximação de alvo, sem que este último exerça prioridade sobre um comportamento para evitar colisões com obstáculos.

A Figura 5.1 ilustra um exemplo simples de implementação da arquitetura *subsumption* (adaptado de (MURPHY, 2000)) para um robô móvel que se move em um ambiente, desviando-se de obstáculos quando necessário, mas sem perder a direção do seu movimento. O robô tem como sensores um conjunto de oito sonares distribuídos uniformemente em sua periferia, capazes de estimar a distância de obstáculos nas direções correspondentes. Um módulo de processamento da informação sensorial *Calcula Vetor* interpreta as leituras dos sensores como vetores e os soma, produzindo um vetor resultante que indica a direção e magnitude rela-

tivas à posição e distância de um obstáculo imaginário, que resume estas leituras. O movimento do robô é comandado por dois motores independentes: um motor 1 para produzir velocidades iguais nas rodas, e um outro motor 2 para produzir um giro do eixo das rodas. O comportamento *Parar* não faz parte da hierarquia de comportamentos e tem prioridade absoluta sobre todos os outros e atua da seguinte forma: ao detectar, pelas leituras dos sonares, a proximidade de algum obstáculo (leitura de algum sonar maior do que o limiar L_{sr}), força a parada do robô enviando um comando apropriado para o motor responsável pelo controle de velocidade das rodas. Na camada mais baixa da hierarquia de comportamentos está o comportamento básico *Afastar*, que produz a) um giro do eixo das rodas do robô até que este fique com sua frente na direção oposta à do vetor (via comando enviado para o motor 2) e b) um posterior avanço proporcional à magnitude do vetor (via comando enviado para o motor 1). Numa camada mais alta, um comportamento *Explorar* combina um vetor de magnitude fixa e direção aleatória, produzido a intervalos regulares, a um vetor de direção oposta ao daquele produzido pelo módulo *Calcula Vetor*. Desta forma, obtém-se uma nova direção para o movimento do robô, correspondente a um desvio de obstáculos mais suave, que leva em consideração uma direção preferencial de movimento (aquela estabelecida pelo comportamento *Explorar*). Este comportamento suprime (ou seja, substitui) a saída produzida pelo comportamento *Afastar*, que no entanto continua ativo: caso o comportamento *Explorar* seja desativado (possivelmente por algum outro comportamento em nível mais alto), o robô ainda terá a capacidade de desviar de obstáculos satisfatoriamente.

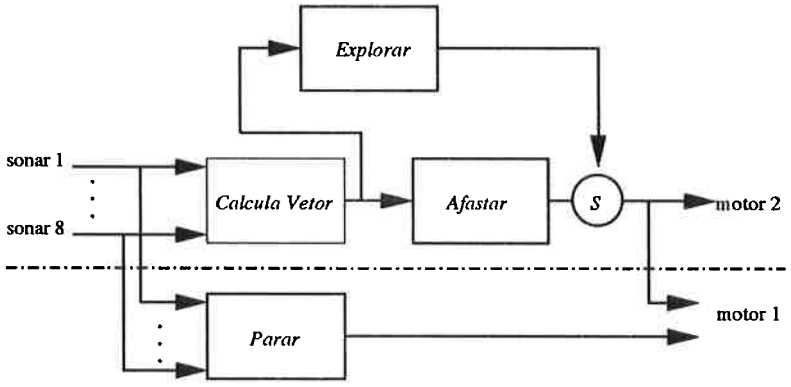


Figura 5.1: Um exemplo simples da arquitetura *subsumption*.

5.1.2 Arquitetura Baseada em Campos Potenciais

Esta arquitetura corresponde à implementação de comportamentos representados como campos potenciais (ver Seção 4.5.3), coordenados de acordo com um mecanismo de cooperação de soma vetorial (coordenação cooperativa de comportamentos). Diferentemente de robôs projetados com base na arquitetura *subsumption*, um robô de arquitetura baseada em campos potenciais tem todos os comportamentos em um mesmo nível de prioridade, sem uma hierarquia explícita entre eles. A cada comportamento corresponde uma ação (expressa como um vetor), produzida em qualquer situação (ou seja, todos os comportamentos estão sempre ativos). O comportamento efetivamente produzido é o resultante da soma dos vetores correspondentes à contribuição de cada comportamento. A magnitude dos vetores em pontos diferentes do ambiente em que o robô atua pode variar, o que equivale na prática a um mecanismo de inibição: um vetor de magnitude grande somado a um de pequena magnitude e direção oposta efetivamente inibe a ação deste último.

Uma arquitetura baseada em campos potenciais requer que a ação correspondente a cada comportamento seja expressa como um entre cinco possíveis tipos de campo:

Uniforme que corresponde a vetores de mesma intensidade e direção em qualquer ponto do ambiente. Um comportamento *Avançar*, que faz o robô se mover para a frente independentemente da informação sensorial, pode ser expresso como um campo deste tipo.

Perpendicular que orienta o robô na direção perpendicular a alguma fronteira (normalmente uma parede).

Atrativo que corresponde a vetores de magnitude inversamente proporcional à distância relativa a um ponto atrator e orientados em direção a este, em um efeito similar ao da gravidade ou atração eletrostática. Este tipo de campo é útil para expressar o efeito de *tropismo* em sistemas biológicos (atração por um objetivo, luz ou "comida").

Repulsivo que é o oposto do campo atrativo. Útil para modelar comportamentos para evitar obstáculos.

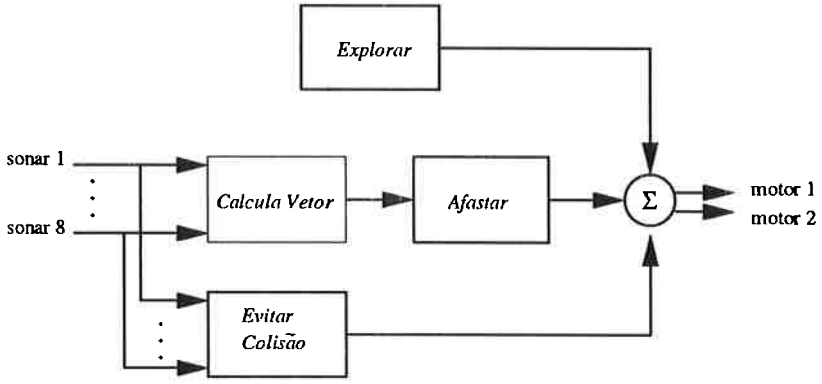


Figura 5.2: Um exemplo simples da arquitetura baseada em campos potenciais.

Tangencial que corresponde a vetores perpendiculares a linhas radiais a partir do centro de um objeto. Este tipo de campo é usado em comportamentos de desvio de obstáculos ou investigação de objetos.

Uma vantagem de uma arquitetura baseada em campos potenciais é a facilidade de visualização do comportamento global do robô, ainda na fase de projeto: a observação do campo resultante da combinação dos vários comportamentos permite prever com relativa facilidade o que o robô fará em cada situação.

Como exemplo de arquitetura baseada em campos potenciais, reconsidere o exemplo de desvio de obstáculos descrito na última Seção. A Figura 5.2 ilustra esta arquitetura.

Diferentemente da versão baseada na arquitetura *subsumption*, neste caso o vetor produzido pelo comportamento *Explorar* corresponde à direção aleatória de movimento, não somada à direção prevista pelo módulo *Calcula Vetor*. A ação correspondente é então diretamente somada à saída produzida pelo comportamento *Afastar*, sem que ocorra uma inibição deste último. O comportamento *Evitar Colisões* produz uma ação repulsiva proporcional à distância a um obstáculo próximo. Maiores detalhes de implementação e estudos de ambas arquiteturas são descritos no Capítulo 11.

5.2 Arquiteturas Deliberativas

As arquiteturas deliberativas incorporam um estágio intermediário de planejamento, quebrando a ligação direta entre sensoreamento e atuação e forçando a execução de um plano antes da seleção de qualquer ação. Neste tipo de arquitetura, as informações sensoriais são fundidas numa estrutura de dados global, chamada *modelo do mundo*, a qual é acessada pelo estágio de planejamento.

Geralmente, tem-se um controle hierárquico, onde o planejamento é subdividido em módulos funcionais, tipicamente dependentes tanto de informações espaciais, quanto de restrições temporais. A Figura 5.3 ilustra este modelo, onde se tem quatro níveis hierárquicos no planejador: nível de planejamento global estratégico, de planejamento intermadiário tático, de planejamento local de curto prazo e de controle do atuador. Além da estratificação do módulo de planejamento, o modelo do mundo também pode ser organizado em uma hierarquia de níveis de abstração crescente. Nota-se que, do nível superior da hierarquia (referente ao planejamento global) ao inferior (referente ao controle dos atuadores), cresce a restrição de tempo na resposta e diminui o espaço físico de interesse.

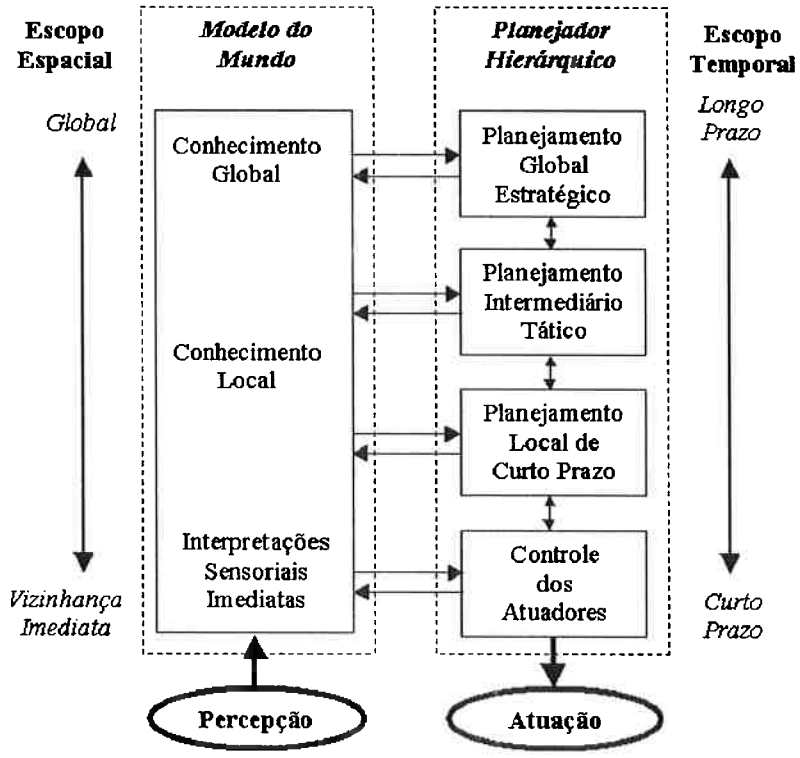


Figura 5.3: Planejamento hierárquico usado em arquiteturas deliberativas.

Uma desvantagem da arquitetura deliberativa consiste em desmembrar o planejamento de forma hierárquica para outras tarefas, diferentes de navegação, uma vez que nem sempre os diferentes níveis de representação e abstração da tarefa são tão intuitivos.

5.3 Arquiteturas Híbridas

As arquiteturas híbridas incorporam um elemento de planejamento sobre a definição e seleção de comportamentos individuais. Assim, uma arquitetura híbrida corresponde a uma arquitetura reativa controlada por um plano de execução e sequenciamento de comportamentos. Através da incorporação da habilidade de raciocínio baseado em modelos internos do mundo (deliberação), estas arquiteturas permitem a *reconfiguração dinâmica* de sistemas de controle reativo.

A integração de deliberação e controle reativo é um problema complexo. Contudo, existe um consenso que nenhuma abordagem – reativa ou deliberativa – é completamente satisfatória isoladamente, e ambas devem ser levadas em consideração para produzir um sistema flexível robusto e inteligente.

Um robô deve ser capaz de responder rapidamente e de forma eficiente às mudanças dinâmicas e não modeladas que ocorrem no mundo. Se um sistema puramente deliberativo tentar modelar e pré-planejar todas as eventualidades, corre-se o risco de que o processo de planejamento nunca termine. Também não é seguro para o robô fazer suposições grosseiras sobre o mundo, que não reflitam a sua natureza dinâmica, traduzindo-as num plano inalterável, que guiará todas as suas ações ou decisões futuras. Por outro lado, uma abordagem reativa responde eficientemente a dados sensoriais imediatos, mas é menos eficaz na integração de conhecimentos sobre o mundo.

Para o desenvolvimento de um sistema híbrido deve-se saber qual é a fronteira apropriada para subdivisão de funcionalidades e como deve ser conduzida a coordenação entre estas.

5.3.1 Percepção nas Arquiteturas Híbridas

A organização da percepção numa arquitetura híbridas é mais complexa, conforme ilustra a Figura 5.4. Em relação aos comportamentos, a percepção permanece como nas arquiteturas reativas: locais, específicos a cada comportamento. No entanto, o módulo de planejamento e deliberação, como precisa de modelos globais do mundo, pode não dividir os mesmos sensores usados pelos comportamentos, como também pode possuir seus próprios sensores e monitorar os comportamentos, extraindo informações pertinentes (sensores virtuais).

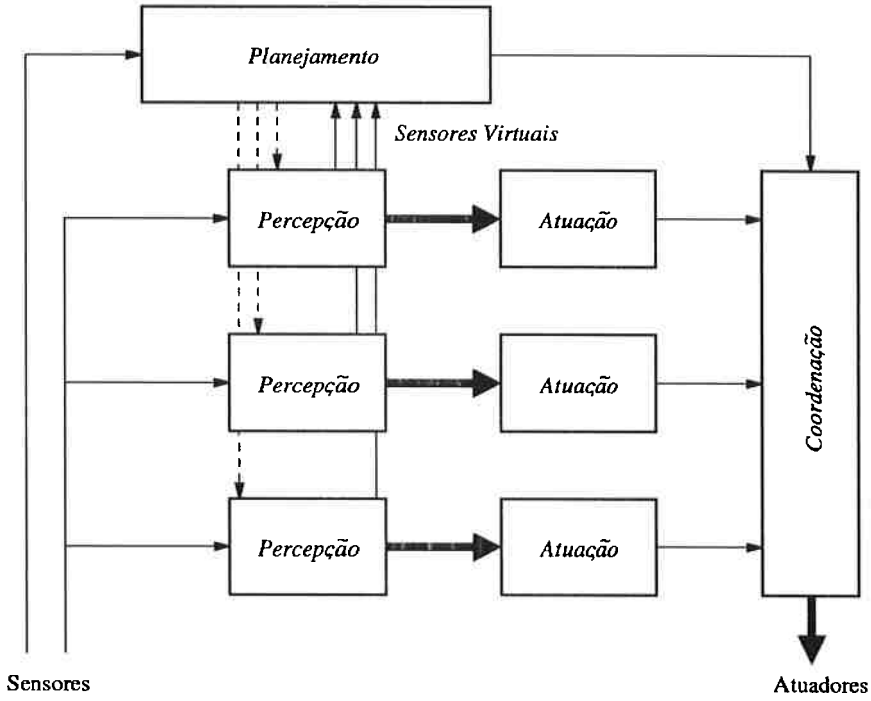


Figura 5.4: Organização da percepção na arquitetura híbrida.

5.3.2 Componentes das Arquiteturas Híbridas

Na maioria das arquiteturas híbridas a funcionalidade reativa é representada pelos comportamentos, que são vistos como sem conhecimento do estado global do robô, sua tarefa e ambiente, atuando exclusivamente em função do *Presente*. Já a funcionalidade deliberativa preocupa-se com o *Passado* e o *Futuro* do robô. A funcionalidade deliberativa normalmente é constituída pelos seguintes componentes:

Sequenciador Define o conjunto de comportamentos para a realização de uma

dada subtarefa, e determina sua sequência de operação.

Gerenciador de Recursos Responsável pela alocação de recursos (sensores, processamento para fusão sensorial, tempo, etc.) para os comportamentos ativos. Em arquiteturas puramente reativas, esses recursos são alocados permanentemente, quando da definição dos comportamentos em si.

Cartógrafo Cria, utiliza e mantém um mapa ou informação espacial do ambiente no qual o robô móvel se desloca, e o atualiza à medida que novas informações sensoriais são recebidas e ações são executadas pelo robô.

Planejador de Missão Codifica a descrição do domínio em uma linguagem apropriada, e define um plano para a execução da tarefa, possivelmente através de uma decomposição em vários subplanos.

Monitorador de Desempenho Avalia o desempenho do robô na execução da tarefa.

Estes componentes são organizados conforme ilustra a Figura 5.5.

A pesquisa na área de sistemas híbridos assume que a representação do conhecimento é necessária para enriquecer e expandir os comportamentos reativos, e adequá-los a domínios referentes a problemas mais interessantes e complexos.

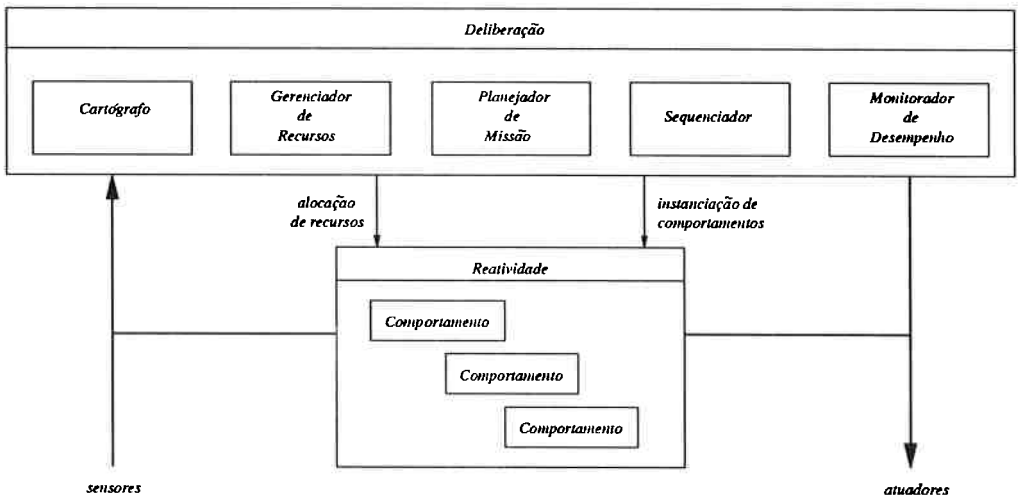


Figura 5.5: Organização das funcionalidades deliberativas e reativas na arquitetura híbrida.

6 Navegação

Navegação é uma habilidade crítica a um robô móvel, e envolve diferentes competências: percepção, atuação, planejamento, arquiteturas, hardware, eficiência computacional e resolução de problemas. Robôs reativos mostram comportamentos adequados para se movimentar no mundo, evitando colisões, mas a tarefa de navegação é mais propositada e requer deliberação, para que o robô possa planejar como atingir determinada localização.

Navegação envolve funções que podem ser expressas em termos de quatro questões básicas (MURPHY, 2000):

- **Para onde vou?** Normalmente esta questão é respondida por um humano ou um planejador de missão. Assim, assume-se que o robô tenha conhecimento *a priori* de seu alvo.
- **Qual o melhor caminho?** Este é o problema básico do planejamento de trajetória – ou caminho, quando somente as posições no ambiente são de interesse.
- **Onde já estive?** Possuir este tipo de informação é objetivo da tarefa de construção de mapas.
- **Onde estou?** Responder esta pergunta é tarefa da localização. Para que um robô possa seguir um caminho ou construir um mapa, ele precisa saber onde se encontra.

Desta forma, pode-se definir navegação como a combinação de três competências fundamentais: localização, planejamento de trajetórias e construção e uso de mapas.

Localização denota a competência do robô em determinar sua própria posição dentro de um plano de referência. Planejamento de trajetória consiste em determinar um caminho, velocidades e acelerações, dentro do plano de referência, para atingir sua meta, a partir de sua localização atual. Construção e interpretação de mapas consiste em determinar o mapa do ambiente – representado internamente em qualquer notação, métrica ou não, definindo locais de referência no plano de referência – assim como em definir a habilidade do robô em interpretar e utilizar o mapa.

O plano de referência para navegação, normalmente, é definido como sendo um sistema cartesiano de coordenadas, onde posições são armazenadas como coordenadas neste sistema – no entanto, qualquer outro tipo de representação poderia ser usado. Considerando que a jornada inicie em uma posição conhecida e que a velocidade e direção de navegação do robô sejam precisamente medidas, o sistema de navegação simplesmente integra no tempo o movimento do robô. Este processo é conhecido como *dead reckoning*. Assim, dado um sistema de coordenadas cartesiano, a posição do robô – e todas as outras – são sempre definidas. Entretanto, há um grave problema: o sistema de referência não está de fato fixo no mundo real. Para que um sistema de *dead reckoning* funcione, o robô deve medir de forma *precisa* seu movimento, o que é impossível, devido a problemas como escorregamentos das rodas; assim, determina-se os movimentos ocorridos *no* sistema de referência, mas não aqueles *do* próprio sistema. Na prática, sistemas de navegação baseados puramente em odometria só funcionam adequadamente em curtas distâncias.

Uma alternativa para sistemas de *dead reckoning* é o uso de navegação direcionada por marcos de referência do mundo real, que é baseada na percepção do ambiente pelo robô. Escorregamentos não são problemas nesta abordagem; no entanto, se o ambiente contiver poucos marcos perceptíveis ou caso os marcos não sejam suficientemente discriminantes, o desempenho deste tipo de abordagem também se deteriora.

Em robótica móvel, muitos mecanismos e competências navegacionais têm sido desenvolvidos. Alguns tipos comumente de navegação utilizados atualmente em robôs móveis são descritos.

6.1 Veículos Guiados

A forma mais simples de fazer com que um robô móvel atinja um alvo – uma posição especial no ambiente – é guiá-lo por meio de linhas pintadas no caminho, por ímãs e elos magnéticos colocados no solo, por marcos específicos (código de barras, marcos geométricos, etc) colocados no ambiente, etc. Tais veículos guiados automaticamente – AGVs, *Automated Guided Vehicles* – são muito usados em cenários industriais, para transportar peças e materiais. No entanto, AGVs não apresentam autonomia.

6.2 Navegação Baseada em *Dead Reckoning*

Devido aos erros – por escorregamento – dos odômetros, é muito raro que um sistema de navegação de um robô móvel seja baseado exclusivamente em *dead reckoning* (técnica de integração de caminho). Assim, nesta classe encontram-se aqueles robôs que usam *dead reckoning* predominantemente, mas que também usam alguma informação sensorial.

Um bom exemplo desta classe é o sistema de navegação de um robô móvel baseado em grades de ocupação (*certainty grids* (ELFES, 1987)), que utiliza um sistema cartesiano de referência, sonares e *dead reckoning*. Neste sistema, o ambiente é dividido em células de tamanho fixo, inicialmente rotuladas de *desconhecida*. À medida que o robô explora o ambiente – partindo de uma posição conhecida e usando os odômetros para estimar sua nova posição – as células vão sendo rotuladas de *livre* ou *ocupada*, dependendo da informação captada pelos sonares. Desta forma, o ambiente é mapeado e o mapa pode ser usado para outras navegações.

6.3 Navegação Baseada em Marcos Perceptuais

Uma alternativa ao sistema navegacional que usa fundamentalmente um sistema global cartesiano de referência, no qual o robô integra seu movimento através de informação dos odômetros, seria um sistema que navegue usando marcos do ambiente. Marcos, aqui, referem-se a percepções sensoriais dependentes da loca-

lização. Entretanto, a percepção sensorial está sujeita a ruídos e variações, tornando necessário o uso de alguma forma de *generalização* que procure conservar os atributos salientes de cada marco. Uma forma de adquirir esta generalização é através do uso de mecanismos de agrupamento (*clustering*): dados similares são agrupados em uma classe representativa.

Em algumas aplicações, o robô deve visitar somente um número fixo de locações previamente definidas; desta forma, pode-se determinar uma assinatura perceptual única para cada locação. Um problema nesta abordagem consiste em como distinguir locações perceptualmente similares – o problema do *perceptual aliasing*, onde discretizações perceptuais confundem as informações adquiridas pelos sensores, tornando as assinaturas indistinguíveis entre si. Em muitos casos reais, pode-se usar diferentes tipos e número de sensores, administrando o problema (que sempre existe, no nível teórico). Pode-se também incorporar a história das seqüências sensoriais locais no reconhecimento das locações ou mesmo combinar percepção e *dead reckoning* para eliminar a ambigüidade de locações similares, aumentando a complexidade do sistema navegacional.

6.4 Discussão

Cada princípio de navegação, seja por *dead reckoning*, seja por marcos perceptuais do ambiente, possui suas vantagens e desvantagens: o primeiro possui erros incorrigíveis dos odômetros, derivados de deslizamentos, enquanto o último é afetado por *perceptual aliasing*, onde duas ou mais locações perceptualmente indistinguíveis requerem ações distintas do robô.

Uma característica fundamental da navegação de seres biológicos – humanos, pássaros, insetos, etc – é o uso de múltiplas fontes de informação: bússola magnética em pombos, rastros no ambiente em insetos, fluxo ótico em gansos, estéreo binocular em humanos, etc (NEHMZOW, 2000). Seguindo os modelos biológicos, a combinação e uso concorrente de diversos mecanismos tornam mais confiável e robusto o sistema de navegação de robôs móveis. Sistemas híbridos, que combinam múltiplas fontes de informação, mostram-se como solução adequada, podendo preservar as vantagens de cada princípio e minimizar suas desvantagens.

7 Times de Robôs

O aumento na complexidade da tarefa designada aos robôs provoca um aumento na dificuldade de se atribuir a um único agente robótico a incumbência de executá-la eficientemente. Assim, o uso de múltiplos Robôs - também chamados de times de robôs ou multi-agentes robóticos - pode ser bastante vantajoso em uma série de aplicações compostas basicamente por tarefas muito complexas e/ou que sejam inerentemente distribuídas, tanto em espaço, quanto em tempo ou funcionalidade.

Pesquisas na área de times de robôs recaem na área de Inteligência Artificial Distribuída (IAD) e Sistemas Multi-Agentes (SMA). Assim como múltiplos comportamentos atuando concorrentemente, no paradigma comportamental, levam a um comportamento emergente, nos sistemas multi-agentes as atuações concorrentes e independentes dos agentes levam a um *comportamento social emergente.*

Neste capítulo são discutidas as vantagens e desvantagens do uso de times de robôs, relacionando tarefas tipicamente apropriadas para realização por múltiplos robôs. São consideradas características básicas dos times, tais como homogeneidade, granularidade, estrutura de controle, entre outras. De acordo com as características atribuídas aos times, pode-se agrupá-los em dois tipos básicos: times com muitos robôs idênticos que pouco interagem durante a execução da tarefa e times compostos por poucos robôs, cada um com habilidades próprias, que necessitam de grande interação entre eles para executar a tarefa de modo eficiente. Finalmente, são descritos exemplos de cada um destes tipos de times, assim como o domínio de futebol de robôs, usado para incentivar pesquisas nesta área, e suas abordagens mais frequentes.

7.1 Por que usar Times de Robôs?

O uso de múltiplos robôs pode ser desejável por diversas razões. Considere uma grande área que deva ser percorrida ou explorada, como no caso de exploração planetária, deteção e combate a incêndios (em florestas, por exemplo), busca e resgate de pessoas e/ou objetos, remoção de minas terrestres, etc. Seria bastante razoável o uso de um time de robôs simples e baratos para executar a tarefa, oferecendo vantagens tanto em termos de tempo de execução (um time de robôs cobre uma área num tempo muito menor que o dispendido por um único robô), quanto em termos de custo (um único robô para executar uma tarefa complexa exige maior capacidade de processamento, maior autonomia e maior robustez do que seria necessário para cada robô de um time). Além disso, uma grande vantagem no uso de múltiplos robôs consiste na redundância oferecida: caso um robô falhe ou seja destruído, o restante do time pode continuar a execução da tarefa. Assim, o uso de múltiplos robôs pode ser bastante vantajoso em uma série de aplicações compostas basicamente por tarefas que são inerentemente distribuídas, tanto em espaço, quanto em tempo ou funcionalidade.

No entanto, não é nada fácil a construção de um time de robôs eficiente e robusto, que opere de modo coerente na solução de uma tarefa. Para que um robô possa atuar num time e interagir de forma coordenada com os outros robôs, uma capacidade extra deve ser acrescida: a capacidade de comunicar. Isso envolve a definição de uma linguagem comum entre os membros do time e de um protocolo de comunicação a ser seguido. Além disso, deve-se garantir um perfeito entendimento entre os robôs durante a exeção da tarefa, eliminando o uso de diferentes termos para um mesmo conceito ou o uso de um mesmo termo para diferentes conceitos, isto é, deve-se estabelecer e disponibilizar a todos os membros do time uma *ontologia* comum, contendo a representação do conhecimento necessário sobre o domínio da tarefa.

Além da questão referente à comunicação, o projeto de um time de robôs precisa ainda endereçar difíceis problemas, tais como:

- Como decompor uma tarefa e alocar as subtarefas entre os membros de um time?
- Como e quanto devem os robôs comunicar e interagir entre si?

- Como garantir que o time agirá coerentemente?
- Como os robôs reconhecerão e resolverão conflitos?
- Como dotar um time com capacidade de adaptação, mudando seu comportamento em resposta às características dinâmicas do ambiente, às mudanças nas metas da tarefa ou às mudanças nas capacidades ou composição do time, de tal forma que melhore ou que evite uma degradação no seu desempenho?

Estas são questões que, embora extensivamente estudadas nos últimos anos, ainda oferecem muitos desafios a serem superados.

7.2 Características dos Times de Robôs

O projeto e construção de times de robôs ainda se configura como um campo muito novo, onde pouco está estabelecido e formalizado. Para entender a organização e interação entre seus membros e tentar prever o comportamento do time como um todo, torna-se importante definir alguns aspectos para classificar o time. Aqui serão considerados aspectos referentes à sua granularidade, heterogeneidade, estrutura de controle, nível de cooperação entre os membros e tipo de comunicação.

A *granularidade* de um time é o seu tamanho, isto é, o número de robôs que o compõem. Dois ou mais robôs configuram um time.

Heterogeneidade corresponde à diversidade entre os robôs que fazem parte do time, podendo classificá-lo como homogêneo ou heterogêneo. Times heterogêneos possuem pelo menos dois membros com características diferentes de *software* e/ou *hardware*, enquanto que em times homogêneos todos seus membros são idênticos.

A *estrutura de controle* de um time define um espectro que varia de controle centralizado a controle distribuído. No controle centralizado, os robôs se comunicam com um computador central, que distribui atribuições, metas, informações, etc; os robôs são essencialmente semi-autônomos, dependentes do computador central para decidir sobre suas ações. Por outro lado, na estrutura de controle distribuído os robôs tomam suas próprias decisões e agem de maneira independente. Naturalmente, existem diversas estruturas que podem ser definidas dentro

do espectro de controle completamente centralizado a controle totalmente distribuído. Por exemplo, os robôs podem interagir com um computador central para definirem suas metas e, depois, atuarem de modo distribuído para cumprí-las.

A *cooperação* entre os membros de um time pode ser ativa ou não-ativa. Cooperação não-ativa indica que os robôs não compartilham explicitamente uma meta comum, ou seja, cada membro tem sua própria (sub-)meta. O time demonstra uma cooperação ativa quando pelo menos alguns de seus membros se reconhecem e trabalham conjuntamente para atingir uma meta comum explícita, caracterizando intencionalidade na cooperação. Normalmente torna-se necessário, na cooperação ativa, incorporar sensores aos robôs de forma a permitir que estes façam a distinção dos membros do time em relação a outros aspectos do ambiente. Por exemplo, num jogo de futebol de robôs, um jogador de um time pode reconhecer o atacante de seu time e passar a bola para ele, para que a meta do time, que é fazer gols e ganhar o jogo, seja atingida, demonstrando, desta forma, uma cooperação ativa. No entanto, considere um time formado por dois robôs idênticos, cujos sensores somente detectam paredes e que têm como metas (individuais) derrubar paredes; caso estes robôs se posicionem em partes diferentes de uma mesma parede, ambos irão derrubá-la, provavelmente finalizando a tarefa num tempo muito menor; porém, como a ajuda mútua foi puramente acidental, os robôs de fato atuam segundo cooperação não-ativa.

Quanto à *comunicação*, existem dois tipos: explícita ou implícita. Na comunicação explícita, informações são *intencionalmente* trocadas entre dois ou mais membros do time. Na comunicação implícita a informação é adquirida pela observação das ações realizadas pelos outros membros ou por rastros não intencionalmente deixados, observados no ambiente.

Um time homogêneo que apresenta alta granularidade, controle descentralizado, cooperação não-ativa entre seus membros e sem comunicação explícita, frequentemente refere-se a um time onde cada robô segue o paradigma comportamental, agindo concorrente e independentemente, propiciando o surgimento de um comportamento social emergente. Este tipo de time oferece o grande atrativo de ser formado por robôs bastante simples, com poucas necessidades de sensoreamento e que executam poucos (e simples) comportamentos. Esta abordagem pode ser interessante em aplicações que não impõem restrições drásticas no tempo

de execução e que requerem numerosas repetições de uma mesma atividade numa área relativamente grande, tais como em tarefas de limpeza (em praias, grandes estacionamentos, etc), de coleta de material (em missões espaciais, subaquáticas, etc), de resgate (busca por pessoas, animais, objetos), entre outras. Deve-se ressaltar, no entanto, que comportamentos sociais emergentes são muito difíceis de serem previstos e controlados.

Por outro lado, em times heterogêneos nem todos os robôs possuem as mesmas capacidades, habilidades ou estrutura de controle. Desta forma, nem todos os robôs podem executar as mesmas tarefas e, mesmo que o façam, estas serão realizadas de formas diferentes, uma vez que alguma diferença existe entre os robôs (em hardware ou software, pois são heterogêneos). Assim, o mapeamento apropriado das subtarefas aos robôs é dependente das capacidades e desempenho de cada robô membro do time. Times heterogêneos geralmente apresentam baixa granularidade e a interação entre seus membros deve ser endereçada de forma clara e explícita. Normalmente, as tarefas designadas a este tipo de time apresentam claras restrições em relação à eficiência de sua execução e, para satisfazê-las, o time frequentemente apresenta cooperação ativa, intencional. Esta restrição adicional traz muitas complicações extras ao projeto de uma arquitetura apropriada que possibilite cooperação ativa, e esta restrição precisa ser endereçada explicitamente de forma a alcançar o nível desejado de cooperação.

Os dois tipos de times descritos acima categorizam, de forma bastante geral, dois grupos básicos de times de robôs móveis. No primeiro grupo (grandes times homogêneos), um fator central consiste na determinação do projeto apropriado para as leis de controle locais que possibilitarão com que o time de robôs execute adequadamente a tarefa a ele designada. Já no segundo grupo (times heterogêneos com cooperação ativa entre seus membros), um fator central consiste em robustamente determinar qual robô deve executar qual tarefa, de forma a maximizar a eficiência do time, além de garantir uma coordenação apropriada entre seus membros para que a tarefa seja executada com sucesso.

7.3 Comportamento Social Emergente

Nesta Seção, exemplos dos dois tipos básicos de times de robôs são apresentados: um deles, com uma abordagem baseada na criação de regras sociais em grandes times homogêneos e outro, baseado na criação de motivações internas aos robôs, para que possam adaptar seus comportamentos aos problemas e conflitos que surjam durante a execução da tarefa.

7.3.1 Regras Sociais em Times Homogêneos

Mataric (MATARIC, 1992), concentrou-se em definir como pode se manifestar a dinâmica de um time homogêneo composto por um grande número de robôs simples, operando em controle totalmente distribuído. No seu trabalho, cada robô foi implementado usando a arquitetura *Subsumption*, composta pelos comportamentos de evitar colisões e de mover para uma posição específica pré-estabelecida.

Inicialmente foram utilizados 20 robôs, posicionados aleatoriamente numa partição do campo de experimentação composto por duas partições, separadas por uma porta estreita, por onde pode passar somente um robô por vez. A todos os robôs foi dada a mesma meta de atingir uma posição localizada na outra partição (para satisfazê-la, todos os robôs devem atravessar a porta). Todos os robôs iniciam o movimento num mesmo instante.

Num primeiro experimento, os robôs atuavam sob *coexistência ignorada*, isto é, coexistem num mesmo time porém sem nenhum conhecimento mútuo, tratando os outros membros como obstáculos. Assim, ao atingir a porta, passavam a maior parte do tempo tentando evitar colisões, demorando muito para atravessá-la. Quanto maior o número de robôs, maior o congestionamento na região da porta e maior o tempo que levaram para concluir a tarefa.

Num segundo experimento, os robôs atuavam sob *coexistência informada*. Assim, algumas modificações foram realizadas em cada robô: podiam agora reconhecer uns aos outros, passaram a possuir também o comportamento de evitar robôs e a respeitar uma regra social simples. A regra social estipulada foi: "caso algum outro robô obstrua a passagem, parar movimento e esperar durante um tempo t; após t, caso o robô continue bloqueando a passagem, virar à esquerda

e reiniciar movimento em direção à meta". O resultado deste experimento foi a diminuição do congestionamento e um tempo muito menor para a execução da tarefa.

Num terceiro e último experimento, o time atuou sob *coexistência inteligente*, onde existia uma repulsão entre os robôs mas, ao se afastar, cada robô deveria tentar se mover na mesma direção que a maioria dos outros robôs estavam indo. Para isso, os robôs transmitiam via rádio, a todos os outros membros, sua direção de movimentação (por inabilidade de determinar isso por visão ou outro sensor). Como resultado, os robôs tendiam a formar uma única fila (apesar deste comando não estar explícito no projeto), diminuindo sensivelmente tanto o congestionamento quanto o tempo de execução da tarefa.

Nos experimentos conduzidos por Mataric, a interfência dos robôs foi minimizada pelo uso de regras sociais simples, sem comunicação explícita entre eles. No entanto, caso algum robô apresente falhas, nenhum outro membro poderia ajudá-lo e nem mesmo as tarefas poderiam ser mudadas dinamicamente. Estes pontos foram endereçados por Parker (PARKER, 1998), que propôs a arquitetura ALLIANCE para cada robô de um time cooperativo, de tal forma a permitir que o time considere e resolva falhas e incertezas, derivadas tanto da seleção quanto da execução das ações.

7.3.2 Cooperação em Times Heterogêneos

ALLIANCE é uma arquitetura que facilita o controle cooperativo tolerante a falhas de times heterogêneos de robôs móveis, os quais executam tarefas compostas por subtarefas que podem, eventualmente, apresentar uma relação de ordem entre si, isto é, a finalização de uma subtarefa pode ser condição para que outra subtarefa seja realizada. ALLIANCE permite que os robôs, cada um com diferentes competências, possam individualmente selecionar ações apropriadas durante a realização da tarefa, baseados não somente nas exigências da tarefa, mas também nas atividades dos outros robôs, nas condições do ambiente e nos seus próprios estados internos. Trata-se de uma arquitetura distribuída baseada em comportamentos, que incorpora o uso de *motivações* em cada robô (como impaciência e aquiescência), de forma a adquirir seleção adaptativa de ações.

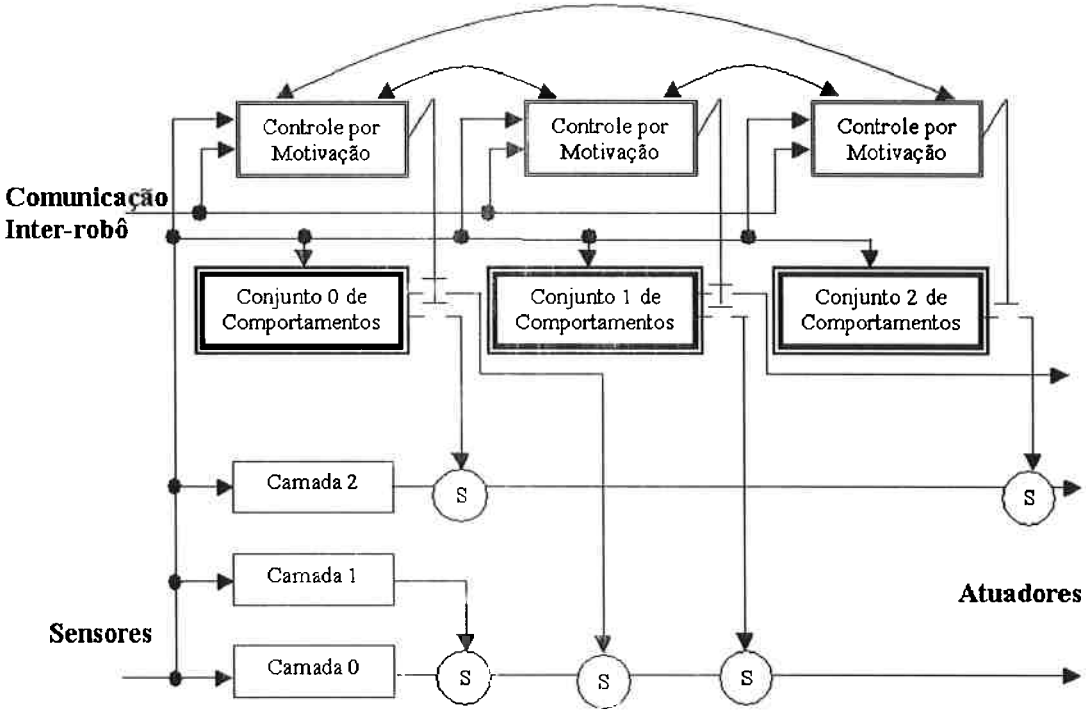


Figura 7.1: Esquema da arquitetura ALLIANCE, implementada em cada robô de um time cooperativo.

Esta arquitetura requer que os robôs membros do time sejam capazes de detectar o efeito de suas próprias ações, assim como o de outros robôs que tenham competências iguais. Além disso, considera-se que os sensores e atuadores dos robôs não são perfeitos e que qualquer subsistema dos robôs pode falhar.

ALLIANCE é uma extensão da arquitetura *Subsumption* (Figura 7.1), onde foram definidos conjuntos de comportamentos que podem estar ativos ou não, correspondendo às competências dos robôs. Os robôs ativam estes conjuntos baseados nas subtarefas a ele designadas e na sua *motivação* para a execução de cada subtarefa.

Dois tipos de motivação interna são modelados em ALLIANCE: *impaciência*, que permite com que o robô lide com falhas dos outros membros relativas à execução de uma subtarefa, e *aquiescência*, que permite com que o robô lide com suas próprias falhas na execução apropriada de uma subtarefa.

Inicialmente, é baixa a motivação de um robô para ativar qualquer conjunto de comportamentos. Com o passar do tempo, a motivação para ativar um de-

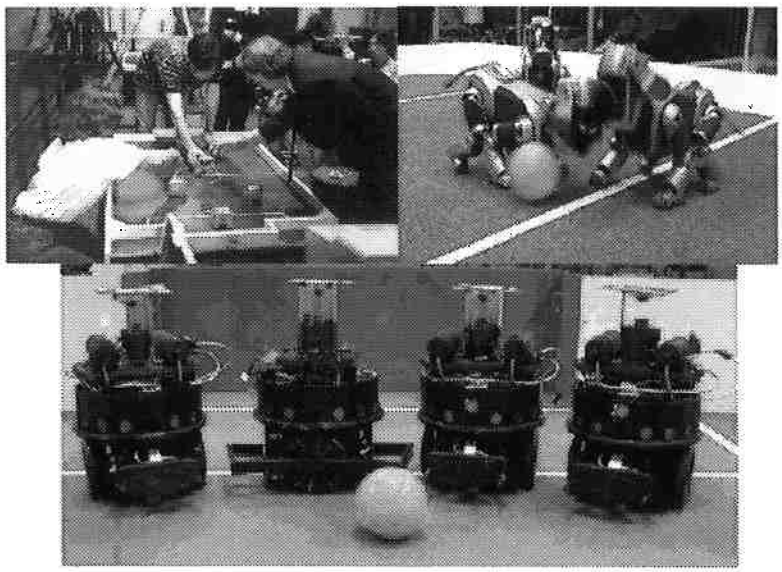
terminado conjunto de comportamentos cresce rapidamente conforme aumenta sua impaciência em relação à não execução, pelos outros membros do time, da subtarefa correspondente àquele conjunto. Por outro lado, se o robô detectar que outro robô está trabalhando naquela subtarefa, sua impaciência deve crescer numa taxa bem mais baixa por um certo período de tempo e deve, durante este tempo, cuidar de outras subtarefas. Isto previne que ações sejam replicadas pelos membros do time. Caso seja atingido o limite de impaciência de um robô relativamente à execução de uma subtarefa, este deve então ativar o respectivo conjunto de comportamentos para que possa executar a referida subtarefa.

Uma motivação complementar e que atua de forma similar à impaciência é a aquiescência. Durante a execução de uma subtarefa, o desejo do robô de desistir de tal execução cresce conforme seus sensores indicam que a sua subtarefa não está sendo propriamente realizada. Após um período de tempo aceitável, se a subtarefa ainda não foi realizada com sucesso, o robô desiste dela e procura executar alguma outra, onde ele possivelmente será mais produtivo. O robô também pode desistir de realizar uma subtarefa (antes do período de tempo dito aceitável), caso detecte que outro robô começou a execução da mesma (pois o nível de impaciência do outro robô ultrapassou seu limite).

Desta forma, a motivação de cada robô muda dinamicamente, adequando-se às condições de execução da tarefa como um todo pelo time. Note que os parâmetros que controlam os níveis motivacionais de cada robô podem ser adaptados automaticamente. ALLIANCE foi demonstrada na implementação de um time de robôs móveis que executava, em laboratório, uma versão de uma tarefa de limpeza de lixos perigosos (PARKER, 1998).

7.4 Futebol de Robôs

O domínio de futebol de robôs é um domínio bastante motivador e que tem atraído grande interesse, no mundo todo, desde que foi proposto em 1995 (KITANO, 1995). Diferentes modalidades de competição foram propostas (micro-robôs, cães robôs, humanóides, simulador, etc), visando estimular pesquisas em diversas áreas, tais como robótica, sensores, fusão sensorial, controle inteligente, inteligência artificial, cooperação entre times, entre outras (Figura 7.2).



Exemplo de competições de futebol de robôs: 1. Esquerda superior: FIRA MiRoSoT, 2. Direita superior: RoboCup - Liga *Legged*, com cães robôs, 3. Inferior: RoboCup - Liga *Middle-size*.

Figura 7.2: Exemplo de competições de futebol de robôs.

O futebol de robôs é uma tarefa bastante complicada devido a basicamente dois motivos:

1. A dinâmica dos jogos dificulta tanto a definição prévia da organização dos robôs quanto a centralização do controle do jogo.

2. As atuações do time oponente são imprevisíveis e, portanto, exigem um alto nível de adaptação em tempo real.

Nas competições tanto da FIRA (*Federation of International Robot-Soccer Association*) quanto da RoboCup pode-se encontrar times com controle centralizado e com controle distribuído. Por exemplo, na Liga MiRoSoT (*Micro-Robot Soccer Tournament*), cada time é constituído por 3 robôs, com dimensões que não excedam 7,5cm X 7,5cm X 7,5cm, sistema de visão global (uma câmera colocada a 2m do campo capta a imagem de todo o campo) e um único computador central que envia comandos aos robôs via rádio. Normalmente, os times que jogam nesta Liga possuem controle centralizado, que definem a atuação de cada robô de acordo com a informação adquirida do sistema de visão computacional, previsão de movimentação do adversário e tática de jogo adotada. Os times são heterogêneos,

com os robôs assumindo os papéis de goleiro, atacante e defensor.

No time Guaraná, vice-campeão mundial da Liga em 1998 (COSTA; PEGORARO, 2000), os robôs possuem hardware idênticos, porém o atacante e o defensor podem trocar de papéis caso o defensor detecte que está em condições adequadas para atacar. O time pode também mudar de tática durante o jogo: caso esteja perdendo por uma diferença grande de gols, o time atua de forma mais ofensiva, com dois atacantes; da mesma forma, se estiver ganhando, passa a usar uma "retranca", atuando com dois defensores.

Por outro lado, as competições das Ligas *Middle-Size* e *Legged* impõem um controle distribuído aos times. Cada robô é constituído por sensores locais, que fornecem informações parciais do ambiente, e por processadores embarcados que possibilitam raciocínio e tomadas de decisão individuais. Os robôs podem se comunicar de forma explícita ou não. Geralmente, os times adotam uma arquitetura híbrida deliberativa/reativa em cada membro.

A cada ano novos desafios são introduzidos nas regras das competições, procurando incentivar e propiciar progressos na área.

8 Aprendizado e Adaptação

Aprendizado possibilita que um agente estabeleça um procedimento, uma habilidade ou conhecimento, não disponíveis na etapa de projeto. Ele permite introduzir novos conhecimentos, fatos, comportamentos, regras no agente, generalizar conceitos, aumentar a eficiência sensorial, aprimorar políticas de comportamentos, criar explicações de como um processo funciona, reutilizar experiências passadas, etc. Assim, o aprendizado pode ser visto como sendo um processo de modificação de parâmetros do agente, de modo a maximizar uma medida de desempenho.

O aprendizado em robôs consiste, portanto, em fazer com que o robô execute tarefas, com sucesso, sem a necessidade de programá-los explicitamente. Esta programação pode ser muito difícil ou mesmo impossível de ser realizada na etapa de projeto por diversas razões: pode haver conhecimento incompleto sobre a tarefa, agente ou ambiente; a tarefa, agente ou ambiente podem mudar com o decorrer do tempo, de forma imprevisível; o projetista pode não ser capaz de implementar uma estrutura de controle adequada, fixa, na etapa de projeto, devido à percepção diferente que humanos e robôs têm do mundo – conhecida como o *problema da discrepância perceptual.* Por estas razões há grande interesse em adicionar a capacidade de aprendizado em robôs móveis.

As técnicas de aprendizado podem ser categorizadas, em função da informação disponível para treinamento, em três classes: aprendizado supervisionado, não-supervisionado e por reforço.

No aprendizado supervisionado, define-se um conjunto de treinamento de entradas e correspondentes saídas desejadas. A distância entre a saída atual e a desejada serve como uma medida de erro que é usada para corrigir os parâmetros do robô aprendiz, minimizando este erro. Um exemplo típico desta classe são

as redes neurais do tipo *Perceptron Multi-Camadas* treinada com o algoritmo de *Retropropagação de Erro*.

No aprendizado não supervisionado, o aprendiz atualiza seus parâmetros sem o uso de pares entrada-saída desejados e sem indicação sobre a adequação das saídas produzidas. O aprendizado se dá por observação das saídas produzidas ou da saída desejada (nunca as duas simultaneamente) ou por determinação de propriedades e regularidades nos dados sensoriais. As técnicas *k-Means* de classificação não supervisionada são exemplos desta classe.

Finalmente, no aprendizado por reforço, para cada entrada apresentada, é produzida uma indicação (reforço) sobre a adequação das saídas correspondentes produzidas pelo robô. Não há medida explícita entre saídas atual e desejada. O aprendiz busca maximizar os reforços recebidos. O algoritmo *Q-learning* é um exemplo clássico desta classe.

8.1 Redes Neurais Artificiais

As Redes Neurais Artificiais – RNA, também chamadas de arquiteturas conexionistas de computação, são algoritmos matemáticos capazes de aprender mapeamentos entre entradas e saídas através de aprendizado supervisionado ou de agrupar dados de entrada em classes através de aprendizado não supervisionado.

RNAs são compostas por diversas unidades de processamento independentes – os neurônios artificiais – interconectadas entre si, trabalhando de forma concorrente, distribuída, paralela. Devido à sua capacidade de aprender mapeamentos de entradas em saídas, de generalizar informação de entrada, de interpretar (classificar) dados de entrada sem supervisão, sua resistência a ruído e sua robustez a falhas de neurônios individuais, a RNA é extremamente atraente para uso em robótica.

8.1.1 Neurônio Padrão

RNAs são baseadas na estrutura e comportamento do sistema nervoso. A estrutura básica desse sistema é o neurônio, que desempenha o papel de difusor de impulsos elétricos. A propagação do impulso ocorre através das sinapses (pontos

de contato entre as terminações de neurônios). Este modelo foi transferido para uma estrutura computacional, onde cada neurônio se torna um processador e a cada informação trocada entre esses processadores está associado um peso (sinapse). A Figura 8.1 apresenta um modelo computacional simples de neurônio que tem sido muito utilizado, onde x denota as entradas, y é a saída do neurônio e f é uma função de ativação ou transferência.

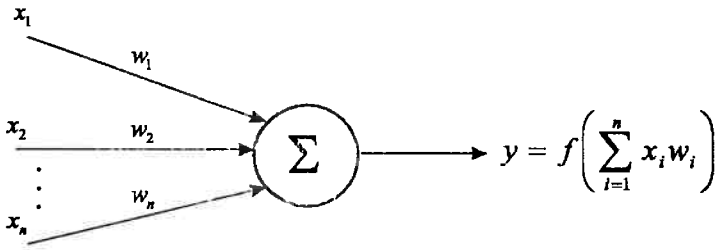


Figura 8.1: Representação de um Neurônio Padrão.

8.1.2 Perceptron e Regra Delta

Suponha que a função de ativação f para o neurônio da Figura 8.1 seja a função limiar $f(.) = 1$ se $\sum_{i=1}^{n} w_i x_i \geq \textit{limiar}$ e $f(.) = -1$ (ou 0) no caso contrário. Assim, o neurônio está ativado ($f(.) = 1$) ou desativado ($f(.) = -1$), produzindo uma saída binária. Uma rede de uma única camada de neurônios deste tipo é denominada rede Perceptron. Suponha que d seja a resposta desejada no tempo t, quando um único perceptron recebe a entrada $x = (x_1, x_2, \ldots, x_n)$, de sinais ponderados pelos respectivos pesos $w = (w_1, w_2, \ldots, w_n)$. Seja y a saída fornecida pelo perceptron em resposta a esta entrada:

$$y(t) = f(\sum_{k=1}^{n} w_k x_k) \tag{8.1}$$

Usualmente, os pesos são inicialmente arbitrários e, provavelmente, a resposta atual y deste neurônio será diferente da resposta desejada d. O erro é dado por: $e(t) = d(t) - y(t)$ e uma medida de erro pode então ser definida como:

$$E[w] = \frac{1}{2}[d(t) - y(t)]^2 = \frac{1}{2}[d(t) - \sum_{k=1}^{n} w_k x_k]^2 \tag{8.2}$$

A função $E[w]$ é maior ou igual a zero e tende a zero quanto mais a resposta da saída do neurônio se aproximar da resposta desejada para o padrão de entrada.

O erro total, E_T, após a apresentação de vários exemplos p de um dado conjunto de treinamento é:

$$E_T = \sum_p E[w] = \frac{1}{2}\sum_p [d^p(t) - y^p(t)]^2 \tag{8.3}$$

O objetivo do aprendizado é minimizar esta medida de erro. Como há informação sobre a resposta desejada, este aprendizado é supervisionado. Embora o erro total E_T seja definido pela soma dos erros para todos os padrões, será assumido, sem perda de generalidade, que a minimização do erro para cada exemplo, dada pela Equação 8.2, levará à minimização do erro total.

Assim sendo, observe que a função de custo depende somente dos pesos sinápticos e dos exemplos apresentados ao neurônio. Então, minimizando-se a função $E[w]$ pode-se obter um conjunto de pesos w_k caminhando em direção oposta ao gradiente de $E[w]$. Usando o método da descida do gradiente (HAYKIN, 1999), os pesos w_k são modificados por uma quantia proporcional ao gradiente de E, isto é:

$$\Delta w_k = -\eta \frac{\partial E[w]}{\partial w_k} \tag{8.4}$$

$$\Delta w_k = -\eta(d(t) - y(t))x_k \tag{8.5}$$

que corresponde às modificações que devem ser realizadas nos pesos em relação ao exemplo p, ou ainda,

$$\Delta w_k = -\eta\delta(t)x_k \tag{8.6}$$

ou

$$w_k(t+1) = w_k(t) + \eta\delta(t)x_k; \quad k = 1, 2, \ldots, p \tag{8.7}$$

onde η é a taxa de aprendizado e $\delta(t)$ é definido por:

$$\delta(t) = (d(t) - y(t)) \tag{8.8}$$

Este resultado dado pelas Equações 8.5, 8.6 e 8.8 é conhecido como *Regra Delta*, regra de Widrow-Hoff ou regra LMS.

8.1.3 Perceptron Multi-Camadas e Retropropagação de Erro

Um perceptron multi-camadas – MLP *"Multi-Layer Perceptron"* – é um conjunto de perceptrons arranjado em diversas camadas, contendo pelo menos uma camada intermediária – também chamada de camada escondida – conforme ilustra a Figura 8.2.

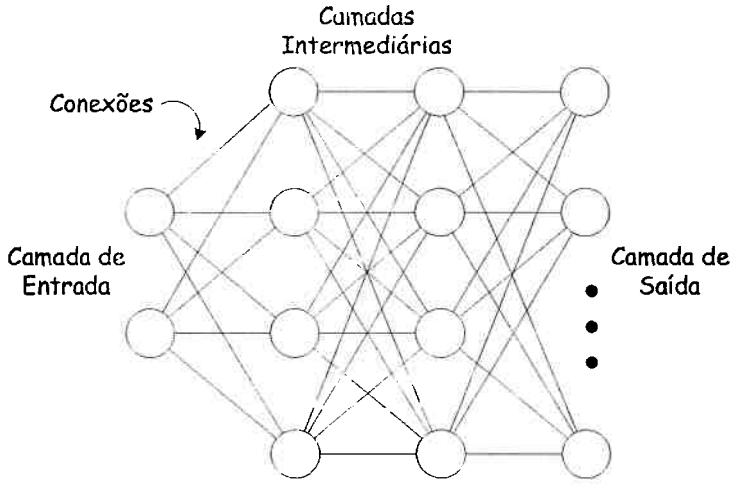


Figura 8.2: Rede Neural Multi-Camadas.

A generalização da Regra Delta, conhecida como algoritmo de Retropropagação do Erro (*Backpropagation*) ou Regra Delta Generalizada, pode ser aplicada a um MLP. Neste caso, a função de custo (erro total a ser minimizado) é dada por:

$$E_T[w] = \frac{1}{2}\sum_p \sum_{i=1}^{m} (d_i^p(t) - y_i^p(t))^2 \tag{8.9}$$

onde m é o número de unidades de saída, d_i é a i-ésima saída desejada e y_i é a i-ésima saída gerada pela rede, no instante t.

Diferentemente da Regra Delta, a Regra Delta Generalizada requer que a função de ativação dos neurônios seja contínua e diferenciável. O cálculo da saída de um neurônio y_j pode ser expresso como:

$$y_j = f_j(net_j^p) \tag{8.10}$$

onde $net_j^p = \sum_k w_{jk} x_k^p$.

Usualmente, os pesos são inicializados aleatoriamente e, através da aplicação do método de descida do gradiente são ajustados pela seguinte regra de atualização, durante o processo de treinamento da rede:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j(t)x_i(t) \tag{8.11}$$

onde: $\delta_j = f'(net_j)(d_j - y_j)$ para um neurônio j na camada de saída, $\delta_j = f'(net_j)\sum_k \delta_k w_{jk}$ para um neurônio j na camada intermediária e $f'(net_j)$ representa a derivada da função de ativação f em relação ao peso w_{ij}.

Os erros são propagados da camada de saída para as camadas anteriores. Maiores detalhes sobre este algoritmo podem ser encontrados em (HAYKIN, 1999).

8.1.4 Mapas Auto-Organizáveis

A rede Perceptron e o MLP, treinados respectivamente pelo algoritmo da Regra Delta e da Retropropagação de Erro, seguem a abordagem de treinamento supervisionado, uma vez que possuem uma medida de erro dada pela diferença entre a saída produzida e a desejada, para cada entrada. A saída desejada é fornecida externamente, por um supervisor.

No entanto, há diversas aplicações em robótica onde se torna necessário a redução da dimensionalidade dos dados, de modo não-supervisionado. Esta redução de dimensionalidade é uma forma de generalização, reduzindo a complexidade do espaço de entrada enquanto retém atributos relevantes deste espaço de entrada. Os mapas auto-organizáveis – SOM *"Self-Organizing Maps"* – também chamados de Redes de Kohonen, constituem um mecanismo que realiza o mapeamento não-supervisionado de um espaço de entrada de alta dimensão para, tipicamente, um espaço bidimensional de saída.

Tipicamente, a SOM consiste de uma grade bidimensional de unidades. Todas as unidades recebem um vetor de entrada i. Inicialmente, todos os vetores de pesos w_j são inicializados aleatoriamente e normalizados para o comprimento da unidade. A saída o_j de cada unidade j na rede é: $o_j = w_j.i_j$

Devido à inicialização aleatória dos pesos, as saídas das unidades irão diferir entre si, sendo que uma unidade irá responder mais fortemente a um particular vetor de entrada. A "unidade vencedora" e sua vizinhança será então treinada

de forma a responder ainda mais fortemente àquele particular vetor de entrada, de acordo com a seguinte função de atualização:

$$w_j(t+1) = w_j(t) + \eta(i - w_j(t)) \tag{8.12}$$

onde η é a taxa de aprendizado.

A vizinhança ao redor da unidade vencedora é, normalmente, definida grande no início do treinamento e com diminuição gradual durante o treinamento.

Conforme o aprendizado prossegue, certas áreas da SOM respondem mais e mais a um certo estímulo de entrada, agrupando, desta forma, o espaço de entrada em um espaço bidimensional. O agrupamento ocorre de forma topológica, mapeando entradas similares em regiões vizinhas no mapa.

8.2 Classificador Não Supervisionado: *k-means*

Considere um problema onde os dados D são um conjunto de exemplos gerados por uma distribuição de probabilidades dada pela mistura de k distintas distribuições gaussianas.

A tarefa do aprendizado consiste em fornecer uma hipótese $h = (\mu_1, \mu_2, \ldots \mu_k)$ que descreva a média de cada uma das k distribuições, dado D. Deseja-se encontrar a hipótese mais provável de tais médias, isto é, a hipótese h que maximize $p(D|h)$.

Para uma única distribuição observada pelos exemplos $x_1, x_2, \ldots x_m$, a hipótese mais provável pode ser dada pela média μ_{ML} que maximize a soma dos quadrados dos erros nos m exemplos de treinamento:

$$\mu_{ML} = argmin_{\mu} \sum_{i=1}^{m} (x_i - \mu)^2 = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{8.13}$$

No entanto, quando se tem uma mistura de k distribuições, o problema se complica. Considere que a mistura seja de duas distribuições gaussianas distintas e que cada exemplo seja dado por (x_i, z_{i1}, z_{i2}), onde x_i é o valor observado do i-ézimo exemplo, z_{i1} e z_{i2} indicam qual das duas distribuições foi usada para gerar o valor; $z_{ij} = 1$ indica que x_i foi gerado pela j-ézima distribuição e $z_{ij} = 0$ no

caso contrário. Assim, todos x_i que possuem $z_{ij} = 1$ formam um agrupamento ou partição ("*cluster*") c_j.

A tarefa do aprendizado agora consiste em buscar pela hipótese mais provável por meio de repetidamente reestimar o valor esperado para z_{ij}, dada a corrente hipótese $(\mu_1, \mu_2, \ldots \mu_k)$ e então recalcular a hipótese usando estes valores estimados.

O algoritmo de aprendizado não supervisionado para resolver esta tarefa com a mistura de duas distribuições inicia com uma hipótese arbitrária $h = (\mu_1, \mu_2)$ e executa:

Passo 1 Calcula o valor esperado $E[z_{ij}]$ de cada variável z_{ij}, dada a hipótese corrente $h = (\mu_1, \mu_2)$:

$$E[z_{ij}] = \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)} \tag{8.14}$$

Passo 2 Calcula nova hipótese $h' = (\mu'_1, \mu'_2)$, dados os valores esperados de z_{ij} calculados no Passo 1. Assumir $h' = (\mu'_1, \mu'_2)$ como hipótese corrente e repetir Passo 1, até atingir convergência.

$$\mu'_j = \frac{\sum_{i=1}^{m} E[z_{ij}] x_i}{\sum_{i=1}^{m} E[z_{ij}]} \tag{8.15}$$

Pode-se provar que, em cada iteração destes passos, o algoritmo aumenta a probabilidade $p(D|h)$ até que atinja um máximo local, determinando a hipótese $h = (\mu_1, \mu_2)$ (MITCHELL, 1997). O procedimento descrito constitui a base do algoritmo de classificação não supervisionada conhecido por *k-means*.

8.3 Aprendizado por Reforço

Considere um agente aprendiz interagindo com o ambiente: em cada passo de interação, o aprendiz observa o estado corrente s do ambiente e escolhe a ação a para realizar. A ação a altera o estado s do ambiente e um sinal escalar de reforço r (penalização ou recompensa) é fornecido ao aprendiz, indicando quão desejável é o estado resultante. O objetivo de um agente que aprende por reforço, na maioria das formulações de problemas que usam o Aprendizado por Reforço

(AR), é aprender uma política ótima de atuação que maximize a soma cumulativa esperada do sinal de reforço, para qualquer estado inicial.

O problema de decisão seqüencial a ser solucionado pode ser modelado como um Processo Markoviano de Decisão – MDP (*"Markov Decision Process"*). Formalmente, um MDP é uma quádrupla $\langle S, A, R, T \rangle$ onde (KAELBLING; LITTMAN, 1996):

- S é um conjunto finito de estados,
- A é um conjunto finito de ações,
- R é uma função escalar de recompensa, $R : S \times A \rightarrow \Re$,
- T é uma função de transição de estados, $T : S \times A \rightarrow \Pi(S)$, onde um membro de $\Pi(S)$ é uma distribuição de probabilidades sobre S. $T(s, a, s')$ representa a probabilidade de transitar do estado s para s' com a execução da ação a.

O objetivo do aprendiz consiste em determinar a política $\pi : S \rightarrow A$ que maximiza o reforço recebido, ao longo do tempo.

Considere o valor cumulativo (descontado) $V^{\pi}(s_t)$ determinado por seguir uma política arbitrária π a partir de um estado inicial arbitrário s_t, dado por:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{8.16}$$

sendo r_{t+i} a seqüência de reforços recebidos a partir de s_t, usando repetidamente a política π para selecionar ações, e γ o fator de desconto, com $0 \leq \gamma < 1$. Outras definições, que não esta com desconto, são possíveis.

O aprendiz, no entanto, tem por objetivo aprender a política ótima π^* que maximiza $V^{\pi}(s)$, para todo estado $s \in S$:

$$\pi^* = argmax_{\pi} V^{\pi}(s), \forall s \tag{8.17}$$

No estado s_t, a política ótima $\pi^*(s_t)$ será escolher a ação a_t que maximize a soma do reforço $r(s_t, a_t)$ com o valor $V^{\pi^*}(s_{t+1})$ do estado sucessor, descontado de γ:

$$\pi^*(s_t) = argmax_a [r(s_t, a_t) + \gamma V^{\pi^*}(s_{t+1})] \tag{8.18}$$

Define-se a função de estimação Q como sendo a soma do reforço recebido com o valor (descontado de γ) de seguir a política ótima daí por diante:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi^*}(s_{t+1}) \tag{8.19}$$

com

$$\pi^*(s) = argmax_a Q(s, a) \tag{8.20}$$

e

$$V^{\pi^*}(s) = max_a Q(s, a) \tag{8.21}$$

8.3.1 O Algoritmo *Q-learning*

Como foi mostrado, aprender a função Q corresponde a aprender a política ótima. Assim, reescrevendo a equação 8.19, vem:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma max_a Q(s_{t+1}, a) \tag{8.22}$$

Esta definição recursiva de Q fornece a base para o algoritmo $Q - learning$, que iterativamente aproxima Q. A regra $Q - learning$ é:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{8.23}$$

onde s é o estado corrente, a é a ação realizada em s, $r(s, a)$ é o reforço recebido após realizar a em s, s' é o novo estado, γ é o fator de desconto ($0 \leq \gamma < 1$), e α é a taxa de aprendizagem ($0 < \alpha < 1$), podendo ser definida por $\alpha = 1/(1 + visitas(s, a))$, sendo $visitas(s, a)$ o número de visitas já feitas ao estado s, com ação a escolhida e realizada.

Note que as ações de treino, efetuadas durante o processo iterativo de aproximação da função Q, podem ser escolhidas livremente. A convergência do $Q - learning$ é garantida (MITCHELL, 1997), porém esta convergência é extremamente lenta.

Parte II

Pesquisas e Desenvolvimentos

9 Pesquisa e Obra da Autora

Nesta segunda Parte desta tese, composta pelos Capítulos 10, 11, 12 e 13, procura-se dar uma visão panorâmica da pesquisa e obra da autora, desenvolvidas nos últimos dez anos, através da investigação teórica, da experimentação e da simulação.

Em seu trabalho, a autora adota uma visão integradora de paradigmas e tecnologias, abordando um espectro concreto de temas relevantes para a Robótica Móvel Inteligente, a saber: percepção, arquiteturas, navegação, controle, aprendizado e uso de múltiplos robôs.

Na percepção para robôs móveis, as contribuições desenvolvidas até o momento pela autora e seu grupo residem basicamente na Visão Computacional, por esta oferecer uma das percepções mais poderosas aos robôs: a percepção visual. As pesquisas e desenvolvimentos realizados, descritos no Capítulo 10, abordam um dos tópicos mais difíceis do processamento de imagens, que é a segmentação de imagens.

A segmentação é responsável por particionar uma imagem em regiões que correspondam a objetos de interesse na cena imageada. Nesta linha, a autora e seu grupo exploram diversas técnicas que envolvem a propriedade cor dos objetos, a modelagem e extração de cenas de fundo, a detecção de movimento na cena através da análise de seqüências de imagens, incluindo, em algumas delas, técnicas de aprendizado autônomo para aumentar a eficiência, autonomia e qualidade das respostas.

Outra habilidade de extrema importância na percepção é a capacidade de determinar a distância dos objetos na cena em relação ao observador. Esta habilidade pode ser conseguida por meio da estereoscopia binocular. Esta linha também é explorada pela autora, através da proposta e desenvolvimento de técnicas de

calibração e de reconstrução tridimensional, visando precisão nas medidas.

Para o projeto e desenvolvimento de robôs móveis, a autora propõe o uso do paradigma comportamental por este oferecer maiores facilidades e flexibilidade de projeto, construção, manutenção e modificação dos robôs, além de serem bastante adequados à atuação em tempo real. Assim, arquiteturas para comportamentos foram investigadas, tanto arquiteturas reativas com coordenação competitiva e cooperativa dos comportamentos, quanto artuiteturas híbridas onde um módulo de controle baseado em aprendizado por reforço foi proposto e desenvolvido. Estas contribuições são descritas no Capítulo 11.

Na navegação e controle, os principais tópicos envolvidos são abordados na pesquisa e obra da autora: a construção autônoma e uso de mapas do ambiente; a determinação autônoma da localização do robô no ambiente; a navegação direcionada por marcos visuais de referência; e o controle para navegação e atuação por meio de aprendizado por reforço. Estas contribuições são descritas no Capítulo 12. O uso de aprendizado por reforço na tarefa de navegação e controle da atuação de robôs móveis se mostra bastante apropriado, uma vez que explora, na sua formulação e modelagem, as repetidas interações do robô com o ambiente.

Finalmente, no Capítulo 13, a autora explora o uso de múltiplos robôs na execução de tarefas mais complexas, que possuam características inerentemente distribuídas, tanto em espaço, quanto em tempo ou funcionalidade. Situações de jogos são investigadas, resultando numa importante contribuição da autora e seu grupo na área de aprendizado multi-agentes, com a proposta de um novo algoritmo, Minimax-QS, que explora o uso de generalização da experiência em ambientes de jogos com dois jogadores, soma zero, através do uso do conhecimento *a priori* do domínio. Este algoritmo tem prova de convergência ao equilíbrio, sob condições apropriadas.

Todas as pesquisas e desenvolvimentos realizados foram aplicados nas mais diversas áreas, incluindo não só a navegação de robôs móveis, como também o futebol de robôs, a montagem em células flexíveis de montagem utilizando manipuladores, a classificação automática de laranjas, a segmentação e análise dos mais diversos tipos de imagens, o monitoramento de terminais rodoviários e de plataformas metroviárias, a medição autônoma de placas e painéis, o descarregamento de navios. Estas aplicações são descritas nesta Parte da tese. As platafor-

mas de hardware utilizadas nos experimentos realizados encontram-se descritas no Apêndice I.

10 Visão Computacional

Uma das habilidades perceptivas mais completa e que fornece uma rica variedade de informação é a visão computacional, que possibilita a percepção visual a máquinas (e robôs). A autora e seu grupo têm dedicado grande esforço nesta área, uma vez que, para atingir autonomia, um robô precisa de boa e ampla capacidade perceptiva.

As pesquisas e desenvolvimentos realizados concentram-se principalmente na segmentação de imagens, na detecção de movimento na cena e na estereoscopia binocular, muitas vezes incluindo técnicas de aprendizado autônomo nas abordagens realizadas.

Na segmentação de imagens, a propriedade cor é explorada para efetuar a classificação de pixels como pertencentes ou não a uma determinada classe de cor, para posteriormente efetuar procedimentos de conexão de pixels adjacentes que possuam as mesmas cores. Isso porque objetos na cena normalmente estão relacionados a pixels adjacentes, que compartilham propriedades similares. Contribuições da autora e seu grupo na segmentação de imagens por cores encontram-se descritas na Seção 10.1 e as aplicações das técnicas propostas, na Seção 10.5.

O objetivo final da segmentação de imagens consiste em particionar a imagem em unidades que correspondam a objetos de interesse na cena. Segmentação de imagens é uma tarefa bastante difícil, pois envolve diversos conceitos e processos: a radiometria da formação das imagens, a geometria envolvida no processo de imageamento, as propriedades e características da cena e seus objetos constituintes. Assim, outra forma de se extrair os conjuntos de pixels que correspondam a objetos de interesse na cena é por meio da definição de um modelo de cena de fundo para posterior extração do mesmo das imagens, restando, desta forma,

somente os objetos de interesse. Esta abordagem é explorada na Seção 10.2, com o uso de modelos fixos e modelos adaptativos de cena de fundo. Aplicações das técnicas propostas encontram-se descritas na Seção 10.5.

A abordagem dual desta última também foi explorada pela autora, onde os objetos de interesse estão em movimento na cena e, assim, procura-se extrair os conjuntos de pixels correpondentes a estes objetos em movimento através da análise de uma seqüência de imagens da cena, detectando movimento na cena imageada. As técnicas exploradas para este fim são descritas na Seção 10.3 e as aplicações desenvolvidas são apresentadas na Seção 10.5.

Finalmente, a estereoscopia binocular tem sido estudada, pois permite que se estime a distância de objetos de interesse em relação ao observador. As formulações dos métodos de calibração do sistema de estereoscopia binocular e as técnicas envolvidas na reconstrução tridimensional do espaço da cena são descritas na Seção 10.4. Aplicações com uso de estereoscopia binocular e descrição do processo de calibração usado são apresentados na Seção 10.5.

10.1 Segmentação por Cores

Segmentar uma imagem consiste em agrupar partes (pixels) desta imagem em unidades que sejam homogêneas em relação a uma ou mais características (atributos ou propriedades). A segmentação por cores define que a homogeneidade de tais unidades deve ser em relação às suas cores: pixels de mesma cor devem ser agrupados nas mesmas unidades. Desta forma, o processo de classificação de cada pixel na imagem como pertencente ou não a um certo grupo (ou classe) que possui determinada cor é tarefa de extrema importância na segmentação.

Normalmente, uma restrição extra, conectividade, também é estipulada: essas partes com mesmas características, para serem agrupadas numa mesma unidade, devem ainda ser vizinhas entre si na imagem. Esta restrição vem do fato de que o que se deseja, no processo de segmentação de uma imagem, é dividi-la em partes constituintes que representem objetos ou propriedades da cena. O nível no qual esta segmentação é conduzida depende da aplicação. Assim, o processo de segmentação deve parar logo que os objetos de interesse em uma aplicação tiverem sido separados. Normalmente, o processo autônomo de segmentação de

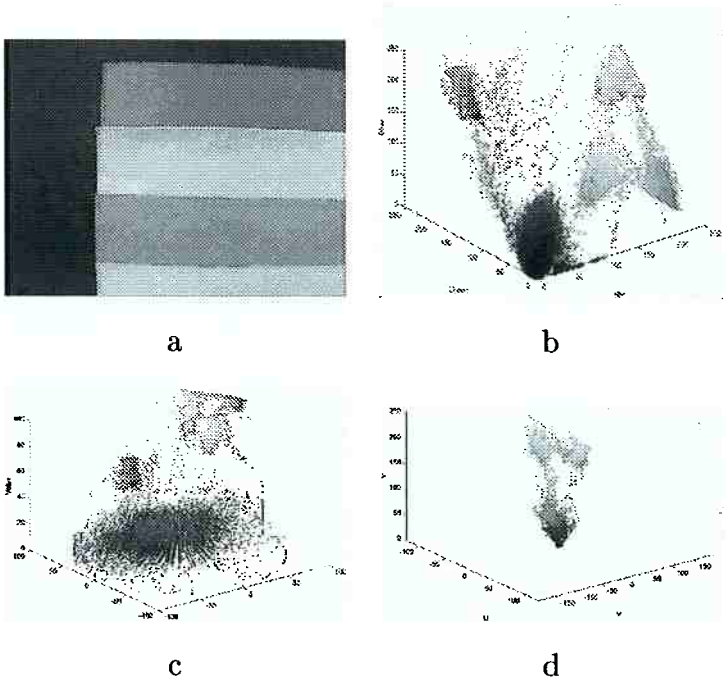


Figura 10.1: Exemplo de (a) uma imagem representada no (b) espaço RGB de cores, (c) espaço HSV e (d) espaço YUV.

imagens é uma das tarefas mais difíceis do processamento de imagens.

10.1.1 Classificacão de cores

Embora aparentemente imediato, o conceito que envolve a palavra cor esconde muitas noções de ordem física e psicológica, o que torna a cor um conceito difícil de modelar. A classificação consiste em particionar um espaço n-dimensional, dado pela transformação da imagem de interesse num espaço apropriado de representação de cores.

Os dispositivos digitais para representação de imagens coloridas utilizam o princípio da tricromaticidade, onde quase todas as cores podem ser reproduzidas como combinação linear de três cores básicas, chamadas cores primárias. Os espaços tricromáticos de representação de cores mais usados são o *Hue Saturation Value* (HSV), o *Red Green Blue* (RGB) e o YUV. A Figura 10.1[1] ilustra a distribuição das cores nestes três espaços.

A escolha do espaço de cores adequado para classificação depende de diversos

[1]Veja esta figura colorida no Apêndice II.

fatores, incluindo disponibilidade do hardware de digitalização empregado e da utilidade para uma aplicação particular.

A classificação então se resume a decidir se um pixel é membro de uma classe de cor particular. Diversos métodos propõem maneiras diferentes de definir os limites e formas destas classes, procurando adequá-las melhor à distribuição das cores no espaço de cores utilizado, considerando as cores de interesse.

Imposição de Limiares

A abordagem conhecida como técnica de imposição de limiares é uma das abordagens para particionamento do espaço de cores, que define cada classe de cor como um bloco retangular (paralelepípedo) no espaço de cores. Esta abordagem oferece bom desempenho, mas não explora potenciais dependências entre as dimensões do espaço de cores.

Na técnica de imposição de limiares, cada classe de cor é especificada como um conjunto de seis valores de limiar, dois para cada dimensão do espaço de cores. Assim, um ponto (x, y) é classificado como fazendo parte de uma determinada classe de cor se $\vec{T_1} < \vec{f}(x, y) \leq \vec{T_2}$, onde $\vec{T_1}$ e $\vec{T_2}$ representam, respectivamente, os limiares inferior e superior que definem a classe, em cada uma das três dimensões do espaço de cores, e $\vec{f}(x, y)$ indica o valor do ponto (pixel) (x, y), em cada dimensão.

Classificação por Redes Neurais

Outra abordagem bastante explorada pela autora e seu grupo é a classificação por Redes Neurais.

Utilizou-se a rede neural artificial do tipo Percentron Multicamada (MLP), treinada com o algoritmo de Retropropagação de Erros (veja Capítulo 8), como ferramenta para a classificação de cores por apresentar algumas características de grande relevância: pode aproximar qualquer função (MITCHELL, 1997); é capaz de aprender através de exemplos sem a necessidade de modelagem matemática do sistema; é dotada de poder de generalização para situações não treinadas; é robusta a erros de treinamento.

Embora o MLP treinado com o algoritmo de retropropagação de erros ofereça

características interessantes para aplicação em tarefas de classificação, os grupos de cores podem se apresentar mais ou menos propícios à separação de acordo com o sistema de representação de cores utilizado. Outras características relevantes, como o tipo de generalização encontrada e o tempo de treinamento necessário para a convergência da rede, devem ser observados.

Assim, foram realizados estudos detalhados buscando um sistema de classificação de cores que fosse robusto à variação de iluminação da cena e de brilho da cor (saturação da cor), além de ser tolerante a erros nas amostras de treinamento. As questões levantadas referiam-se a qual espaço de representação de cores seria o mais adequado, RGB, HSV ou YUV; qual a arquitetura do MLP seria mais adequada à tarefa de classificação de cores e como deveria ser o treinamento, quantos exemplos deveriam ser fornecidos (SIMÕES, 2000; SIMÕES; COSTA, 2000a; SIMÕES; COSTA, 2000b; SIMÕES; COSTA, 2000c; SIMÕES; COSTA, 2001).

Os testes foram realizados utilizando duas imagens como padrões, mostradas na Figura 10.2[2]: uma paleta com cores contínuas e uma paleta com cores em degradê, para representar as diferenças de saturação da cor.

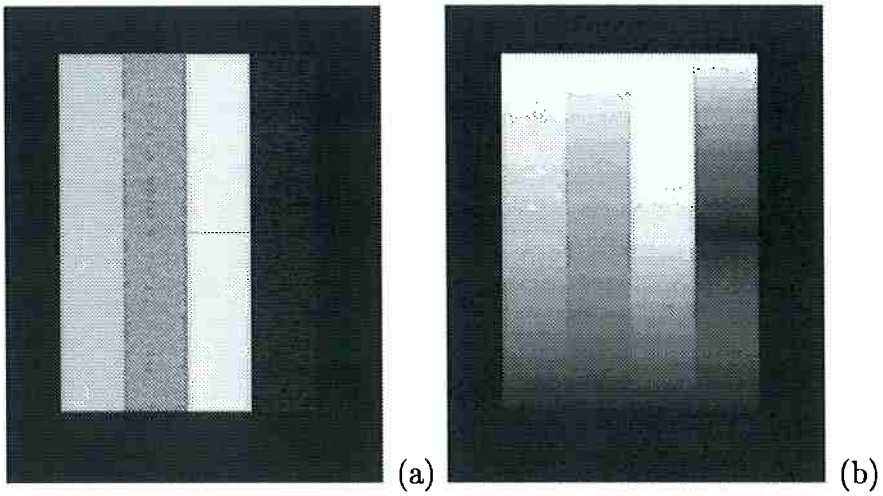
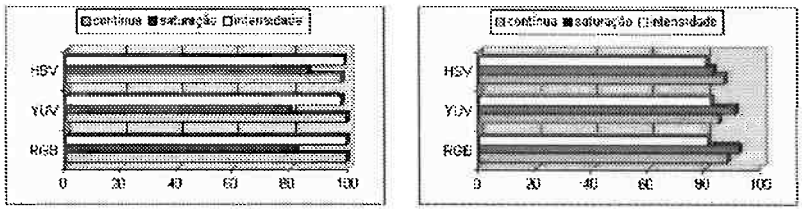


Figura 10.2: Imagens utilizadas nos experimentos: (a) paleta contínua e (b) paleta degradê.

Foram efetuados diversos testes variando o número de neurônios na camada escondida do MLP, variando o sistema de representação de cores, o número de amostras de treinamento, o tipo de amostras do treinamento (com saturações

[2]Veja esta figura colorida no Apêndice II.



a b

Porcentagem média de acerto das redes neurais, para cada sistema de representação de cores (HSV, YUV e RGB) e para cada forma de treinamento, utilizando para validação a) a imagem da paleta contínua e b) a imagem da paleta degradê.

Figura 10.3: Porcentagem média de acerto das redes neurais.

bastante variadas ou não, e com iluminação variada ou não). A rede usada possui três neurônios na camada de entrada, um para cada dimensão do espaço de representação de cor do pixel utilizado (RGB, HSV ou YUV). Variando-se o número de neurônios na camada escondida, a rede não apresentou muita alteração na qualidade da resposta quando este número ficou na faixa de dez a vinte neurônios. A rede, no estudo realizado, continha cinco neurônios na camada de saída, uma para cada cor das paletas: preta, rosa, laranja, amarela e azul. Estas cores foram definidas de acordo com as regras dos campeonatos de futebol de robôs autônomos.

A porcentagem média de acerto do MLP com 12 neurônios na camada oculta e 25 exemplos de treinamento, retirados da paleta de cores contínua (faixas inferiores), degradê (faixas centrais) e com variações na iluminação (faixas superiores), para cada sistema de representação de cores (HSV, YUV e RGB), é apresentada na Figura 10.1.1

Concluiu-se que, ao contrário das expectativas, o oferecimento de exemplos simples (saturação média de cada cor-exemplo, sob condições normais de iluminação – por volta de 1000 a 1500 lux) das cores foi a melhor solução.

Aplicações destes resultados no domínio de futebol de robôs e de classificação de laranjas são apresentadas na Seção 10.5.4.

Classificação por Estimação Nebulosa

As imagens digitais são constituídas por regiões nem sempre bem definidas, portanto situações incertas na classificação ocorrem na análise da imagem. Uma decisão tomada em uma fase do processamento terá conseqüências nas fases seguintes. Uma representação adequada da incerteza permite a manutenção de mais informações durante todo o processamento. A técnica de agrupamento nebuloso permite efetuar uma segmentação baseada em classificação de pontos, de forma que pontos possam pertencer a mais de uma região da imagem com o devido grau de pertinência – entre 0 (certamente não pertence) e 1 (certamente pertence).

A técnica de agrupamento nebuloso mais empregada é uma derivação do algoritmo *k-means* (veja Seção 8.2), o *Fuzzy C-Means* (FCM). Em ambos, a medida de similaridade normalmente empregada é a distância euclidiana entre dois pontos no espaço de atributos escolhido:

$$d_{ik}^2 = (\vec{v}_i - \vec{x}_k)^2 \tag{10.1}$$

onde $\vec{v}_i$ é a posição do centro do grupo (classe) e $\vec{x}_k$ é o valor do elemento no espaço do atributo considerado. Assim, busca-se minimizar esta distância em relação a todos os pontos do mesmo grupo. A função a ser minimizada é:

$$J(\vec{v}, U) = \sum_{i=1}^{c} \sum_{k=1}^{N} (u_{ik}^m (\vec{v}_i - \vec{x}_k)^2) \tag{10.2}$$

onde U é o conjunto de partições nebulosas a ser obtido, u_{ik} é o valor de pertinência do ponto k no grupo i, c é o número de grupos pré-determinado, N é o número de pontos no espaço de atributos e m chamado *fator de nebulosidade*, cuja influência está no grau de superposição dos grupos, onde $1 \leq m \leq \infty$. Para $m \sim 1$, a partição resultante é binária, e para $m \to \infty$, não há partição resultante – todos os elementos pertencem a todos os grupos, com grau de pertinência $1/c$.

No *k-means* clássico (binário), a pertinência u_{ik} é estritamente 0 ou 1; mas no caso nebuloso varia continuamente de 0 a 1. Esta minimização ainda deve possuir os seguintes vínculos:

$$\sum_{i=1}^{c} u_{ik} = 1 \quad \text{e} \quad 0 < \sum_{k=1}^{N} u_{ik} < N \tag{10.3}$$

O cálculo dos centros $\vec{v}_i$ e dos valores de pertinência u_{ik} dependem iterativamente uns dos outros. Então, é necessária uma inicialização das posições dos centros e dos valores de pertinência na primeira iteração. Geralmente, não há *a priori* nenhuma informação sobre a natureza das partições e dos centros, sendo comum uma inicialização aleatória. Algumas considerações heurísticas podem melhorar a convergência do algoritmo, pois os centros vão se deslocando da sua posição aleatória inicial para o seu valor verdadeiro durante as iterações.

No trabalho desenvolvido (BONVENTI Jr.; COSTA, 2002), a imagem é representada no espaço RGB. Observou-se que a distribuição de pontos neste espaço muitas vezes é alongada, sendo melhor a aproximação por elipsóides para caracterizar os grupos. Assim, para esta caracterização, optou-se por usar a distância de Mahalanobis como medida de similaridade.

Para avaliar a qualidade dos grupos resultantes do FCM, adotou-se um índice dado pela relação entre a distância média dos dados aos seus respectivos centros e a mínima distância entre os centros. Uma boa definição dos grupos deve minimizar este índice, correspondendo a grupos mais compactos e mais separados. A compactação é dada por:

$$Comp = \frac{J(\vec{v}, U)}{N} = \frac{1}{N} \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^2 d_{ik}^2 \tag{10.4}$$

e a separação:

$$Sep = min_{i,k}(\vec{v}_i - \vec{x}_k)^2 \tag{10.5}$$

e o índice a ser minimizado é:

$$S = \frac{Comp}{Sep} \tag{10.6}$$

com o número ótimo de grupos c^* dado por $c^* = min_c S$, com $c = 2, \ldots, c_{max-1}$.

O estudo desenvolvido resultou no estabelecimento de uma seqüência de procedimentos para realizar o agrupamento nebuloso de acordo com o melhor número de grupos e representá-los no espaço RGB, descrito no Algortimo 1.

Esta seqüência de procedimentos proposta é relativamente custosa em termos computacionais, mas este aspecto foi considerado menos importante, em vista da qualidade do resultado final (veja Seção 10.5.5).

Ler o arquivo da imagem
Identificar cada ponto e seu valor RGB de cor;
para o número de grupos $c = 2 \ldots c_{max}$ **faça**
  Aplicar o algoritmo FCM até sua convergência;
  Calcular fator $S = Comp/Sep$ para o agrupamento realizado;
fim-para
Obter o melhor número de grupos c^* proveniente do menor S.

ALGORITMO 1: Agrupamento Nebuloso de Padrões de Cores.

10.1.2 Agrupamento em Regiões

O objetivo das técnicas de agrupamento em regiões é usar características da imagem para mapear pixels individuais de uma imagem de entrada em conjuntos de pixels, denominados regiões. Uma região da imagem geralmente corresponde a um objeto do mundo real (ou parte significativa deste) e, desta forma, resulta em uma área bidimensional conectada na imagem – isto é, corresponde a um agrupamento de pixels conectados.

Um pixel p_i de uma região R é dito *conectado* a p_j se existir uma seqüência $\{p_i, \ldots, p_j\}$ de tal forma que p_k e p_{k+1} são conectados e todos os pixels pertencem a R. R é uma região conectada se todo par de pixels é conectado no conjunto de pixels em R.

A imagem completa $I = \bigcup_{k=1}^{m} R_k$, com $R_i \bigcap R_j = \emptyset$ e $i \neq j$. Estas regiões (disjuntas) são chamadas partições e correspondem a áreas homogêneas, isto é, para alguma função booleana $H(R)$ que mede a homogeneidade da região:

$$H(R_k) = \begin{cases} true & \text{se todo par } (p_i, p_{i+1}) \text{ de pixels adjacentes} \\ & \text{em } R \text{ satisfaz } f(p_i) - f(p_{i+1}) < T. \\ false & \text{em caso contrário.} \end{cases}$$

e $H(R_i \bigcup R_j) = false$, sendo $i \neq j$ e R_i e R_j regiões vizinhas, $f(p)$ o valor do pixel p na imagem e T um valor de limiar. No caso de segmentação por cor, $f(p)$ corresponde a um vetor $\vec{x}_i = \{C_i^1, C_i^2, C_i^3\}$, sendo i a posição (x, y) do pixel p na imagem e C^j cada uma das três dimensões da representação tricomática da imagem, correspondendo à cor do pixel p.

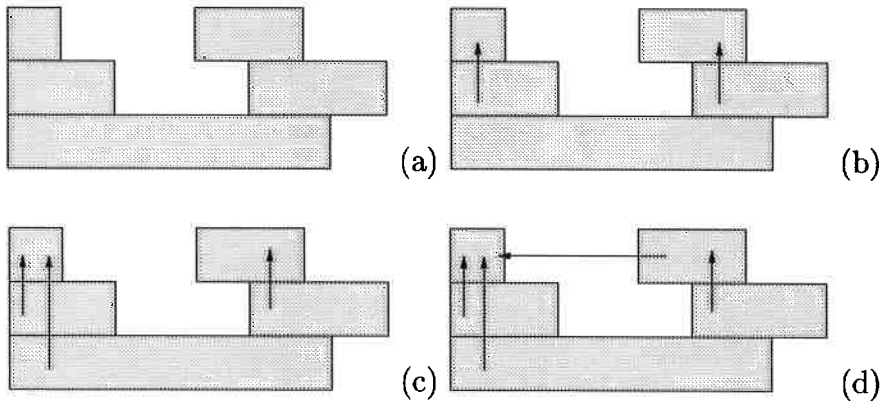
Um algoritmo típico de agrupamento em regiões consiste em percorrer sistematicamente a imagem, classificando a cor de cada pixel e agrupando pixels

adjacentes com mesma cor em regiões distintas. Ao final deste processo, a imagem é particionada em um conjunto de regiões disjuntas, cujos pixels sejam conectados e possuam a mesma cor. Tipicamente, esta é uma operação custosa e que impacta negativamente no desempenho em tempo real dos sistemas de visão computacional.

Um algoritmo bastante eficiente foi desenvolvido, com a participação da autora: CMVision (BRUCE; BALCH; VELOSO, 2000). O algoritmo possui duas fases. Na primeira fase, computa-se o código *run-length* (RLE – *Run Length Encoded*) da imagem classificada pelas cores. Normalmente, no mundo real, não há mudanças bruscas nos pixels adjacentes da imagem; assim, o RLE permite que se manipule grandes porções de linhas da imagem, ao invés de pixels individuais. Além disso, no processo de agrupamento em regiões, basta buscar por conectividade na vertical da imagem, uma vez que componentes horizontais já foram agrupados pelo RLE.

A segunda fase consiste em efetuar uma busca em árvore para o agrupamento de regiões. Inicialmente, cada elemento do RLE é nomeado como seu próprio antecessor, resultando numa floresta (completamente disjunta). O processo de agrupamento então rastreia elementos em linhas adjacentes e agrupa aqueles que possuírem a mesma cor, desde que algum par de pixels (um de cada elemento distinto do RLE) sejam adjacentes. O primeiro elemento encontrado na imagem é o antecessor e os outros, sucessores. Finalmente, um novo rastreamento é realizado, para que todos elementos que pertençam a uma única região apontem para uma única raiz (primeiro elemento encontrado na cor da região em questão).

Os passos do algoritmo estão ilustrados na Figura 10.4, onde é mostrado cada elemento do RLE, inicialmente desconectado. Durante o rastreamento da imagem, buscando elementos conectados, cada elemento encontrado aponta para o antecessor de mesma cor. Finalmente, a imagem é novamente rastreada, buscando agrupar todos elementos conectados de mesma cor, todos possuindo uma mesma raiz.



(a) Inicialmente, elementos disjuntos na imagem; (b) e (c) novos elementos de mesma cor são agrupados, numa busca por linhas adjacentes na imagem, e (d) num último passo, todos os elementos conectados que possuam a mesma cor são agrupados numa única região, com uma única raiz.

Figura 10.4: Agrupamento em regiões do CMVision.

10.2 Extração de Cena de Fundo

Em diversas aplicações baseadas em técnicas de visão computacional, como sistemas de inspeção, monitoramento, navegação e futebol de robôs, a segmentação dos objetos de interesse numa cena dinâmica é uma etapa crítica e primordial para todo o processamento subsequente. Um procedimento básico incluído neste processo de segmentação consiste na comparação de uma imagem observada com uma estimativa da imagem se a mesma não contivesse nenhum dos objetos de interesse. Tradicionalmente, para a realização desta idéia, utiliza-se a subtração da cena corrente por um modelo de cena de fundo.

A grande dificuldade desta abordagem, entretanto, está na construção de um modelo robusto de cena de fundo, adaptável a variações de iluminação, a sombras e reflexões criadas pelos objetos de interesse e às diferentes texturas das regiões da própria cena. Este modelo deve, eventualmente, ainda poder lidar com situações como cenas de fundo dinâmicas e alto tráfego de objetos de interesse na cena.

Explorando este tema, a autora e seus orientados utilizaram uma técnica de segmentação de objetos de interesse por extração de cena de fundo, usando um modelo fixo de cena de fundo no time GUARANÁ de futebol de robôs e, numa derivação de aplicação deste algoritmo, na primeira versão de um sistema de

monitoramento de terminais rodoviários baseado em visão computacional (veja Seções 10.5.2 e 10.5.6. No entanto, nesta segunda aplicação, o uso de um modelo fixo de cena de fundo não se mostrou apropriado e uma nova proposta foi feita, com o uso de um modelo adaptativo.

10.2.1 Modelo Fixo de Cena de Fundo

Uma idéia básica e intuitiva do processo de subtração de cena de fundo consite em realizar a subtração dos valores de cada pixel da imagem corrente por uma imagem de referência (chamado modelo de cena de fundo), operação esta seguida por uma limiarização. Esta subtração deve ser realizada em todas as bandas, caso a imagem seja colorida. Desta forma, torna-se necessário a estimação da imagem de referência, que é usada como modelo de cena de fundo.

A estimativa do modelo da cena de fundo pode ser feita pela média de N quadros:

$$\overline{B}(C) = \frac{\sum_{i=1}^{N} B_i(C)}{N} \tag{10.7}$$

com $\overline{B}(C)$ é a estimativa da cena de fundo, para cada banda de cor da imagem: para imagens RGB, $C \in \{R, G, B\}$, N é o número de quadros considerados, $B_i(C)$ é cada quadro de cena de fundo, em cada banda.

Um pixel da imagem corrente é classificado como pertencente a um objeto de interesse (ou seja, como não pertencente ao modelo de cena de fundo) se:

$$\sum_{C \in \{R,G,B\}} \mid I_t(C) - \overline{B}(C) \mid > kT \tag{10.8}$$

onde $I_t(C)$ é a imagem corrente na cor C, $\overline{B}(C)$ é o modelo de cena de fundo na cor C, T é um limiar que estima o desvio padrão do ruído, k é uma constante pré-estabelecida.

10.2.2 Modelo Adaptativo de Cena de Fundo

O uso de um modelo fixo de cena de fundo dificulta a adaptação a variações de iluminação, sombras, texturas, etc, que podem ocorrer na cena de fundo e impossibilitam sua aplicação em situações de cena de fundo dinâmica. Assim, investigou-se um algoritmo para a estimativa de um modelo robusto de cena de

fundo a fim de realizar, em tempo real, a segmentação de objetos de interesse em ambientes externos dinâmicos, sujeitos a diferentes condições de iluminação.

O atributo cor é um atributo poderoso na segmentação de imagens. Para sistemas onde a estimativa ou a remoção das cenas de fundo seja fundamental, deseja-se uma forma de representação de cores onde a luminância e as componentes cromáticas estejam claramente separadas, com o objetivo de obter informações mais apuradas no que se refere à iluminação e a refletância da cena. Desta forma, utiliza-se, neste estudo, a representação YUV para o espaço de cores da imagem da cena.

O algoritmo aqui proposto baseia-se na utilização de uma mistura de distribuições gaussianas adaptativas para modelar cada pixel da cena de fundo. Este modelo, contudo, não é totalmente adaptável a todas diferentes condições de uma cena de ambiente externo, apresentando falhas na ocorrência de sombras e reflexões, além de erros quando implementado em cenas com objetos de interesse quase estáticos ou em alto tráfego. Assim, para minimizar estes erros, propõe-se a adoção de três níveis de processamento: *(i)* nível pontual, onde são realizados processamentos envolvendo somente o pixel em questão; *(ii)* nível local, onde são realizados processamentos numa vizinhança do pixel de interesse, para refinar a classificação já realizada; *(iii)* nível global, envolvendo toda a imagem (quadro), com o objetivo de verificar variações súbitas de iluminação da cena.

Processamento Pontual

O valor de um pixel resulta da aquisição da energia refletida por uma certa superfície sob uma condição de iluminação particular, podendo, assim, ser modelado por uma distribuição gaussiana simples, considerando algum ruído que por ventura ocorra na aquisição da imagem.

Porém, em aplicações onde há variações das superfícies que compõem a cena (objetos em movimento), além da variação de iluminação, faz-se necessário a utilização de múltiplas distribuições gaussianas adaptativas (mistura de K gaussianas), para representar as diferentes superfícies.

Define-se o histórico de um pixel como sendo uma série temporal dos seus valores, que são vetores $\vec{X}_{i,t} = \{Y_{i,t}, U_{i,t}, V_{i,t}\}$. A cada instante t, para cada pixel

$i = \{x_0, y_0\}$, pode-se representar seu histórico como:

$$\{\vec{X}_{i,1}, \ldots, \vec{X}_{i,t-1}\} = \{\vec{I}(x_0, y_0, j) : 1 \leq j \leq t-1\} \tag{10.9}$$

onde I é a seqüência de imagens (quadros).

Com isso a probabilidade de observação do pixel atual pode ser estimada como:

$$P(\vec{X}_{i,t}|\vec{X}_{i,1}, \ldots, \vec{X}_{i,t-1}) = \sum_{k=1}^{K} w_{i,t-1,k}\, \eta(\vec{X}_{i,t}, \vec{\mu}_{i,t-1,k}, \Sigma_{i,t-1,k}) \tag{10.10}$$

onde $w_{i,t-1,k}$ é o peso associado à k-ésima distribuição gaussiana na mistura no tempo $t-1$, o qual indica a proporção relativa às observações passadas modeladas por cada distribuição gaussiana, $\vec{\mu}_{i,t-1,k}$ e $\Sigma_{i,t-1,k}$ são os valores do vetor-média e matriz de covariância dessa k-ésima distribuição gaussiana e η é a função densidade de probabilidade normal dada por:

$$\eta(\vec{X}_{i,t}, \vec{\mu}_{i,t-1,k}, \Sigma_{i,t-1,k}) = \frac{1}{2\pi^{\frac{n}{2}} |\Sigma_{i,t-1,k}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}_{i,t}-\vec{\mu}_{i,t-1,k})^T \Sigma_{i,t-1,k} (\vec{X}_{i,t}-\vec{\mu}_{i,t-1,k})} \tag{10.11}$$

Por razões de custo computacional, convencionou-se utilizar a seguinte matriz de covariância:

$$\Sigma = diag \begin{bmatrix} \sigma_Y^2 & \sigma_U^2 & \sigma_V^2 \end{bmatrix} \tag{10.12}$$

onde σ_Y^2, σ_U^2 e σ_V^2 são, respectivamente, as variâncias da luminância Y e das componentes cromáticas U, V, assumindo que estas componentes são independentes.

Tradicionalmente a escolha do valor K é igual para todos os pixels, sendo tipicamente na faixa de 3 a 5. Quanto mais distribuições forem usadas, melhor será a representação das cenas pelo modelo, acarretando, entretanto, maior custo computacional.

A cada nova observação, a mistura de distribuições gaussianas utilizada para modelar a história de observações de cada pixel deve ser atualizada. Idealmente, a cada instante t, dever-se-ia reestimar todos parâmetros da mistura de distribuições gaussianas de cada pixel aplicando um algoritmo de Maximização da Esperança (*Expectation Maximization*) a alguma janela de recentes observações, já incluindo a última. Este procedimento, porém, é demasiadamente custoso, podendo-se empregar sem maiores prejuízos uma aproximação do algoritmo k-

means (veja Seção 8.2). Essa aproximação pode ser vista como a busca pela distribuição que melhor representa o pixel atual a fim de atualizar os parâmetros de η_k utilizando a observação corrente.

A busca é realizada através do cálculo da distância de Mahalanobis entre a observação corrente e as k distribuições do modelo. Se o valor da luminância da observação corrente for menor que um valor mínimo de luminância pré-fixado, as componentes cromáticas não são utilizadas no cálculo da distância de Mahalanobis, o mesmo ocorrendo para casos em que a distribuição gaussiana tenha valor médio da luminância menor que o valor pré-fixado. Isso se deve ao fato de que as componentes cromáticas tornam-se instáveis quando os níveis de luminância são baixos.

No caso em que duas ou mais distribuições tenham a mesma distância em relação à observação atual, escolhe-se a distribuição gaussiana com a maior relação peso/variância da luminância (já que esta é válida em todos os casos). Este critério foi adotado porque se espera que uma distribuição gaussiana que represente a cena de fundo tenha grande peso (já que ocorre freqüentemente) e baixa variância (já que varia pouco no tempo).

A busca falha quando as distâncias calculadas entre as k distribuições e a observação corrente forem maior que um valor β, geralmente da ordem de 2.5, a fim de englobar 95% da função densidade de probabilidade normal. Se a busca falhar, a distribuição com a menor relação peso/variância é substituída por uma nova que representa a observação atual, sendo os seus parâmetros inicialmente ajustados para uma alta variância e um baixo peso.

No caso de sucesso da busca, a atualização dos parâmetros da distribuição gaussiana que melhor modela a observação corrente é dada pelas equações abaixo:

$$\vec{\mu}_{i,t,k} = (1-\alpha)\vec{\mu}_{i,t-1,k} + \alpha\vec{X}_{i,t} \tag{10.13}$$

$$\sigma^2_{Y,i,t,k} = (1-\alpha)\sigma^2_{Y,i,t-1,k} + \alpha(Y_{i,t} - \mu_{Y,i,t-1,k})^2 \tag{10.14}$$

$$\sigma^2_{U,i,t,k} = (1-\alpha)\sigma^2_{U,i,t-1,k} + \alpha(U_{i,t} - \mu_{U,i,t-1,k})^2 \tag{10.15}$$

$$\sigma^2_{V,i,t,k} = (1-\alpha)\sigma^2_{V,i,t-1,k} + \alpha(V_{i,t} - \mu_{V,i,t-1,k})^2 \tag{10.16}$$

onde α é chamada de taxa de aprendizagem.

Efetivamente, a constante de tempo $1/\alpha$ determina a velocidade com a qual os parâmetros da distribuição mudam, podendo ser ajustada de acordo com o objetivo da implementação.

Uma observação importante é que todas as variâncias não podem ser decrescidas abaixo de um valor mínimo, como forma de evitar instabilidade na busca em regiões da cena que permaneçam estáticas por um longo período de tempo. Além disso, o valor mínimo da variância da luminância deve ser mantido num nível substancial, com o objetivo de que sombras e reflexões não interfiram no processo de busca. Os parâmetros das outras distribuições gaussianas que não modelam a observação corrente permanecem inalterados.

Outro importante parâmetro é os pesos de cada distribuição na mistura, que devem ser ajustados como segue:

$$w_{i,t,k} = (1 - \alpha)w_{i,t-1,k} + \alpha M_{i,t,k} \tag{10.17}$$

onde $M_{i,t,k}$ é 1 (verdadeiro) para a distribuição que melhor modela a observação corrente e 0 (falso) para as demais. Tanto para o caso de sucesso na busca, quanto para o caso de falha, os pesos devem ser renormalizados.

A cada instante, uma ou mais distribuições gaussianas de cada mistura são selecionadas como modelos de cena de fundo para um determinado pixel, enquanto as outras são classificadas como modelos de objetos de interesse. A escolha das distribuições que representam o modelo de cena de fundo é realizada primeiramente ordenando-se decrescentemente as k distribuições gaussianas pela relação peso/variância. Em seguida, escolhe-se as B primeiras distribuições como modelo de cena de fundo de acordo com o critério abaixo:

$$B = argmin_b \left(\sum_{k=1}^{b} w_{i,k} > T \right) \tag{10.18}$$

onde T é um limiar previamente fixado. Se a distribuição que melhor modela a observação corrente, no caso de sucesso da busca, for uma dessas B primeiras distribuições, o pixel atual é classificado como pertencente ao modelo de fundo. Se a busca falhar ou a distribuição não pertencer a estas B primeiras, o pixel é classificado como objeto de interesse.

O procedimento anterior, entretanto, é extremamente dependente da taxa de

aprendizagem α. Para valores elevados desta taxa, os objetos de interesse que por ventura permaneçam praticamente estáticos na cena, por exemplo, duas pessoas paradas conversando, serão rapidamente incorporados ao modelo de fundo. Já no caso de valores baixos desta taxa, variações bruscas de iluminação em algumas regiões da cena, poderiam resultar na classificação errônea de determinadas partes da cena de fundo como objetos de interesse por um longo período de tempo.

Com o objetivo de contornar estas dificuldades, tornou-se a taxa de aprendizagem adaptativa baseada no nível de atividade da cena. Especificamente, computa-se uma medida de atividade da cena A para cada pixel e reduz-se a taxa de aprendizagem α por um fator ε nos pixels onde A exceda um limiar H. Com isso, variações bruscas na luminosidade de uma determinada região seriam mais rapidamente incorporadas ao modelo de cena de fundo e objetos de interesse que não estejam completamente estáticos seriam incorporados mais lentamente ao modelo de cena de fundo. Inicialmente, para cada pixel, o nível de atividade da cena é ajustado em zero e gradativamente é computado recursivamente do nível anterior de atividade e da diferença de luminância Y entre o frame atual e o passado:

$$A_{i,t,k} = (1 - \lambda)A_{i,t-1,k} + \lambda\,|Y_{i,t} - Y_{i,t-1}| \tag{10.19}$$

onde λ é a taxa de aprendizagem da medida de atividade, cuja função é suavizar ruídos decorrentes da luminância. É importante destacar que o valor de α não é reduzido a zero, permitindo que o modelo de cena de fundo seja sempre ajustado corretamente.

Processamento local e global

A abordagem com misturas de distribuições gaussianas modelando os pixels pode ser estendida de forma a incorporar camadas supervisoras que considerem as regiões da imagem ou quadros inteiros, a classificação de objetos ou outras propriedades da cena que sinalizem a qualidade da classificação dos pixels. Estas camadas devem ter a habilidade de detectar a correta (ou incorreta) classificação dos objetos de interesse e também ter a capacidade de gerar mapas que apontem quais pixels geraram esta classificação.

Harville (2002) sugere dois tipos de realimentação para o processamento pontual:

1. Realimentação positiva, com a função de reforçar os acertos na detecção dos objetos de interesse;

2. Realimentação negativa, com o objetivo de ajustar o modelo de cena de fundo gerado no processamento pontual a fim de prevenir a recorrência de erros de classificação.

A interface destas camadas com o processamento pontual são duas imagens com os resultados da realimentação negativa e da positiva nas respectivas posições dos pixels. A criação destas duas imagens tem como objetivo permitir a correlação direta entre os pixels das imagens de realimentação e os do modelo obtido no processamento pontual. É importante ressaltar também que deve haver uma política de decisão dependente da aplicação, para os casos de conflito entre as duas realimentações. Deve-se, no entanto, preferencialmente privilegiar a realimentação positiva, já que erros neste caso não prejudicam efetivamente o modelo de cena de fundo.

A realimentação positiva proposta por Harville consiste na detecção dos acertos da segmentação dos objetos de interesse realizada no processamento pontual. Esta detecção é realizada por um ou mais módulos das camadas de alto nível, que podem ser, por exemplo, qualquer implementação de um detector de objetos de interesse.

Nos desenvolvimentos feitos pela autora e seu grupo foi utilizada uma implementação simples de um detector de objetos, baseada no agrupamento em regiões (veja a Seção 10.1.2), que verifica a consistência do objeto segmentado com um padrão estipulado, como áreas (em pixels) mínima e máxima.

Os pixels corretamente classificados são marcados na imagem com valor igual a um e a observação corrente não entra na atualização do modelo de mistura de distribuições gaussianas. O intuito desta realimentação é impedir que objetos de interesse sejam incorporados ao modelo de fundo de cena.

A realimentação negativa funciona nos casos onde os pixels classificados como partes dos objetos de interesse seriam melhores descritos como parte do modelo de cena de fundo. Basicamente os módulos responsáveis pela detecção dos erros de classificação podem ser os mesmos da realimentação positiva, além de um módulo de detecção de variação súbita de iluminação. A realimentação negativa

proposta por Harville consiste na criação de uma mistura de distribuições gaussianas (igual a do processamento pontual) para representar os pixels onde houve erro de segmentação, eventualmente substituindo (na verdade, é realizada uma fusão) o modelo de mistura de distribuições da cena de fundo. Esta abordagem, embora tendo sido proposto um algoritmo otimizado, é muito custosa para aplicações comerciais com requisitos de funcionamento em tempo real. No entanto, a realimentação negativa pode ser realizada modificando-se apenas os pesos das distribuições gaussianas nos pixels onde houve seguidos erros de segmentação. A Seção 10.5.6 mostra aplicações das técnicas aqui propostas, desenvolvidas pela autora e seu grupo.

10.3 Detecção de Movimento no Ambiente

O estudo da movimentação em seqüência de imagens permite responder perguntas como: quantos objetos em movimento existem, em que direções eles estão se movendo, com que velocidade estão se movendo, sua movimentação é linear ou rotacional.

De uma seqüência de imagens (mínimo duas), calcula-se a função de fluxo ótico (ou campo de movimento, dependendo do tipo de algoritmo), para cada pixel. Um vetor de velocidade $\vec{v} = (u, v)$ é encontrado, o qual define:

Módulo do vetor Quão rápido o pixel monitorado, que corresponde a um objeto de interesse na cena, está se movendo através das imagens.

Direção do vetor Direção do movimento do pixel na imagem.

Com estes dados de módulo e direção do movimento de cada pixel da imagem, pode-se então segmentar a imagem, utilizar subtração de cena de fundo, agrupamento de regiões para isolar certas partes e determinar a velocidade de diferentes objetos.

A forma mais comum de representar um campo de movimento ou um campo de fluxo ótico é o *needle map*. O *needle map* é uma figura com a representação dos vetores $\vec{v}$ de todos ou apenas alguns pontos da imagem. A Figura 10.5 ilustra um exemplo de *needle map*, onde são desenhados o módulo (tamanho do segmento) e a direção (direção do segmento) de cada vetor de velocidade encontrado.

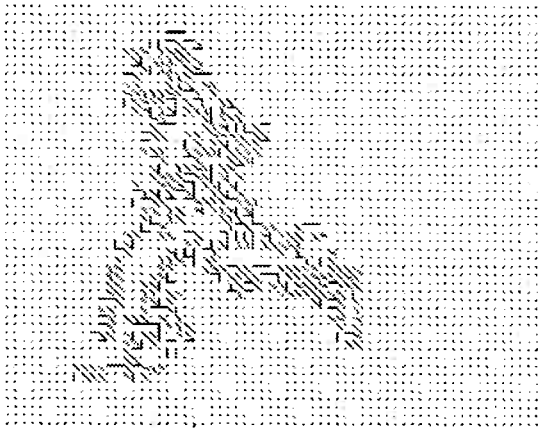


Figura 10.5: Um exemplo de *needle map*.

O problema principal na análise do movimento é a estimação das componentes tridimensionais do movimento dos objetos sendo observados. Isto é de grande relevância em muitos problemas como reconstrução 3D, rastreamento de objetos e navegação de robôs. A informação disponível em um sistema de visão baseado em análise de movimento é relacionada com a projeção da velocidade 3D real no plano da imagem.

Três principais dificuldades afetam o problema de estimação do campo de movimento:

1. Descontinuidades no campo de movimento que são originados pela presença de ruído no brilho da imagem.

2. A presença de oclusões entre diferentes objetos em movimento (que usualmente têm diferentes velocidades) e entre os objetos em movimento e o fundo estático.

3. O chamado problema de abertura (*aperture*) que é relacionado com a impossibilidade de recuperar a direção do movimento se o objeto é observado através de uma abertura que é menor que o tamanho do objeto, e as referências ao objeto sob observação (como textura) não são suficientes para perceber a componente transversal do movimento do objeto.

10.3.1 Tipos de Algoritmos de Análise do Movimento

Na literatura, duas abordagens principais para estimação de campo de movimento em tempo real podem ser identificadas: baseado em correspondência (*matching-based*), e baseado em gradiente (*gradient-based*) (LAPLANTE; STOYENKO, 1997). Por sua vez, as abordagens baseadas em gradiente podem ser classificadas em regularização (*regularization-based*) e multi-restrição (*multiconstraint-based*).

Abordagens baseadas em correspondência utilizam-se da identificação de um conjunto de características esparsas e facilmente identificáveis dos objetos em movimento. Pelo rastreamento destas características, uma correspondência entre os quadros é buscada para estimar o movimento das características no plano da imagem. Essas características facilmente identificáveis podem ser classificadas em alto nível (ex.: linhas e formas) e baixo nível (ex.: padrões, curvaturas). Estas técnicas são muito úteis para rastrear objetos em movimento no espaço 3D utilizando-se características de alto nível, porque as características rastreadas podem ser partes relevantes do objeto. Mas o rastreamento destas características requer uma fase de pré-processamento (ex.: filtragem, extração de bordas) nas imagens, com os correspondentes custos computacionais envolvidos. Além disso, as técnicas de correspondência devem ser independentes de rotação e escala e, deste ponto de vista, melhores soluções são obtidas utilizando-se características de baixo nível.

Abordagens baseadas em gradiente fornecem uma solução para estimação de movimento da observação no tempo das mudanças no brilho da imagem. Essas mudanças são modeladas por equações diferenciais parciais, que são normalmente chamadas de equações de restrição (*constraint equations*). O campo de vetores de velocidade obtido pela solução dessas equações é normalmente chamado de fluxo ótico (*optical flow*) ou fluxo da imagem (*image flow*). Genericamente, o conceito de campo de fluxo ótico difere do conceito de campo de movimento porque o campo de movimento é um conceito puramente geométrico, enquanto que o conceito de fluxo ótico é baseado na observação de mudanças no brilho da imagem.

A equação diferencial parcial mais importante para a modelagem do campo de fluxo ótico é obtida pela consideração das mudanças no brilho da imagem

$E(x_t, y_t, t)$ como estacionário em respeito a t ($dE/dt = 0$):

$$Exu + Eyv + Et = 0 \tag{10.20}$$

na qual u e v correspondem à dx/dt e dy/dt, respectivamente, representando os componentes do vetor de velocidade local $\vec{v}$ em toda extensão das direções x e y da imagem. Esta equação é normalmente chamada de *Optical Flow Constraint* (OFC) ou Restrição do Fluxo Ótico. Além disso, a OFC pode também ser aproximada à equação de uma reta no plano (u, v):

$$\vec{v} = mu + c \tag{10.21}$$

onde $m = -(Ex/Ey)$ é a inclinação e $c = -(Et/Ey)$ é a raiz. Qualquer ponto ao longo dessa reta é uma possível solução para o problema de estimação do fluxo ótico. No estudo realizado foram utilizados dois algoritmos de cálculo de fluxo ótico (CAMUS, 1997; PROESMAN, 1994), assim como outros mais simples para tratamento de imagem relativos à extração de bordas, convolução e operadores morfológicos *open* e *close* (GONZALEZ; WOODS, 1992). O principal objetivo é o estudo do uso dos dois algoritmos de fluxo ótico para detecção de movimento, analisando o desempenho, a calibragem e a aplicabilidade dos mesmos.

10.3.2 Fluxo Ótico por Difusão Não Linear

O algoritmo de Fluxo Ótico por Difusão Não Linear (PROESMAN, 1994) consiste em extrair um campo denso de velocidades de uma seqüência de imagens. Inicialmente, com o uso de um pré-processamento de convolução com máscaras de Sobel (GONZALEZ; WOODS, 1992), são extraídas as bordas verticais e horizontais da imagem, o que gera respectivamente os gradientes Ex e Ey.

O gradiente no tempo Et é obtido por uma simplificação da diferenciação, que consiste na subtração dos valores (intensidades) dos pixels de duas imagens consecutivas na seqüência de imagens.

Após a obtenção dos gradientes, alguns parâmetros que definem a calibragem do algoritmo são usados durante o cálculo. São eles:

Níveis Define a escala de redução da imagem para resoluções menores. A menor figura obtida utilizando-se essa escala será utilizada para fazer a primeira es-

timação do fluxo ótico. A escala de redução é dada por: $Escala = 1/4^{níveis}$.

Iterações O número de iterações de refinamento do fluxo ótico em cada nível de resolução.

λ Define a relação entre gradientes do espaço e o gradiente do tempo. Para imagens com ruído, o λ deve ser próximo de zero e para imagens sintéticas ou sem ruído o λ deve ter um valor alto (acima de 1000).

Estimação Variável booleana que define se resultados dos quadros anteriores devem ser usados nos cálculos atuais.

A análise da imagem para extração do fluxo ótico por difusão não linear é feita em diferentes níveis de resolução, do nível mais baixo de resolução para o mais alto, utilizando os dados extraídos em resoluções inferiores como base para extrair o fluxo ótico em resoluções superiores, com mais detalhes.

No nível de resolução mais baixa o algoritmo faz a primeira tentativa em obter o fluxo ótico. O resultado desta primeira tentativa é usado nas outras resoluções, sendo o mesmo sempre refinado de acordo com o número de iterações definido.

10.3.3 Campo de Movimento de Camus

O algoritmo de Camus (CAMUS, 1997) extrai o campo de movimento e não o campo do fluxo ótico. Para isso o algoritmo de Camus utiliza um entre seis métodos diferentes para determinar se ocorreu modificação nos valores de pixels, entre duas imagens consecutivas na seqüência de imagens. São eles:

1. SAD: Sum of Absolute Differences;
2. ZSAD: Zero-mean Sum of Absolute Differences;
3. SSD: Sum of Squared Differences;
4. ZSSD: Zero-mean Sum of Squared Differences;
5. NCC: Normalized Cross Correlation;
6. ZNCC: Zero-mean Normalized Cross Correlation;

Uma máscara percorre as imagens, efetuando um dos cálculos selecionados entre elas. É um processamento bem mais simples que o fluxo ótico por difusão não linear, por isso é muito mais rápido. Entretanto, os resultados alcançados são bem mais imprecisos.

A Seção 10.5.7 mostra resultados comparativos da aplicação de ambos os métodos aqui descritos. Esforços atuais da equipe da autora consistem em propor uma nova abordagem, com procedimento integrado de ambas as propostas. Assim, o algoritmo de Camus seria utilizado inicialmente, gerando uma primeira aproximação do cálculo do fluxo ótico; após, esta primeira estimação de movimento seria fornecida ao algoritmo de cálculo de fluxo ótico por difusão não linear, buscando uma solução mais precisa.

10.4 Estereoscopia Binocular

Inúmeras aplicações de visão computacional requerem algum tipo de inferência sobre a informação tridimensional de uma cena. Aplicações de medição, assim como as de navegação robótica, são exemplos de casos em que é necessário recuperar informações métricas do objeto de interesse – ou da sua localização no ambiente – a partir de imagens capturadas por câmeras de vídeo.

Uma técnica freqüentemente utilizada para recuperar, a partir de imagens, a informação tridimensional de uma cena – especialmente quando há a necessidade de alta precisão – é a estereoscopia. Esta técnica permite inferir a profundidade de um ponto de interesse no ambiente a partir de duas ou mais imagens deste ponto, capturadas a partir de diferentes pontos de observação. A captura é geralmente feita por duas ou mais câmeras simultaneamente ou por uma mesma câmera em diferentes posições (caso a cena seja absolutamente estática).

A inferência sobre a profundida é possível devido ao deslocamento entre a posição do pixel em cada uma das duas imagens, correspondente a um único ponto na cena. Este deslocamento (conhecido como *disparidade*) é proporcional à distância entre o ponto do ambiente e a câmera, possibilitando a recuperação da profundidade do ponto por triangulação entre as duas câmeras e o ponto. Quanto maior a distância entre as câmeras, maior será a precisão da triangulação; no entanto, maior será a dificuldade de incluir no campo visual das câmeras a

mesma região da cena, dificultando o processo de correpondência entre elas.

Para realizar a triangulação, é preciso conhecer as características geométricas internas das câmeras utilizadas, bem como a relação entre a localização e orientação das câmeras que capturaram as imagens, ou entre as posições e orientações na cena a partir das quais as imagens foram capturadas (no caso de apenas uma câmera estar sendo utilizada). A este conjunto de parâmetros é dado o nome de *modelo geométrico da câmera.* As características geométricas (físicas) internas da câmera são chamadas de *parâmetros intrínsecos*, e a relação geométrica entre as posições e orientações de captura são chamados de *parâmetros extrínsecos.* O processo de obtenção destes parâmetros é conhecido como *calibração da câmera.*

10.4.1 Calibração de Câmeras: modelos e métodos

O modelo geométrico que se deseja obter com a calibração é derivado do modelo *pinhole* geralmente adotado para as câmeras. Neste modelo, o processo de formação da imagem na câmera é aproximado por uma projeção em perspectiva, onde as coordenadas de um ponto do ambiente e a sua projeção no plano de imagem são relacionados pela seguinte equação:

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{10.22}$$

onde x, y são as coordenadas (em pixels) do ponto no plano de imagem, X, Y, Z são as coordenadas do ponto no ambiente, R e T são a matriz de rotação e o vetor de translação, respectivamente, entre o sistema de coordenadas do ambiente e o sistema de coordenadas da câmera, cuja origem coincide com o centro de projeção da mesma, e s é um fator de escala. A matriz A, conhecida como matriz intrínseca da câmera, é dada por:

$$A = \begin{bmatrix} \alpha_u & \theta_c.\alpha_u & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

onde α_u e α_v são os fatores de escala horizontal e vertical, respectivamente, u_0 e v_0 são as coordenadas do ponto principal da imagem e o θ_c codifica o ângulo

formado pelos eixos da matriz CCD da câmera.

No entanto, o modelo *pinhole* é apenas uma aproximação da formação da imagem na câmera. Quando alta precisão é um requisito, um modelo mais fiel deve ser utilizado. Normalmente, o modelo pinhole é estendido para considerar a distorção causada pelas lentes. As principais correções utilizadas são para as componentes radial e a tangencial da distorção, através da adição de alguns coeficientes ao equacionamento exposto acima. O número de coeficientes varia de acordo com a complexidade do modelo matemático utilizado para a câmera. Os modelos mais complexos prevêem até quatro coeficientes para a distorção radial e dois para a tangencial. Com a introdução da distorção, o mapeamento entre as coordenadas do ponto no ambiente e no plano da imagem deixa de ser linear e pode ser descrito como segue.

Seja $[X\ Y\ Z]^T$ um ponto do ambiente no sistema de coordenadas da câmera (obtido aplicando-se a rotação R e a translação T sobre o ponto no sistema de coordenadas do ambiente). $[X/Z\ Y/Z]^T = [x\ y]^T$ são as coordenadas normalizadas (projeção *pinhole*) no plano da imagem. Ao se considerar a distorção, as novas coordenadas normalizadas serão:

$$x' = (1+kc(1).r^2+kc(2).r^4+kc(5).r^6).x+2.kc(3).x.y+kc(4).(r^2+2.x^2) \quad (10.23)$$

$$y' = (1+kc(1).r^2+kc(2).r^4+kc(5).r^6).y+kc(3).(r^2+2.y^2)+2.kc(4).x.y \quad (10.24)$$

onde $r^2 = x^2 + y^2$ e kc é o vetor contendo os coeficientes de distorção radial e tangencial. As coordenadas em pixels seriam então obtidas por:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (10.25)$$

Há controvérsia sobre o impacto da complexidade do modelo de distorção sobre a precisão da calibração. Alguns autores (ZHANG, 1999) argumentam que a distorção é dominada pela componente radial, mais especialmente pelo seu primeiro termo, e que o uso de modelos mais complexos não aumentaria a precisão, além de causar instabilidade numérica.

Diversos métodos para obter este conjunto de parâmetros podem ser encontrados na literatura (ZHANG, 1999; HEIKKILÃ; SILVÉN, 1997; TSAI, 1987; FAU-

GERAS, 1987), e um ótimo histórico sobre o desenvolvimento destes métodos pode ser encontrado em (CLARKE; FRYER, 1998). No entanto, com exceção dos métodos de calibração automática (*self-calibration*), os quais ainda não estão suficiente maduros e nem sempre fornecem resultados confiáveis (ZHANG, 1999; ZHANG; FAUGERAS; DERICHE, 1997), invariavelmente há a necessidade de arranjos ou procedimentos experimentais relativamente complexos e acurados para a realização da calibração. Recentemente, foi proposto um método de calibração (ZHANG, 1999) que dispensa a complexidade do arranjo e dos procedimentos experimentais antes necessários. Através do uso deste método, é possível realizar a calibração a partir da observação, pela câmera, de um gabarito de calibração plano a partir de diversos (pelo menos dois) pontos de observação diferentes (um gabarito de calibração é geralmente um padrão de quadrados, como um tabuleiro de xadrez, impresso e colado sobre alguma superfície plana). A única informação que deve ser conhecida com precisão é o tamanho das arestas dos quadrados que compõem o gabarito e o número de quadrados nas direções horizontal e vertical.

Neste método, o conjunto de parâmetros é obtido através da minimização iterativa da distância entre as coordenadas dos pontos do gabarito nas imagens utilizadas na calibração e as coordenadas dos pontos reprojetados nestas mesmas imagens, segundo um conjunto de valores iniciais para os parâmetros, geralmente obtido pela solução analítica de um sistema linear derivado da equação 10.22, solução que somente é possível se os coeficientes de distorção não forem considerados. Este método vem sendo utilizada com sucesso pela autora e seus orientados.

Os métodos acima referenciados dizem respeito apenas à calibração de uma câmera. No caso da aplicação destes métodos para a calibração de um arranjo estéreo binocular, cada câmera deve ser calibrada separadamente. Este procedimento tem uma deficiência grave (ZHANG; FAUGERAS; DERICHE, 1997), uma vez que não faz uso do fato de que os pontos presentes nas imagens capturadas pelas duas câmeras correspondem aos mesmos pontos do ambiente. Isto impõe fortes restrições sobre a geometria epipolar do arranjo estéreo, e a sua utilização resultaria na obtenção muito mais precisa desta geometria, o que não ocorre nos métodos citados anteriormente, uma vez que estes métodos estimam a geometria epipolar a partir dos parâmetros intrínsecos e extrínsecos obtidos na calibração. Isto implica na obtenção de um modelo geométrico para a câmera que apenas é válido na região do ambiente onde foi colocado o gabarito de calibração.

Foi proposto um método (ZHANG; FAUGERAS; DERICHE, 1997) que tem por objetivo a calibração do arranjo estéreo como um conjunto, e não apenas das câmeras separadamente. Neste método, o primeiro passo é estimar a geometria epipolar do arranjo com base nos pontos correspondentes presentes nas imagens capturadas pelas duas câmeras. Nos próximos passos obtêm-se um conjunto de parâmetros do modelo geométrico que seja compatível não apenas com os pontos do gabarito, mas também com a geometria epipolar estimada anteriormente. Aparentemente o modelo geométrico obtido através deste método é razoavelmente válido para regiões fora do espaço do gabarito de calibração.

O algoritmo usado pela autora e seu grupo, apesar de não estimar a geometria epipolar antes da calibração propriamente dita, realiza a calibração do conjunto estéreo através da obtenção dos dois modelos geométricos como se fossem um só, simultaneamente. Este procedimento tem impacto sobre a precisão do modelo obtido para as câmeras – especialmente sobre os parâmetros extrínsecos, mas não aumenta a validade dos modelos fora da região do gabarito de calibração.

10.4.2 Reconstrução Estéreo 3D

Os modelos geométricos obtidos para as câmeras de um arranjo estéreo binocular fazem o mapeamento entre as coordenadas de um ponto no ambiente e as suas coordenadas (em pixels) na imagem. Devido à projeção perspectiva, diversos pontos no ambiente podem ser mapeados em um mesmo ponto no plano de imagem. Portanto, o mapeamento inverso – do plano de imagem para o ambiente – funciona da mesma maneira, sendo um ponto do plano de imagem inversamente mapeado para o ambiente como uma reta (a reta formada pelo ponto no plano de imagem e o centro de projeção da câmera, ou o *pinhole*) (FAUGERAS, 1987). A reconstrução tridimensional através da estereoscopia é baseada neste príncipio: o mapeamento inverso das projeções de um mesmo ponto do ambiente nos planos de imagens de duas câmeras resulta em duas retas que se cruzam no ponto do ambiente que gerou aquelas projeções. É fácil perceber que quanto mais câmeras forem utilizadas, mais retas serão obtidas pelo mapeamento inverso, e mais precisamente o ponto de cruzamento será determinado no ambiente.

Equacionamento do mapeamento inverso

Se o modelo linear derivado da equação 10.22 estiver sendo utilizado, isto é, se a compensação da distorção dada pelas equações 10.23 e 10.24 não estiver sendo considerada, existe uma solução analítica para o problema do mapeamento inverso. Caso contrário, se o modelo utilizado for o derivado da equação 10.25, o modelo deixa de ser linear e não existe solução analítica para o problema. Neste caso, soluções iterativas foram propostas (HEIKKILÃ; SILVÉN, 1997). Uma destas soluções encontra-se implementada no ToolBox do MatLab (BOUGUET, 2003), na forma da função *normalize.m*. Esta função, que recebe como parâmetro as coordenadas do ponto na imagem e os parâmetros intrínsecos da câmera utilizada, retorna as coordenadas normalizadas do ponto no plano de imagem, já citadas anteriormente e designadas por $[X/Z\ Y/Z]^T$. Estas coordenadas, como foi dito, definem uma reta no sistema de coordenadas da câmera. No entanto, estas coordenadas se encontram em sistemas de coordenadas diferentes (cada câmera têm o seu próprio sistema). Para que a intersecção das retas definidas por estas coordenadas possa ser computada, é necessário obter estas coordenadas em um mesmo sistema. Esta transformação não é parte do ToolBox utilizado e precisou ser formalizada e implementada como segue abaixo.

No caso do arranjo estéreo, para cada câmera foi obtido, através da calibração, o conjunto de parâmetros extrínsecos correspondentes, mais especificamente a matriz de rotação R e o vetor de translação T. De posse destes dados a recuperação do ponto no ambiente se dá da seguinte maneira:

Seja $[X_1/Z_1\ Y_1/Z_1]^T = [a_1\ b_1]^T$ e $[X_2/Z_2\ Y_2/Z_2]^T = [a_2\ b_2]^T$ as coordenadas normalizadas do ponto de interesse nas câmeras 1 e 2, respectivamente. Seja R_1, T_1, R_2 e T_2 as matrizes de rotação e os vetores de translação destas câmeras. $R = R_1/R_2$ e $T = T_1 - T_2$ são a matriz de rotação e o vetor de translação do sistema de coordenadas da câmera 2 com relação à câmera 1, de maneira que:

$$\begin{bmatrix} X_2/Z_2 \\ Y_2/Z_2 \end{bmatrix} = R \begin{bmatrix} X_1/Z_1 \\ Y_1/Z_1 \end{bmatrix} + T \tag{10.26}$$

Se

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \text{ e } T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

então:

$$X_2 = r_{11}.X_1 + r_{12}.Y_1 + r_{13}.Z_1 + T_x$$

Porém,

$$X_2 = Z_2.a_2, X_1 = Z_1.a_1, Y_1 = Z_1.b_1$$

segue-se então que:

$$Z_2.a_2 = r_{11}.Z_1.a_1 + r_{12}.Z_1.b_1 + r_{13}.Z_1 + T_x$$

Além disso,

$$Z_2 = r_{31}.X_1 + r_{32}.Y_1 + r_{33}.Z_1 + T_z$$

Analogamente,

$$Z_2 = r_{31}.Z_1.a_1 + r_{32}.Z_1.b_1 + r_{33}.Z_1 + T_z \tag{10.27}$$

Aplicando-se 10.27 em 10.26, tem-se:

$$A_2.(r_{31}.Z_1.a_1 + r_{32}.Z_1.b_1 + r_{33}.Z1 + T_z) = r_{11}.Z_1.a_1 + r_{12}.Z_1.b_1 + r_{13}.Z_1 + T_x$$

Resolvendo-se para Z1:

$$Z_1 = T_x - a_2.T_z/(a_1.a_2.r_{31} + a_2.b_1.r_{32} + a_2.r_{33} - (a_1.r_{11} + b1.r_{12} + r_{13}))$$

Resolvido Z1, têm-se:

$$X_1 = a_1.Z_1 e Y_1 = b_1.Z_1$$

Obtendo-se, assim, as coordenadas do ponto no sistema de coordenadas da câmera 1. Para se obter as coordenadas no sistema de coordenadas do ambiente, designada aqui por $[X_w Y_w Z_w]^T$, basta aplicar a transformação abaixo, finalizando

a reconstrução tridimensional do ponto:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = R_1^{-1} \left(\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} - T \right)$$

O problema da correspondência

Até o momento, supôs-se conhecidas as coordenadas, nas duas imagens, do mesmo ponto de interesse do ambiente para realizar a reconstrução estéreo. No entanto, este é a maior dificuldade encontrada na reconstrução. Este problema, na verdade, divide-se em duas partes: como identificar, em uma das imagens, um ponto ou um conjunto de pontos significativos para a aplicação em questão e, dado este ponto ou conjunto de pontos, como encontrar, na segunda imagem, o mesmo ponto ou conjunto de pontos selecionados na primeira. A primeira parte do problema é relativamente dependente da aplicação, e portanto será tratada juntamente com a descrição da mesma, nas Seções 10.5.3 e 10.5.8. A segunda parte, no entanto, tem caráter mais geral.

Supõe-se que, além de selecionado o ponto de interesse na primeira imagem, existe uma coordenada inicial próxima à coordenada correspondente ao ponto de interesse na segunda imagem, ou seja, um ponto inicial cuja localização sabe-se, por restrição do domínio da aplicação, estar próxima da localização do ponto que se procura na segunda imagem.

Uma das técnicas utilizadas para a busca do ponto correspondente na segunda imagem, dada as suas coordenadas na primeira imagem, é a *adequação por modelo* (*template matching*). Nesta técnica, o modelo corresponde à região em torno do ponto de interesse na primeira imagem. É feita então uma busca na segunda imagem, a partir do ponto inicial disponível, por uma região cuja diferença dos valores de intensidade dos pixels em relação ao modelo seja a menor possível. Esta busca geralmente tenta minimizar o quadrado das diferenças entre os valores dos pixels do modelo e da região sendo procurada.

Para esta busca ser bem sucedida, o ponto de interesse ou o modelo, na primeira imagem, deve se sobressair, em termos de valores de intensidade de pixels, em relação ao resto da imagem ou pelo menos em relação à região próxima

do ponto de interesse. Se o modelo da primeira imagem ocorrer mais de uma vez nesta mesma imagem, corre-se o risco de a busca na segunda imagem resultar em uma região que não corresponde aos mesmos pontos do modelo, gerando ambigüidades. Este risco é maior quando não existe um ponto inicial ao redor do qual a busca deve ser realizada, obrigando a busca ser realizada em toda a segunda imagem.

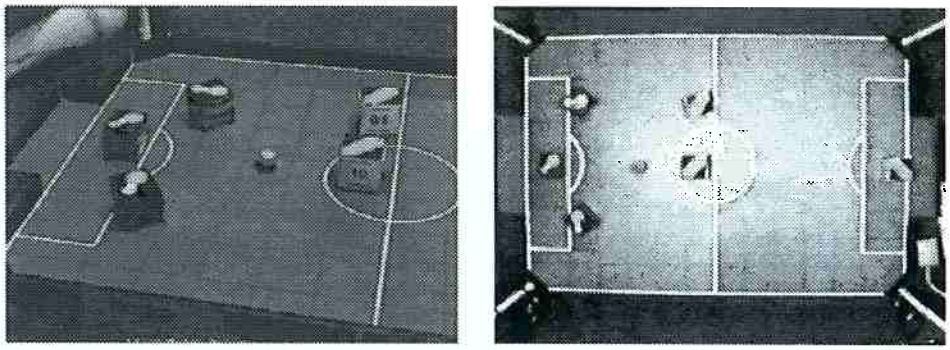
10.5 Aplicações Desenvolvidas

As várias técnicas de visão computacional descritas neste Capítulo têm sido empregadas pela autora em diferentes aplicações, nas mais diversas áreas, de modo integrado ou isolado.

Nesta Seção são descritos os algoritmos usados, suas particularidades e resultados alcançados nos sistemas de visão dos times de futebol de robôs, FUTEPOLI (BIANCHI; COSTA, 2000) e GUARANÁ (COSTA, 1999a; COSTA, 1999b; COSTA; PEGORARO, 2000), e no sistema de visão do robô Pioneer, no qual novos módulos estão sendo integrados.

Algumas técnicas de segmentação de imagens por padrões de cores, além de mostrarem resultados no domínio da Robótica Móvel (SIMÕES; COSTA, 2000b; SIMÕES; COSTA, 2000c; SIMÕES; COSTA, 2001; BONVENTI Jr.; COSTA, 2000b; BONVENTI Jr.; COSTA, 2000a; BIANCHI; SIMÕES; COSTA, 2001; BONVENTI Jr.; COSTA, 2002), também foram aplicadas com sucesso em outras áreas, como classificação de laranjas (SIMÕES; COSTA; ANDRADE, 2001; SIMÕES, 2002; SIMÕES; COSTA, 2003), segmentação de imagens de vistas aéreas (BONVENTI Jr.; COSTA, 2002), etc.

Da mesma forma, técnicas de extração de cena de fundo e de análise de movimento na imagem estão sendo aplicadas com sucesso em monitoramento de terminais rodoviários (SEIXAS; COSTA, 2002) e de plataformas metroviárias (MAIA Jr.; COSTA, 2002), respectivamente. Outra aplicação muito interessante, com resultados animadores, refere-se à utilização de estereoscopia binocular para medição de painéis. Estas aplicações e alguns resultados são mostrados a seguir.



O time FUTEPOLI em posição de defesa (esquerda) e uma imagem capturada pelo sistema de aquisição depois de ter o seu histograma equalizado (direita).

Figura 10.6: O time FUTEPOLI.

10.5.1 Visão do Time Futepoli

Esta Seção descreve o projeto e implementação do sistema de visão computacional utilizado no time de futebol de robôs FUTEPOLI, que participou da 1ª Copa Brasil de Futebol de Robôs. O projeto da FUTEPOLI envolveu a construção dos robôs, do sistema de comunicação, do sistema de visão, do sistema de controle e das estratégias usadas. A Figura 10.6 mostra imagens do time FUTEPOLI.

A contribuição principal deste trabalho é a proposta e desenvolvimento de um algoritmo simples e eficiente para a localização e identificação dos robôs do time e da bola de maneira rápida e simples, visando uma atuação em tempo real. O algoritmo consiste no rastreamento das cores dos objetos no espaço de cores e na localização de centros de objetos circulares.

O sistema descrito é responsável por obter, a partir da imagem adquirida da câmera disposta sobre o campo de jogo, a posição dos robôs adversários, a posição da bola e a identificação, posição e orientação dos robôs do time, através do reconhecimento de identificadores coloridos colocados sobre cada robô.

Os elementos principais do sistema de visão computacional são: aquisição da imagem, calibração do sistema, rastreamento das cores dos objetos no espaço de cores e localização dos robôs e da bola na cena.

A aquisição da imagem é feita através de uma placa digitalizadora padrão e uma câmera de vídeo padrão RGB. As imagens são adquiridas a uma taxa de 30

quadros por segundo. O tamanho usado das imagens é de 640 x 480 x 24 bits de cores. Um exemplo de imagem típica trabalhada pode ser vista na Figura 10.6-direita.

Para que o sistema possa conhecer a posição real dos objetos a partir da imagem capturada é necessário realizar uma calibração do sistema. Para tanto, basta definir, no início do jogo, a localização do campo na imagem, o que é feito com 2 *clicks* de mouse, um no canto superior esquerdo do campo na imagem e outro no canto inferior direito. A calibração das cores também é realizada ao início de uma partida, quando o usuário utiliza o mouse para, a partir das imagem capturadas em tempo real, definir as cores que serão utilizadas.

O maior problema de um sistema de visão computacional para o jogo de futebol de robôs, é o fato da luminosidade no campo não ser homogênea, como pode ser observado na Figura 10.6-direita, que mostra uma imagem após ter seu histograma equalizado.

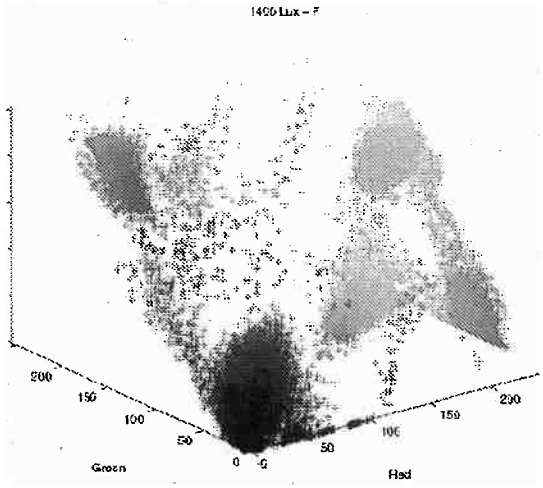


Figura 10.7: Cores de interesse para o futebol de robôs, no espaço RGB.

A técnica de imposição de limiares para classificação de cores define regiões retangulares no espaço de cores para limitação de pertinência em cada classe de cor. No entanto, como pode ser visto na Figura 10.7, paralelepípedos no espaço tricomático não são as representações de classes de cores mais apropriadas para o domínio de interesse.

Para solucionar este problema, foi proposto e desenvolvido um sistema de visão que varia dinamicamente as dimensões dos paralelepípedos que definem as classes de cores. A idéia consiste em definir um pequeno paralelepípedo que

representa cada cor definida pelo operador, na posição de início de jogo – portanto, cada cor estará sob uma determinada condição de iluminação, de acordo com sua posição inicial no campo de futebol. À medida que cada objeto se move no campo, o mesmo é rastreado e os limiares que definem sua cor, modificados para adaptação às novas condições, uma vez que estas condições se modificam de forma suave, permitindo seu rastreamento na taxa de 30 quadros por segundo.

Assim, este sistema inicializa a cor da bola, dos 3 robôs do time e a cor do time adversário durante a calibração e atualiza o valor da cor destes objetos a cada quadro com a média das 3 últimas medidas.

De posse da cor de cada objeto com precisão, é fácil determinar a sua localização durante a partida. Basta realizar uma varredura na imagem à procura da cor desejada, com passos menores que a metade do tamanho do menor objeto de interesse na cena.

Para calcular a posição da bola foi implementado um algoritmo simples e eficiente: a partir do ponto inicialmente encontrado (P_ini) com a cor da bola, trace uma reta horizontal e encontre o ponto de cruzamento desta reta com as bordas do círculo (P_1 e P_2). O ponto médio desta reta (P_{cx}) define a posição x do centro do círculo. Trace outra reta a partir do ponto médio encontrado, na vertical, e encontre as bordas do círculo (P_3 e P_4). O ponto médio desta segunda reta é o centro do círculo ($P_{central}$). A Figura 10.8-esquerda ilustra este algoritmo.

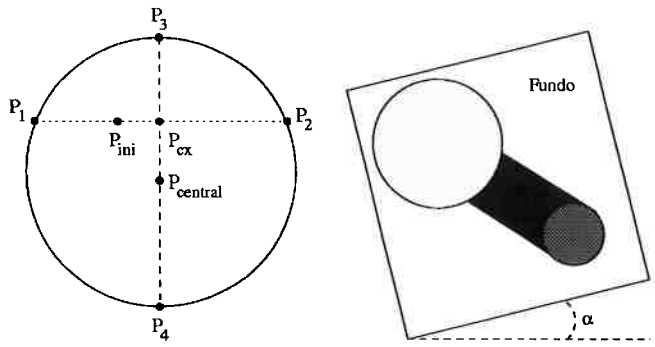


Figura 10.8: Cálculo do centro da bola (esquerda) e da posição e orientação do robô (direita).

Para identificar e localizar os robôs, cada um possui em sua parte superior uma identificação colorida baseada em círculos, mostrada na Figura 10.8-direita.

Como pode ser visto, esta identificação é composta por dois círculos de ta-

manhos e cores diferentes ligados por um retângulo de cor preta. Cada robô do time é identificado pela combinação das cores dos seus círculos. A cor de fundo é a cor identificadora do time.

Para encontrar a posição e orientação de um robô utiliza-se o seguinte algoritmo: *(i)* encontre o centro do círculo maior (mesmo algoritmo usado para a bola); *(ii)* procure o círculo menor, que está a uma distância conhecida do círculo maior; *(iii)* encontre o centro do círculo menor; *(iv)* considerando a distância (conhecida) entre os centros dos círculos maiores e menores nos robôs, defina pares como pertencentes a um único robô; *(v)* caso haja mais de uma combinação possível entre pares (círculo maior, círculo menor) – por alinhamento acidental dos robôs na cena – verifique se existe a conexão (cor preta) entre eles, definindo um único par para cada robô do time; *(vi)* trace a reta entre os 2 centros: seu ponto médio define o centro do robô e sua inclinação, a orientação do robô.

A avaliação do time FUTEPOLI durante a 1ª Copa Brasil de Futebol de Robôs permitiu concluir que o sistema de visão computacional utilizado foi robusto às mudanças de luminosidade, rápido, confiável e de fácil calibração, sendo um dos poucos times que sempre conseguia realizar a calibração no tempo estabelecido pelo regulamento (BIANCHI; COSTA, 2000). Ainda, o sistema foi capaz de separar cores que outros times não faziam distinção, por exemplo, rosa e laranja. No entanto, a definição de cores adequadas para fácil classificação – e que não fossem de uso restrito pelas regras dos jogos – tornou-se um problema, na medida em que os times adversários também poderiam usar rótulos das mais diversas cores. Assim, o sistema de visão foi modificado para uso no time GUARANÁ, descrito a seguir.

10.5.2 Visão do Time Guaraná

Assim como no time FUTEPOLI, o reconhecimento dos objetos de interesse no campo – jogadores e bola – é baseado nas cores da imagem capturada pela câmera, sendo que um conjunto de pixels adjacentes de cor laranja identifica a bola; de cor azul, os robôs de um time e de cor amarela, os robôs de outro time. Para completar identificação, localização e determinação da orientação dos robôs do time GUARANÁ, além da etiqueta da cor do time, exigida pelas regras, uma segunda etiqueta de cor rosa foi utilizada, ambas margeadas por uma moldura

preta, conforme apresentado na Figura 10.9. A cor rosa foi escolhida por não ser uma das cores reservadas na categoria MIROSOT e por apresentar razoável facilidade de separação no espaço de cores restantes, no espaço RGB utilizado.

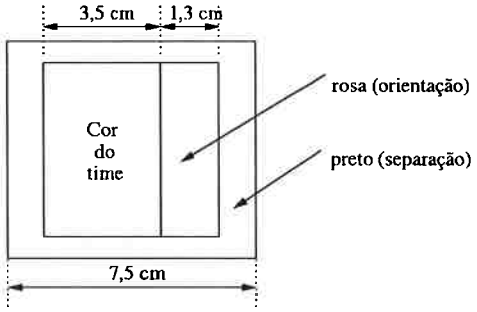


Figura 10.9: Etiqueta de identificação dos robôs do time GUARANÁ.

Nos vinte minutos que antecedem o início de uma partida, as equipes devem preparar os times, instalando equipamentos e calibrando o módulo de visão computacional. Nesta fase de calibração, a visão computacional é preparada para adaptar-se à iluminação do ambiente, reconhecer cores, localizar objetos, relacionar o espaço-imagem (pixels) com o espaço-campo (centímetros), definir cores do time e do adversário, identificar campos de ataque e de defesa, etc. Duas destas tarefas são de extrema importância e serão detalhadas a seguir: definição dos valores limiares para classificação das cores e construção de um modelo do campo vazio, que será a cena de fundo do jogo.

Classificação das cores

A técnica utilizada para classificação de cores foi a de imposição de limiares (veja Seção 10.1). A definição dos valores limites para a classificação das cores utilizadas no jogo – azul, amarela, rosa e laranja – é feita através de interação direta com um operador humano. Numa interface apresentada ao operador, deve-se inicialmente selecionar qual cor será calibrada. A seguir, imagens do campo são continuamente apresentadas, onde a cor de interesse deve ser selecionada pelo operador. Para cada seleção, ou seja, para cada pixel indicado pelo operador, são calculadas as relações R/G e G/B do valor RGB do pixel. Esses valores são mostrados em duas faixas correspondentes, abaixo da imagem do campo, para acompanhamento visual pelo operador. Após diversas seleções, os limites superior e inferior das relações R/G e G/B são estabelecidos como valores limites para

serem usados na classificação da cor calibrada. Este processo deve ser repetido para cada cor de interesse.

Para maior robustez na classificação, os elementos contendo a cor que estiver sendo calibrada devem ser dispostos em diferentes posições no campo, para que os valores calculados reflitam a não homogeneidade normalmente existente na iluminação. O operador deve repetir a calibração – ou redefinir limites – caso observe superposição dos limites de aceitação para diferentes cores.

Modelo de Cena de Fundo

Durante uma partida, para acelerar o processamento de imagens, é realizada uma fase preliminar de subtração da imagem atual do campo com uma imagem modelo do campo vazio – sem jogadores e sem a bola – chamado modelo de cena de fundo. Esta subtração é realizada somente nas bandas R e G da imagem. A definição da imagem que será usada como modelo fixo de cena de fundo (veja Seção 10.2) também é realizada na fase de calibração do sistema. São adquiridas n imagens do campo vazio, a um intervalo Δt entre as aquisições, para melhor refletir as oscilações de luminosidade devidas tanto à iluminação não homogênea quanto aos ajustes automáticos de algumas câmaras utilizadas. No time GUARANÁ, os valores de n e Δt foram definidos empiricamente. Para determinar o modelo de cena de fundo, calcula-se a média entre as n imagens:

$$\overline{p}(x,y) = \frac{\sum_{i=1}^{n} p_i(x,y)}{n} \tag{10.28}$$

onde $\overline{p}(x,y)$ é o valor do pixel na coordenada (x,y) do modelo de cena de fundo; $p_i(x,y)$ é o valor do pixel na coordenada (x,y) da $i-ésima$ imagem do campo vazio; n é o número considerado de imagens; $x_i \leq x \leq x_f$ e $y_i \leq y \leq y_f$, sendo (x_i, y_i) e (x_f, y_f) as coordenadas superior esquerda e inferior direita que definem os limites do campo na imagem, refletindo a área de interesse na imagem. Esses limites dependem da fixação da câmera em relação ao campo, a qual deve ser feita de forma que todo o campo possa ser visualizado na imagem e que longitudinalmente o campo esteja alinhado com o eixo horizontal da imagem.

A partir do modelo, calcula-se os desvios dos valores dos pixels em relação às

outras n imagens, definindo limites de variação em R e G:

$$L_R = max(\mid r(\overline{p}(x,y)) - r(p_i(x,y)) \mid) \quad (10.29)$$

$$L_G = max(\mid g(\overline{p}(x,y)) - g(p_i(x,y)) \mid) \quad (10.30)$$

onde L_R e L_G são os limites dos componentes vermelho e verde da imagem, respectivamente, que serão utilizados na subtração das imagens, durante a fase de execução do sistema; $r(.)$ e $g(.)$ são os componentes vermelho e verde do pixel, respectivamente; $\overline{p}(x,y)$ é o valor do pixel na coordenada (x, y) do modelo de cena de fundo, com $1 \leq i \leq n$ e $x_i \leq x \leq x_f$ e $y_i \leq y \leq y_f$; n é o número de imagens consideradas; (x_i, y_i) e (x_f, y_f) são as coordenadas superior esquerda e inferior direita que definem os limites do campo na imagem.

Identificação e Rastreamento

Para enviar um comando a um determinado robô, o sistema deve ser capaz de identificar este robô, além de determinar sua posição, orientação e velocidade atual. No entanto, no time GUARANÁ, as etiquetas de todos os robôs são idênticas e, portanto, não podem ser utilizadas para distinguir um determinado robô dos outros membros do time. Desta forma, é necessário o uso de um processo de rastreamento dos robôs, a partir de uma identificação realizada por um operador humano no início do jogo.

Subtração inicial de imagens

Quando uma imagem é capturada, gera-se inicialmente a imagem diferença $p_d(x,y)$ da subtração desta imagem atual $p_a(x,y)$ com o modelo $\overline{p}(x,y)$:

$$r(p_d(x,y)) = \mid r(\overline{p}(x,y)) - r(p_a(x,y)) \mid \quad (10.31)$$

$$g(p_d(x,y)) = \mid g(\overline{p}(x,y)) - g(p_a(x,y)) \mid \quad (10.32)$$

Se $r(.) > L_R$ e $g(.) > L_G$, o pixel em questão deve corresponder a elementos de interesse na imagem (podendo também ser ruído). Na imagem diferença são determinados conjuntos de pixels *conectados* (veja Seção 10.1) de valores *similares*. A medida de similaridade é definida pelos valores limites das relações R/G e G/B das cores de interesse, determinadas na fase de calibração.

Determinação do Centróide

Pixels adjacentes conectados com cores similares são considerados componentes de um mesmo elemento, desde que o número de pixels conectados esteja na faixa de limiares previamente estabelecidos na fase de calibração do sistema (em função do tamanho do elemento naquela respectiva cor, da resolução da imagem e da posição da câmera). Os que estiverem fora dos limiares são considerados ruídos e são desprezados. Para localizar estes elementos na imagem, a coordenada (x, y) do centróide (centro de massa que, no caso, corresponde ao centro de área) de cada elemento é calculada:

$$x = \frac{\sum_{i=1}^{k} x_i}{k} \quad e \quad y = \frac{\sum_{i=1}^{k} y_i}{k} \tag{10.33}$$

onde (x_i, y_i) são as coordenadas dos pixels e k é número de pixels conectados que compõem o elemento.

A finalidade da borda preta usada nas etiquetas dos robôs do time GUARANÁ é a de evitar que duas etiquetas de uma mesma cor, porém colocadas em dois robôs distintos que estejam em contato, possam ser identificadas como um único elemento, evitando um indesejado alinhamento acidental.

Rastreamento

Quando uma partida é iniciada, cada robô é identificado por um operador humano. A partir deste instante, a identificação se dá por intermédio de um algoritmo de rastreamento, que usa um método de otimização exaustiva para encontrar a solução de mínimo custo do seguinte sistema:

$$s(i, j, r) = d(e_{time}(i), e_{orient}(j)) + d(e_{time}(i), e_{ant}(r)) \tag{10.34}$$

sujeito a: $li_{time} \leq ne_{time}(i) \leq ls_{time}$, $li_{orient} \leq ne_{orient}(i) \leq ls_{orient}$, $0cm \leq d(e_{time}(i), e_{ant}(r)) \leq 14cm$ e $4cm \leq d(e_{time}(i), e_{ant}(r)) \leq 7cm$. Sendo que $r = 1, 2, 3$ é o número do robô considerado, $i = 1, 2, \ldots, n$ é o número de um dos n elementos da cor do time encontrado na imagem; $j = 1, 2, \ldots, m$ é o número de um dos m elementos da cor rosa na imagem; $s(i, j, r)$ é a função objetivo a ser minimizada; $d(e_1, e_2)$ é a distância cartesiana entre os elementos e_1 e e_2 no sistema de coordenadas (em cm); $e_{time}(i)$ é o centróide do $i - ésimo$ elemento

com a cor do time; $e_{orient}(j)$ é o centróide do $j-ésimo$ elemento de orientação – etiqueta de cor rosa; $e_{ant}(r)$ é o centróide do elemento da cor do time do $r-ésimo$ robô identificado na imagem anterior; $ne_{time}(i)$ é o número de pixels no $i-ésimo$ elemento da cor do time; $ne_{orient}(j)$ é o número de pixels no $j-ésimo$ elemento de orientação (cor rosa); li_{time} e ls_{time} são os limites inferior e superior de número de pixels, que definem a faixa válida do tamanho do elemento (etiqueta) da cor do time; li_{orient} e ls_{orient} são os limites inferior e superior de número de pixels, que definem a faixa válida do tamanho do elemento (etiqueta) de orientação (cor rosa).

Os três pontos de mínimo (combinação de i, j e r que resultam nos três menores valores da função s) encontrados através deste sistema representam o relacionamento da etiqueta da cor do time identificado por i com a etiqueta de orientação (rosa) identificada por j, referentes ao robô r. Os valores li_{time}, ls_{time}, li_{orient} e ls_{orient} mudam de acordo com o tamanho da etiqueta, da resolução da imagem e da posição da câmara. O sistema utilizado, na prática, tem se mostrado muito eficaz. Não se observou perdas ou trocas de identificação no rastreamento dos robôs em situações normais de jogo.

Orientação e Posição

A orientação do robô é definida pelo ângulo α medido do eixo x do campo até o vetor (c_p, c_t), com c_p sendo o centróide rosa e c_t, o centróide da cor do time, conforme Figura 10.10:

$$\alpha = \arctan(\frac{y_t - y_p}{x_t - x_p}) \tag{10.35}$$

Para as dimensões das etiquetas usadas pelo robô e a resolução da imagem usada, foi observado um desvio máximo de ± 10 graus. Esse desvio foi medido experimentalmente comparando o resultado apresentado pelo sistema de visão e o valor real medido na orientação do robô.

A posição (x, y) de cada robô do time resulta da média ponderada das posições dos centróides da etiqueta do time e da etiqueta rosa:

$$x = \frac{c_1 x_t + c_2 x_p}{c_1 + c_2} \quad e \quad y = \frac{c_1 y_t + c_2 y_p}{c_1 + c_2} \tag{10.36}$$

onde (x_t, y_t) é a coordenada do centróide do elemento da cor do time; (x_p, y_p) é

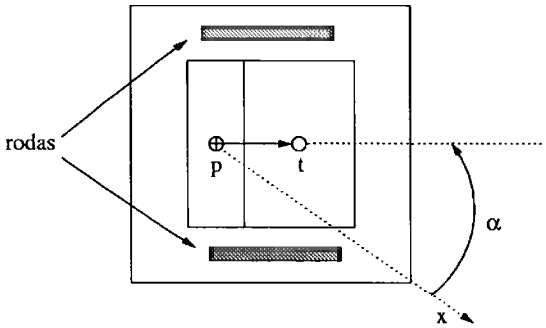


Figura 10.10: Orientação do robô a partir das etiquetas.

a coordenada do centróide do elemento rosa e c_1 e c_2 são fatores dependentes do tamanho das etiquetas.

A posição dos robôs adversários e da bola são definidos pelas coordenadas dos centróides dos elementos das cores do time adversário e da bola, respectivamente. Suas orientações são inferidas pela direção do movimento, dada pela diferença de coordenadas entre a posição dos elementos na imagem atual e na anterior, uma vez que não existe outra informação disponível sobre os adversários.

Após identificar cada objeto na imagem, o sistema de coordenadas da imagem (em pixels) é transformado em sistema cartesiano do campo.

O módulo completo de visão computacional implementado é bastante rápido (7 ms por quadro) e eficaz, permitindo algumas variações nas condições de iluminação e nas cores do campo, etiquetas e bola. A fase de calibração, nas condições normais de jogo, é sempre rápida e simples de ser executada graças à interface amigável entre homem-máquina desenvolvida. Esta interface possibilita facilidades tanto na captura das imagens do campo vazio quanto na identificação das faixas válidas para as cores. Descrições detalhadas das propostas, desenvolvimentos e resultados podem ser encontrados em (COSTA, 1999a; COSTA, 1999b; COSTA; PEGORARO, 2000).

10.5.3 Visão do robô Pioneer

Tarefas fundamentais para a navegação em robótica móvel são a detecção de obstáculos e de alvos em ambientes dinâmicos e imprevisíveis. Estas tarefas dividem-se em duas partes fundamentais: a identificação do objeto de interesse (seja um alvo ou um obstáculo), e sua localização no ambiente, determinando sua

posição em relação ao robô.

Identificação dos obstáculos e alvos

Um dos atributos da imagem de um objeto muito utilizado para a sua identificação é a sua cor, ou a cor de marcadores posicionados no mesmo para a sua identificação. Isto porque os algoritmos para a segmentação de regiões baseada em cores são relativamente simples, permitindo resposta em tempo real do robô (veja Seção 10.1). Neste trabalho o CMVision (*Color Machine Vision*) (BRUCE; BALCH; VELOSO, 2000) está sendo utilizado para segmentar, de acordo com suas cores, os objetos de interesse na cena.

Foi necessário o desenvolvimento de um programa para possibilitar a captura das imagens e o controle dos limiares, e integrá-lo com o CMVision, que foi escrito em C++ . A plataforma escolhida para o desenvolvimento foi o JAVA, pelo seu caráter multi-plataforma e por possuir uma API de captura e tratamento de vídeo em tempo real (JMF – *Java Media Framework*) que facilitou significativamente o desenvolvimento. Além disso, há uma outra API do Java para o interfaceamento com código C/C++ , denominada JNI – *Java Native Interface*, que possibilitou sua integração com o código do CMVision.

Os resultados da segmentação de cores são os centróides das regiões encontradas na imagem, acompanhados dos identificadores das cores detectadas e das quantidades de pixels presentes em cada região. Estas informações permitem determinar se a região é um obstáculo ou um alvo (com base na cor), se não se trata de ruído (regiões com pequeno número de pixels) e qual é o ponto de referência do objeto na imagem (centróide). Experimentos realizados com o robô Pioneer utilizaram como alvo uma bola de cor laranja e os obstáculos, cilindros azuis dispostos aleatoriamente na sala onde o robô executa tarefas de navegação, buscando aproximar-se do alvo, enquanto evita colisões.

Estereoscopia para a localização de alvos e obstáculos

Além de identificar o alvo e os obstáculos, é necessário localizá-los no ambiente, ou seja, determinar quais as suas posições relativas ao robô para que este possa reagir adequadamente a eles. É nesta parte do problema que a estereoscopia

se encaixa (veja Seção 10.4). Para a correspondência entre as duas imagens da estereoscopia binocular, as regiões em torno dos centróides detectados na primeira imagem serão os modelos, e os centróides detectados na segunda imagem serão os pontos iniciais para a busca iterativa de adequação do modelo. No que diz respeito à calibração, a validade do modelo obtido para as câmeras não são um problema nesta aplicação. A região em torno do robô onde se deseja localizar objetos de interesse tipicamente não ultrapassa 3 ou 4 metros de distância, região que pode ser modelada precisamente (menos de 5% de erro) com gabaritos de calibração não maiores que uma folha tamanho A0.

A Figura 10.11[3] ilustra o resultado da aplicação do processo de segmentação por cores, determinação dos centróides e estereoscopia para estimar distâncias do sistema desenvolvido.

Divergente do Fluxo Ótico para Navegação

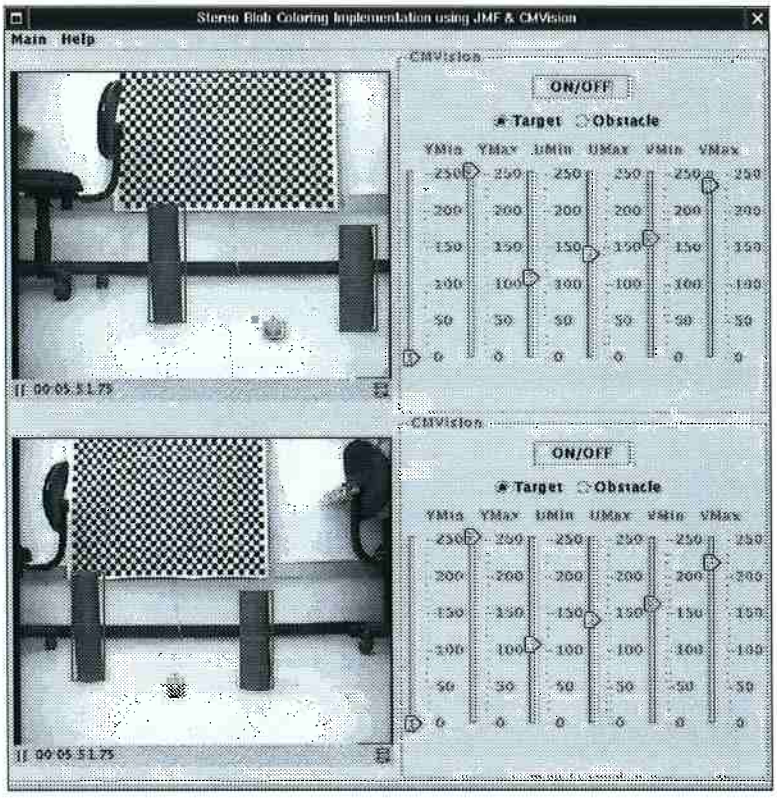
Outro módulo de percepção visual que está sendo implementado no robô Pioneer consiste naquele que introduz um comportamento denominado *moveToFree* (veja Seção 11.1), que busca espaços livres no ambiente para permitir a movimentação do robô. Este comportamento inicialmente calcula o fluxo ótico da seqüência de imagens adquiridas por uma câmera instalada sobre o dorso do robô móvel, realizando previamente uma sub-amostragem das imagens, e então calculando o divergente em todos os pontos.

Prova-se que o divergente do fluxo ótico em um ponto é aproximadamente proporcional ao inverso do tempo de colisão τ entre a câmera e a projeção deste ponto na cena real (CAMUS, 2002).

A vantagem no método do divergente consiste no fato de, teoricamente, não ser afetado pela rotação da câmera sobre seu eixo, fornecendo uma informação robusta da estrutura da cena para um observador em movimento.

Um filtro (mediano) é aplicado tanto espacialmente quanto temporalmente no campo do divergente para reduzir flutuações momentâneas. Então, a partir da seqüência filtrada do divergente e da percepção do alvo, um mapa de riscos é elaborado, onde é identificado um caminho a seguir.

[3] Veja esta figura colorida no Apêndice II.



O programa desenvolvido, resultante da integração das técnicas de segmentação por cores, determinação dos centróides e estereoscopia. Dois obstáculos (cilindros) e um alvo (bola) são segmentados nas imagens das duas câmeras. Graças ao deslocamento dos centróides de uma imagem para a outra, é possível recuperar a informação tridimensional dos objetos. Os controles deslizantes ao lado das imagens têm a função de definir os limiares, no espaço de cores YUV, para a segmentação.

Figura 10.11: Interface do sistema de visão do Pioneer.

Este mapa nada mais é do que a identificação, na seqüência de informações filtradas do divergente, de áreas onde ele é pequeno – o que significa grande tempo de colisão τ, ou seja, maior distância a algum obstáculo. Neste caso, o objetivo do comportamento é apenas avançar por onde for possível. No caso de ter algum outro objetivo, deve-se compor o mapa de risco advindo do divergente do fluxo ótico com um mapa de risco elaborado a partir da distância ao alvo (dada por estereoscopia ou fornecida por um operador humano).

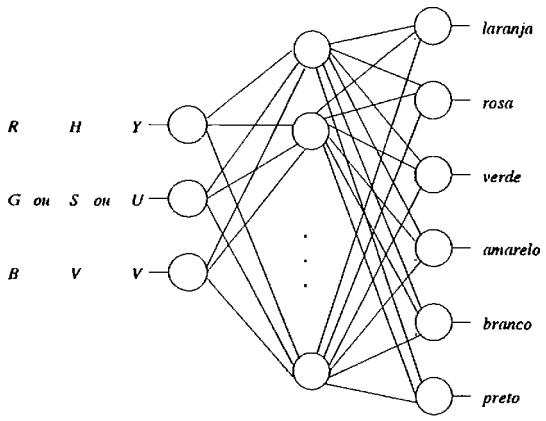
Estuda-se atualmente a possibilidade de representar o mapa de risco em termos de campos potenciais, para inserir na arquitetura REACT desenvolvida pela autora e seu grupo (veja Seção 11.1).

10.5.4 Aplicações de Classificação de Padrões de Cores por Redes Neurais

De acordo com os estudos realizados e descritos na Seção 10.1.1, foi observado que, para classificação de cores, o uso de redes neurais do tipo MLP, treinadas com o algoritmo de retropropagação de erros se mostrou bastante eficiente. Assim, redes neurais foram aplicadas na classificação de cores no domínio do futebol de robôs e da classificação de laranjas.

Futebol de robôs

Para a implementação da classificação foi usado o MLP mostrado esquematicamente na Figura 10.12 com 3 neurônios na camada de entrada, 10 neurônios na camada oculta e 6 neurônios na camada de saída, correspondentes às 6 classes de interesse – laranja, rosa, verde, amarelo, branco e preto (SIMÕES; COSTA, 2000a; SIMÕES; COSTA, 2000b; SIMÕES; COSTA, 2000c; SIMÕES; COSTA, 2001). À entrada do MLP apresenta-se uma trinca expressa em algum sistema de representação de cor (RGB, HSV ou YUV) de cada um dos pixels pertencentes a uma imagem (um a um), tanto no treinamento quanto na execução. Na saída da rede obtém-se a cor classificada como pertencente a uma das 6 classes de cores para as quais o sistema foi treinado.



MLP usado: 3 neurônios de entrada, 10 neurônios na camada intermediária e 6 neurônios de saída, representando as classes de interesse.

Figura 10.12: Rede Neural usada.

Os testes foram realizados com imagens comuns nesse domínio, como a mos-

trada na Figura 10.13[4]. Nessa imagem pode-se identificar classes distintas de cores de interesse: laranja, rosa, verde, amarelo, branco, azul. Algumas outras cores presentes no fundo da imagem não são de interesse para o jogo e foram aqui reunidas em uma única classe genérica: preto.

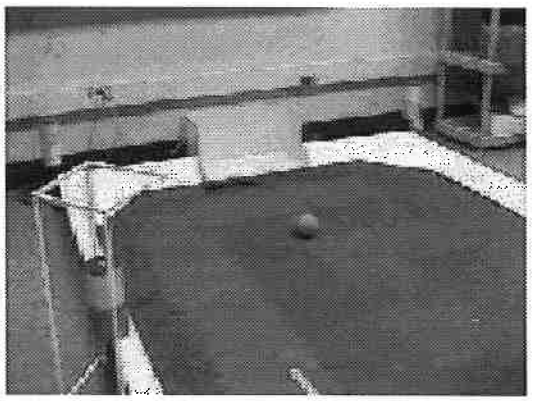


Figura 10.13: Imagem do campo de futebol de robôs utilizada para testes.

Um único pixel desta imagem foi escolhido aleatoriamente para representar cada classe (e dois para representar a classe fundo, devido às diferenças entre as cores dos pixels englobados nesta classe). Os pixels usados no treinamento para a classificação de cores representados no sistema RGB encontram-se listados na tabela abaixo.

R	G	B	Classe
208	84	24	Laranja
242	42	188	Rosa
53	129	97	Verde
213	192	11	Amarelo
209	228	246	Branco
158	146	184	Preto
29	24	32	Preto

As trincas no sistema de representação RGB apresentadas na tabela foram fornecidas à rede para treinamento. Depois de treinado o MLP, a imagem original (Figura 10.13) foi reapresentada a esta, na totalidade de seus pixels, para execução da classificação. Este processo foi repetido, transformando o sistema

[4]Veja esta figura colorida no Apêndice II.

de representação de cores para HSV e YUV (para treinamento e validação). Os resultados obtidos são mostrados na forma de imagens, sendo cada pixel apresentado de acordo com a classe a ele atribuída pela rede, conforme observado na Figura 10.14[5].

Imagens de resposta resultantes da apresentação da imagem da Figura 10.13 às redes neurais treinadas utilizando os sistemas de representação de cores: (a) RGB; (b) HSV e (c) YUV.

Figura 10.14: Resultados do uso de MLPs para o futebol de robôs.

A função de mapeamento da rede neural será tão melhor quanto maior o número de exemplos apresentados à rede. No entanto, buscou-se a melhor aproximação para a função com o menor número de exemplos possível, devido às restrições no tempo de calibração antes dos jogos.

As cores adotadas no futebol de robôs são escolhidas para que sejam mais facilmente separáveis no sistema RGB, daí o melhor resultado conseguido neste sistema de representação de cores. O sistema YUV ficou, em qualidade, um pouco abaixo deste, apresentando ainda uma boa classificação, mas com um nível de ruído maior. A classificação utilizando o sistema de representação HSV forneceu os piores resultados, tipicamente por possuir uma função de mapeamento mais complexa e descontínua. Vale observar que apenas 1 (um) exemplo de cada classe foi utilizado. Certamente a resposta da rede para todos os sistemas de representação de cores tenderia a ser homogênea quando treinada à exaustão.

Classificação de laranjas

A classificação e a seleção são problemas comums em sistemas de produção de frutos. O procedimento usual para a realizar esta tarefa é a inspeção visual humana considerando atributos gerais da fruta. Os atributos relevantes no contexto da classificação são padronizados por instituições em diferentes países.

[5]Veja esta figura colorida no Apêndice II.

O padrão brasileiro de classificação da laranja segundo o *Centro de Qualidade em Horticultura* (CENTRO..., 2000) – adotado no contexto deste trabalho – propõe a classificação de laranjas com base em dois aspectos distintos: cor e qualidade. Com relação à cor, as laranjas são agrupadas em cinco diferentes classes (C1 a C5), e toda a informação fornecida sobre estas classes é a informação visual mostrada na Figura 10.15[6]. Já o parâmetro *qualidade* é observado segundo a ocorrência de defeitos de ordem mecânica, patológica, presença e intensidade de manchas e podridão.

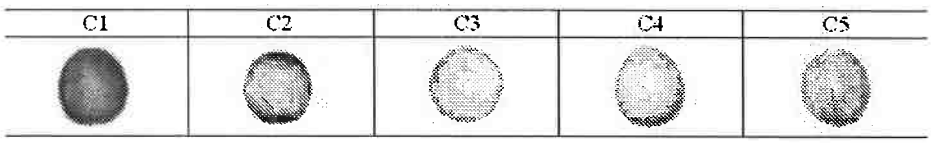


Figura 10.15: Padrão brasileiro para classificação de laranjas segundo sua coloração.

Diferentes abordagens têm sido propostas para a automação da classificação de frutas (NAKANO, 1997; ZHANG, 1997; KONDO, 2000). A arquitetura de muitos desses sistemas apresenta quatro etapas bem definidas: *i)* classificação dos pixels da imagem; *ii)* segmentação da imagem; *iii)* análise do objeto segmentado e *iv)* comparação dos padrões obtidos com padrões previamente conhecidos. Contudo, as estratégias e ferramentas usadas para realizar cada uma destas subtarefas não são consensuais. Buscando um sistema capaz de realizar a tarefa da classificação visual de laranjas com robustez às variaçõe das condições do ambiente e baseado em padrões com descrições vagas e imprecisas, propôs-se a abordagem apresentada de forma simplificada na Figura 10.16 (SIMÕES; COSTA; ANDRADE, 2001; SIMÕES, 2002; SIMÕES; COSTA, 2003).

A tarefa da *classificação de pixels* consiste em separar adequadamente os pixels da imagem de cada laranja em um espaço de cores. As redes neurais artificiais apresentam-se como uma ferramenta bastante propícia a este trabalho, já que o perceptron multicamada (MLP) treinado com o algoritmo da retropropagação do erro é capaz de expressar uma rica variedade de superfícies não-lineares de decisão. As classes de cores relevantes para o treinamento, no problema, são as classes de cores de pixels identificadas pela quase totalidade das pessoas

[6]Veja esta figura colorida no Apêndice II.

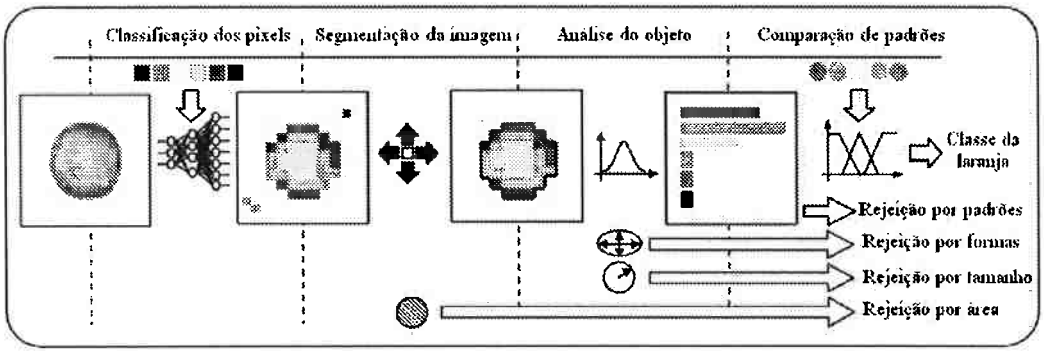


Figura 10.16: Visão geral da abordagem proposta.

observando a Figura 10.15: *verde-escuro, verde-claro, amarelo, laranja-claro* e *laranja-escuro.*

A rede neural proposta é mostrada na Figura 10.17. Este MLP mapeia o nível RGB de cada pixel de uma imagem para uma das cinco classes, acrescidas de branco (cor do fundo) e uma cor destinada à identificação de manchas.

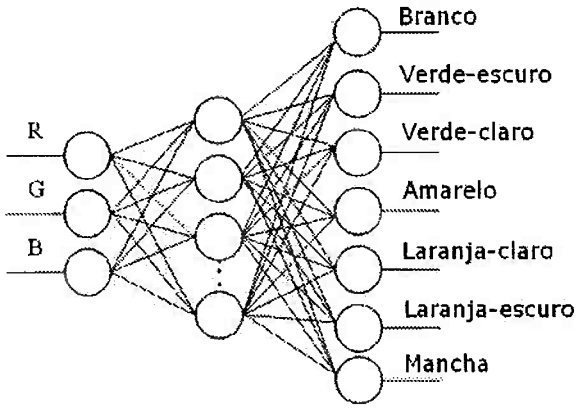


Figura 10.17: A rede neural artificial utilizada para a classificação de pixels.

Considerando uma imagem com seus pixels classificados por uma RNA, a *segmentação da imagem* – no problema, a identificação dos pixels pertencentes a cada laranja – pode ser obtida pela aplicação do bem conhecido algoritmo de agrupamento em regiões (veja Seção 10.1.2) para as cores de interesse. Este procedimento mapeia os pixels individuais em conjuntos de pixels chamados regiões tomando como base suas cores e as conexões entre estes.

A fase de *análise dos objetos* consiste em avaliar as laranjas segundo suas características de forma e cor. A análise da forma dos frutos pode ser realizada

através da verificação, em cada laranja, de algumas propriedades tais como o número de pixels da fruta, altura, largura, razão altura-largura, menor e maior raio, etc. A análise feita nesta fase permite ao menos três tipos de rejeições: *i)* rejeição por tamanho; *ii)* rejeição por formas inesperadas e *iii)* rejeição por excesso de manchas. Esta última pode ser obtida pela simples observação de um vetor (denominado vetor de atributos) que contém o número de pixels pertencentes a cada classe de cores.

De forma a finalizar a análise, o sistema precisa ser robusto no sentido de realizar uma *comparação dos padrões* entre as laranjas observadas e as definidas pelos padrões (classes C1 a C5 na Figura 10.15). Baseado na natureza vaga e imprecisa das descrições das laranjas típicas de cada classe, foi proposto um classificador nebuloso para esta tarefa. Nesta abordagem, a entrada do sistema nebuloso é o vetor de atributos obtido na fase anterior, e sua saída, a classe a que pertence a laranja apresentada. A Figura 10.5.4 apresenta um esquema simplificado deste estágio.

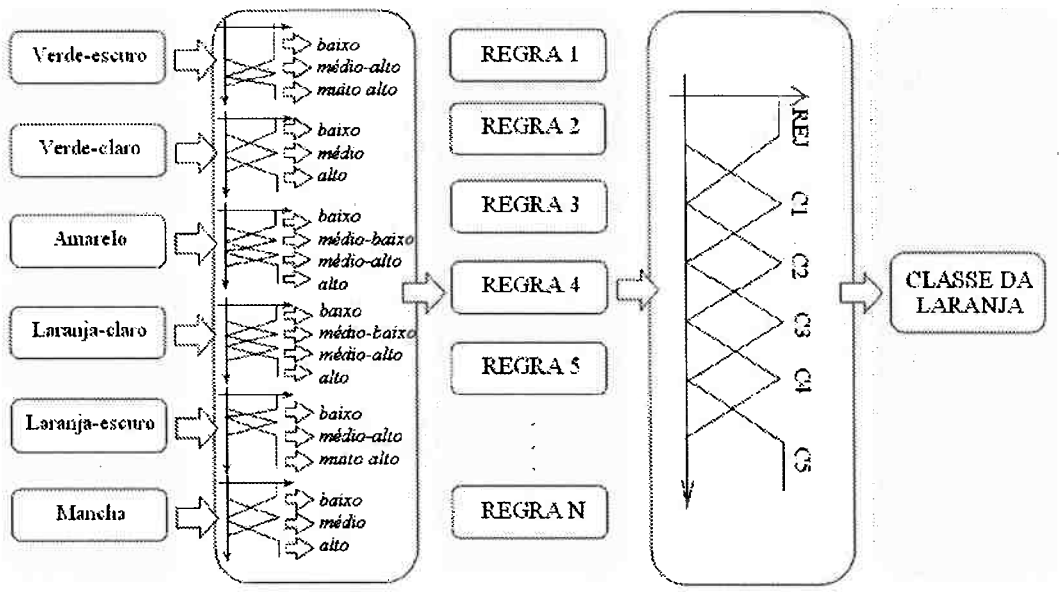


Figura 10.18: O sistema nebuloso para a comparação de padrões.

Na fase de fuzificação, os graus de pertinência de cada variável de entrada para cada variável lingüística são determinados pela aplicação do vetor de atributos às funções de pertinência do sistema nebuloso. Então, regras nebulosas expressando o conhecimento de especialistas humanos são aplicadas sobre tais valores, e depois da fase de defusificação, uma classe pode ser atribuída para cada laranja. As

regras nebulosas utilizadas são mostradas na tabela abaixo.

Regra	**Antecedentes**	**Conseqüentes**
1.	Se (verde-escuro = *muito alto*)	classe = C1
2.	Se (verde-escuro = *médio-alto*) e (laranja-claro = *médio-baixo*)	classe = C2
3.	Se (amarelo = *médio-alto*) e (laranja-claro = *médio-baixo*)	classe = C3
4.	Se (amarelo = *médio-baixo*) e (laranja-claro = *médio-alto*)	classe = C4
5.	Se (laranja-claro = *médio-baixo*) e (laranja-escuro = *alto*)	classe = C5
6.	Caso contrário	classe = REJ

Os testes da abordagem deram-se em duas linhas principais, de forma a analisar: *i)* o poder de classificação dos pixels e sua robustez às variaçõe das condições de iluminação; *ii)* o poder de classificação do sistema. Para o teste de *classificação dos pixels*, uma imagem com uma laranja típica de cada classe (C1 a C5 e rejeitada) foi oferecida à RNA e a um especialista. A classificação do especialista para cada pixel foi utilizada para determinar a taxa de acerto na classificação feita pela rede, definida como: $T = (P_C/P_T) \cdot 100$, onde P_C é o número de pixels classificados pela rede na classe considerada e P_T é o total de pixels classificados pelo especialista humano na classe considerada. A tabela abaixo apresenta os dados obtidos. O treinamento da rede foi realizado utilizando como exemplos pixels amostrados de imagens do domínio a aproximadamente 800 Lux.

Classificação do MLP (em %)							
	F	**V-E**	**V-C**	**A**	**L-C**	**L-E**	**M**
F	**95,0**	0,9	1,9	1,3	0,0	0,8	0,0
V-E	8,0	**86,0**	6,0	0,0	0,0	0,0	0,0
V-C	2,4	7,8	**89,0**	1,1	0,0	0,0	0,0
A	3,4	0,1	2,6	**92,0**	0,0	2,3	0,0
L-C	0,0	0,0	0,2	3,3	**92,0**	4,0	0,4
L-E	2,0	0,6	0,9	4,1	0,5	**92,0**	0,0
M	0,0	0,0	4,1	1,4	14,0	0,0	**81,0**

Sem realizar qualquer alteração no treinamento da rede, pixels da imagem de uma laranja C1 típica sob condições de iluminação diversas foram submetidos à classificação da rede, que apresentou resultados consistentes para uma faixa de 550 a 1400lux. Um exemplo da classificação de pixels realizada pelo sistema pode ser visto na Figura 10.19[7].

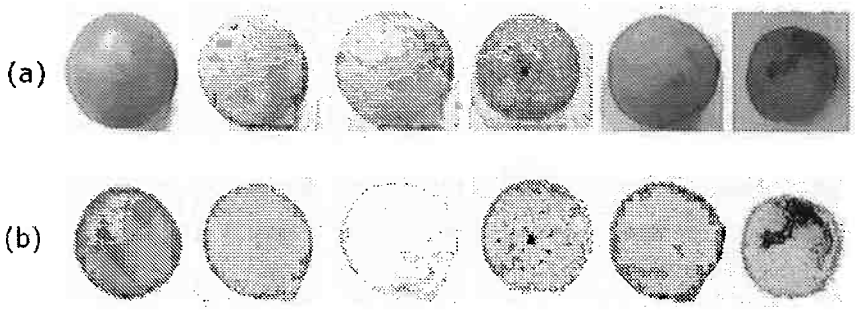


Figura 10.19: a) Classes C1 a C5 e rejeitada; b) Resultado do MLP.

Para o teste do *poder de classificação do sistema*, um conjunto de 120 laranjas igualmente distribuídas em todas as classes válidas e rejeitadas de laranjas foram apresentadas ao sistema e a um especialista humano. Novamente, tomando como corretas as classificações realizadas pelo especialista, observou-se uma taxa acerto de 100% na classificação das laranjas, mostrando que o sistema nebuloso é robusto aos pequenos erros apresentados pela classificação das cores dos pixels, feita pelo MLP. Os resultados mostram que a abordagem feita para o problema é bastante apropriada, reservando grande potencial para aplicação na inspeção automática de laranjas e frutas em geral.

10.5.5 Aplicações de Classificação de Padrões de Cores por Agrupamento Nebuloso

Os procedimentos descritos na Seção 10.1.1 foram empregados em imagens coloridas de domínios diferentes: cenas de ambiente, fotografia aérea, imagens de satélite e cenas específicas de futebol de robôs (BONVENTI Jr.; COSTA, 2000b; BONVENTI Jr.; COSTA, 2000a; BONVENTI Jr.; COSTA, 2002).

Para cada imagem, foi aplicado o algoritmo FCM de classificação de cores por agrupamento nebuloso e obtido o melhor número de grupos, através do índice

[7]Veja esta figura colorida no Apêndice II.

$S = Comp/Sep$ (veja Seção 10.1.1). O número de classes foi variado de 2 a 9, para todos os exemplos. Alguns resultados do agrupamento são mostrados em seguida, com as imagens correspondentes a cada grupo e ainda projeções da distribuição de cores no espaço RGB com os centros dos grupos selecionados. Na seqüência, projeções dos elipsóides que caracterizam os respectivos grupos são mostrados.

Em todas as imagens foi escolhido o fator de nebulosidade $m = 1,3$ arbitrariamente. Com isto, espera-se que apenas os pontos próximos às regiões de separação de cores (em cada imagem particular) apresentem pertinência em mais de uma região.

Futebol de robôs

A Figura 10.13[8] mostra uma cena ambiente do campo de futebol de robôs. O melhor resultado de particionamento alcançado foi para seis classes, conforme o índice S.

A Figura 10.20 mostra as coordenadas das cores da imagem da Figura 10.13 no espaço RGB, projetados nos planos RG (Figura 10.20 (a), esquerda), BG (Figura 10.20 (b), esquerda) e BR (Figura 10.20 (c), esquerda). Os pequenos rótulos numerados são projeções dos centros dos grupos obtidos (seis em cada plano). Ao lado direito de cada uma delas, os elipsóides projetados nos respectivos planos da esquerda. Vale observar que os semi-eixos dos elipsóides não necessariamente estão paralelos aos planos onde são projetados.

A Figura 10.21 mostra as imagens obtidas pela separação dos pontos nas seis classes de cores, segundo o melhor particionamento dado pelo menor valor de S. Cores diferentes de branco indicam pertinência à classe em questão.

Conclui-se, no caso do campo de futebol, que a separação foi bem nítida, de onde pode-se destacar as classes gol, gramado, bola, paredes, laterais do campo e partes escuras. No caso da bola, aparece também o contorno do gol e as balizas coloridas, que também deveriam ser classificadas com o gol, mas provavelmente sua menor luminosidade não permitiu esta caracterização. Observa-se também que os pares de elipsóides que se superpõem em uma projeção não se sobrepõem em outra, demonstrando que a grande maioria dos pontos foi bem separada no processo de agrupamento.

[8] Veja esta figura colorida no Apêndice II.

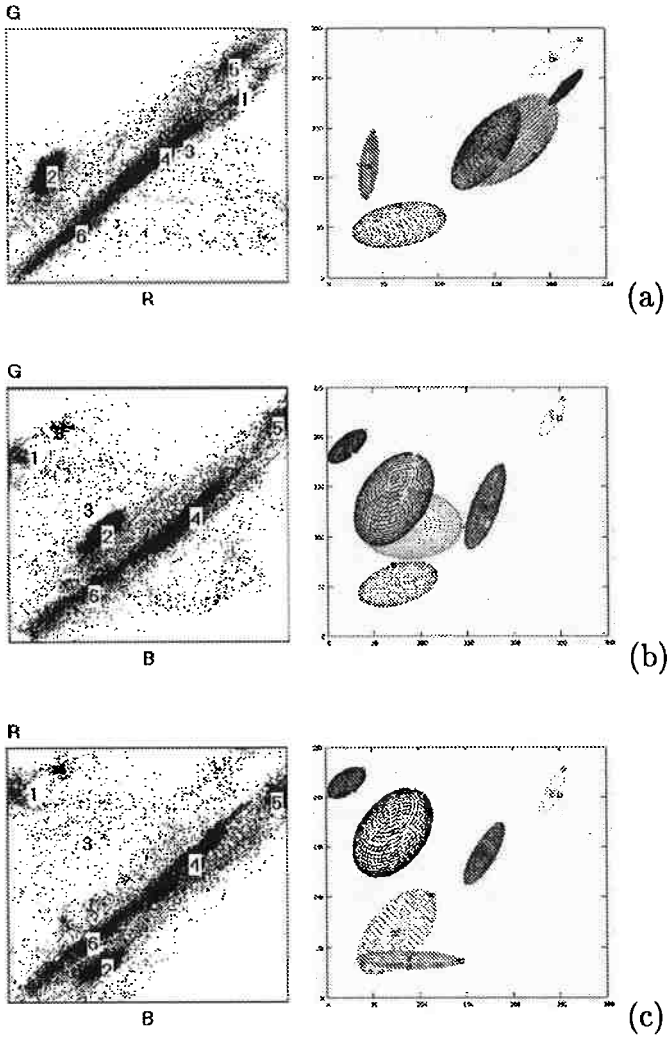


Figura 10.20: Nuvens para o campo de futebol.

Imagem de satélite da cidade do Rio de Janeiro

Esta é uma imagem do satélite Spot10, de baixa resolução, nas bandas 432 (cores naturais). Pode-se notar a lagoa Rodrigo de Freitas e o Oceano Atlântico rodeados por vegetação, como a floresta da Tijuca e morros. As áreas mais claras correspondem a praias e edificações – nota-se inclusive o autódromo de Jacarepaguá na Figura 10.22[9]. O algoritmo FCM usado separou três classes de cores, identificadas como: água, areia+edifícios e vegetação (Figura 10.23).

A imagem do Rio possui três cores bem distintas, aproximadamente: azul, verde e branco. Nota-se no espaço RGB que os centros ficaram a uma boa distância entre si, e as variâncias expressas pelos elipsóides estão relativamen-

[9] Veja esta figura colorida no Apêndice II.

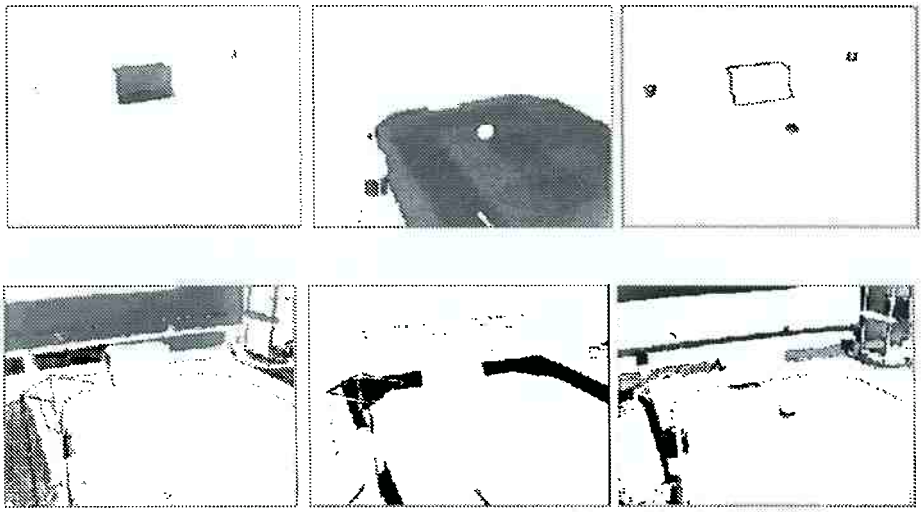


Figura 10.21: Resultados para o campo de futebol.

Figura 10.22: Imagem de satélite do Rio de Janeiro.

te pequenas, mostrando boa separabilidade entre as classes.

Cena ambiente de pimentões

Uma variedade de pimentões, com boa iluminação, pode ser vista na Figura 10.24[10]. Devido às suas superfícies lisas, aparecem intensos reflexos, onde a cor fica descaracterizada (saturação no brilho). Neste caso, foram obtidas quatro classes: dois matizes (vermelho e verde-amarelado) e duas cores com intensidades extremas (branco intenso e sombras). O resultado da classificação pode ser observado na Figura 10.25 , que foram separadas em: pimentões verde-amarelados, reflexos de luz, pimentões vermelhos e sombras e partes escuras.

Os resultados obtidos com o agrupamento nebuloso FCM, com o emprego da distância de Mahalanobis revelaram que seu uso mostrou-se bastante apropriado para a segmentação de imagens levando em conta o atributo da cor, no

[10]Veja esta figura colorida no Apêndice II.

Em ordem, da esquerda para a direita, areia ou edificacoes, água e matas.

Figura 10.23: Resultados para a imagem do Rio de Janeiro.

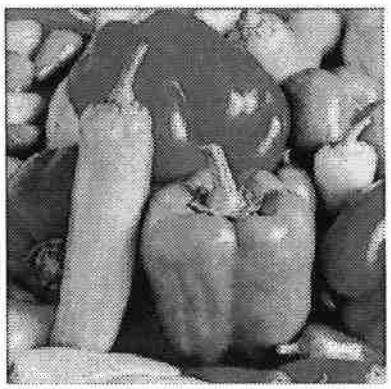


Figura 10.24: Cesta com pimentões verdes e vermelhos. Os vermelhos aparecem em tom cinza mais escuro.

espaço de representação RGB. Outros espaços de representação de cores podem ser empregados. Atualmente, o espaço YUV tem sido investigado pelo grupo da autora. Ainda, o comportamento do algoritmo em função da variação do fator de nebulosidade, comparado com os resultados esperados de classificação para cada imagem em particular, é um alvo de investigação para determinar se há alguma dependência do domínio. Em geral, variações suaves na cor dificilmente estão associadas a objetos diferentes na cena, a menos em casos de animais que

Figura 10.25: Resultado da classificação FCM-GK por padrões de cores dos pimentões.

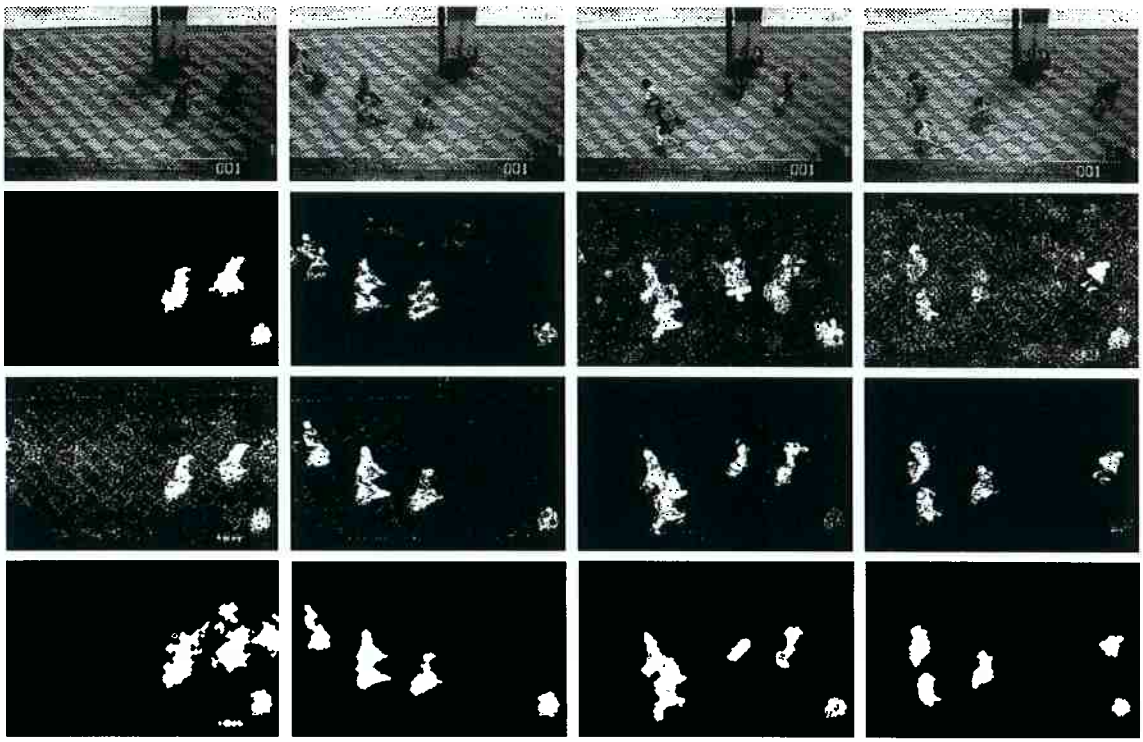
se camuflam na paisagem, por exemplo.

10.5.6 Monitoramento de Terminais Rodoviários

O algoritmo de estimação de um modelo robusto de cena de fundo que realiza, em tempo real, a segmentação de objetos de interesse em ambientes externos dinâmicos, sujeitos a diferentes condições de iluminação, proposto na Seção 10.2.2, foi utilizado no monitoramento de terminais rodoviários baseado em visão computacional (SEIXAS; COSTA, 2002). Neste sistema, busca-se estimar a área ocupada nas diferentes plataformas do terminal, com o objetivo de adequar em tempo real os fluxos dos ônibus às demandas específicas. As imagens de entrada do sistema são obtidas a partir de câmeras fixas perpendiculares ao solo, o que proporciona vistas superiores da área total. O sistema deve ainda funcionar 24 horas por dia para pessoas em movimento e estáticas na plataforma, mostrando a necessidade de um modelo de fundo robusto a variações de iluminação e a objetos de interesse estáticos ou em alto tráfego.

A Figura 10.26 apresenta os resultados da segmentação de pessoas numa plataforma rodoviária, onde o sistema opera. Pode-se perceber a complexidade da cena de fundo, que possui piso heterogêneo e está sujeita a variações de iluminação causadas pelas diferentes condições meteorológicas. A primeira linha da figura contém as imagens da plataforma a serem analisadas em diferentes instantes. A segunda linha mostra o resultado do algoritmo de subtração tradicional, onde o modelo (fixo) de cena de fundo foi obtido no início do processamento, pela aquisição da cena sem objetos de interesse. Já na terceira linha, estão os resultados do processamento pontual do algoritmo proposto, com modelo adaptável (veja Seção 10.2.2). Por fim, na quarta linha, estão os resultados do algoritmo completo, com as camadas de processamento local e global.

Pode-se perceber na Figura 10.26 que, nos instantes iniciais de processamento do sistema, os resultados do algoritmo de subtração tradicional são melhores que os resultados do algoritmo proposto. Entretanto, isto já era esperado, uma vez que o nível de iluminação do modelo de cena de fundo da subtração tradicional ainda era bem próximo ao nível daquele instante. Por outro lado, o algoritmo proposto requer alguns segundos para que as distribuições gaussianas do modelo de fundo tenham amostras sufientes para realizar a segmentação, eliminando-se



Imagens da plataforma rodoviária (primeira linha) e resultados dos algoritmos: subtração com modelo fixo (segunda linha), algoritmo proposto com processamento pontual (terceira linha) e algoritmo proposto completo, com processamento pontual, local e global (quarta linha). A primeira coluna mostra os resultados da seqüência de teste após 20 segundos do início do tempo de processamento; a segunda, após 2 minutos e 10 segundos do início; a terceira, após 4 minutos e 35 segundos; e a quarta, após 9 minutos e 22 segundos. Os parâmetros utilizados nesta seqüência de testes foram: taxa de aquisição das imagens de 5 *quadros/s*, $K = 3$, $\alpha = 0.001$, $\beta = 2.5$, $Y_{min} = 16$, $T = 0.62$, $\lambda = 0.11$ e $H = 6$.

Figura 10.26: Imagens da plataforma rodoviária (primeira linha) e resultados dos algoritmos propostos.

assim a necessidade de cenas vazias para estimar o modelo de cena de fundo.

A grande vantagem do algoritmo proposto, porém, começa a ficar bem clara alguns minutos após o início do processamento. Como numa aplicação deste tipo não é possível obter freqüentemente imagens sem objetos de interesse (pessoas), o modelo de cena de fundo da subtração tradicional, com o tempo, passa a ficar inconsistente, causando muitos erros de segmentação. O algoritmo proposto, no entanto, tende a gerar um modelo de cena de fundo mais robusto com o tempo, já que as distribuições são adaptativas, como pode ser visto nas colunas seguintes da Figura 10.26.

É interessante notar, também, as diferenças entre o algoritmo proposto com e sem as camadas de processamento local e global. O processamento pontual gera alguns erros de classificação, originados por sombras, reflexões ou mesmo objetos de interesse estáticos, e muitas vezes são corrigidos pelos processamentos local e global. Um exemplo claro da necessidade de outras camadas de processamento está nas últimas colunas da Figura 10.26. No canto inferior direito das imagens, um passageiro permanece estático, sentado num banco, por alguns minutos. No algoritmo de processamento pontual, este passageiro gradualmente foi incorporado ao modelo de cena de fundo, o que não aconteceu no funcionamento do algoritmo completo, já que as relimentações positivas nos pixels que descreviam a segmentação deste passageiro não permitiram sua inclusão no modelo de cena de fundo.

10.5.7 Monitoramento de Plataformas Metroviárias

Este item descreve um sistema de monitoramento inteligente de plataformas metroviárias, em tempo real, que apresenta as seguintes funcionalidades: detecção de situações de risco para o usuário ou impedimento de circulação dos trens; distinção entre pessoas e objetos na área de risco (próximo aos trilhos); emissão de alarmes correspondentes aos riscos detectados (níveis de alarmes); direcionamento da atenção do operador ao risco.

Situações consideradas de risco para os usuários e bloqueio da circulação de trens são: pessoas que acidentalmente caiam na via; pessoas tentando invadir a área de circulação de trens; objetos que venham a cair na via.

Para estudos futuros, pretende-se acrescentar a detecção de excesso de usuários na plataforma (para efetuar bloqueio de catracas) e de tumulto ou agitação excessiva na plataforma.

O programa desenvolvido (MAIA Jr.; COSTA, 2002) utiliza os algoritmos de difusão não linear e de Camus para extrair o fluxo ótico e o campo de movimento, respectivamente, de uma seqüência de imagens (veja Seção 10.3).

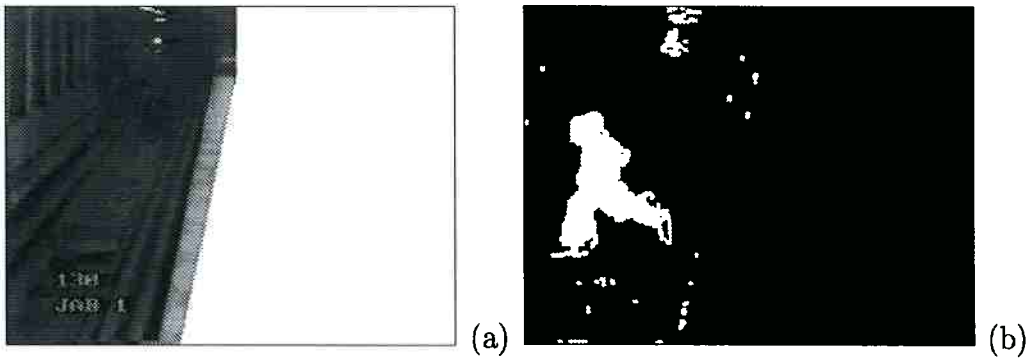
De uma seqüência de entrada são mostrados os seis primeiros quadros (veja Figura 10.27). A taxa de captura do vídeo para essa seqüência é de 5 quadros/segundo, daí pode-se concluir que o tempo entre as capturas é de 200 ms.

Esse dado será útil para o cálculo da velocidade relativa de um objeto na cena.

Imagens obtidas a partir do circuito fechado de TV de estações de metrô da cidade de São Paulo.

Figura 10.27: Seis primeiras imagens da seqüência de entrada utilizada nos testes.

Nos testes utilizou-se uma máscara que define a região de interesse nas imagens, ilustrada na Figura 10.28-esquerda. Para os *needle maps* (veja definição na Seção 10.3) que foram obtidos utilizando os dois algoritmos, o limiar utilizado de subtração dos valores dos pixels entre imagens consecutivas na seqüência de dados (para o cálculo aproximado da diferenciação) foi igual a 20. O resultado obtido de tal subtração é mostrado na Figura 10.28-direita. Vale ressaltar que esta operação somente é realizada na região de interesse na imagem. No resultado desta subtração foram aplicados os operadores morfológicos *open* e *close* (GONZALEZ; WOODS, 1992) para que eliminar pontos isolados e depois aumentar ligeiramente a área de processamento. Existe a opção de uso ou não desses operadores na interface homem-máquina desenvolvida.



(a) (b)

Figura 10.28: (a) Máscara que define a região de interesse na imagem e (b) Exemplo do resultado obtido.

Para cada novo *needle map* que se deseja obter (de um conjunto de dois ou mais quadros), um novo resultado é obtido utilizando-se a operação de subtração.

A Figura 10.29 mostra a aparência da interface atual do sistema. A interface

permite escolher o algoritmo, as figuras que serão utilizadas como entrada e as regiões de interesse. Podem ainda ser configurados os parâmetros da execução de cada algoritmo, o limiar do gradiente e o limiar da subtração. O usuário pode ainda escolher quais figuras serão salvas entre *needle maps*, resultados intermediário e gradientes. Existem ainda duas outras janelas para auxiliar a visualizar dados durante a execução. Uma delas mostra a ocupação do processador e a outra, dados como o tempo de execução de um comando (veja Figura 10.30).

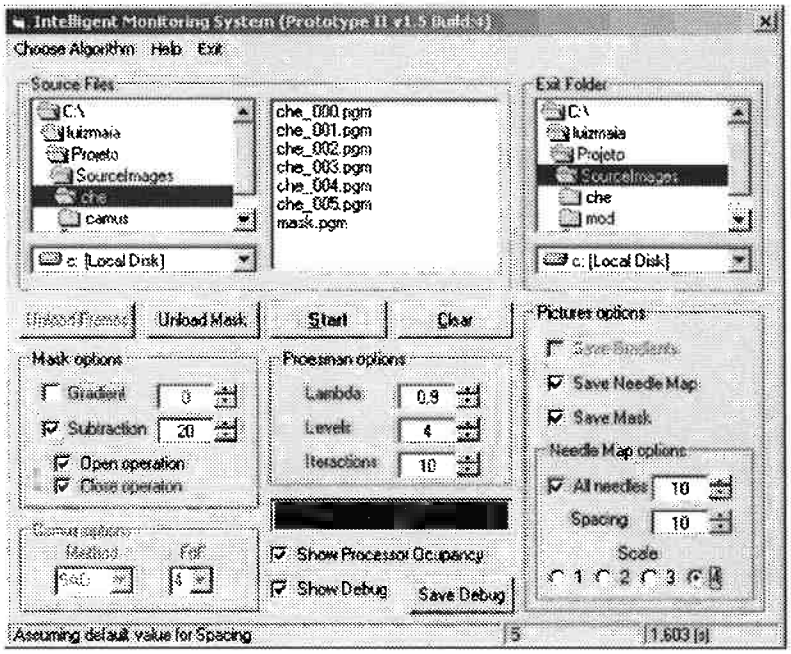
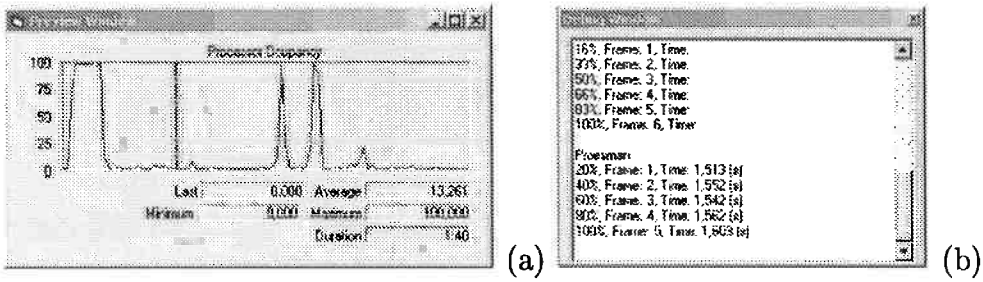


Figura 10.29: Interface homem-máquina desenvolvida.

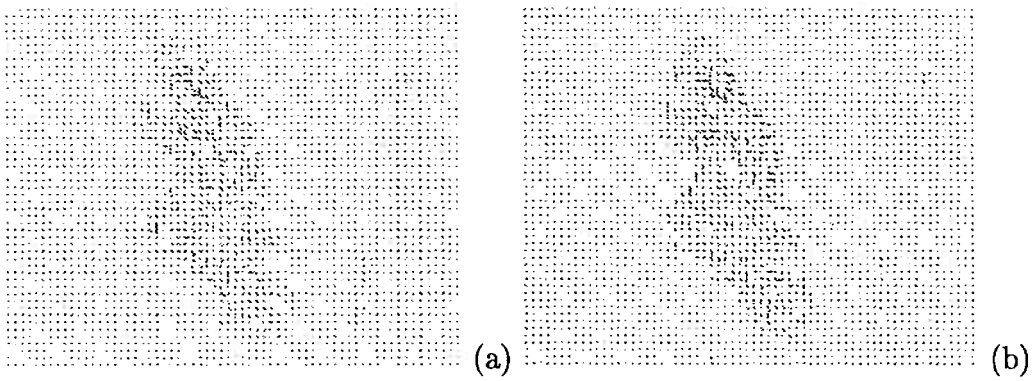


(a) (b)

Figura 10.30: (a) Janela que informa a ocupação do processador e (b) Janela que informa as operações sendo executadas e o tempo que demoram.

O algoritmo de cálculo do fluxo ótico por difusão não linear utiliza quatro parâmetros de entrada. Destes quatro, três podem ser variados livremente (λ, Níveis e Iterações) e um deles (Estimação) é uma variável booleana. As faixas de valores para cada parâmetro foram obtidas empiricamente, definindo aquelas

que definiram os melhores resultados considerando-se o desempenho de quadros processados por segundo: λ, de $0,7$ a $0,9$; Níveis, de 2 a 4; Iterações, de 4 a 16; Estimação, sempre 0. A Figura 10.31 apresenta dois exemplos de fluxo ótico extraídos da seqüência de quadros mostrada na Figura 10.27, usando o algoritmo de fluxo ótico por difusão não linear.



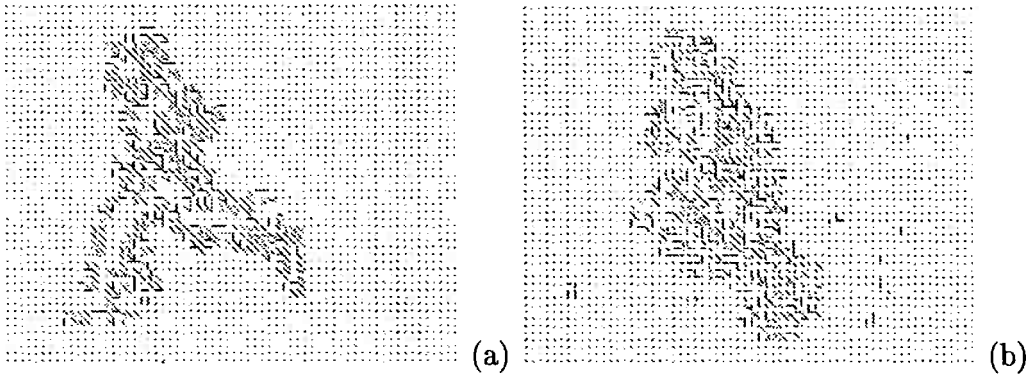
$\lambda = 0.9$, Níveis = 2 e Limiar de subtração = 20.

Figura 10.31: Resultados do algoritmo de fluxo ótico por difusão não linear com (a) Iterações = 16 e (b) Iterações = 8.

O algoritmo de Camus utiliza dois parâmetros de entrada, o método para obtenção do campo de movimento e o número de quadros processados em conjunto. Os melhores resultados foram obtidos utilizando-se o método NCC para dois ou três quadros processados em conjunto. Esse método foi escolhido porque apresentou melhores resultados com texturas e porque é o mais rápido. Os resultados são baseados nesse método. A Figura 10.32 apresenta exemplos de campos de movimento extraídos da seqüência de quadros mostrada na Figura 10.27, usando o algoritmo de Camus.

Em termos de tempo de processamento, o algoritmo de Camus é muito mais rápido que o de difusão não linear, porém o último apresentou resultados com melhor qualidade.

Para que bons resultados sejam obtidos com o algoritmo de cálculo do fluxo ótico por difusão não linear é necessária uma calibragem do sistema, principalmente no que diz respeito ao λ e o número de iterações e, secundariamente, à quantidade de níveis de resolução. Essa calibragem foi feita empiricamente através de várias tentativas e análise dos resultados.



(a) Campo de Movimento utilizando o Método NCC e somente dois quadros de entrada e (b) Campo de Movimento utilizando o Método NCC e três quadros de entrada.

Figura 10.32: Resultados do algoritmo de campo de movimento de Camus.

Em relação ao algoritmo de Camus o que foi observado é que ele apresenta resultados coerentes rapidamente mas que, para refinar esse resultado, ter-se-ia o mesmo custo computacional do algoritmo anterior. Mesmo o método NCC sendo bom para imagens com textura, o que foi observado é que mais de três quadros devem ser utilizados em conjunto por processamento, para que os vetores do campo apresentem resultados mais precisos.

Os esforços atuais residem em reunir os dois algoritmos testados num único, onde a primeira estimação de movimento seria feita pelo algoritmo de Camus e, daí por diante, seria aplicado o algoritmo por difusão não linear. O algoritmo de Camus é eficiente para determinar a direção inicial e apresenta o campo de movimentação com conjuntos de vetores que fornecem uma indicação razoável da direção do movimento de uma região, já o outro algoritmo determina velocidades e direções dos vetores de forma mais precisa.

10.5.8 Medição de Painéis

As técnicas de estereoscopia descritas na Seção 10.4 também estão sendo utilizadas para a medição de painéis (*OutDoors*) a partir de grandes distâncias. A Figura 10.33 mostra um painel típico de interesse.

Nesta aplicação deseja-se, a partir de duas imagens capturadas por um arranjo estéreo, realizar a medição das dimensões do painel. A identificação dos limites

Figura 10.33: Painel típico de interesse para a aplicação.

dos painéis não é automática, ficando o operador do sistema responsável por indicar, através de comandos do mouse, os vértices do painel nas duas imagens. De acordo com o que foi discutido na Seção 10.4.2, as regiões em torno dos pontos definidos pelo operador na primeira imagem serão os modelos, e os pontos definidos na segunda imagem serão os pontos iniciais para a busca iterativa de adequação do modelo. Como os painéis podem e geralmente têm grandes dimensões, tipicamente em torno de $5m$x$10m$, é necessário posicionar o arranjo estéreo a uma grande distância destes para que seja possível enquadrá-los nas duas câmeras. Como foi dito anteriormente, a validade do modelo geométrico da câmera obtido com a calibração se restringe à região do espaço onde foram colocados os gabaritos de calibração. Isto consiste num problema grave para esta aplicação, pois para obter uma reconstrução 3D precisa, tudo indica ser necessário realizar a calibração utilizando um gabarito de calibração das mesmas proporções que o maior painel que se deseje medir. Diante deste problema, iniciou-se um estudo da validade e da precisão do modelo geométrico obtido com a calibração em função dos seguintes parâmetros: *(i)* quantidade de imagens do gabarito utilizadas na calibração; *(i)* faixas de distância onde foram colocados os gabaritos durante a calibração;

Este estudo tem por motivação responder algumas perguntas – cujas respostas

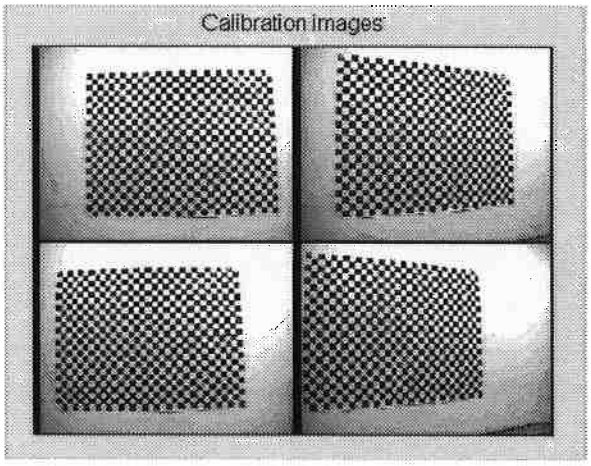
não se pôde obter na literatura – que surgem quando se deseja, como é o caso desta aplicação, calibrar um arranjo estéreo para realizar a reconstrução tridimensional em regiões do espaço que não podem ser incluídas na calibração:

1. Dado que se deseja fazer a reconstrução tridimensional em uma determinada região do espaço, com uma determinada precisão, qual região do espaço deve ser incluída na calibração?
2. Qual a quantidade de imagens que deve ser capturada nesta região?
3. É possível obter um modelo preciso válido para uma grande faixa de distâncias ou o aumento da validade do modelo implica na diminuição da precisão de um modo geral?
4. Qual o impacto da distância entre as câmeras do arranjo estéreo (conhecida como *baseline*) na precisão da reconstrução?

Para tentar responder estas perguntas, conduziu-se um experimento onde foram capturados 95 pares de imagens dos gabaritos de calibração, numa região do espaço de 1m até 10m de distância do arranjo estéreo. A cada 0.5 metro foram capturadas 5 imagens a partir de diferentes orientações em relação ao gabarito: frontal, lateral esquerda, lateral direita, superior e inferior. A Figura 10.34 abaixo mostra as vistas frontal e lateral esquerda capturadas pelas duas câmeras à distância de 5 metros do arranjo.

Três calibrações do arranjo estéreo foram computadas: a primeira, utilizando os pares de imagens capturados a partir das distâncias 1, 2, 3, 4 e 5 metros, a segunda a partir das distâncias 1, 2, 3, 4, 5, 6 e 7 metros e a terceira a partir das distâncias 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 metros. A partir dos três modelos obtidos nestas calibrações, foi feita a reconstrução utilizando todos os pares de imagens capturados. A Figura 10.35 mostra a precisão da reconstrução utilizando-se o primeiro modelo. Os pontos correspondem ao erro (média e desvio padrão) das medidas das dimensões (vertical e horizontal) dos painéis em função da distância a partir da qual o par de imagens do painel foi capturado.

Pode-se observar que a reconstrução é bastante precisa (erro inferior a 5%) na região incluída na calibração (de 1 a 5 metros), e que o erro aumenta gradualmente com a distância, chegando à casa dos 10% (o maior erro obtido foi de 13%) para as



As imagens superiores foram capturadas pela câmera esquerda, e as inferiores pela direita.

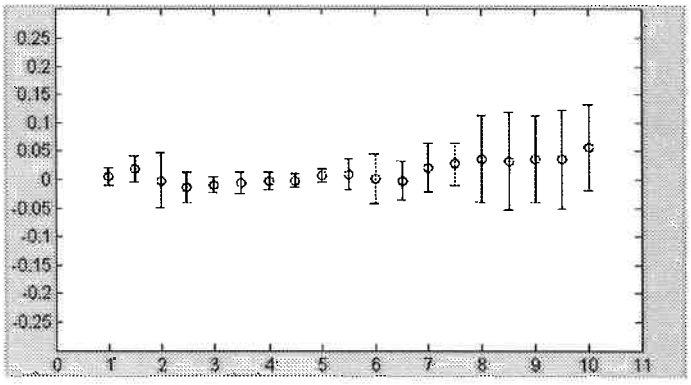
Figura 10.34: Vistas frontal e lateral esquerda do gabarito de calibração à distância de 5 metros.

maiores distâncias. A Figura 10.36 mostra a precisão da reconstrução utilizando-se o segundo modelo.

Pode-se observar que o erro da reconstrução nas maiores distâncias diminuiu (o maior erro foi de 8 %), mas o erro nas menores distâncias aumentou ligeiramente. A Figura 10.37 mostra a precisão da reconstrução utilizando-se o terceiro modelo.

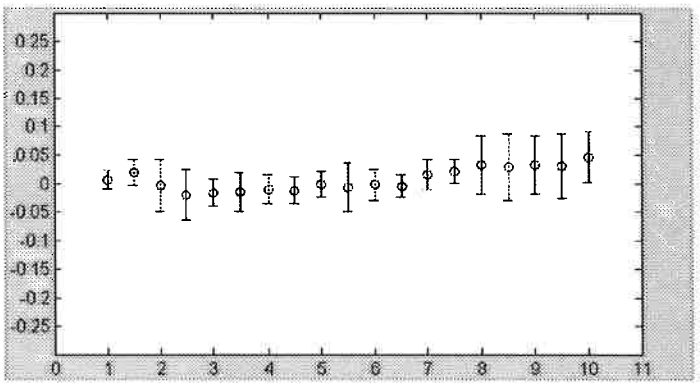
Pode-se observar que a precisão da reconstrução para as maiores distâncias aumentou significativamente (o maior erro obtido foi de 5%), mas a precisão do modelo diminuiu de um modo geral. O primeiro gráfico indica que se a aplicação permitir uma certa margem de erro (no caso 10%), é possível extrapolar um modelo obtido na faixa de 1 a 5 metros para uma faixa de 1 a 10 metros. O segundo e o terceiro gráfico indicam que realmente há um compromisso entre a validade do modelo e a sua precisão, uma vez que, ao se incluir maiores distâncias na calibração, a precisão do modelo diminui como um todo.

Os resultados obtidos até o momento respondem a primeira e a terceira questão, isto é, é possível extrapolar um modelo obtido para uma determinada faixa de distâncias se a aplicação permitir uma determinada margem de erro, e não é possível obter um modelo preciso e válido para uma grande faixa de distâncias, uma vez que a precisão do modelo é inversamente proporcional à sua



As barras de erro têm comprimento de 2 vezes o desvio padrão.

Figura 10.35: Erro e desvio padrão da reconstrução dos pontos do gabarito, com modelo obtido da calibração a partir das distâncias 1, 2, 3, 4 e 5 metros.

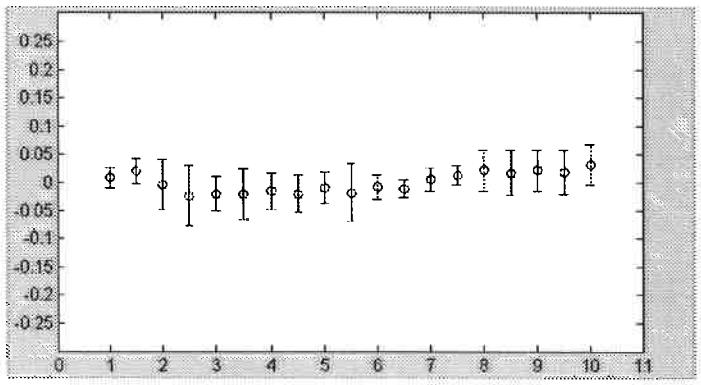


As barras de erro têm comprimento de 2 vezes o desvio padrão.

Figura 10.36: Erro e desvio padrão da reconstrução dos pontos do gabarito, com modelo obtido da calibração a partir das distâncias 1, 2, 3, 4, 5, 6 e 7 metros.

validade. Testes estão em andamento para determinar o impacto da quantidade de imagens utilizada na qualidade do modelo obtido, e um novo experimento está sendo conduzido com um novo arranjo estéreo, onde será utilizada uma distância de *baseline* 50% maior que a utilizada no experimento anterior.

Como para a aplicação de medição de painéis admite-se uma margem de erro de até 10% em cada eixo, e a faixa de distâncias onde será realizada a reconstrução não ultrapassará os 10 metros, tudo indica que a calibração até 5 metros será suficiente. No entanto, se houver a necessidade de uma maior extrapolação (na faixa de 15 ou 20 metros), ou se houver o desejo de simplificar a calibração (a calibração de 1 a 5 metros exige um gabarito de $1,5m$x$2,0m$), o método de



As barras de erro têm comprimento de 2 vezes o desvio padrão.

Figura 10.37: Erro e desvio padrão da reconstrução dos pontos do gabarito, com modelo obtido da calibração a partir das distâncias 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 metros.

calibração (ZHANG; FAUGERAS; DERICHE, 1997) para arranjos estéreo – que estima a geometria epipolar do arranjo antes da calibração propriamente dita – poderá ser considerado. Como foi dito, este método estima a geometria epipolar do arranjo estéreo com base num conjunto de pontos correspondentes nas duas imagens e realiza a calibração de modo a obter um modelo compatível com os dados de calibração e com a geometria epipolar estimada anteriormente. Isto permitiria, a princípio, refazer a calibração do arranjo estéreo com um conjunto básico de imagens do gabarito de calibração e um conjunto de pontos correspondentes do par de imagens do painel que se deseja medir num determinado instante. A geometria epipolar do arranjo seria estimada apenas com base nos pontos correspondentes do par de imagens do painel, e a calibração geraria então um modelo compatível com esta geometria, modelo que teoricamente seria válido para a região onde se encontra o painel naquele determinado instante.

10.6 Discussões

Neste Capítulo diversas técnicas de processamento de imagens foram descritas, abordando importantes tópicos da visão computacional, como o uso da cor e do movimento para segmentar imagens em partes correspondentes a objetos de interesse na cena, determinação e modelagem de cenas de fundo e estereoscopia binocular para determinar a distância do objeto ao observador. Várias aplicações

foram descritas, em Robótica e em outras áreas.

Particularmente na área da Robótica Móvel Inteligente, visão é uma habilidade sensorial extremamente poderosa. A autora acredita ser essencial a exploração de novas formas de organizar a visão e integrá-la de modo efetivo aos robôs móveis, para que estes possam interagir com o ambiente em tempo real.

11 Arquiteturas Desenvolvidas

A definição de arquiteturas é muito importante, uma vez que elas estabelecem não só os módulos constituintes do projeto de um robô ou sistema, mas também suas interações (veja Capítulo 5). As arquiteturas reativas baseadas em agentes ou comportamentos são bastante apropriadas para tarefas executadas em tempo real, tarefas estas tipicamente envolvidas na navegação de robôs móveis.

A autora e seu grupo têm investigado este tipo de arquitetura no robô móvel Pioneer, permitindo um maior entendimento e aperfeiçoamento da abordagem, além de conseguir resultados bastante motivadores. A Seção 11.1 apresenta a arquitetura REACT, uma arquitetura reativa com coordenação cooperativa de comportamentos, e a Seção 11.2 descreve uma arquitetura reativa com coordenação competitiva dos comportamentos.

No entanto, para um desempenho global eficiente em tarefas mais complexas, que envolvam deliberação, as arquiteturas puramente reativas não apresentam resultados satisfatórios. Neste caso, as arquiteturas híbridas, reativas e deliberativas, tornam-se necessárias.

A autora e seu grupo propuseram uma nova arquitetura, baseada em Sistemas Multi-Agentes e técnicas de aprendizado autônomo como solução para problemas robóticos que envolvam reatividade e deliberação. Estas propostas são discutidas nas Seções 11.3 e 11.4 deste Capítulo.

11.1 Arquitetura Reativa com Coordenação Cooperativa

A contribuição deste trabalho consiste na definição de modelos e parâmetros adequados ao robô Pioneer (modelo básico), cuja capacidade sensorial é limitada

(uma vez que há sonares apenas na parte frontal e os odômetros são imprecisos), de tal forma a implementar uma arquitetura reativa baseada em comportamentos modelados por *Motor-Schemas*, utilizando o método de Campos Potenciais (ARKIN, 1999). Estes modelos foram inseridos na arquitetura REACT, implementada no robô Pioneer (PACHECO; COSTA, 2002a; PACHECO; COSTA, 2002b).

Um *Motor-Schema* permite, essencialmente, a definição e implementação de um comportamento em dois módulos principais: o módulo de percepção, que é responsável por extrair dos estímulos sensoriais as informações relevantes para o comportamento em questão; e o módulo de codificação do comportamento que, alimentado pelo módulo de percepção, executa o mapeamento dos estímulos sensoriais nas respostas motoras. Os parâmetros que definem uma ação – a qual corresponde à saída de cada *Motor-Schema* – são a magnitude e a direção do movimento. Estes parâmetros são representados por um vetor, e correspondem, respectivamente, à velocidade e à rotação que o robô deve executar, segundo o comportamento em questão.

A coordenação dos vários comportamentos é feita de forma cooperativa, onde a ação resultante que o robô deve executar deriva das contribuições – em maior ou menor grau – advindas de todos os comportamentos ativos. O Método de Campos Potenciais sugere uma maneira muito simples para a coordenação dos comportamentos: a superposição dos campos gerados pelos mesmos. Na superposição, as contribuições (vetores) de cada comportamento são somadas, de maneira que o problema da coordenação se resume a dimensionar corretamente os campos dos diferentes comportamentos de forma que a sua soma resulte no comportamento global desejado para o robô.

11.1.1 Descrição dos Comportamentos

Três comportamentos compõem a arquitetura REACT atual do robô Pioneer: *avoidCollision*, *moveAhead* e *moveToGoal*.

Comportamento *avoidCollision*

Este comportamento tem por objetivo evitar colisões em relação a múltiplos obstáculos. O módulo de percepção deste comportamento interpreta as leitu-

ras dos sonares de modo a localizar os obstáculos presentes no ambiente. Esta interpretação adota a restrição de continuidade espacial do ambiente, no intuito de evitar que duas leituras advindas de um mesmo obstáculo sejam interpretadas como dois obstáculos diferentes. Tal interpretação faria com que o robô reagisse de modo muito intenso ao obstáculo, dificultando a navegação por caminhos estreitos.

A continuidade espacial é determinada por dois fatores: a vizinhança dos sonares e a diferença entre os valores absolutos das leituras. Deste modo, duas leituras correspondem a um espaço contínuo – e, portanto, representam um mesmo obstáculo – se tiverem sido obtidas por sonares vizinhos e se a diferença entre os valores absolutos das leituras (que correspondem às distâncias medidas) não ultrapassar um limiar pré-estabelecido. Este limiar consiste em um parâmetro do comportamento e pode ser ajustado conforme o tipo de ambiente em que o robô deve navegar. No caso da detecção de um obstáculo por dois ou mais sonares, a posição deste obstáculo é definida como sendo a leitura que indicar maior proximidade ao robô.

O módulo de codificação deste comportamento, alimentado pelo módulo de percepção, determina qual a ação motora adequada àquela percepção, ou seja, constrói o campo potencial do comportamento, associando cargas repulsivas aos obstáculos.

Como se trata de um campo potencial, este módulo deve fornecer a direção e a magnitude do campo no ponto em que se encontra o robô. A direção é definida como radial. Deste modo, na presença de um obstáculo, o robô sofre uma força repulsiva na direção da reta que une o seu centro de massa ao obstáculo detectado. O resultado final do comportamento é a soma vetorial de cada uma das forças devidas a cada obstáculo detectado. A magnitude da força é determinada por uma função de decaimento que depende apenas da distância do centro de massa do robô ao obstáculo em questão. Deste modo, o módulo de codificação deste comportamento é composto pelas seguintes equações:

$$V(d) = e^{\frac{(-d+S)}{T}} \qquad \phi = -\phi_{robô-obstáculo} \tag{11.1}$$

onde V é a magnitude do comportamento, d é a distância do centro de massa do robô ao obstáculo, S é o *"stand-off"* do robô – distância abaixo da qual a

magnitude do comportamento satura em seu valor máximo, normalizado –, T é a constante de tempo da curva de decaimento, ϕ é a direção do vetor resultante do comportamento e $\phi_{robô-obstáculo}$ é a direção definida pela reta que une o obstáculo ao centro de massa do robô. O uso da exponencial se justifica por esta função apresentar um decaimento controlado (parâmetro T) e por ter a característica de crescer rapidamente em um ponto também controlado (parâmetro S), características bastante adequadas a um comportamento de evitar colisões.

Comportamento *moveAhead*

Este comportamento é responsável por fazer com que o robô sempre tenha uma forte tendência de continuar na direção atual, ou seja, que ele tenha um certo tipo de inércia. Apesar de aparentemente ingênuo, este comportamento tem grande importância no sistema, pois ele compensa a oscilação que pode resultar do comportamento *avoidCollision* e suaviza significativamente a trajetória do robô, uma vez que contribui para a diminuição da magnitude da rotação executada. Este comportamento também demonstrou importância especial quando o robô navega por corredores. Neste caso, dado que os sonares abrangem apenas a parte frontal do robô, verificou-se que, sem o comportamento *moveAhead*, o robô executava um movimento de "zig-zag"entre as paredes. Isto porque, ao perceber uma das paredes laterais, o robô girava para o lado contrário bruscamente, devido à repulsão à parede detectada; no entanto, ao executar este movimento, os sonares agora detectariam a outra parede, causando nova repulsão e gerando a oscilação. Com a introdução do comportamento *moveAhead*, as rotações foram suavizadas, permitindo que, a cada giro para um dos lados, o giro seguinte fosse menor. Desta maneira, a trajetória se estabiliza rapidamente.

O comportamento *moveAhead* não possui módulo de percepção e o seu módulo de codificação é elementar, consistindo em um campo onde a magnitude é constante (parâmetro do comportamento) e a direção corresponde sempre à direção do robô naquele instante. Portanto:

$$V(d) = cte \qquad \phi = \phi_{robô} \tag{11.2}$$

Comportamento *moveToGoal*

Este comportamento tem o objetivo de atrair o robô para um determinado ponto no ambiente. A posição inicial do alvo – no sistema global de coordenadas – é informada ao robô por meio de um agente externo ou detectada por meio de um sistema de visão computacional baseado em estereoscopia binocular (veja Seção 10.5.3). A posição atual do robô (também no sistema global de coordenadas) é estimada pelo módulo de percepção, tomando como base a posição inicial do robô e integrando as informações fornecidas pelos seus odômetros. O módulo de codificação fica encarregado de determinar a direção do movimento, que será a mesma direção dada pela reta que une os pontos que definem as posições do robô e do alvo. A magnitude é constante até uma determinada proximidade do robô ao alvo, e depois se torna uma exponencial decrescente. Este limiar de proximidade é um parâmetro do comportamento. Portanto, quando o robô estiver distante do alvo, vale:

$$V(d) = cte \qquad \phi = \phi_{robô-alvo} \tag{11.3}$$

e quando o robô estiver próximo do alvo:

$$V(d) = e^{\frac{d+S}{T}} \qquad \phi = \phi_{robô-alvo} \tag{11.4}$$

O resultado desta codificação é um campo radial atrativo de magnitude constante até uma determinada distância do robô ao alvo, e em seguida a magnitude decai exponencialmente até o alvo.

Comportamento Global

O comportamento global é dado pela coordenação cooperativa dos comportamentos primitivos, que será a soma vetorial das ações definidas por cada um deles. Os Motor-Schemas dos três comportamentos e a respectiva composição vetorial são ilustrados na Figura 11.1.

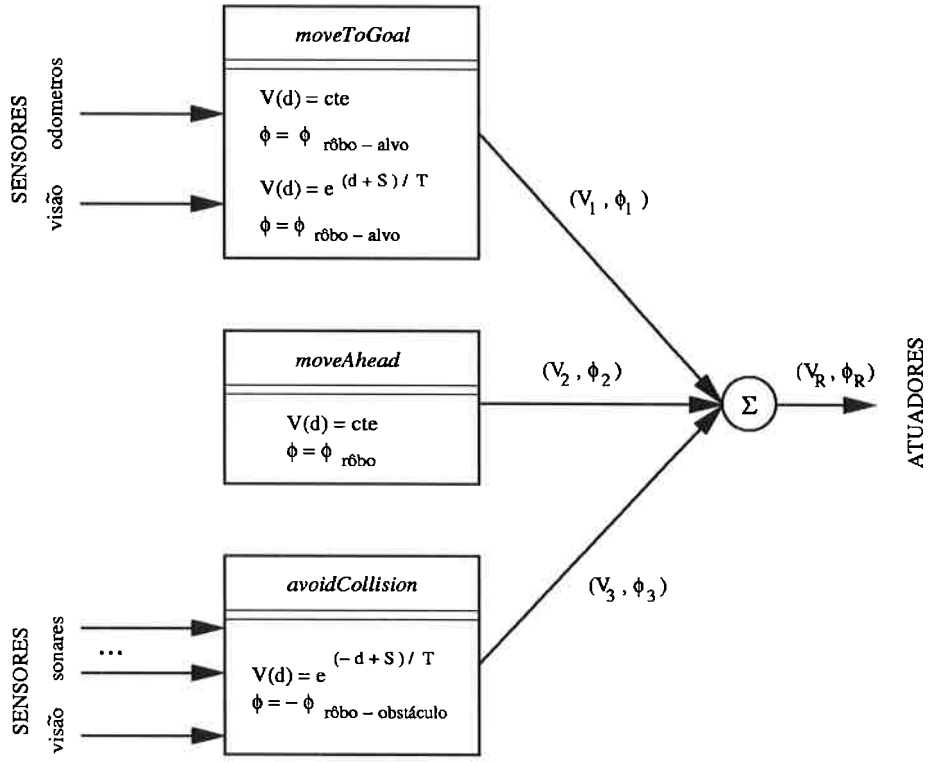


Figura 11.1: *Motor-Schema* dos comportamentos *avoidCollision*, *moveAhead*, *moveToGoal* e a composição vetorial.

11.1.2 Resultados e Discussões

A arquitetura REACT foi testada tanto no simulador do Pioneer quanto no robô real, exibindo em ambos os casos uma atuação bastante eficiente.

A arquitetura reativa proposta foi desenvolvida na linguagem C, através do uso da biblioteca Saphira 6.2 de funções do robô.

Os parâmetros utilizados no experimento foram os seguintes:

- *avoidCollision*: $S = 0.3$, $T = 0.3$, limiar de detecção de obstáculos $d = 50cm$;
- *moveAhead*: magnitude $V = 0.6$;
- *moveToGoal*: magnitude $V = 0.4$;
- velocidade máxima: $150mm/s$.

A escolha dos parâmetros dos comportamentos se mostrou bastante dependente da velocidade máxima permitida para o robô.

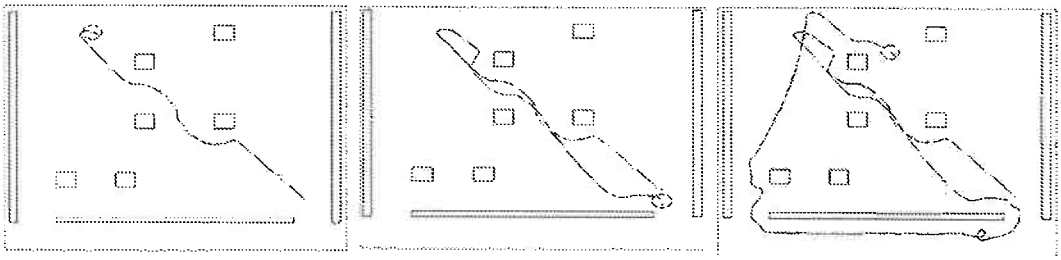
Os valores da magnitude dos comportamentos podem assumir valores entre 0 e 1, onde 1 corresponde à velocidade máxima e 0, à mínima. As magnitudes dos comportamentos *moveAhead* e *moveToGoal* devem ter sua soma igual a 1, pois desta maneira é possível garantir que, no caso em que o robô se dirige para o alvo e se depara frontalmente com um obstáculo (pior caso), a inversão da direção do campo resultante se dará na distância especificada pelo parâmetro S do comportamento *avoidCollision*, uma vez que neste ponto a magnitude deste comportamento será máxima (explosão da exponencial), e a direção será diretamente oposta ao vetor resultante da soma dos comportamentos *moveAhead* e *moveToGoal*.

O experimento no ambiente de simulação consistiu em colocar o robô inicialmente em uma posição próxima à porta no canto inferior direito de uma sala hipotética com obstáculos, com orientação para o interior da sala, e estabelecer a posição alvo no canto superior esquerdo, extremo oposto da sala. Quando o robô atinge a posição alvo, as posições inicial e alvo são invertidas, de modo que o robô volte para a posição inicial do experimento. Ao atingir o novo alvo, as posições inicial e alvo são novamente invertidas, e assim sucessivamente. Deste modo, o robô se movimenta ciclicamente entre os dois extremos da sala.

Na primeira parte do experimento, o robô parte do canto inferior direito da sala e atinge o canto superior esquerdo. Vale notar que o robô desviou com sucesso dos obstáculos, executando uma trajetória bastante eficiente. O ambiente com obstáculos e a trajetória do robô são mostrados na Figura 11.2-esquerda.

Na segunda parte do experimento, o robô parte do canto superior esquerdo da sala e atinge o canto inferior direito. Observa-se, na Figura 11.2-centro, que o caminho percorrido indica sucesso no desvio dos obstáculos. É importante observar que, conforme esperado, as trajetórias de ida e de volta são distintas, uma vez que o sistema é reativo e que, devido à orientação e sensores do robô, as condições de ida e de volta são ligeiramente diferentes.

Um resultado interessante foi obtido na terceira parte do experimento, ilustrado na Figura 11.2-direita. Devido aos erros acumulados por seu sistema de odometria, o robô tem uma percepção incorreta da posição alvo da segunda parte do experimento, localizando-a fora da sala. Ao atingir o ponto que acredita ser a posição alvo, as posições inicial e alvo são invertidas e o robô tenta se dirigir



Resultados: Esquerda – alvo no canto superior esquerdo e robô no canto inferior direito, o robô executa trajetória eficiente, desviando dos obstáculos e atingindo o alvo. Centro – robô volta à posição original, executando trajetória diferente da ida. Direita – devido ao acúmulo de erros por seu sistema de odometria, o robô localiza a posição do alvo (canto inferior direito) fora da sala; devido à atração ao novo alvo (canto superior esquerdo) e repulsão à parede, uma trajetória estável, paralela à parede, é executada até a porta, no canto inferior esquerdo, quando o robô consegue entrar novamente na sala e se dirigir ao alvo.

Figura 11.2: Resultados da arquitetura REACT.

à nova posição alvo, encontrando um obstáculo (parede) à sua frente. Embora o comportamento *moveToGoal* atraia o robô para o alvo, o comportamento *avoid-Collision* evita a colisão com a parede, resultando em uma trajetória estável, paralela à parede, até a porta do canto inferior esquerdo do ambiente, quando o robô consegue novamente entrar na sala e se dirigir para o alvo. A trajetória executada nesta última parte do experimento ilustra bem o caráter não-ótimo das trajetórias obtidas em sistemas reativos.

Durante os testes e as simulações, os sonares do robô demonstraram ineficiência na detecção de quinas. Estas só são detectadas quando o robô já está muito próximo das mesmas, de modo que a colisão não pode ser evitada. Para minimizar este problema foi inserido um mecanismo que verifica, a cada nova percepção do ambiente, se a magnitude da rotação a ser executada pelo robô excede um determinado limiar. Se isto ocorrer, entende-se que houve falha na detecção do obstáculo e este já se encontra demasiadamente próximo do robô, caso contrário a trajetória teria sido mais suave e tal magnitude de rotação não seria necessária. O sistema faz então com que a velocidade translacional seja igual a zero e apenas a rotação é executada, até que uma nova percepção do ambiente seja processada e a magnitude da rotação não ultrapasse o limiar. Com esse mecanismo o robô consegue desviar eficientemente de obstáculos, mesmo quando a sua detecção não é apropriada.

Discussão

A arquitetura REACT se mostrou, através dos testes, bastante ajustável, devido ao grande número de parâmetros dos comportamentos. Isto era esperado, pois houve a preocupação, durante todo o decorrer do desenvolvimento, de permitir o ajuste dos comportamentos a diferentes ambientes, bem como permitir que estes comportamentos primitivos sirvam de base para tarefas mais complexas. No entanto, esta flexibilidade traz uma dificuldade no dimensionamento dos parâmetros, principalmente no tocante a velocidade máxima estipulada para o robô. Durante os testes, pôde-se perceber que os parâmetros são extremamente dependentes desta velocidade, o que indica que estes devem ser equacionados também em relação a esta velocidade. Há a intenção, em trabalhos futuros, de equacionar esta dependência, bem como eliminar a dificuldade de dimensionar os parâmetros, através do uso de mecanismos baseados em algoritmos adaptativos para gerá–los automaticamente.

O Método de Campos Potenciais visa suavizar a trajetória executada pelo robô, mas para isso é necessário que a capacidade sensorial do robô cubra 360 graus ao seu redor, caso contrário o método tem a sua eficácia diminuída. A capacidade sensorial do robô Pioneer usado é apenas frontal e sujeita a falhas, como no caso das quinas. Isto se refletiu, algumas vezes, em mudanças bruscas de direção e variações igualmente indesejadas na velocidade. Pretende-se também incluir um novo comportamento na arquitetura: *moveToFree*, baseado no cálculo, pelo sistema de visão computacional, do divergente do fluxo ótico para determinar espaços livres para locomoção (veja Seção 10.5.3).

11.2 Arquitetura Reativa com Coordenação Competitiva

A arquitetura reativa *subsumption* foi implementada e testada no robô Pioneer, uma vez que a mesma é adequada para robôs que operam em tempo real, já que a simplicidade dos comportamentos reativos favorece rápido processamento computacional (BRAMBILA; COSTA, 2002). Essa arquitetura prevê, para cada estímulo captado do ambiente pelo robô, uma reação pré-definida (veja Capítulo 5).

A arquitetura *subsumption* organiza os comportamentos reativos em camadas de competência e, através de um mecanismo de prioridades, estabelece-se uma hierarquia entre essas camadas, fazendo com que comportamentos de níveis mais altos correspondam a comportamentos mais deliberativos da tarefa especificada, enquanto aqueles de níveis mais baixos sejam direcionados a tarefas mais simples, reativas. Cria-se então uma coordenação competitiva entre os comportamentos, que disputam entre si pelo envio de um comando aos atuadores.

Foi utilizado o mecanismo de supressão entre os comportamentos, onde o comportamento de nível mais alto define uma ação e suprime a ação estipulada por outro de nível mais baixo, que permanece ativo mas sua ação não produz efeito diante de um comportamento prioritário. Entretanto, a prioridade de um comportamento de nível mais alto ocorre apenas sobre comportamentos específicos de níveis mais baixos e não sobre todos de forma indiscriminada. Por exemplo, comportamentos que preservam a integridade física do robô, como parar ao detectar um obstáculo muito próximo, não são suprimidos por outros comportamentos.

Até o momento, foram implementados três comportamentos, que têm por objetivo fazer com que o robô se mova em um ambiente em direção a um alvo pré-estabelecido, desviando-se dos obstáculos quando necessário: *avoidCollision*, *moveAhead* e *moveToGoal*.

11.2.1 Descrição dos Comportamentos

O comportamento *avoidCollision* inclui ações relativas à preservação da integridade física do robô e tem prioridade absoluta de atuação; o comportamento *moveAhead* consiste em preservar o robô na mesma velocidade e direção de movimento em que se encontra, e corresponde à camada mais baixa da hierarquia de prioridades, e o comportamento *moveToGoal*, que implementa o objetivo primordial do robô, no caso, atingir um alvo, correspondendo a uma camada mais alta na hierarquia.

O comportamento *avoidCollision* recebe como parâmetros a distância de máxima aproximação de obstáculos *stopDistance* e o ângulo de giro $\Delta\phi$ que o robô deve girar para evitar colisões. O comportamento recebe as leituras dos oito sonares, determina a distância d_{min} do obstáculo mais próximo e o lado do robô

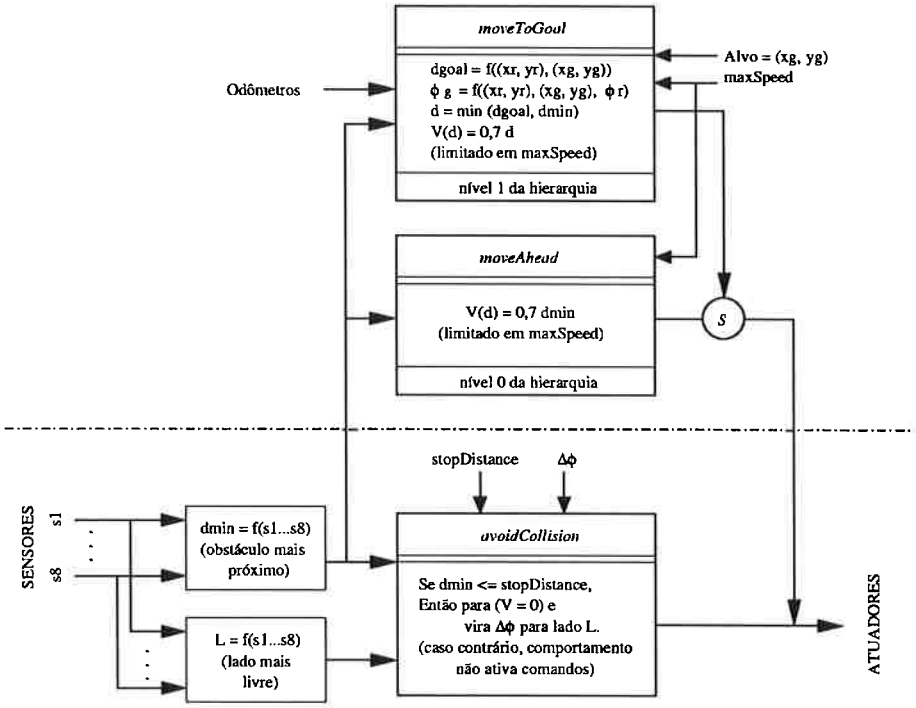


Figura 11.3: Implementação realizada da arquitetura *subsumption*.

L mais livre de obstáculos. Se $d_{min} \leq stopDistance$ então o comportamento é ativado e envia ao robô (com prioridade máxima) comando de parar e girar de $\Delta\phi$ no sentido definido por L; caso contrário, o comportamento não é ativado.

O comportamento *moveAhead* recebe como parâmetro a velocidade máxima *maxSpeed* de navegação do robô, efetua as leituras dos oito sonares e determina a distância d_{min} do obstáculo mais próximo. O comportamento então envia ao robô um comando de avançar, na direção em que se encontra, com velocidade $v = 0,7d_{min}$, caso $v > maxSpeed$, então faz $v = maxSpeed$.

Finalmente, o comportamento *moveToGoal* recebe como parâmetros a velocidade máxima *maxSpeed* de navegação do robô e as coordenadas do alvo (x_g, y_g); efetua a leitura do sistema de odometria, determinando suas coordenadas atuais (x_r, y_r) e a orientação atual do robô ϕ_r, e calculando sua distância do alvo $d_{goal} = f((x_r, y_r), (x_g, y_g))$ e a orientação ao alvo $\phi_g = f((x_r, y_r), (x_g, y_g), \phi_r)$; efetua as leituras dos oito sonares, determinando a distância d_{min} do obstáculo mais próximo e calculando a distância d, dada por $d = min(d_{goal}, d_{min})$. O comportamento então envia ao robô um comando de avançar, na direção ϕ_g, com velocidade $v = 0,7d$, caso $v > maxSpeed$, então faz $v = maxSpeed$. A Figura 11.3 ilustra a implementação realizada da arquitetura *subsumption*.

11.2.2 Resultados e Discussões

Os resultados obtidos até agora mostraram-se eficientes, principalmente o comportamento *avoidCollision*, já que nos testes realizados o robô não colidiu com os obstáculos.

Os parâmetros mais freqüentemente utilizados nos testes foram os seguintes: $stopDistance = 15cm$, $\phi = 10graus$, $maxSpeed = 1m/s$, posição (x_g, y_g) qualquer.

Com esses parâmetros o robô locomove-se com boa velocidade, que diminui de forma suave ao se aproximar de um obstáculo. Para determinar a velocidade em função da distância ao obstáculo, a função $v = 0.7d$ desenvolve velocidade suficientemente rápida sem colidir.

No comportamento *avoidCollision*, a princípio o robô girava de $\Delta\phi = 90$ graus para o lado oposto à menor leitura (lado mais livre). Entretanto, esse ângulo fazia com que o robô girasse de forma brusca, dificultando a tarefa de encontrar caminhos alternativos para desviar do obstáculo, principalmente em configurações com passagens estreitas (como corredores, por exemplo). Portanto, o melhor ângulo encontrado empiricamente, que permite ao robô sair por passagens estreitas foi $\Delta\phi = 10$ graus.

O problema mais freqüente, típico desta arquitetura, é a oscilação entre comportamentos, sob certas condições do ambiente. Problemas intrínsecos ao *dead reckoning* também foram observados. Espera-se diminuir estes erros com o uso de visão computacional nos comportamentos. Nova série de testes, tanto no simulador quanto no robô real, está sendo realizada.

11.3 Arquitetura Reativa Multi-Agentes: Vibra

A autora e seu grupo propuseram uma nova arquitetura reativa, VIBRA, baseada na metáfora de Sistemas Multi-Agentes. A arquitetura VIBRA – Vision Based Reactive Architecture – pode ser vista como uma sociedade de Agentes Autônomos (AAs), cada qual exibindo um comportamento para solucionar um problema de sua competência específica, que colaboram entre si para alcançar suas metas (BIANCHI; COSTA, 1996; BIANCHI; COSTA, 1997a; BIANCHI; COSTA,

1997b; BIANCHI; COSTA, 1997c; COSTA; BARROS; BIANCHI, 1998; BIANCHI, 1998).

A teoria dos Sistemas Multi-Agentes (SMA) define a organização de agentes em uma sociedade onde relações de autoridade, comunicação, controle e fluxo de informação são descritas. A arquitetura VIBRA é um sistema multi-agente que processa pedidos assíncronos recebidos por sensores, os prioriza, suspende temporariamente as ações de mais baixas prioridade, e intercala comportamentos compatíveis.

Nesta arquitetura os Agentes Autônomos possuem: *(i)* capacicades de comunicação, com um módulo responsável pela interação entre os AAs e uma linguagem de interação que define a maneira pela qual os AAs se comunicam; *(ii)* planejamento e execução, que possibilita a geração e execução de um plano para atingir a tarefa definida para o AA, exibindo um comportamento esperado, controlando a aquisição de dados sensoriais e a atuação do agente sobre o mundo.

Um agente especial na sociedade pode criar, mudar ou destruir a sociedade, adicionando ou removendo agentes e controlando os recursos nas fases de início e término da sociedade. Assim, novos agentes podem ser implementados e ativados na sociedade durante a execução do sistema para lidar com mudanças de condições no ambiente ou domínio. Por outro lado, agentes que não estão executando suas tarefas de maneira satisfatória podem ser removidos. Os AAs se comunicam entre si através de uma rede descentralizada e totalmente conectada (veja Figura 11.4).

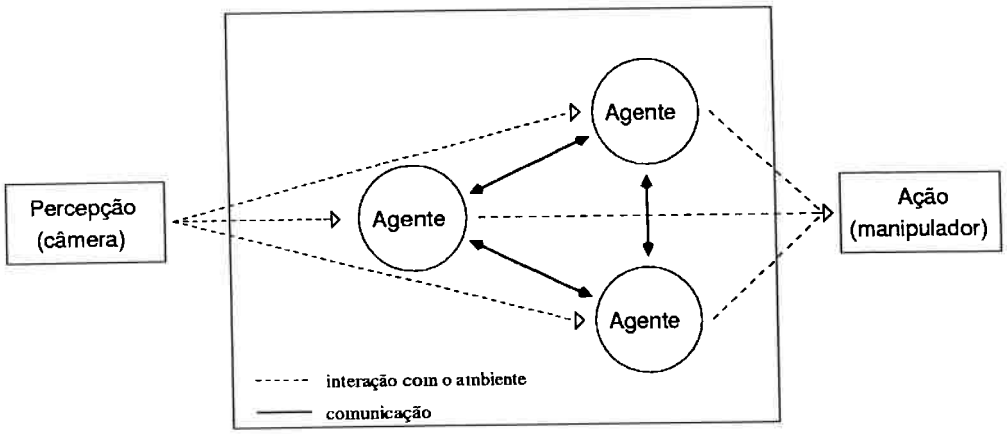


Figura 11.4: Sociedade de agentes implementada na arquitetura VIBRA.

Para gerenciar a interação entre os agentes, a sociedade é controlada por uma

política que envolve regras de comportamento e uma estrutura de autoridade. Esta política possibilita que os agentes decidam qual deles deve possuir o controle de um recurso em um determinado momento.

Um recurso é definido como sendo parte do sistema que tem seu uso compartilhado pelos agentes e que pode ser controlado por somente um agente em um determinado momento. Por exemplo, o sistema de direção de um robô móvel ou um manipulador robótico em uma célula de montagem são recursos, assim como uma câmera (e seu hardware e software de aquisição de dados). Existe uma diferença entre o nível de prioridade nos quais os recursos são compartilhados. Isso ocorre porque alguns recursos são mais disputados que outros. Por exemplo, uma câmera que consegue fornecer imagens sem criar conflito entre os agentes é menos disputada que um manipulador robótico. Por outro lado, caso essa câmera faça parte de um sistema de visão ativa, torna-se mais disputada, uma vez que os agentes competem pelo controle do processo de aquisição de dados. Por isso, existe a necessidade de uma política para a solução de conflitos na alocação de recursos entre os agentes.

A estrutura de autoridade define o nível de prioridade que um agente possui para usar um determinado recurso. A estrutura de autoridade da sociedade é o que torna possível a decisão sobre como os recursos do sistema devem ser alocados, decidindo qual agente deve ter o controle de um recurso em um determinado momento, evitando conflitos. A maneira pela qual essa decisão é tomada depende das regras de comportamento definidas para a sociedade em uma implementação específica e pode ser, por exemplo, o resultado de uma competição entre agentes. Desse modo, as regras de comportamento definem como a estrutura de autoridade deve ser usada para o controle das comunicações e da alocação dos recursos.

As regras de comportamento e a estrutura de autoridade são dependentes de domínio: as regras podem mudar e os níveis de autoridade variam para cada agente e cada recurso.

A estrutura de autoridade de um sistema possui uma profunda relação com a precedência e a dependência entre os comportamento dos agentes presentes na sociedade. Em geral, os comportamentos reativos devem ter precedência sobre os deliberativos, uma vez que exigem atuação em tempo real. Dessa maneira, a estrutura de autoridade definida para os AAs é geralmente uma hierarquia que

vai do agente mais reativo ao mais deliberativo.

O uso explícito de regras sociais na definição de um agente permite a ele alcançar suas metas determinadas dinamicamente sem interferir com outros agentes. Na arquitetura de VIBRA foram adotadas três regras simples:

- **Regra # 1:** somente um agente pode controlar um recurso em um determinado momento.
- **Regra # 2:** qualquer agente pode requisitar o controle de um recurso para um agente com menor autoridade que ele, em qualquer momento.
- **Regra # 3:** um agente só pode requisitar o controle de um recurso para um agente de maior autoridade se este estiver cedendo o controle.

11.3.1 Domínio de aplicação

A aplicação desta arquitetura foi desenvolvida na célula flexível de montagem da EPUSP. O domínio de montagem pode ser caracterizado como uma tarefa de planejamento complexa e reativa, onde os agentes têm que gerar e executar planos, coordenar suas atividades para alcançar uma meta comum e executar alocação dinâmica de recurso. A dificuldade da execução dessa tarefa se encontra na necessidade de realizar um processamento adequado das imagens e de uma capacidade de compreensão que permita ao sistema lidar adequadamente com interrupções para evitar colisões e interações de um operador humano com a configuração da mesa de trabalho. Este domínio foi o assunto de diversos trabalhos de grupo na Célula de Montagem Flexível mostrada na Figura 11.5 (COSTA; BARROS; BIANCHI, 1998; BIANCHI, 1998).

Um exemplo de uma tarefa de montagem é o empacotamento. Dado que peças chegam em uma mesa (por meio de uma correia transportadora, por exemplo), a meta é selecionar peças, limpá-las e empacotá-las. As peças, que podem ser de metal ou plástico moldado, geralmente possuem rebarbas do próprio processo de fabricação. Limpar uma peça significa remover estas rebarbas não desejadas ou quaisquer outros objetos que obstruam o empacotamento. Deste modo, não existe a necessidade de se limpar todas as peças antes de empacotá–las, apenas aquelas que devem ser guardadas e não estão limpas.

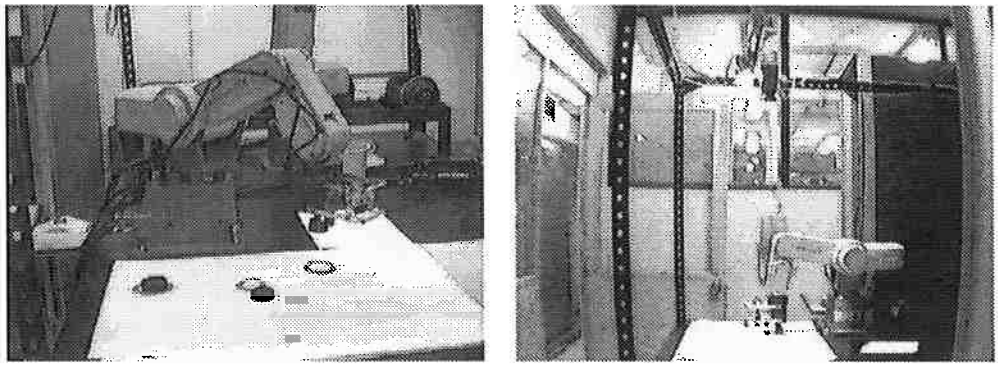


Figura 11.5: O manipulador utilizado da Célula de Montagem.

Enquanto a tarefa principal está sendo executada, interações humanas podem acontecer a qualquer momento interferindo na tarefa de montagem de modos diferentes e imprevisíveis: podem ser adicionados objetos não desejados ou novas peças para a montagem e o robô tem que ser capaz de lidar corretamente com esses eventos. Para evitar colisões, a limpeza e o empacotamento devem ter sua execução interrompida até que a área de trabalho esteja livre da possibilidade de colisão.

O domínio de montagem é um caso típico de uma tarefa que pode ser decomposta em um conjunto de tarefas independentes. Para reduzir a complexidade do problema, pode-se assumir que cada tarefa da solução corresponde a um comportamento que é independente e interage com outros comportamentos. Assim, a solução desta tarefa de montagem pode ser decomposta em três subtarefas:

1. Montagem: se uma peça sobre a mesa está limpa, ela deve ser pega com o manipulador e colocada na localização desejada, empacotando-a.

2. Limpeza: se uma peça possui rebarbas, estas devem ser removidas. Além disso, objetos não desejados que uma pessoa ou outro manipulador possam ter colocado na área de trabalho devem ser retirados.

3. Evitar colisões: evitar colisões do manipulador com objetos que se movam na área de trabalho, com o objetivo de preservar a integridade física do sistema.

Seguindo a estrutura proposta da arquitetura VIBRA, estas três tarefas foram mapeadas em três Agentes Autônomos, o *Montador*, o *Limpador* e o *Evitador*

de Colisões, respectivamente. Os recursos compartilhados nesta aplicação de montagem são a câmera e o manipulador robótico. Nessa aplicação, para qualquer pedido dos três agentes, a câmera pode prover uma imagem sem criar qualquer conflito entre as tarefas. Por outro lado, o manipulador é altamente disputado e, conseqüentemente, as tarefas têm que obedecer uma política específica para resolução de conflito. Assim, a estrutura de autoridade foi definida para os agentes deste domínio da seguinte maneira: o agente *Evitador de Colisões* é o agente com maior autoridade, o *Limpador* é o segundo e o *Montador* é o agente com menor autoridade. Pode-se notar que a estrutura de autoridade tem por objetivo principal a preservação da integridade física do sistema e, secundariamente, a tarefa global a ser realizada. Quando um sistema cresce, a estrutura é modificada para incorporar outros agentes.

Na tarefa de montagem o objetivo e o tipo de peças envolvidas podem mudar, por exemplo, da tarefa de embalagem para a montagem de um objeto conhecido ou a seleção de objetos por forma ou cor. Aplicações freqüentes são a produção de produtos industriais pequenos como partes mecânicas ou brinquedos. Vale a pena notar que uma célula de montagem pode ser parte de uma linha de montagem maior, fabricando parte de um produto mais complexo. Apesar do uso do domínio de uma célula de montagens, a arquitetura proposta pode ser aplicada a outros domínios, por exemplo, robôs móveis autônomos.

11.3.2 Resultados e Discussões

A aplicação implementada na Célula de Montagem Flexível foi testada extensivamente, tendo desempenhado de maneira satisfatória todas as tarefas de montagem.

Por ser historicamente distribuída e modular, a abordagem de Sistemas Multi-Agentes é uma tendência natural para a modelagem de uma arquitetura de controle distribuído, sendo uma solução eficaz para o problema da integração de diversos módulos de comportamentos diferentes em um mesmo sistema robótico que utiliza visão computacional e diversos outros sensores para perceber o mundo.

11.4 Arquiteturas Híbridas: L-VIBRA e ANT-VIBRA

Numa arquitetura reativa (multi-agentes ou não), a seqüência de ativação de comportamentos primitivos – ou a coordenação de comportamentos – é fixa, pré-definida e construída manualmente no projeto. Esta foi uma desvantagem da solução adotada na arquitetura VIBRA (COSTA; BARROS; BIANCHI, 1998), que usa uma estrutura de autoridade pré-definida e fixa. Assim, uma vez estabelecido que um agente tem precedência sobre outro, o sistema sempre se comportará da mesma maneira, não importando se isto resulta em um desempenho ineficiente. Porém, em uma aplicação real, se um objeto não desejado não está impedindo uma ação de montagem, não é necessário executar uma ação de limpeza *a priori*.

Para solucionar este problema, foi desenvolvida uma arquitetura híbrida, onde existe um módulo seqüenciador que adapta dinamicamente o conjunto de comportamentos (agentes) e suas condições de ativação e seqüenciamento. Assim, a autora e seu grupo de pesquisa propuseram L-VIBRA e ANT-VIBRA (COSTA; BIANCHI, 2000; BIANCHI; COSTA, 2002d; BIANCHI; COSTA, 2002a; BIANCHI; COSTA, 2002b), que são arquiteturas híbridas multi-agentes nas quais foi introduzido um agente seqüenciador capaz de aprender, através de aprendizado por reforço, a melhor seqüência de ativação dos comportamentos (agentes) para a execução da tarefa, segundo as condições sensoriadas.

Uma vez que evitar colisões é uma tarefa extremamente reativa, sua precedência sobre as tarefas de montagem e limpeza é preservada nas arquiteturas propostas. Foi usado o algoritmo Q-Learning (veja Seção 8.3) para aprender a coordenação entre os agentes, decidindo entre as tarefas de montagem e limpeza (Figura 11.6).

A introdução do algoritmo de aprendizado por reforço Q-Learning na arquitetura L-VIBRA (COSTA; BIANCHI, 2000) resultou em um sistema capaz de criar o plano de montagem otimizado como desejado, mas cujo desempenho na produção de novos planos não se mostrou suficiente, decorrente das inúmeras interações necessárias do robô com o ambiente para que a coordenação seja aprendida (o que demanda muito tempo). Toda vez que a configuração da área de trabalho da célula de montagem é alterada, o sistema tem que aprender um novo plano. Des-

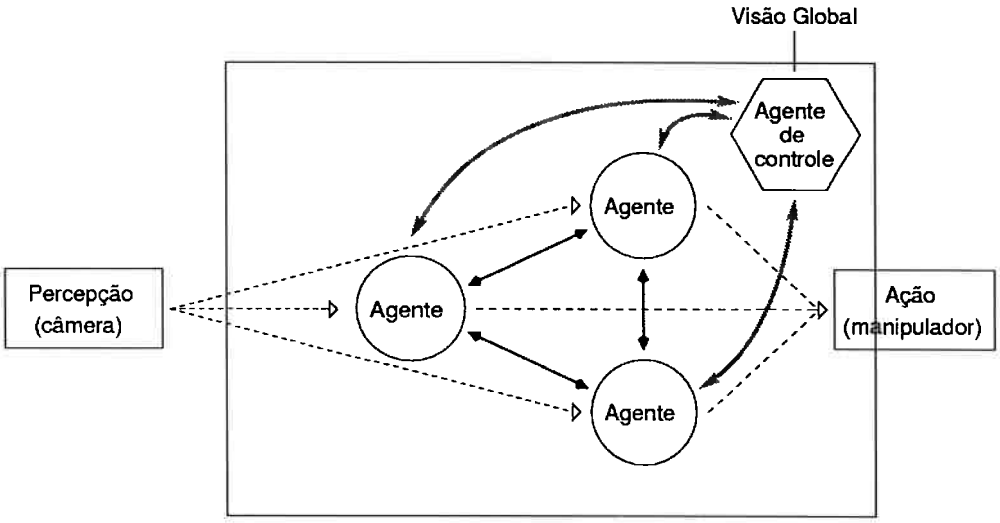


Figura 11.6: Esboço da sociedade implementada na arquitetura L-VIBRA.

ta maneira, é necessário um algoritmo de aprendizado de alto desempenho.

Uma nova versão da arquitetura – chamada ANT-VIBRA (BIANCHI; COSTA, 2002d; BIANCHI; COSTA, 2002a, 2002a) – foi proposta, adaptando o algoritmo de *Ant Colony System* (ACS) (DORIGO; GAMBARDELLA, 1997) para lidar com sub-tarefas diferentes e para planejar a rota que minimiza o deslocamento total feito pelo manipulador ao realizar seus movimentos durante a tarefa de montagem (incluindo limpeza).

11.4.1 O Algoritmo ACS

Baseado na metáfora do inseto social, a Inteligência de Enxame (*Swarm Intelligence*) estuda a emergência da inteligência coletiva em grupos de agentes simples, enfatizando a flexibilidade, robustez, autonomia e interação (direta ou indireta) entre agentes. Os Métodos de Enxame mais comuns estão baseado na observação do comportamento de colônias de formiga. Nestes métodos, um conjunto de agentes simples, chamados de formigas, cooperam para encontrar boas soluções para problemas de otimização combinatorial.

O Algoritmo ACS é um algoritmo de Inteligência de Enxame proposto por Dorigo e Gambardella (1997) para solução de problemas. Ele foi aplicado a vários problemas de otimização combinatorial como os problemas do Caixeiro Viajante simétricos e assimétricos (TSP e ATSP respectivamente), roteamento em redes de computadores, robótica, entre outros, sendo atualmente considerado o melhor

algoritmo para a solução do problema do Caixeiro Viajante. O ACS pode ser interpretado como um tipo particular de técnica de Aprendizado por Reforço distribuída, em particular uma abordagem distribuída aplicada ao Q-Learning.

O ACS representa a utilidade de se mover de uma cidade s quando o agente se encontra na cidade r como $\tau(r,s)$, chamado *feromônio*, que é um valor real positivo associado ao arco (r,s) em um grafo. O feromônio é similar ao valor-Q no algoritmo Q-Learning. Existe também uma *heurística* $\eta(r,s)$ associada ao arco (r,s). Ela representa uma avaliação heurística de qual movimento é melhor. No problema do Caixeiro Viajante, $\eta(r,s)$ é o inverso da distância δ de r à s, $\delta(r,s)$.

O agente k posicionado na cidade r se move para a cidade s usando a seguinte regra de transição de estado:

$$s = \begin{cases} \arg \max_{u \in J_k(r)} \tau(r,u) \cdot \eta(r,u)^\beta & \text{se } q \leq q_0 \\ S & \text{se } q > q_0 \end{cases} \tag{11.5}$$

onde:

- β é um parâmetro que pondera a importância relativa dos valores de feromônio apreendidos e da heurística ($\beta > 0$).
- $J_k(r)$ é a lista de todas as cidades que devem ser visitadas por um agente k, onde r é a cidade atual. Esta lista é usada para restringir os agentes, permitindo que visitem cada cidade apenas uma vez.
- q é um valor escolhido de maneira aleatória com probabilidade uniforme entre [0,1] e q_0 é o parâmetro que define a taxa de exploração/exploitação: quanto maior q_0 menor a exploração ($0 \leq q_0 \leq 1$).
- S é uma variável com valor aleatório.

Esta regra de transição favorece arcos com uma quantia grande de feromônio e que representam uma distância curta.

Os agentes no ACS atualizam o valor de $\tau(r,s)$ em duas situações: na atualização local (quando visitam os arcos) e na global (quando terminam de percorrer o grafo, completando as visitas a todas as cidades). A regra de atualização local é:

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \Delta\tau(r,s) \tag{11.6}$$

onde: $0 < \rho < 1$ é a taxa de aprendizado local, e $\Delta\tau(r,s) = \gamma \cdot \max_{z \in J_k(s)} \tau(s,z)$, sendo γ um fator de desconto ($0 \leq \gamma \leq 1$), simulando uma "evaporação" do feromônio depositado.

A regra de atualização global é dada por:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s) \tag{11.7}$$

onde: α é a taxa de aprendizado global, e $\Delta\tau(r,s)$ é um reforço, geralmente o inverso da distância do menor percurso. O reforço é dado apenas para o melhor caminho, feito pelo agente que percorreu a menor distância global.

As fórmulas de atualização do feromônio visam depositar uma quantidade maior de feromônio nos caminhos mais curtos, e simulam tanto a adição de feromônio por formigas quanto sua evaporação.

Para ser capaz de tratar os problemas apresentados no domínio da arquitetura ANT-VIBRA, modificações no algoritmo ACS se fizeram necessárias. A primeira foi a introdução de diversas tabelas de feromônios, uma para cada operação que o sistema realiza. E a segunda foi a extensão da lista de lugares a serem visitados para uma lista $J_k(s,a)$ de pares estado/ação que podem ser aplicados na próxima transição e estado, no domínio de montagem.

A arquitetura ANT-VIBRA foi testada em um domínio simulado que é representado por uma área de trabalho discreta de 10x10 posições, onde cada posição pode apresentar uma das seis seguinte configurações: uma peça, um encaixe, uma obstrução, uma peça com obstrução, um encaixe obstruído, uma peça sobre um encaixe ou uma posição livre.

Foram realizados diversos experimentos considerando configurações diferentes da área de trabalho, onde foram aprendidos com sucesso as melhores políticas de ação para cada experiência.

11.4.2 Resultados e Discussões

Para ilustrar os resultados são apresentados três exemplos. Em todos, o objetivo é encontrar uma seqüência de acões de montagem que minimize a distância percorrida pelo manipulador robótico. Uma iteração termina quando não restam peças a empacotar e o processo de aprendizado pára quando o resultado se

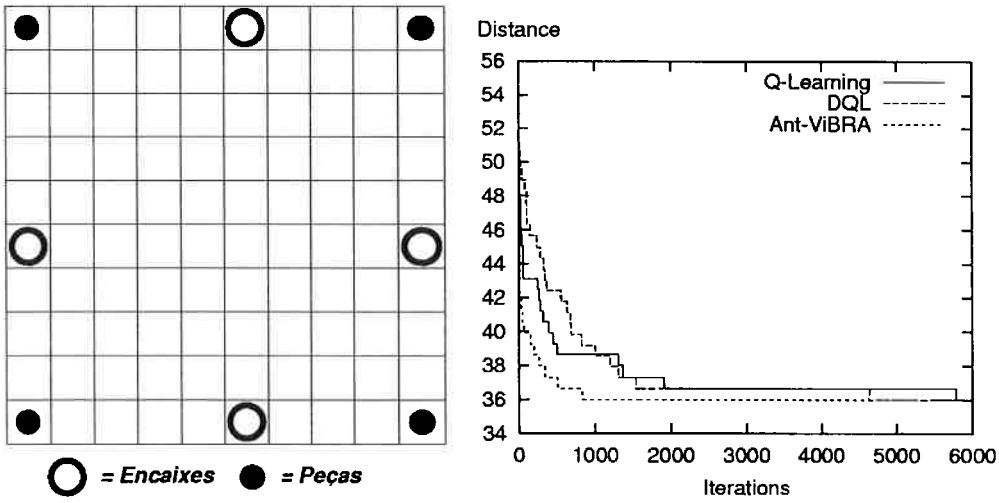


Figura 11.7: Configuração do exemplo 1 e seu resultado.

estabiliza ou quando se atinge um número máximo de iterações permitidas.

Nos exemplos são apresentados os resultados do ANT-VIBRA comparados aos resultados obtidos pelo sistema L-VIBRA – que utiliza o algoritmo de aprendizado por reforço Q-Learning –, e com uma nova versão do L-VIBRA, onde foi implementado o algoritmo *Distributed* Q-Learning (DQL), uma versão distribuída do mesmo algoritmo Q-Learning (BIANCHI; COSTA, 2002a).

No primeiro exemplo (Figura 11.7) há 4 peças e 4 encaixes separados na borda de uma área de trabalho 10x10. Como não há nenhum lixo (obstrução), as operações que podem ser executadas são: *(i)* pegar uma peça e *(ii)* colocar a peça sobre um encaixe. A posição inicial (e final) do manipulador é sobre a célula (1,1).

Neste exemplo, a média de 25 episódios do ANT-VIBRA levou 844 iterações para convergir à solução ótima, que é 36 unidades de deslocamento (a distância total entre peças e encaixes). O mesmo problema levou 4641 passos em média para alcançar o mesmo resultado usando o DQL e 5787 passos usando o Q-Learning. Isto mostra que a combinação de aprendizado por reforço e heurísticas rende bons resultados.

O segundo exemplo (Figura 11.8) é similar ao primeiro, porém com 8 peças e 8 encaixes distribuídos de maneira aleatória na área de trabalho. Os resultados do ANT-VIBRA novamente são melhores que os do DQL ou do Q-Learning,

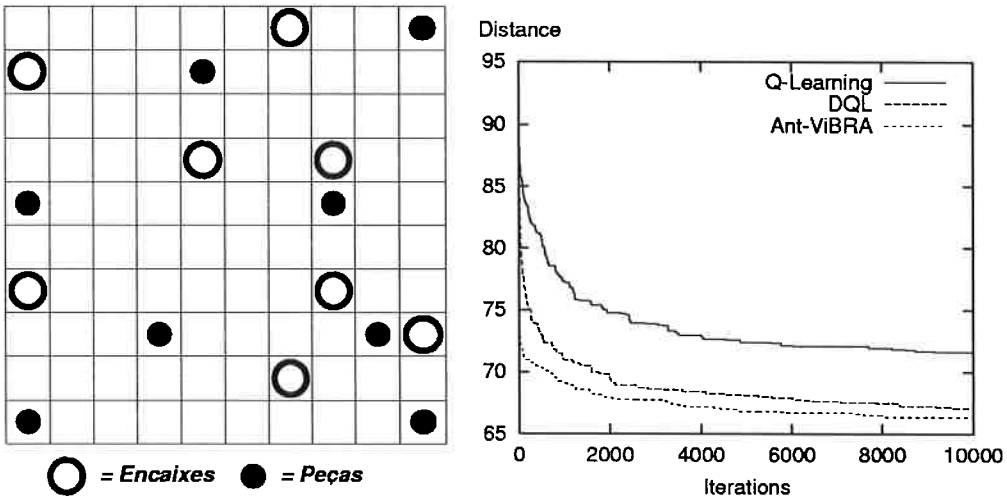


Figura 11.8: Configuração do exemplo 2 e seu resultado.

computados em uma média de 25 episódios.

Finalmente, o exemplo 3 apresenta uma configuração onde o sistema deve realizar a limpeza de algumas peças antes de realizar a tarefa de montagem (Figura 11.9). A posição das peças é a mesma do exemplo 1, mas a peça na posição (1, 10) e o encaixe na posição (6,1) devem ser previamente limpos. A ação de limpeza move o manipulador para a posição a ser limpa, pega o objeto indesejado e o leva para a lata de lixo, na posição (1,11). Pode-se ver que novamente o ANT-VIBRA apresenta os melhores resultados.

Nos três exemplos os parâmetros usados foram os mesmos: taxa de exploração/exploitação de 0.9, fator de desconto γ de 0.3. A taxa de aprendizado (α para ambos o Q-learning e o DQL e ρ para o ANT-VIBRA) usada é de 0.1. Os experimentos foram realizados em um microcomputador AMD K6-2 500MHz com 256 Mbytes de memória RAM, usando Linux e compilador gcc. nesta configuração, a ANT-VIBRA convergiu ao valor ótimo, no exemplo 1, na ordem de milissegundos de tempo de processamento, demonstrando sua eficiência na resolução do problema.

Das experiências realizadas foi possível concluir que o uso de um agente de controle baseado em aprendizado por reforço permitiu a inserção de uma estrutura de controle dinâmica para a coordenação dos agentes em um Sistema Multi-Agentes.

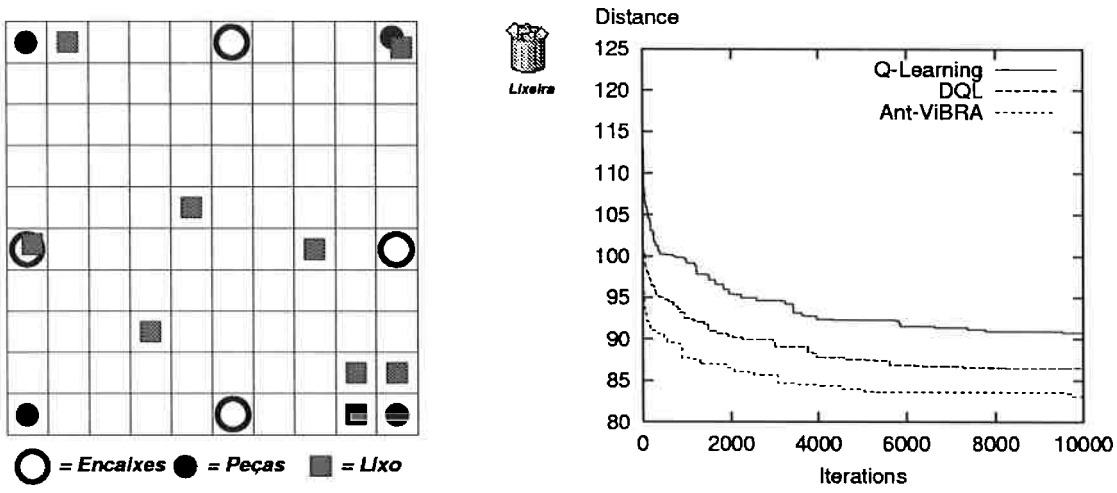


Figura 11.9: Configuração do exemplo 3 e seu resultado.

Os resultados obtidos mostraram que o algoritmo ANT-VIBRA pôde minimizar o tempo de execução da tarefa (ou a distância total percorrida pelo manipulador) em várias configurações da área de trabalho. Além isso, o tempo de aprendizagem do ANT-VIBRA também foi menor quando comparado ao DQL e Q-Learning.

Porém, o tamanho do espaço de busca cresce exponencialmente com o número de posições. Para tratar problemas do mundo real, uma alternativa para as tabelas Q é o uso de alguma forma de representação compacta para a função de valor, como uma rede Neural de Perceptron Multicamada. Estudos neste sentido têm sido feitos pela autora e seu grupo (veja Seção 12.4.2). A autora também conduz pesquisas visando agilizar as técnicas de aprendizado por reforço, embutindo conhecimento do domínio na formulação (veja Seções 12.4.1 e 13.3).

11.5 Discussões

Pelos estudos e desenvolvimentos realizados, comprova-se a adequação do uso de arquiteturas reativas para tarefas que exigem execução em tempo real.

No entanto, para tarefas deliberativas mais complexas, arquiteturas reativas apresentam dificuldades. O número de comportamentos(complexos) necessários para a execução de tais tarefas pode ser muito grande, dificultando – ou mesmo

impossibilitando – uma coordenação efetiva e eficiente entre elas.

A autora acredita que a solução se encontra no uso do paradigma multi-agentes, e pesquisas e desenvolvimentos continuarão sendo feitos nesta direção.

12 Navegação e Controle

Navegação é uma tarefa fundamental na Robótica Móvel e envolve determinar a localização do robô no ambiente, planejar atividades e trajetórias a seguir no ambiente, além da capacidade de construir e usar adequadamente mapas do ambiente, de forma autônoma. A autora tem conduzido pesquisas e desenvolvimentos nesta linha.

Um algoritmo de construção de mapas usando sonares e o robô Pioneer é descrito na Seção 12.1. O ambiente de atuação é um ambiente interno bem estruturado (como escritórios, por exemplo). A proposta apresentada consiste em um método híbrido para armazenagem do mapa, utilizando uma grade de tamanho variável.

A seguir, na Seção 12.2, um método de localização utilizando grades probabilísticas é apresentado, para uso no robô Pioneer, com informações sensoriais advindas dos sonares.

Tanto a construção de mapas, quanto a localização e locomoção de um robô no ambiente podem ser mais eficientes caso se tenha disponibilidade de melhores e maior quantidade de informações sensoriais. Assim, técnicas de visão computacional em conjunto com aprendizado autônomo estão sendo investigadas pela autora e seu grupo, buscando formas para que o robô seja capaz de aprender por si só marcos visuais do ambiente, podendo, assim, utilizá-los para sua localização e representação do ambiente. Pesquisas e desenvolvimentos nesta linha encontram-se descritos na Seção 12.3.

Finalmente, a autora e seu grupo têm explorado o uso de aprendizado por reforço na definição de políticas ótimas (ou quase ótimas) de atuação, descrito na Seção 12.4 deste Capítulo. Investiga-se o uso de técnicas de generalização da experiência para agilizar o desempenho do algoritmo Q-learning em tarefas

de navegação. Uma outra linha desta pesquisa utiliza agregação de estados para generalizar a representação dos valores de custo de cada estado, para cada atuação. Normalmente esta representação é tabular, inviabilizando sua aplicação em sistemas de controle reais mais complexos, que possuem um grande número de estados e/ou ações. Assim, o uso de redes neurais para representar valores é investigado, com aplicações no domínio de futebol de robôs e no domínio real de um descarregador de navios.

12.1 Construção de mapas usando sonares e o robô Pioneer 2DX

Atualmente, quando se realiza a geração de mapas, são utilizados dois tipos básicos de algoritmos, classificados pelo modo de armazenar as informações coletadas e processadas: aqueles baseados em grade de pontos, que armazenam o mapa do ambiente como uma grade de ocupação (matriz de pontos), e os que armazenam o mapa em estruturas topológicas (grafos), baseados em marcos (ou pontos de referência) no ambiente.

Os mapas em grades de pontos são mais fáceis de serem construídos, porém ocupam muita memória e a tarefa de escolher trajetórias a partir dos mapas resultantes é muito custosa. Os mapas topológicos, por possuírem uma estrutura em grafo, tornam fácil a escolha de trajetórias, porém a seleção e identificação de marcos não é fácil e é uma área em estudo, não possuindo ainda algoritmos eficientes e robustos para este fim.

O algoritmo proposto pela autora e seu grupo (BARRA; DOMENECCI; COSTA, 2002; BARRA; DOMENECCI, 2002) propõe um método híbrido destas duas abordagens, utilizando uma grade de tamanho variável para representar o ambiente. Este algoritmo apresenta algumas características interessantes:

- O algoritmo utiliza uma estrutura de dados híbrida que apresenta vantagens sobre os algoritmos topológicos e o baseado em grade de pontos.
- O algoritmo gera o mapa de forma autônoma e durante sua exploração (*on-line*), ou seja, à medida que os dados vão sendo captados pelos sensores, eles vão sendo tratados a fim de gerar o mapa. Assim, não é necessário

armazenar todos os dados coletados para serem trabalhados no final.

- O algoritmo é dividido em módulos, facilitando sua compreensão, desenvolvimento, depuração e alteração.

No entanto, algumas restrições foram assumidas nesta proposta:

- O algoritmo só é capaz de lidar com ambientes compostos por segmentos de retas alinhados com os eixos de coordenadas cartesianas. Essa restrição ainda permite um amplo universo de aplicações, pois a maioria dos objetos, ambientes internos de escritórios e prédios apresentam essa característica.
- Devido a restrições impostas pela capacidade dos sonares, deve-se impor um tamanho mínimo para os segmentos dos objetos e contornos do ambiente. Este tamanho mínimo foi estimado em 50 centímetros para o robô Pioneer usado.
- Os corredores e salas por onde o robô se move devem possuir uma largura mínima entre suas paredes (e entre estas paredes e objetos internos). Adotou-se 1 metro para essa restrição, devido às capacidades sensoriais e motoras, além das dimensões do robô Pioneer.

O algoritmo proposto utiliza uma representação híbrida do ambiente. Inicialmente, o ambiente como um todo é representado por um grande (teoricamente infinito) quadrado, rotulado como desconhecido. À medida que vai sendo explorado, o quadrado vai sendo particionado e suas partições rotuladas como ocupadas ou vazias. O procedimento repete-se até que o mapa seja composto somente por polígonos que representam a área ocupada ou retângulos que representam a área livre.

Para o robô se deslocar pelo mapa, este é transformado num grafo (contendo apenas as partições vazias) e é determinado um caminho entre a partição em que o robô se encontra e a próxima partição escolhida para ser explorada. Enfim, o algoritmo se comporta localmente como um algoritmo de Grade de Pontos, porém possui uma estrutura celular que pode ser considerada como uma estrutura topológica.

A principal vantagem de representar áreas ocupadas por polígonos está na facilidade de trabalhar com marcos. Todos os vértices encontrados podem ser

usados como marcos, pois são locais únicos no mapa que podem ser facilmente identificáveis no ambiente. Isso pode ajudar o robô a se localizar no ambiente durante a exploração, minimizando os erros de odometria e dos sensores do robô, de modo muito eficiente.

12.1.1 Descrição do Algoritmo

O funcionamento geral do algoritmo é descrito abaixo:

1. Ao iniciar, o robô começa explorando o ambiente em linha reta, até encontrar algum obstáculo. Neste momento existe apenas uma partição, que é desconhecida e infinita para todos os lados.

2. Ao encontrar o primeiro obstáculo, o robô inicia a identificação dos vértices do obstáculo. À medida que os vértices são encontrados, eles são usados para gerar o novo objeto no mapa atual, até que o mesmo seja completamente explorado. Uma vez completo, o novo objeto é inserido na representação do ambiente (em forma poligonal, pois representa área ocupada) e a mesma é atualizada a fim de gerar as partições retangulares que cercam o objeto — no lugar da(s) antiga(s) partições que existiam no local — representando os espaços livres ou desconhecidos no ambiente.

3. Logo após, escolhe-se a próxima partição a ser explorada (ou encerra a execução caso não existam mais partições que mereçam atenção). Uma rota entre a partição atual e a partição a ser explorada é traçada e começa a ser percorrida. Se o robô encontrar um obstáculo no caminho definido pela rota, é preciso começar a mapear este novo objeto. Neste caso, o fluxo de execução volta ao passo 2.

4. Se não encontrar nada no caminho, ao chegar na partição determinada começa a exploração exaustiva desta partição. Se o robô encontrar um obstáculo desconhecido na partição, começa-se a mapear o novo objeto e o fluxo de execução volta ao passo 2 dessa descrição. Senão, o fluxo de execução volta ao passo 3 dessa descrição.

12.1.2 Detalhamento dos Módulos do Algoritmo

O algoritmo descrito de construção de mapas globais é composto por sete módulos, a saber: módulo de alinhamento com uma partição ocupada, módulo de identificação de formas, módulo gerenciador do mapa de partições, módulo selecionador de partição, módulo planejador de rotas, módulo de execução de rotas e módulo de exploração exaustiva.

Módulo de alinhamento com uma partição ocupada

A função do módulo de alinhamento com uma partição ocupada é determinar o ângulo de uma parede (ou face de um objeto) em relação ao robô e então fazer a frente do robô ficar paralela a esta parede. A Figura 12.1 ilustra este procedimento: o robô gira sobre seu eixo e efetua medições da distância das paredes, com seus sonares (pontos quadrados na Figura). Com determinado número de pontos (que varia de três a cinco, no caso), aplica-se o método dos mínimos quadrados para encontrar a reta que melhor descreve a parede, determinando, desta forma, seu ângulo de orientação em relação ao robô. Com base neste ângulo, o robô efetua o giro necessário para que fique paralelo à parede em questão, a uma distância pré-definida. O alinhamento nem sempre é muito preciso, mas isto é compensado pelo módulo de identificação de formas, pois este possui métodos de correção do alinhamento enquanto segue uma parede.

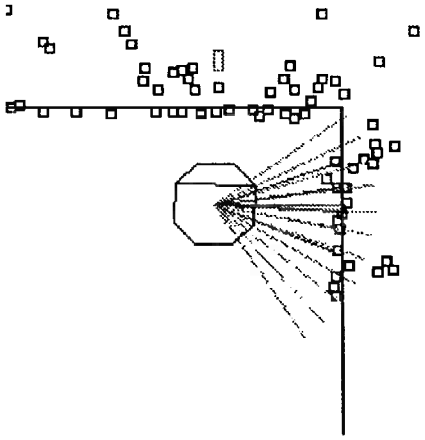


Figura 12.1: Alinhamento com uma parede.

Módulo de Identificação de Formas

O módulo de identificação de formas, por sua vez, tem como tarefas: seguir a parede até encontrar uma quina (ou canto), calcular o ponto da quina e ir até a próxima parede a ser explorada. Para seguir a parede, o módulo efetua constantemente a correção (de forma exponencial em relação ao erro, para se ter uma correção suave) do alinhamento do robô com a parede. O robô segue a parede até que a mesma termine (no caso de quina) ou que apareça outra parede à sua frente (no caso de canto). A Figura 12.2 ilustra este procedimento. Neste procedimento foram usados somente três dos oito sonares do robô: os dois frontais, para verificar a existência de paredes à frente, e um lateral, do lado da parede que estiver sendo seguida.

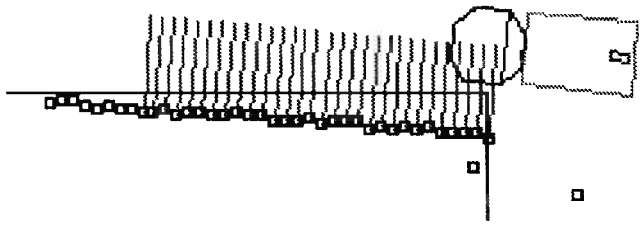


Figura 12.2: Robô seguindo uma parede.

Uma vez determinado o ponto da quina (ou canto), este valor é corrigido. Para isso, tem-se que o valor de uma das coordenadas da quina atual é o mesmo do ponto da quina anterior, pois todas as paredes são paralelas ao eixo horizontal ou ao vertical (restrição da abordagem). O valor da segunda coordenada pode ser calculado como o valor da quina anterior adicionada (ou subtraída, dependendo do caso) da distância entre a quina anterior e a quina atual. A Figura 12.3 ilustra um processo de correção do ponto de uma quina.

Para ir até a próxima parede a ser explorada, basta fazer: se for um canto, virar 90 graus; se for uma quina, é necessário andar para frente a distância D pela qual está se seguindo a parede, virar 90 graus e andar a mesma distância D para a frente. A distância D é definida empiricamente, como uma distância de segurança para distanciamento do robô das paredes. Caso seja necessário,

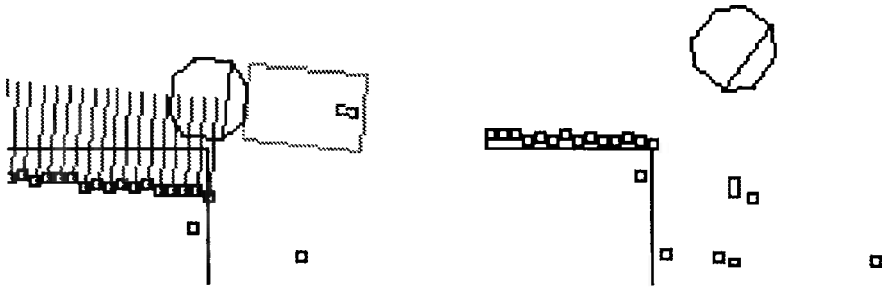


Figura 12.3: Correção do ponto de uma quina.

pode-se alinhar novamente com a parede.

Módulo Gerenciador do Mapa de Partições

O módulo gerenciador do mapa de partições irá armazenar as partições (ocupadas ou vazias) encontradas no ambiente na forma de polígonos, representados por uma lista de vértices. Entre esses polígonos, o espaço livre (partições vazias) será preenchido dinamicamente por uma série de retângulos de diversos tamanhos, representando regiões não ocupadas ou desconhecidas. Esse módulo está sujeito à restrição de só aceitar polígonos (objetos do ambiente) cujas arestas estejam todas alinhadas com o eixo horizontal ou vertical do sistema de coordenadas cartesianas adotadas. A Figura 12.4 ilustra um mapa típico com retângulos representando partições vazias (tons claros) e polígonos representando partições ocupadas e o contorno do mapa (em preto).

Os retângulos gerados dinamicamente para as partições vazias têm por função facilitar a implementação dos algoritmos de navegação do robô. Junto a cada vértice de um dado polígono, é armazenado o erro que o robô acumulou desde que encontrou o primeiro vértice do dado polígono.

Além de gerar e manter essa representação do ambiente em exploração, o módulo é responsável por produzir vistas diferentes do ambiente, conforme outros módulos necessitem. O módulo selecionador de partição, por exemplo, precisa de uma lista das partições vazias para trabalhar. Por sua vez, o módulo planejador de rotas precisa de um grafo que represente as partições vazias e suas conexões.

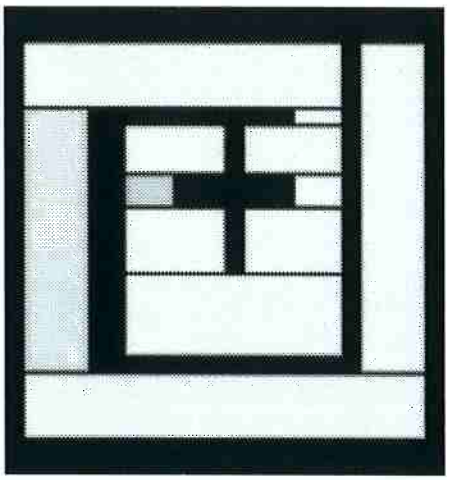


Figura 12.4: Um ambiente com um objeto (polígono preto) e partições não ocupadas (retângulos em tons claros).

Módulo Selecionador de Partição

O módulo selecionador de partição implementa uma função que mapeia um escalar para cada partição existente. A função é atualizada cada vez que uma partição vazia é explorada e cada vez que um novo objeto é encontrado (o que resulta em novas partições criadas e algumas antigas apagadas). O valor da função é calculado por:

$$valor = \frac{logArea^2 \cdot timeFromVisit \cdot internalValue}{numVisits}, \qquad (12.1)$$

onde $logArea$ é o logaritmo da área da partição — foi usado o logaritmo da área para evitar um valor muito grande para partições extensas; $numVisits$ é o número de visitas que a partição já recebeu — é inicializado em um e é incrementado cada vez que a partição é visitada; $timeFromVisit$ é um valor que indica há quanto tempo a partição não recebe uma visita — o valor é incrementado cada vez que alguma outra partição é visitada, se esta partição for visitada, o valor é zerado; $internalValue$ é um valor interno para impor um critério de parada. É escolhida para ser explorada a partição com o maior valor dessa função. Se para todas as partições o valor for zero, então o mapa é considerado pronto.

Módulo Planejador de Rotas

O módulo planejador de rotas deverá criar uma lista de pontos formando uma rota da partição atual até a partição desejada. Para isso, primeiramente ele irá selecionar uma seqüência de partições vazias adjacentes, com base no algoritmo de caminho de menor custo (busca em árvore), sendo, a cada partição, associado um custo (logaritmo de sua área). Após obter uma lista de partições é calculado um caminho ponto a ponto entre partições desejadas. O robô deve sempre navegar seguindo a orientação de algum eixo de coordenada, para minimizar os erros de *dead reckoning*. A Figura 12.5 mostra um ambiente e um caminho para partir do canto inferior esquerdo e atingir o canto superior direito.

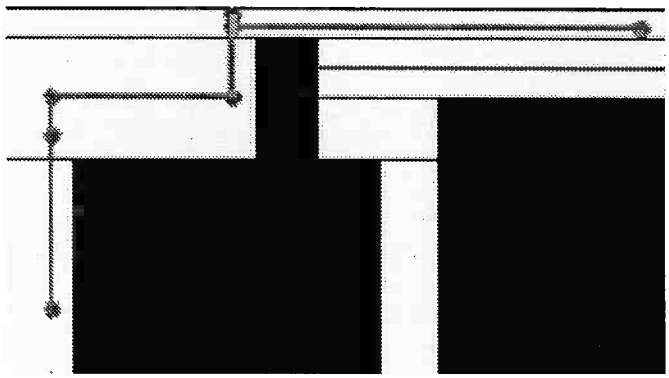


Figura 12.5: Caminho entre duas partições.

Módulo de Execução de Rotas

O módulo de execução de rotas recebe o caminho a ser percorrido como uma lista de pontos, na qual, além da informação das coordenadas, há informações sobre o erro do ponto e informações dos objetos ao redor do ponto. Para percorrer este caminho o robô anda, um a um, os pontos listados (na ordem passada) até o último deles.

Durante este percurso, a informação de quatro sonares é utilizada: os dois frontais e os dois laterais, conforme mostra a Figura 12.6. Estes são utilizados para verificar se existe algum objeto durante o percurso. Se algum objeto for encontrado, é preciso verificar se este objeto existe no mapa. Esta verificação é feita diretamente com o Módulo Gerenciador do Mapa de Partições.

Se um objeto desconhecido aparecer no caminho, é necessário avisar o Módulo

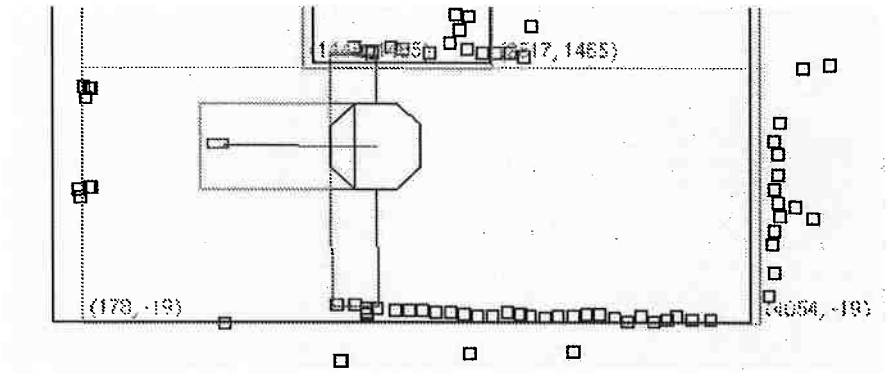


Figura 12.6: Robô executando uma rota.

de Identificação de Formas e interromper a execução da rota. Isto ocorre porque o mapeamento do novo objeto muito provavelmente vai alterar o caminho que deve ser percorrido (e com certeza vai alterar a posição do robô).

Módulo de Exploração Exaustiva

Finalmente, o módulo de exploração exaustiva deverá criar uma rota na partição atual de modo que toda a partição seja varrida pelo robô. Para isso, nenhum ponto da partição pode ficar a uma distância maior que 2 metros devido ao alcance do sonar do robô.

A rota a ser criada é composta de uma seqüência de voltas retangulares na partição, formando uma espécie de caracol, com a restrição de manter um vértice fixo, escolhido como sendo uma quina de algum objeto conhecido, para minimizar os erros de *dead reckoning*.

12.1.3 Resultados e Discussões

Os testes do sistema consistiram em comandar o robô Pioneer para mapear um ambiente desconhecido. Nas Figuras 12.7, 12.8 e 12.9 estão reproduzidos alguns resultados obtidos. Os dois primeiros foram obtidos utilizando o simulador do Pioneer, enquanto que o último foi obtido a partir de um ambiente real. Nestas Figuras, as linhas escuras representam a posição real do objeto e as linhas claras, a posição encontrada na construção do mapa. O robô nas Figuras indica sua posição ao iniciar o mapeamento do ambiente.

Os resultados alcançados foram bastante motivadores. Grande parte dos erros

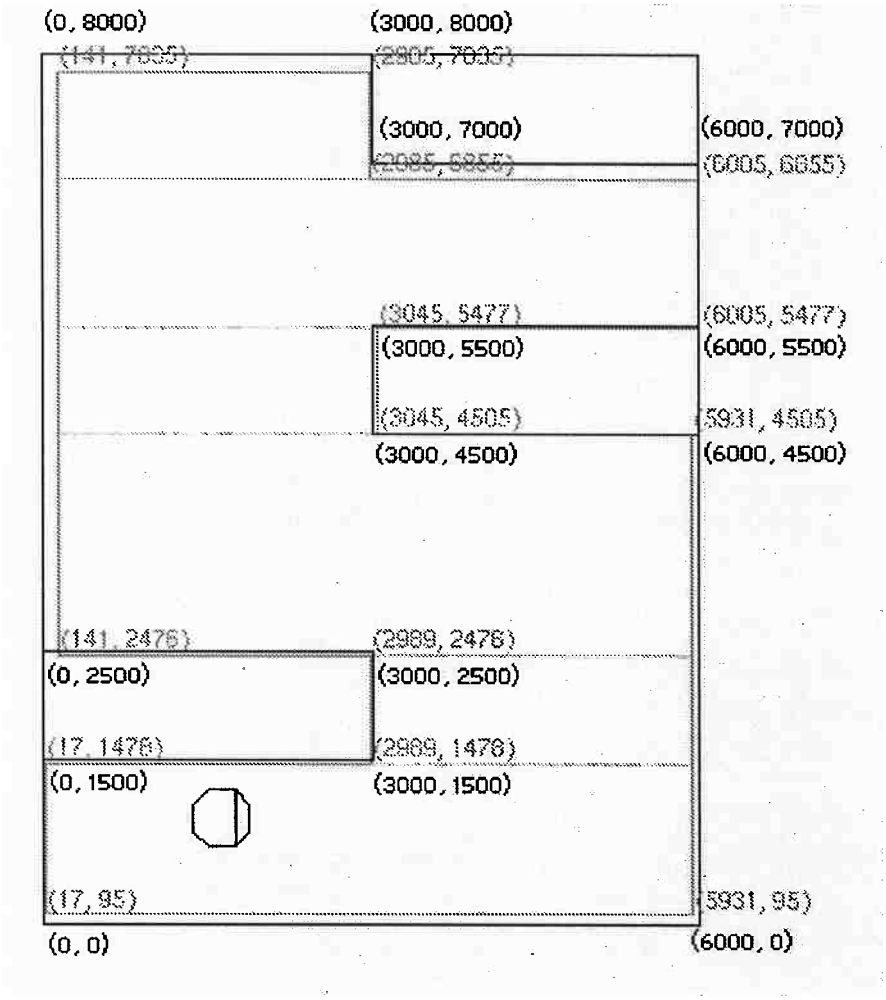


Figura 12.7: Resultado de mapeamento de um ambiente no simulador do Pioneer.

é devida à imprecisão do sistema de odometria do robô. Espera-se conseguir, em trabalhos futuros, resultados significativamente melhores com a introdução de visão computacional nos robôs Pioneers, quando em tarefas de construção autônoma de mapas de ambientes, explorando o uso de marcos visuais de forma a possibilitar a construção de mapas de ambientes mais complexos.

12.2 Localização Markoviana Com o Uso de Sonares

Uma habilidade extremamente necessária em tarefas de navegação consiste em o robô móvel ser capaz de se localizar, determinando sua posição dentro de um sistema de referências adotado. A localização pode ser relativa ou absoluta —

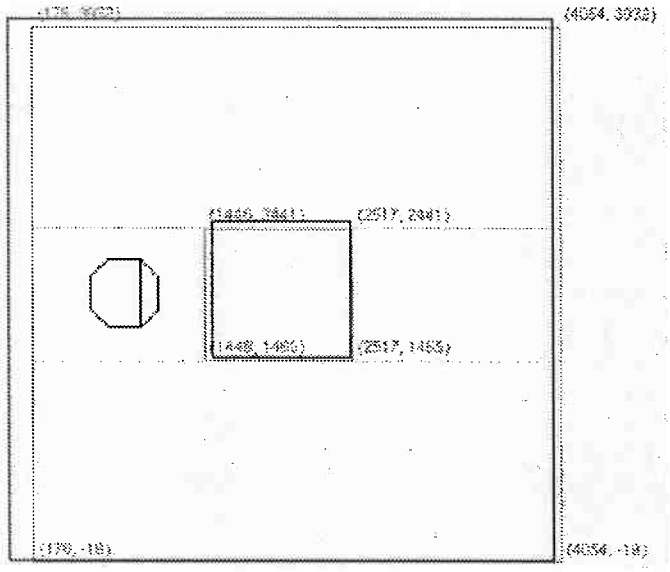


Figura 12.8: Resultado de mapeamento de outro ambiente no simulador do Pioneer.

também chamadas de local e global, respectivamente.

A localização relativa assume que o robô tenha conhecimento de sua posição inicial e rastreie sua posição ao longo de seu percurso, tentando apenas compensar os erros odométricos. Já a localização absoluta fornece a posição do robô no ambiente de forma global. A localização absoluta dá ao robô maior autonomia, pois sua posição inicial não precisa ser conhecida além de tornar a navegação mais segura, pois o robô é capaz de se recuperar de falhas.

O trabalho da autora e seu grupo neste tópico consiste em investigar a abordagem da localização absoluta, por meio do algoritmo de localização markoviana, que dá ao problema da localização uma abordagem probabilística capaz de integrar o mundo real, imprevisível e cheio de incertezas, com os métodos computacionais, permitindo ao robô, desta forma, lidar com ambigüidades quanto a sua localização, além de facilitar a modelagem de erros e imprecisões encontrados nos sensores e atuadores.

A idéia básica desse algoritmo pode ser explicada pela Figura 12.10. Nela pode ser visto que, partindo de uma distribuição inicial, a probabilidade de localização nas regiões onde as leituras são mais prováveis aumenta a cada leitura dos sonares, e a cada movimento do robô essa distribuição é deslocada de acordo com a distância percorrida. Dessa forma é possível restringir as prováveis posições do

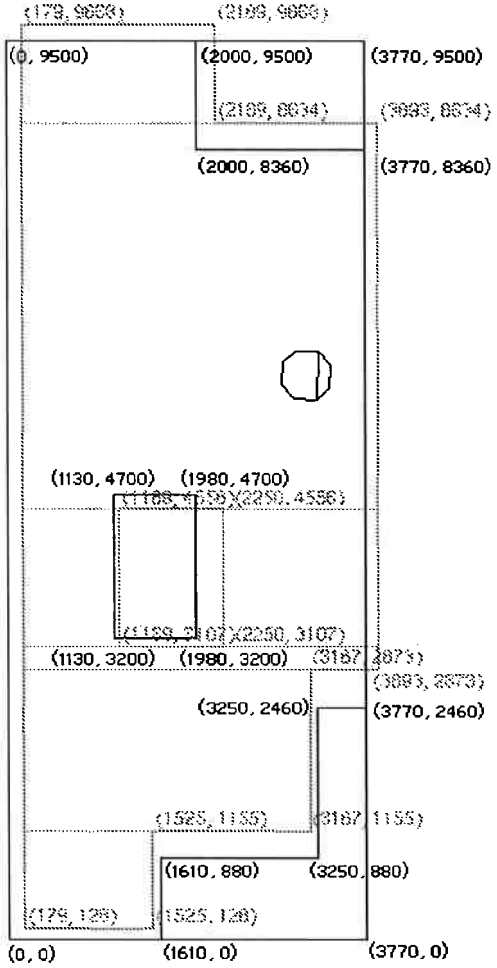


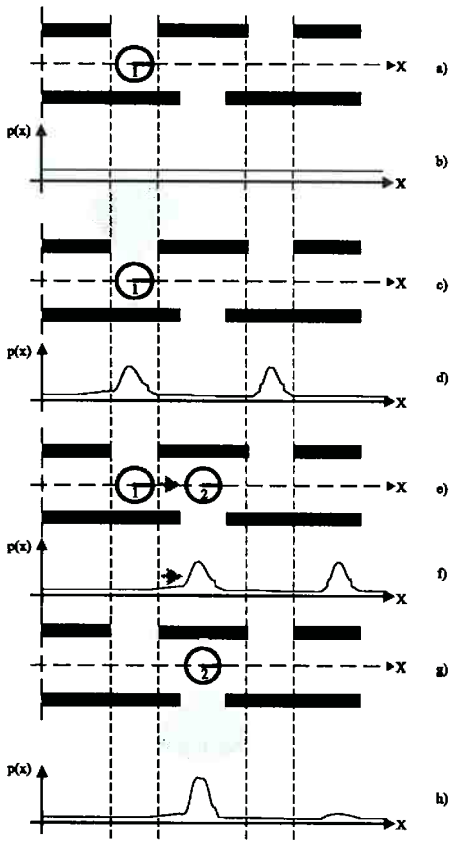
Figura 12.9: Resultado de mapeamento de um ambiente real.

robô a pequenas áreas e rapidamente determinar a sua posição. Esse algoritmo e suas variações têm sido extensamente aplicados, com sucesso (BURGARD, 1998b; BURGARD, 1998a; THRUN, 1999; KAWAI; COSTA, 2002).

Alguns conceitos básicos muito importantes, utilizados no algoritmo de localização markoviana, envolvem noções do teorema da probabilidade total, a regra de Bayes, processos markovianos e os processos parcialmente observáveis de decisão markoviana (*Partially Observable Markov Decision Processes* – POMDP).

O Teorema da Probabilidade Total e a Regra de Bayes encontram-se descritos, de forma resumida, no Anexo A.

Considere um exemplo em que o robô quer determinar se uma porta está aberta, dado que ele obteve de seus sensores a leitura de que não existem obstáculos. Para resolver esse problema, suponha que existam apenas dois estados possíveis



a)ambiente de corredor onde o robô só se desloca na direção de x; b) densidade de probabilidades inicial; c) observação feita na posição 1; d) densidade após observação em 1; e) deslocamento de 1 para 2; f) densidade deslocada de 1 para 2; g) observação feita em 2; h) densidade após observação feita em 2.

Figura 12.10: Exemplo de localização markoviana.

para a porta, que são: aberta (a) ou fechada (f). Além disso, suponha que os sensores do robô só possam retornar duas leituras que são: com obstáculo (c) ou sem obstáculo (s). Então, no exemplo, o que se quer determinar é $P(a|s)$, que é a probabilidade de que a porta esteja aberta dada a leitura, sem obstáculo. Para isso, basta que se tenha $P(s|a)$ e $P(s|f)$, que são, respectivamente, as probabilidades de que se obtenha a leitura, sem obstáculo, dado que a porta está aberta e que a porta está fechada. À primeira vista, isso não parece ter solucionado o problema, no entanto, $P(s|a)$ e $P(s|f)$ podem ser pré-determinadas experimentalmente ou através da modelagem dos sensores. Na verdade, é preciso também ter conhecimento da probabilidade $P(a)$ de que a porta esteja aberta, mas ela pode ser deduzida teoricamente.

Nesse trabalho, no entanto, a regra de Bayes é aplicada a múltiplas variáveis

aleatórias, em uma forma um pouco diferente.

Seja $\mathbf{X} = X_1, \ldots, X_n$ um vetor aleatório, isso é, cada X_i é uma variável aleatória, e Y uma outra variável aleatória, então,

$$P(Y|X_1, \ldots, X_n) = \frac{P(X_n|Y, X_1, \ldots, X_{n-1})}{P(X_n|X_1, \ldots, X_{n-1})}, \tag{12.2}$$

é a probabilidade condicional de Y dado o vetor $\mathbf{X}$.

Outro importante conceito utilizado é o de processos markovianos. Processos markovianos são processos estocásticos em que a probabilidade de um evento futuro, só depende do evento atual, não importando os eventos passados. Isso pode ser expresso matematicamente da seguinte maneira. Seja $X(t)$ um processo estocástico, para que ele seja também um processo markoviano, é preciso que, para qualquer n e $t_1 < t_2 \ldots < t_{n+1}$, tenha-se:

$$P[X(t_{n+1})|X(t_n), \ldots, X(t_1)] = P[X(t_{n+1})|X(t_n)] \tag{12.3}$$

Por outro lado, um POMDP é usado para modelar problemas em que um agente realiza ações sobre um ambiente que está em um determinado estado, levando-o a outros estados, mas tais estados não podem ser determinados com exatidão, pois só podem ser estimados indiretamente através de observações imprecisas ou incompletas.

Um POMDP é definido pela tupla $\langle S, A, Z, T, R, O \rangle$ em que:

- S é um conjunto finito de estados do mundo $S = s_1, \ldots, s_n$;
- A é um conjunto finito de ações possíveis $A = a_1, \ldots, a_n$;
- Z é um conjunto finito de observações possíveis $Z = z_1, \ldots, z_n$;
- $T : S \times A \rightarrow \Pi(S)$ é um modelo de transição de estados que atribui para cada estado a probabilidade de que esse seja alcançado, dado um par estado/ação;
- $R : S \times A \rightarrow \Re$ é a função recompensa que retorna ao robô um número real que representa a recompensa por realizar uma determinada ação em um determinado estado;
- $O : S \times A \rightarrow \Pi(Z)$ é uma função de observação que atribui a cada elemento

de Z a probabilidade de que esse seja observado após a execução da ação $a \in A$ em $s \in S$.

No contexto da navegação, pode-se supor que cada posição possível do robô seja um estado do ambiente — a partir deste ponto, deve-se entender posição ou localização como uma tripla $\langle x, y, \theta \rangle$ que representa as coordenadas x e y do robô e sua orientação θ — dessa forma, o problema é encontrar uma ação, ou conjunto de ações, que o levem à posição desejada. Mas é preciso lembrar que o robô é incapaz de saber sua posição exata, então ele tenta estimá-la a partir da leitura de seus sensores.

Em (KAELBLING; CASSANDRA; LITTMAN, 1994) o problema de agir em ambientes parcialmente observáveis é dividido conforme a Figura 12.11. O componente chamado "SE" (*State Estimator*) é o avaliador de estados, responsável pela avaliação ou estimação do estado atual dadas a avaliação anterior b, a observação s mais recente e a última ação a realizada. O outro componente é a política, assinalada com π (na Figura 12.11), que dado a crença no estado atual, é responsável por decidir as ações a serem realizadas. Além disso, são introduzidos, um conjunto $B = b_1, \ldots, b_n$ de estados de crenças e uma função $b(s_i)$ que atribui a cada estado de crença b_i a probabilidade de que s_i seja o estado atual do mundo.

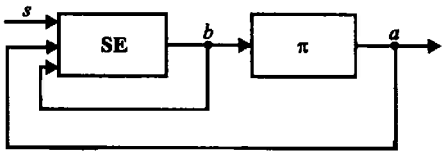


Figura 12.11: Um controlador POMDP.

É possível obter um avaliador de estados pela simples aplicação do teorema de Bayes, da seguinte forma:

$$P(s'|a,z) = \frac{P(z|s',a) \cdot P(s',a)}{P(z|a)}. \tag{12.4}$$

Aplicando a lei da probabilidade total,

$$P(s'|a,z) = \frac{P(z|s',a) \cdot \sum_{s \in S} P(s'|a,s) \cdot P(s)}{P(z|a)}. \tag{12.5}$$

Denomina-se $SE_{s'}(b,a,z)$ a probabilidade de estar no estado s' dados a última ação a e a mais recente observação z (ou $P(s'|a,z)$). $P(s'|a,s)$ é a função de

transição de estados $T(s, a, s')$, $P(s)$ é a função $b(s)$ e finalmente $P(z|a)$ é o termo de normalização. O que resulta na seguinte equação de atualização:

$$SE_{s'}(b, a, z) = \frac{O(a, s', z) \cdot \sum_{s \in S} T(s, a, s') \cdot b(s)}{P(z|a)}. \tag{12.6}$$

Essa equação deve ser aplicada a cada estado do espaço de estados e, dessa maneira, o *avaliador de estados* retorna um vetor contendo as crenças b_i para todos os estados s_i.

Não será explicado aqui o componente referente à política, pois, nesse trabalho, não se está interessado no modelo completo de POMDP, apenas no avaliador de estado, já que o enfoque é na localização, que nada mais é do que a determinação do estado do robô.

12.2.1 Localização Markoviana

O algoritmo de localização markoviana apresentado em diversos trabalhos anteriores (NOURBAKHSH; POWERS; BIRCHFIELD, 1995; SIMMONS; KOENIG, 1995; BUGARD, 1996; FOX, 1998; FOX; BURGARD; THRUN, 1999; KAWAI; COSTA, 2002) é uma aplicação direta do avaliador de estados descrito. Pode-se afirmar que o problema da localização pode ser enquadrado na estrutura da POMDP à medida que a posição do robô pode ser representada por um vetor aleatório de três dimensões $\langle x, y, \theta \rangle$, em que cada posição representa um estado e tais estados não podem ser determinados diretamente, apenas observados através de leituras feitas pelos sensores. E finalmente, as funções de transição de estados podem ser obtidas a partir de modelagem probabilística dos movimentos realizados pelo robô.

No caso da localização markoviana, será representado por L o espaço de todos os possíveis estados ou posições. O espaço de crenças continua a ser representado por B, em que a cada elemento b_l é atribuída a probabilidade de um determinado l ser a posição atual do robô. $Bel(L_t = l)$ é a função que mapeia L em B. As informações provindas dos sensores s_t são as observações sobre os estados e os movimentos realizados pelo robô são as ações a_t. Dessa forma tem-se um avaliador de estados que recebe a leitura mais recente s_t , a última ação a_{t-1} e a crença anterior $Bel(L_{t-1} = l)$ e retorna uma nova crença $Bel(L_t = l)$ como mostra a

equação a seguir, obtida do modelo de avaliador de estados:

$$Bel(L_t = l|s_{1,\ldots,t}, a_{1,\ldots,t-1}) = \frac{P(s_t|L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \cdot P(L_t|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}}{P(s_t|s_{1,\ldots,t-1}, a_{1,\ldots,t-1})}. \tag{12.7}$$

A seguir, cada termo da Equação 12.7 é analisado separadamente.

Modelo de Ação

O termo $P(L_t = l|s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ da Equação 12.7, descreve a probabilidade de se estar na posição l logo após a execução da ação a_{t-1}. Pode-se aplicar a Lei da Probabilidade Total e a propriedade de Markov para obter a função de transição de estados do modelo de avaliador de estados, como segue. Aplicando a Lei da Probabilidade Total, vem:

$$\begin{aligned} P(L_t = l|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) &= \sum_{l' \in L} P(L_t = l|L_{t-1} = l', s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \times \\ & \quad P(L_{t-1} = l'|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \end{aligned} \tag{12.8}$$

e depois, a propriedade de Markov:

$$\begin{aligned} P(L_t = l|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) &= \sum_{l' \in L} P(L_t = l|L_{t-1} = l', a_{t-1}) \times \\ & \quad P(L_{t-1} = l'|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}), \end{aligned} \tag{12.9}$$

onde $P(L_t = l|L_{t-1} = l', a_{t-1})$ é a função de transição de estados ou aqui chamado de *modelo de ação do robô* e $P(L_{t-1} = l'|s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ é a função $Bel(L_{t-1} = l')$.

Modelo de Percepção

$P(s_t|L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ modela implicitamente o mapa do ambiente, pois representa a leitura que seria esperada em cada posição dentro dele. Ele é freqüentemente chamado de *modelo dos percepção.* Pode-se aplicar a propriedade de Markov para simplificar o termo, obtendo:

$$P(s_t|L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) = P(s_t|L_t = l). \tag{12.10}$$

Normalização

O denominador da Equação 12.7, $P(s_t|s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$, serve para garantir que, ao se somar os valores de $Bel(L_t)$ de todas as posições do ambiente, essa soma não ultrapasse o valor 1.

Aplicando o Teorema da Probabilidade Total, vem:

$$P(s_t|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) = \sum_{l \in L} P(s_t|L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \times P(L_t = l|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \quad (12.11)$$

e, com a propriedade de Markov, resulta:

$$P(s_t|s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) = \sum_{l \in L} P(s_t|l) \cdot P(L_t = l|L_{t-1}, a_{t-1}). \quad (12.12)$$

12.2.2 Avaliador de Estados

Finalmente pode-se escrever um algoritmo para o avaliador de estados da maneira descrita no Algoritmo 2.

para cada posição de L **faça**
  se uma ação for executada: **então**
    - usar o modelo de ação:
    $P(L_t|L_{t-1}, a_{t-1}) \leftarrow \sum_{l' \in L} P(L_t = l|L_{t-1} = l', a) Bel(L_{t-1} = l')$
  fim-se
  se uma leitura for recebida: **então**
    - usar o modelo de sensor:
    $Bel(L_t = l) \leftarrow P(s|l) P(L_t|L_{t-1}, a_{t-1})$
  fim-se
  - normalizar:
  $Bel(L_t = l) \leftarrow \frac{Bel(L_t = l)}{\sum_{l \in L} P(s_t|l) P(L_t = l|L_{t-1}, a_{t-1})}$
fim-para

ALGORITMO 2: Forma iterativa do avaliador de estados

Grades Probabilísticas de Posição

Utilizado por (BUGARD, 1996) as grades probabilísticas de posição, ao contrário de métodos baseados em mapas topológicos (NOURBAKHSH; POWERS; BIRCHFIELD, 1995), são capazes de explorar características do ambiente como a largura

de corredores ou a geometria de objetos. Além disso, facilitam a fusão sensorial, atributo esse que aumenta a confiabilidade das informações e reduz a influência de ruídos. As grades probabilísticas de posição são simplesmente matrizes de ordem três em que cada célula é associada à probabilidade de que essa célula se refira à posição e orientação atuais do robô como exemplifica a Figura 12.12. Logo, pode-se perceber que as grades probabilísticas conseguem acomodar o modelo de espaço de estados do POMDP, associando-se a cada célula um elemento do conjunto de posições L_t.

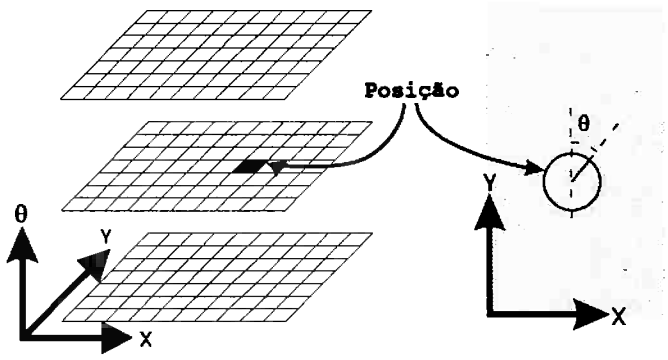


Figura 12.12: Grades probabilísticas de posição.

Os elementos da grade probabilística de posição são atualizados usando-se um *avaliador de estados*. Uma das grandes desvantagens dessas grades probabilísticas é o tamanho da matriz a ser mantida, sendo inviável a atualização de todos os elementos da matriz em tempo real, e portanto, é necessário o uso de ferramentas que possibilitem tal tarefa. Dentre essas ferramentas, as mais importantes são o modelo rápido de percepção e as técnicas de atualização local do espaço de estados. O modelo rápido de percepção será descrito em seguida; no entanto, as técnicas de atualização local ainda são temas de estudos da autora e seu grupo.

Modelo Rápido de Percepção

O modelo de percepção adotado é o modelo rápido de percepção de proximidade, introduzido em (BUGARD; FOX; HENNIG, 1997) e aplicado em (BURGARD, 1998b; BURGARD, 1998a; FOX, 1998; FOX; BURGARD; THRUN, 1999; THRUN, 1999). Este modelo é computacionalmente pouco custoso pois considera apenas a distância ao obstáculo mais próximo. Nessa abordagem, a distribuição de probabilidades inclui a modelagem dos obstáculos classificando-os da seguinte forma:

- Obstáculos conhecidos (ou previamente mapeados): aqui, pode-se associar as distâncias lidas pelos sensores a uma variável aleatória gaussiana com média igual à distância modelada do obstáculo e o desvio dependente da precisão e da acurácia do sensor. Assim, sejam a distância ao obstáculo mais próximo o_l, a distância medida pelo sensor d_i, e o desvio devido a imprecisões nos sensores σ, então pode-se escrever:

$$Pm(d_i|l) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \exp\left(-\frac{(d_i - o_l)^2}{2\sigma^2}\right), \tag{12.13}$$

  como sendo a probabilidade de se fazer a leitura d_i dado que o robô está na posição l.

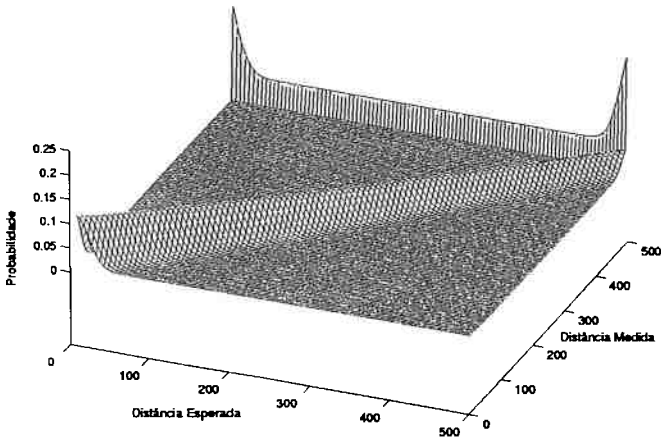
- Obstáculos desconhecidos (não inclusos no mapa): supõe-se que a probabilidade de encontrar um obstáculo desconhecido seja a mesma em qualquer posição visível ao sonar, isso é, existe uma distribuição uniforme de probabilidade, igual a cr, de se encontrar esse tipo de obstáculo. Então, pode-se escrever:

$$Pu(d_i) = cr(1 - \sum_{j<i} Pu(d_j)), \tag{12.14}$$

  em que, $(1 - \sum_{j<i} Pu(d_j))$ é a probabilidade de que não haja nenhum obstáculo antes de d_i.

Para aumentar a velocidade de atualização da grade de probabilidades, dadas as leituras feitas pelos sensores, é armazenada uma matriz cujas linhas representam as distâncias a um obstáculo e cujas colunas representam as possíveis leituras dos sensores. Nessa matriz, cada célula $a_{d,o}$ contém a probabilidade de se obter a leitura d_i dado que o obstáculo está a uma distância o_l. Assim pode-se obter $P(s_t|L_t = l)$ simplesmente consultando o valor dessa célula. A Figura 12.13 ilustra essa matriz.

Nesse ponto, é importante notar que a função densidade de probabilidade encontrada é unidimensional enquanto que na realidade os sonares detectam obstáculos dentro de uma região tridimensional. Apesar disso, o modelo é suficientemente preciso para dar ao robô condições de se localizar. Outros modelos mais complexos que levam em conta múltiplas reflexões e obstáculos bidimensionais (KONOLIGE, 1997) poderão vir a ser adotados futuramente, mas é necessário ter em mente que o modelo escolhido deve possibilitar o processamento dos dados



Matriz de consulta: Obtém-se a probabilidade $P(s_t|L_t)$ por simples consulta, cruzando-se a distância medida pelo sensor com a distância esperada pelo modelo do mapa.

Figura 12.13: Modelo rápido de percepção.

em tempo real.

12.3 Navegação direcionada por marcos de referência

Um marco é qualquer elemento do ambiente que sirva como ponto de referência para a localização do robô. Marcos podem ser artificiais ou naturais. Um marco artificial é aquele inserido no ambiente com a finalidade de ser reconhecido, sendo que sua utilização altera o ambiente — por exemplo, códigos de barras ou círculos concêntricos (ZIVOTÁ; FLUSSER, 1999). Um marco natural é um elemento do ambiente que não foi criado para o reconhecimento, mas que atende a este propósito. Por exemplo, luz do teto, paredes, portas, segmentos de linhas, bordas, cores e níveis de cinza.

Dentre os marcos naturais, existem aqueles dependentes de um ambiente particular como luz do teto ou portas, que são inúteis em ambientes que não possuam tais marcos. Tendo como objetivo a construção de robôs que operem de forma autônoma, busca-se por marcos que não dependam do ambiente.

Marcos podem ser extraídos por diversos sensores. Um dos mais flexíveis e

versáteis, com possibilidade de fornecimento de várias informações relevantes, por exemplo textura, formas, cores, etc., é a visão computacional (veja Capítulo 10). Assim, o uso de marcos visuais para tarefas de localização é altamente relevante.

Entretanto, reconhecer marcos visuais em uma seqüência de imagens é um problema desafiador, pois a aparência de qualquer dado marco varia de uma observação para outra. Além disso, variações devido a aspectos diferentes como mudanças na iluminação, oclusões parciais, imprecisão de sensores na formação de imagens, ruídos na captura, entre outros, são fatores que afetam os marcos observados.

DeSouza e Kak (2002) apresentam um apanhado geral sobre as técnicas de navegação de robô móvel utilizando visão, apontando várias aplicações que obtiveram sucesso ao empregar visão. O trabalho divide a navegação em ambientes internos e externos, e ainda, ambientes internos foram subdivididos em ambientes estruturados, ou seja, com mapa, e não estruturados, sem mapa.

A idéia de navegação baseada em mapa é fornecer um modelo do ambiente para o robô. Se o robô encontra um marco através de sua percepção do ambiente e estabelece uma correspondência entre o marco encontrado e um marco do mapa, então o robô está localizado em relação ao mapa. O mapa pode ser fornecido previamente para o robô. Caso isso não ocorra, o robô deve construí-lo buscando marcos no ambiente. O problema de busca por marcos é normalmente é conhecido por CLM – Mapeamento e Localização Concorrentes (*Concurrent Localization and Mapping*), ou por SLAM – Mapeamento e Localização Simultâneos (*Simultaneous Localization and Mapping*). Trabalhos relacionados com esse problema podem ser encontrados em (CHOSET; NAGATANI, 2001; DAVISON; MURRAY, 2002; THRUN; BURGARD; FOX, 1998). No mapeamento e localização simultâneos o robô corresponde a percepção atual do ambiente a suas observações passadas, que são o próprio mapa. Uma vez que o robô conhece sua posição correta em relação ao mapa, a percepção atual é adicionada ao mapa.

Na navegação sem utilização de mapa, o robô não utiliza uma representação explícita do ambiente, mas reconhece marcos encontrados no ambiente. Este tipo de navegação é conhecido também como método baseado em aparência. Neste método o dado do sensor é utilizado diretamente para formar uma função que mapeia o dado do sensor em uma posição no ambiente, ao invés de utilizar uma

representação do dado do sensor, como por exemplo atributos. Em tal técnica o robô armazena imagens do ambiente e associa essas imagens a posições. Após uma seqüência de imagens ser armazenada, quando o robô repetir a trajetória, o sistema deve comparar a imagem observada no momento com a seqüência de imagens armazenadas. Uma vez que uma imagem é selecionada como pertencente a um lugar, tem-se a estimativa de sua posição. Uma informação mais precisa sobre a posição pode ser obtida computando o deslocamento em pixels entre a imagem armazenada e a imagem vista, projetando esta diferença para o plano de referência de navegação.

Existem muitos marcos utilizados para localização de robôs móveis com diferentes técnicas. Surge então a questão: quais são os melhores marcos para localização? Algumas características desejáveis para marcos são: ser facilmente reconhecidos, ser percebidos de diferentes ângulos de observação e possuir atributos discriminantes.

12.3.1 Análise de Componentes Principais na Determinação de Marcos Visuais

Numa primeira investigação no tema, realizada pela autora e seu grupo, representou-se cada marco por uma cena, e um grupo de marcos similares define um local do ambiente. As cenas foram representadas através de PCA – Análise de Componentes Principais – e o reconhecimento é realizado por um classificador estatístico supervisionado, não paramétrico, que estima as funções densidade de probabilidade condicionais utilizando o estimador de Parzen (MARTINEZ; COSTA, 2002). Para testar a técnica descrita foram utilizadas imagens de três salas do LTI – Laboratório de Técnicas Inteligentes da EPUSP. Uma amostra das imagens utilizadas pode ser vista na Figura 12.14.

O sistema foi treinado para diferenciar os três locais. O treinamento foi realizado utilizando uma seqüência de vídeo obtida posicionando a câmera na entrada das salas e realizando uma rotação de aproximadamente 180 graus na câmera. As imagens da seqüência foram identificadas como pertencentes a cada um dos locais antes de iniciar o treinamento.

Para o reconhecimento, foram avaliadas as probalidades $P(C_i|\mathbf{x})$, onde $\mathbf{x}$ é o vetor de atributos de uma imagem que se quer identificar, obtido através de

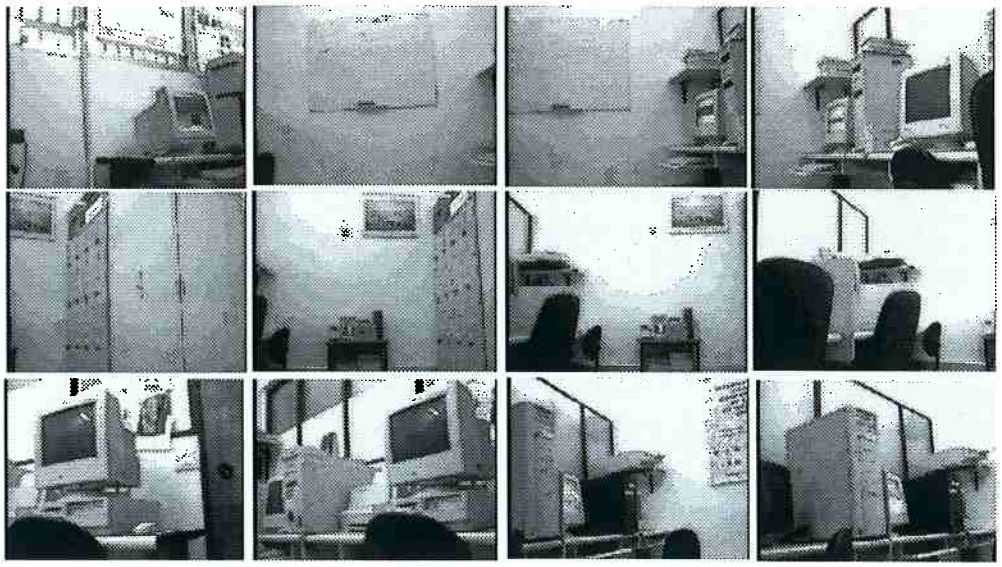


Figura 12.14: Cenas de 3 locais, sendo um local por linha, utilizadas no treinamento do classificador.

PCA, e C_i e cada uma das três classes possíveis (salas distintas do LTI), ou seja:

$$P(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i) \cdot P(C_i)}{\sum_{j=1}^{c} p(\mathbf{x}|C_j) \cdot P(C_j)}, \tag{12.15}$$

onde $p(\mathbf{x}|C_i)$ refere-se a função densidade de probabilidade condicional.

O estimador de Parzen foi utilizado para obter as densidades condicionais $P(\mathbf{x}|C_i)$. A probabilidade a priori de cada classe utilizada foi $P(C_i) = 1/3$, considerando que todas as classes são equiprováveis.

A classificação de uma imagem desconhecida é realizada atribuindo um rótulo para uma classe que possui a maior probabilidade $P(C_i|\mathbf{x})$, se essa probabilidade estiver acima de um limiar pré-definido.

Na estimação de Parzen, a função densidade de probabilidade $p(\mathbf{x})$ pode ser estimada a partir de uma função janela, que deve ser uma função densidade. Uma função janela amplamente utilizada é a gaussiana multivariada.

Assim, a estimação da função densidade de probabilidade $p(\mathbf{x})$ é uma média das densidades normais centradas nas amostras,

$$p(\mathbf{x}) = \frac{1}{t} \sum_{i=1}^{t} \frac{1}{(2\pi)^{d/2} h^d |\Sigma|^{1/2}} \times exp\left(-\frac{1}{2h^2}(\mathbf{x} - \mathbf{x}_i)^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}_i)\right), \tag{12.16}$$

onde h é a largura da janela e t é o número de amostras. A altura da janela h

deve satisfazer $\lim_{n\to\infty} h^t = 0$ e $\lim_{n\to\infty} th^t = \infty$ e para que a estimação de $p(\mathbf{x})$ possa convergir para $p(\mathbf{x})$.

12.3.2 Resultados e Discussões

As Figuras 12.15 e 12.16 ilustram imagens classificadas corretamente.

A seqüência de vídeo utilizada no treinamento possui 174 imagens para a classe 1, 119 para a classe 2 e 130 para a classe 3. Todas as imagens foram usadas para treinamento. Para testar o sistema foi utilizada uma nova seqüência de vídeo.

Nos testes realizados as imagens foram reduzidas para 32×24 pixels, a escolha desse tamanho de imagem foi necessário, pois com imagens maiores, o classificador tornou-se muito lento . Foram utilizados 15 atributos por imagem, o que corresponde a 80% da variância preservada. Variamos o parâmetro h do estimador de Parzen, entre 0.1 e 1. Os melhores resultados encontrados foram com $h = 0.25$. O resultado da classificação foi divido em classificação correta, incorreta ou sem classificação. Os resultados encontram-se na tabela abaixo.

classes	correta	incorreta	sem classificação
1	78%	19%	3%
2	55%	24%	21%
3	20%	71%	9%

Ao obsevar os dados da tabela é possível notar que a classe 3 não foi classificada corretamente em 71% das imagens de teste. Esse resultado foi atribuído ao fato que a aparência da classe 1 é muito semelhante a da classe 3, o que tornou incorreta a classificação. No entanto, um bom classificador depende da relação entre o número e qualidade de atributos utilizados e o número de amostras de treinamento. Dessa forma, acredita-se ser possível melhorar o classificador ajustando os parâmetros envolvidos e escolhendo outros atributos para serem utilizados em conjunção com o PCA.

Uma outra questão colocada no tema de aprendizado de marcos é: os robôs podem definir seus próprios marcos, de forma a melhorar a localização? Entre os trabalhos que tratam das questões levantadas está o de Thrun (1998), que propôs

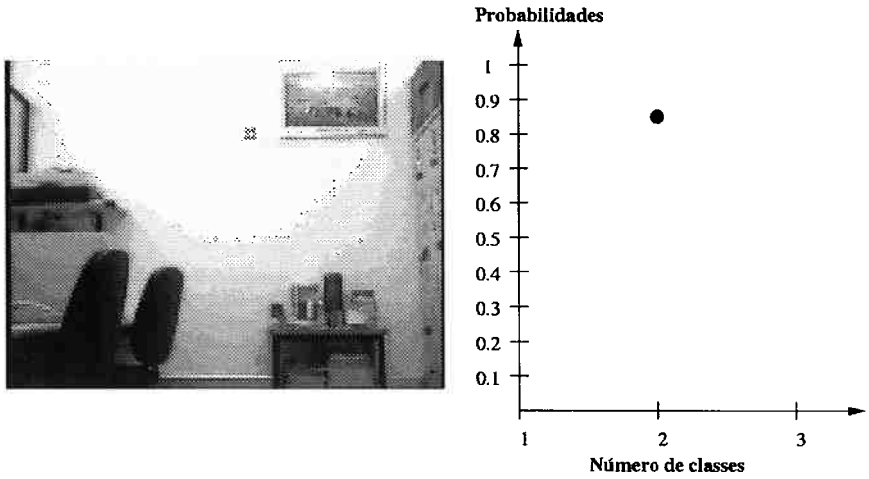


Figura 12.15: Resultado da classificação: uma imagem desconhecida, classificada como pertencente à classe 2.

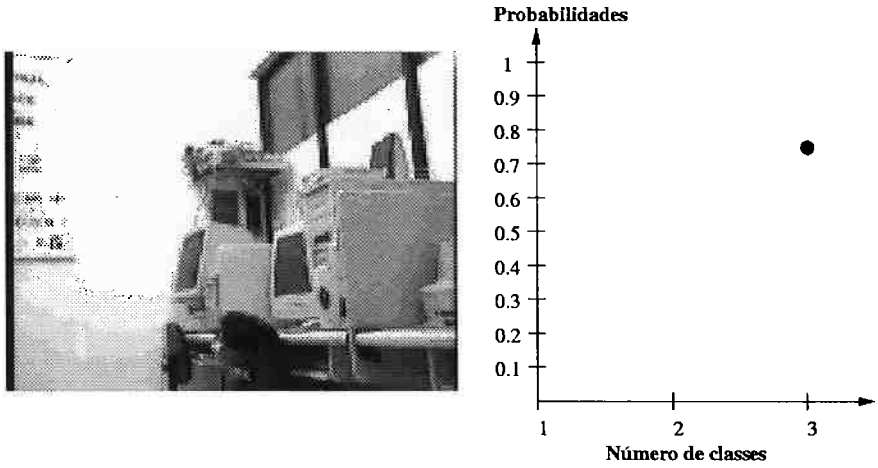


Figura 12.16: Resultado da classificação: uma imagem desconhecida, classificada como pertencente à classe 3.

um algoritmo para que o robô aprenda quais são os melhores marcos para localização. Os marcos são extraídos a partir de redes neurais que minimizam o erro bayesiano da localização. O fato de o robô aprender marcos torna-o mais flexível para adaptar-se a mudanças no ambiente e no sensor, e aumenta a autonomia do robô, uma vez que não é necessária a ajuda de um especialista humano para selecionar os marcos. Um resultado interessante do trabalho de Thrun é que os marcos que parecem apropriados para a orientação humana, não são necessariamente apropriados para robôs. Outros trabalhos relacionados com o aprendizado de marcos podem ser encontrados em (BORENSTEIN; EVERETT; FENG, 1996; GREINER; ISUKAPALLI, 1994).

A autora e seu grupo visam investigar esta área e desenvolver um sistema

de identificação de marcos que possa dotar um robô móvel com a capacidade de reconhecer por si só pontos de referência visual e utilizá-los como meio de estimar seu posicionamento em ambientes internos. O sistema deve ser capaz de estimar a localização do robô no caso em que não se conhece sua localização inicial e também encontrar o robô em caso de falha, em que sua localização é perdida.

A fase de aprendizado de marcos visuais explora alguns atributos da imagem, como cores, níveis de cinza, tipos e números de segmentos, segmentos paralelos, agrupamento de segmentos e bordas, pois acredita-se que estes atributos são aplicáveis a vários ambientes. O objetivo dessa fase é identificar automaticamente quais são os melhores atributos para estimar a posição do robô, buscando explorar representações e seleções eficientes de marcos e rápidas indexações de marcos selecionados.

Na fase de estimação da localização são investigadas técnicas probabilísticas, que têm apresentado bons resultados (FOX, 1998; THRUN, 1998; THRUN, 2000; KRÖSE, 2001; KAWAI; COSTA, 2002). Nesta abordagem são usados um modelo de ação e um modelo de percepção para interação do robô com seu ambiente. Com base no modelo de ação e percepção, espera-se obter a sobreposição das fases de aprendizado e localização da seguinte maneira. O robô recebe uma informação de percepção e atualiza sua localização. O robô executa uma ação e sua posição no ambiente muda, aumentando ou diminuindo a certeza da posição do robô. O valor desta mudança é informado ao robô por um sinal de reforço. O comportamento do robô deve ser escolher os marcos que tendem a aumentar, ao longo do tempo, uma função de soma dos valores do sinal de reforço, seguindo uma abordagem de aprendizado por reforço. A autora e seu grupo dirigem esforços no sentido de realizar esta proposta.

12.4 Aprendizado por Reforço no Controle

O aprendizado por reforço é extremamente útil na Robótica Móvel, pois as repetidas interações do robô com o ambiente permitem a modelagem do processo de decisão do robô como um Processo de Decisão de Markov – MDP (veja Seção 8.3), que pode ser solucionado com algoritmos de aprendizado por reforço.

Solucionar MDPs significa obter meios para escolher qual ação deve ser exe-

cutada em cada situação possível para obter o maior benefício do domínio, isto é, significa escolher políticas ótimas de atuação.

12.4.1 Generalização da Experiência

A obtenção de uma política ótima através do aprendizado por reforço requer um número muito grande de interações do robô com o ambiente, podendo sua aplicação se tornar proibitiva.

A autora e seu grupo têm conduzido pesquisas no sentido de buscarem métodos e técnicas que permitam a redução do tempo necessário ao aprendizado.

A proposta central consiste na generalização da experiência, uma abordagem na qual um única experiência do agente — isto é, uma única interação do robô com o ambiente, que consiste num único ciclo de iteração do algoritmo de aprendizado — pode atualizar um número maior de valor de custo, e não somente um único valor, como é tradicionalmente feito.

Assim, a conseqüência de executar a ação a_t no estado s_t é *espalhada* a outros pares (s, a) como se a experiência de fato ocorrida no tempo t fosse, na realidade, $\langle s, a, s_{t+1}, r_t \rangle$ (PEGORARO; COSTA, 2000; PEGORARO, 2001; PEGORARO; COSTA; RIBEIRO, 2001a; PEGORARO; COSTA; RIBEIRO, 2001b; RIBEIRO; PEGORARO; COSTA, 2002).

O Algoritmo QS-learning

Considere a seguinte variante do algoritmo Q-learning, dado pelo Algoritmo 3.

Sendo $V(s') = \max_a Q(s', a)$ e $\sigma(s, a, s_i, a_i)$, a *função de espalhamento* ($0 \leq \sigma(s, a, s_i, a_i) \leq 1$).

O algoritmo Q-learning padrão corresponde à Equação 12.17, com $\sigma(s, a, s_i, a_i) = \delta(s, s_i)\delta(a, a_i)$, onde $\delta(.,.)$ é a função delta Kronecker – isto é, $\delta(u, v) = 1$ se $u = v$, no caso contrário, $\delta(u, v) = 0$.

Um mecanismo de espalhamento baseado na função g de similaridade no espaço de estados pode ser definido como $\sigma(s, a, s_i, a_i) = g(s, s_i)\delta(a, a_i)$.

O algoritmo abaixo, chamado algoritmo QS-learning, tem sido estudado e

para cada instante t, o agente:
repita
visita o estado s;
seleciona e executa a ação a;
recebe o reforço $r(s,a)$ e observa o próximo estado s';
para todo estado $s_i \in \mathcal{S}$ e ação $a_i \in \mathcal{A}$ **faça**
atualiza $Q(s_i, a_i)$ segundo:

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\sigma(s, a, s_i, a_i)[r(s,a) + \gamma V(s') - Q(s_i, a_i)] \quad (12.17)$$

fim-para
até que a convergência seja atingida.

ALGORITMO 3: O Algoritmo QS-learning.

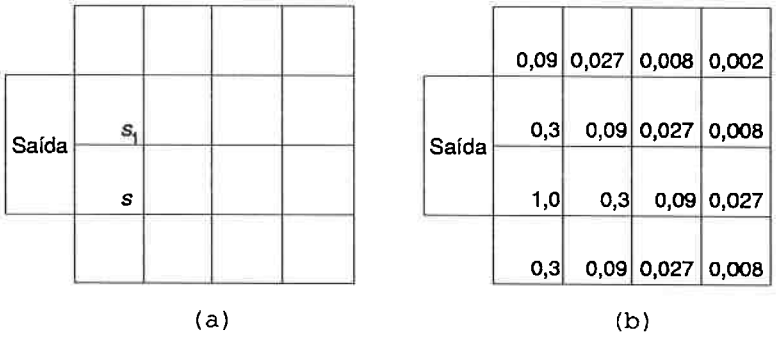
aplicado com sucesso (RIBEIRO; SZEPESVÁRI, 1996; PEGORARO, 2001; PEGORARO; COSTA, 2000; PEGORARO; COSTA; RIBEIRO, 2001a; PEGORARO; COSTA; RIBEIRO, 2001b).

Aplicação na Navegação de um Robô

O algoritmo QS-learning foi aplicado numa série de experimentos simulados de tarefas de navegação de um robô. O objetivo consiste em determinar, por aprendizado por reforço, a política de atuação que, no caso, corresponde a determinar a seqüência de ações que faz com que o robô realize a menor rota para atingir um alvo, partindo de qualquer posição dentro do espaço explorado durante o aprendizado. Em cada um dos casos testados considerou-se um agente (robô) capaz de identificar com precisão o seu estado atual.

A detecção de algumas similaridades, entre as possíveis, pode muitas vezes ser notada pela observação do ambiente. Por exemplo, considere a situação de um agente em uma sala em que ele precisa aprender a encontrar a saída – o alvo é a porta da sala. Considere também que o robô está no estado s onde uma determinada ação será executada, levando-o então imediatamente à saída e recebendo a recompensa máxima. Se isso ocorrer é certo que muitos dos estados vizinho, como apresentado na Figura 12.17(a), poderiam também conduzí-lo diretamente à saída (como o estado s_1) ou fariam parte de um bom caminho para encontrá-la. Na Figura 12.17(b) a função de similaridade foi definida como $g(s, s_i) = k^n$, onde k é uma constante (na Figura e nos testes, $k = 0,3$) e n é o número mínimo

de ações que devem ser realizadas pelo robô para que se desloque de s para s_i, sendo que o robô pode executar, a cada passo, uma das quatro ações disponíveis, correspondentes às quatro direções – Norte, Sul, Leste, Oeste.



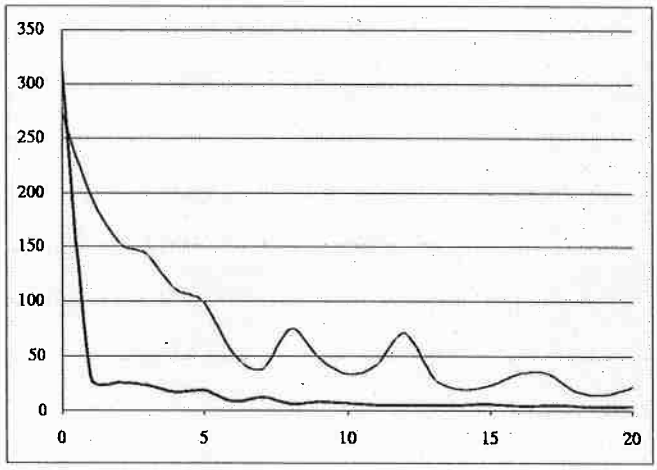
(a) A posição s do agente antes da ação que encontrará a saída. (b) Medidas de similaridade para o estado s na configuração mostrada em (a).

Figura 12.17: Exemplo de função de similaridade.

A verificação de que é possível utilizar essa abordagem para agilizar o aprendizado pode ser observada no gráfico da Figura 12.18. A linha fina representa o aprendizado com o Q-Learning tradicional, a linha grossa representa o aprendizado utilizando o algoritmo QS-Learning, utilizando similaridade entre os estados. O ambiente utilizado é o descrito no exemplo da Figura 12.17.

As curvas no gráfico representam a média em 40 séries, do número de passos de cada uma das 20 épocas consideradas. O número de passos de cada época é contado como o número necessário de ações para alcançar a saída, com o agente iniciando em uma posição aleatória da sala. Durante o aprendizado utilizou-se uma taxa fixa de 20% de exploração, $\gamma = 0,9$ e uma recompensa de 100 unidades, fornecida quando o robô chega à saída.

No início as duas curvas começam praticamente juntas, uma vez que nesse instante não existe nenhuma experiência anterior para indicar a qualidade de uma ação. Após o fim da primeira época, as duas implementações já encontraram a saída uma vez, implicando que o Q-Learning já atualizou o valor de um estado vizinho à saída para a ação que levou o agente para fora, enquanto o QS-Learning, nesse mesmo instante, já apresenta atualizações em todos os estados similares, além do estado vizinho à saída, para aquela ação. Desta forma, o aprendizado pode ser acelerado, como demonstra o gráfico apresentado na Figura 12.18.



Algoritmo Q-Learning (linha fina) e o algoritmo QS-Learning utilizando similaridades entre os estados (linha grossa). O eixo vertical representa o número de ações e o eixo horizontal, as épocas.

Figura 12.18: Comparação entre o algoritmo Q-Learning e o QS-Learning.

12.4.2 Agregação de Estados

A generalização da experiência está relacionada a métodos de agregação de estados (*state aggregation*) (TSITSIKLIS; ROY, 1996; TESAURO, 1995), no contexto de aproximação de funções para aprendizado por reforço — AR. Estes métodos consideram os efeitos de agregação adaptativa quando representações compactas são usadas.

No trabalho conduzido pela autora e seu grupo neste tema, buscou-se explorar o uso de redes neurais como aproximador da função valor, visando sua representação de forma compacta. É um fato conhecido que as redes neurais do tipo Perceptron Multi-Camada — MLP — treinadas pelo algoritmo de retropropagação de erros são aproximadores universais de funções contínuas (HAYKIN, 1999).

Os MLPs têm sido usados com sucesso em aplicações de AR que envolvem espaço de estados de dimensões elevadas, tais como o jogo de gamão (TESAURO, 1995), o controle de sistemas de elevadores (CRITES; BARTO, 1998), o jogo de futebol de robôs (SCÁRDUA; COSTA; CRUZ, 1999a; SCÁRDUA; COSTA; CRUZ, 1999b; SCÁRDUA; COSTA; CRUZ, 2000) e o controle de um desscarregador de navios (SCÁRDUA; CRUZ; COSTA, 2000; SCÁRDUA; CRUZ; COSTA,

2003b; SCÁRDUA; CRUZ; COSTA, 2003a). No estudo e desenvolvimento realizados, um algoritmo AR baseado em diferenças temporais (SUTTON; BARTO, 1998) com um aproximador de funções MLP de três camadas é utilizado para lidar com o enorme número de possíveis estados das aplicações implementadas.

Durante o processo de aprendizado, o algoritmo AR escolhe a ação que ele avalia como sendo aquela capaz de gerar a maior soma de recompensas, a longo prazo. Para cada possível par estado-ação, esta estimativa é traduzida por um número real chamado valor Q (veja Seção 8.3). Em aplicações com um número muito grande de estados, torna-se impraticável o uso de representações tabulares para armazenar os valores Q. A solução adotada utiliza um MLP para representar de forma compacta tais valores.

O MLP usado possui tantos neurônios na camada de entrada quantos forem o número de variáveis de representação de estados somado ao número de possíveis ações do aprendiz AR. O número de neurônios na camada escondida depende da aplicação e do número resultante de neurônios na camada de entrada. A camada de saída é composta por um único neurônio que efetua a soma de todas as saídas dos neurônios da camada escondida.

O aprendizado AR integrado ao aproximador MLP treinado com retropropagação de erro (veja Seção 8.1.3) é descrito pelo Algoritmo 4.

Aplicação no Controle de um Jogador de Futebol de Robôs

O algoritmo abaixo proposto foi utilizado com sucesso no controle de um jogador de futebol de robôs que deveria escolher as ações (de um nível de abstração mais alto) que resultariam nas maiores recompensas, dadas em função dos gols marcados (recompensas por gols marcados pelo aprendiz e punições por gols marcados pelo oponente). As ações eram:

Get-the-Ball

Pré-condições: estar numa distância de chute do jogador.
Efeitos: mover o jogador na direção da bola; ao alcançá-la, parar em posição de chute.

Kick-the-Ball

Pré-condições: estar em posição de chute.

repita
  visita o estado s;
  seleciona a ação a:
  se deve fazer exploração, **então**
    escolhe a ação a aleatoriamente;
  fim-se
  se deve fazer a exploitação, **então**
    para cada ação que pode ser executada no estado s **faça**
      usa o MLP para avaliar o valor $Q(s,a)_t$;
    fim-para
    escolhe aquela ação a que resultou no maior valor;
  fim-se
  executa a ação a escolhida;
  recebe o reforço $r(s,a)$ e observa o próximo estado s';
  atualiza por AR o valor $Q(s,a)_{t+1}$;
  adapta o MLP usando retropropagação de $Q(s,a)_{t+1} - Q(s,a)_t$;
até que a convergência seja atingida.

ALGORITMO 4: O Algoritmo AR com agregação de estados por MLP.

Efeitos: chutar a bola na direção do gol adversário.

Turn-the-Ball

Pré-condições: a bola deve estar em distância de chute do jogador, porém não se encontra alinhada com o jogador e o gol adversário.
Efeitos: mover a bola para que fique entre o jogador e o gol adversário, porém mantendo a distância de chute.

Os testes foram executados em um simulador com um campo discretizado em 105x68 células. As variáveis de estado foram: o ângulo entre a bola e o jogador aprendiz, a distância entre a bola e o aprendiz, o ângulo entre o oponente e o aprendiz, e a distância entre o oponetne e o aprendiz.

Os resultados — detalhados em (SCÁRDUA; COSTA; CRUZ, 1999a; SCÁRDUA; COSTA; CRUZ, 1999b; SCÁRDUA; COSTA; CRUZ, 2000) — foram bastante encorajadores, mostrando que o agente foi capaz de aprender rapidamente sua política de atuação no jogo de futebol e que o uso do MLP foi bastante apropriado — caso contrário, uma representação tabular com 3x105x68 deveria ser manipulada e, com o aumento do número de ações e do campo, esta se tornaria impraticável. Devido aos resultados animadores conseguidos, a autora e seu grupo resolveram aplicar esta abordagem para solucionar um problema real, descrito a seguir.

Aplicação no Controle de um Descarregador de Navios

Um problema importante nas operações de descarga de navios é a otimização do movimento entre o navio e a moega, respeitando restrições impostas pelos equipamentos e obedecendo condições de contorno específicas.

O descarregador de navios é basicamente um sistema carro-pêndulo em que o comprimento do pêndulo pode ser variado independentemente do movimento do carro. Uma caçamba localizada na extremidade do cabo é usada para transportar o material a ser descarregado. Sua dinâmica pode ser modelada como um sistema de equações diferenciais, envolvendo as seguintes variáveis: aceleração do carro, aceleração máxima do carro, magnitude da aceleração da gravidade local, comprimento atual do cabo, valor mínimo e máximo do comprimento do cabo, posição do carro, ângulo de inclinação do cabo em relação à vertical, velocidade de içamento da caçamba e velocidade do carro.

No problema, o ciclo de descarga foi dividido em seis fases: partida do navio, translação em direção à moega, chegada à moega, partida da moega, translação em direção ao navio e chegada ao navio.

Na partida do navio, inicialmente tanto o carro quanto a caçamba são supostos em repouso num dado ponto acima do navio; o comprimento do cabo neste instante é máximo. O içamento do cabo e o movimento do carro ocorrem simultaneamente a partir do instante inicial. O içamento da caçamba é feito a uma velocidade constante até que o comprimento mínimo do cabo seja atingido, definindo o tempo de duração desta fase — tempo total de içamento do cabo, com carro em movimento. O objetivo do sistema de aprendizado por reforço usado, com representação dos valores Q em MLPs, é maximizar a distância percorrida pelo carro durante o içamento da caçamba, admitindo-se que o intervalo de tempo determinado para esta fase seja suficiente para o carro atingir sua velocidade máxima. As seguintes restrições devem ser respeitadas: a aceleração do carro pode tomar apenas valores 0 ou 1, e a velocidade do carro não pode ser superior à velocidade máxima de movimento do carro.

Na fase de translação do carro em direção à moega o carro move se com velocidade máxima em direção à moega e o comprimento do cabo é mantido constante no seu comprimento mínimo. Uma vez que o movimento é retilíneo e

uniforme, a caçamba permanece em repouso com relação ao carro e a vertical.

Na fase da chegada à moega, supõe-se o carro inicialmente se movendo em direção à moega com velocidade máxima e a caçamba é admitida em repouso com relação ao carro na posição vertical. O objetivo da aprendizagem nesta fase é minimizar a distância de frenagem para parar o carro de maneira que, no final, a caçamba esteja em repouso na vertical. O comprimento do cabo é mantido no seu valor mínimo. A duração desta fase não é conhecida *a priori* e deve ser obtida pelo controlador aprendiz. As seguintes restrições devem ser respeitadas: a aceleração do carro pode tomar apenas valores 0 ou -1, e a velocidade do carro não pode ser inferior a 0.

As três últimas fase são similares às três primeiras. Na implementação realizada, com o propósito de acelerar o aprendizado, cada fase do problema é controlada por um sistema AR e um MLP treinado separadamente, o que significa que diferentes pesos são aprendidos para cada fase. Os MLPs usados possuem, cada um, 20 neurônios na camada escondida.

Uma contribuição muito importante na abordagem proposta foi a de realizar um procedimento de treinamento com níveis de dificuldade crescentes, inicialmente com objetivos relaxados (que, quando atingidos, fornecem recompensa ao aprendiz) e, conforme o aprendizado prossegue, estes objetivos vão se tornando cada vez mais exigentes, até atingirem o objetivo final do controlador. Este procedimento se justifica porque, ao estipular um objetivo rigoroso logo no início do aprendizado, o ambiente retorna punições mais severas pelos erros do controlador, o que pode fazer com que os pesos dos MLPs oscilem ou mesmo atinjam a saturação.

O sistema proposto produziu um semiciclo de 15,23 segundos (tempo para realizar três fases do sistema, de ida ou de volta), coincidindo com o valor obtido pela solução do problema por meio de técnicas da Teoria de Controle Ótimo (SCÁRDUA; CRUZ; COSTA, 2000; SCÁRDUA; CRUZ; COSTA, 2003a; SCÁRDUA; CRUZ; COSTA, 2003b). O trabalho desenvolvido mostrou assim que controladores baseados em AR podem ser utilizados de maneira efetiva em problemas reais de controle em que se dispõe de pouca ou nenhuma informação a respeito da dinâmica do sistema.

12.5 Discussões

Navegação, localização, construção e uso de mapas são tarefas essenciais em robôs móveis. Esforços continuarão sendo feitos pela autora e seu grupo para dominar adequadamente estas habilidades.

Outras técnicas de aprendizado, por exemplo, baseadas em algoritmos genéticos, devem ser exploradas para o controle da atuação dos robôs, assim como experimentações no ambiente real deverão ser estimuladas.

Um novo desafio a ser explorado consiste no estudo e desenvolvimento de técnicas de planejamento de trajetórias, para que o robô seja capaz de atingir um alvo, ao mesmo tempo que evita colisões e trata contingências imprevisíveis.

13 Múltiplos Robôs

As pesquisas e desenvolvimentos da autora com múltiplos robôs iniciaram com a construção dos times FUTEPOLI e GUARANÁ de futebol de robôs autônomos. Nestes times, os comportamentos, arquiteturas e políticas de atuação foram totalmente definidas nos respectivos projetos. No entanto, uma contribuição muito interessante realizada no time GUARANÁ foi o fato de haver a possibilidade, de forma independente, de troca de papéis entre o atacante e o defensor do time. Os comportamentos envolvidos em cada robô e a atuação do time como um todo estão descritos na Seção 13.1.

Por outro lado, as investigações conduzidas pela autora utilizando múltiplos robôs têm envolvido diversas áreas, buscando autonomia dos robôs através do uso de aprendizado autônomo. Desta forma, buscou-se *(i)* fazer com que robôs aprendessem o momento adequado de comunicarem entre si, quando atuando de forma coordenada (Seção 13.2); *(ii)* introduzir numa sociedade um agente capaz de aprender a realizar a coordenação entre os outros agentes da sociedade, visando alcançar uma meta comum de forma eficiente (veja Seção 11.4, descrita no Capítulo 11); e *(iii)* fazer com que robôs, quando atuando em ambiente de jogos, pudessem aprender de modo eficiente – através do uso de generalização da experiência – políticas adequadas de atuação (Seção 13.3).

Nesta última, as pesquisas conduzidas pela autora e seu grupo resultaram na proposta de um novo algoritmo, o Minimax-QS, com prova de convergência a valores ótimos, sob condições adequadas (veja Anexo B).

13.1 O Time Guaraná

O time GUARANÁ é um time de robôs heterogêneos, de granularidade três, com controle centralizado, em cooperação não ativa e sem comunicação entre si (veja Seção 7).

O time possui três estratégias de jogo: *(i)* se estiver ganhando com um bom saldo de gols, colocar todos os jogadores na defesa; *(ii)* se estiver perdendo por muitos gols, colocar todos os jogadores no ataque; *(iii)* se estiver ganhando ou perdendo por poucos gols, colocar um jogador na defesa e outro no ataque. A grande maioria dos jogos foi realizada utilizando a última estratégia, que possui um mecanismo interessante de troca de papéis entre o atacante e o defensor, descrito adiante (COSTA, 1999a; COSTA, 1999b; COSTA; PEGORARO, 2000).

13.1.1 Comportamentos do Time Guaraná

Foram definidos três comportamentos estratégicos que os robôs podem assumir de acordo com sua posição no campo: goleiro, defensor e atacante. Um robô fixo é designado para ser o goleiro; já os outros dois alternam de comportamento, de acordo com as circunstância do jogo, sempre mantendo um jogador como defensor e outro, como atacante. O campo é disposto longitudinalmente na imagem, sendo o sistema de coordenadas colocado no canto inferior esquerdo do campo, e, para fins de cálculos, o gol da direita é o gol do adversário.

Goleiro

O comportamento do goleiro está descrito pelo diagrama de transição de estados mostrado na Figura 13.1, o qual consiste de três estados: A_{gol}, que visa posicionar o goleiro no centro do gol; B_{gol}, que posiciona o goleiro na intersecção da direção da bola com a linha defensiva do gol; C_{gol}, que coloca o goleiro alinhado à coordenada y_B da bola (eixo Y de referência do campo).

O estado B_{gol} é o mais importante, pois ele é o responsável pelas defesas. Neste estado, o goleiro é colocado na linha defensiva (coordenada x_{gol} do campo) e na altura y_{inter} da estimativa da linha da direção da bola e a linha de defesa,

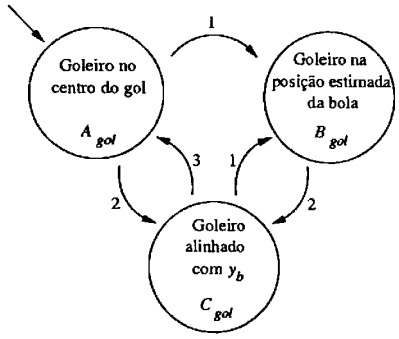


Figura 13.1: Comportamento do goleiro.

com:

$$y_{inter} = y_B - \frac{\Delta y_B \cdot (x_B - x_{gol})}{\Delta x_B} \tag{13.1}$$

sendo (x_B, y_B) as coordenadas da bola; $(\Delta x_B, \Delta y_B)$ os deslocamentos da bola em X e Y, observados nos últimos dois quadros de imagem analisados (veja Seção 10.5.2); x_{gol} a linha de defesa, onde o goleiro deve permanecer para não sair da área nem invadir o gol.

Nos três estados a abscissa do goleiro é sempre x_{gol}, já a ordenada nos estados B_{gol} e C_{gol} é calculada, porém sempre respeitando os limites do gol $y_{gmin} \leq y_{inter} \leq y_{gmax}$, com y_{gmin} e y_{gmax} sendo os limites inferior e superior do gol.

As transições de estado da Figura 13.1 ocorrem quando:

1. $x_B < d_1$ e $\Delta x_B < 0$;
2. $x_B < d_2$ e $\Delta x_B \geq 0$;
3. $x_B \geq d_2$;

sendo d_1 e d_2 limites estratégicos que definem se a bola está muito próxima do gol ou não.

Defensor

A posição alvo do defensor situa-se na coordenada y_B da bola, em uma faixa de defesa a uma certa distância do gol, bloqueando a bola para impedí-la de chegar ao gol – estado A_{def} – ou obstruindo um possível chute do oponente – estado B_{def}; porém, ele sai do caminho para permitir que o atacante do mesmo time conduza a bola ao gol adversário – estado C_{def}.

Quando o defensor e a bola estiverem numa configuração favorável com respeito ao gol adversário, o defensor assume o papel de atacante, obrigando o

atacante a passar a atuar como defensor. A configuração favorável é definida por uma área triangular de pequenas dimensões, dentro dos limites do campo, com vértice na posição da bola e base paralela á linha de defesa do gol do time, sendo que o defensor deve estar dentro desta área e com orientação voltada para o gol adversário – atrás da bola.

Atacante

O atacante pode estar em dois estados distintos: D_{atac}, que é o modo de condução, quando o robô se desloca e corrige sua trajetória para conduzir a bola ao gol adversário; E_{atac}, quando o robô procura se posicionar atrás da bola, na direção do gol adversário. A figura 13.2 ilustra o diagrama de transição de estados do atacante e do defensor, mostrando quando há a troca de papéis.

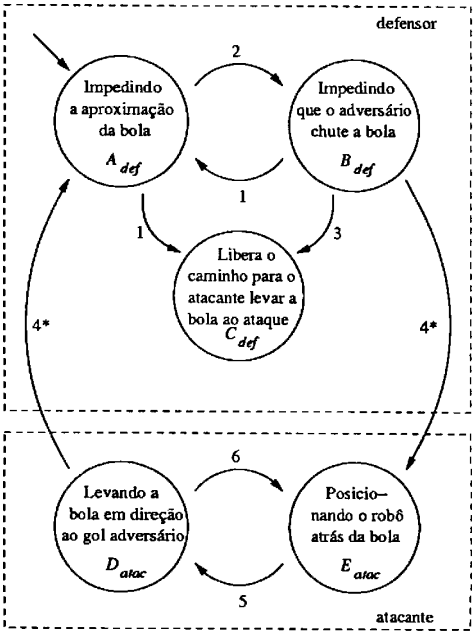


Figura 13.2: Comportamentos do defensor e do atacante.

As transições de estado da Figura 13.2 ocorrem quando:

1. $x_B - d_1 > x_d$;
2. $x_B + d_1 < x_d$;
3. $x_a < x_B < x_d$ e $y_a - y_B < d_2$ e $y_d - y_B < d_2$;
4. $x_B - x_d < d_3$ e $y_{gmin} \leq (y_B + ((y_d - y_B) \cdot (x_B - x_{gadv}))/(x_d - x_B)) \leq y_{gmax}$ – nesta transição ocorre troca de papéis entre atacante e defensor;
5. $x_B - x_a < d_3$ e $y_{gmin} \leq (y_B + ((y_a - y_B) \cdot (x_B - x_{gadv}))/(x_a - x_B)) \leq y_{gmax}$;

6. $x_a - x_B > d_4$;

sendo (x_B, y_B) as coordenadas da bola; (x_a, y_a) as coordenadas do atacante; (x_d, y_d) as coordenadas do defensor; y_{gmin} e y_{gmax} os limites inferior e superior do gol; x_{gadv} a coordenada da linha de gol do campo adversário; e $d_1 \ldots d_4$ valores determinados empiricamente (foram usados 4, 20, 4 e 8 cm, respectivamente).

O time GUARANÁ, desenvolvido sob a coordenação e participação da autora, conquistou o título de vice-campeão mundial na categoria de micro-robôs do Micro-Robot Soccer Tournament – Mirosot, nos jogos da Federation of International Robot-Soccer Association – FIRA, em Paris, França, julho de 1998. O troca de papéis entre o atacante e o defensor do time foi um diferencial, que agilizou em muito os jogos, aproveitando situações favoráveis para a inversão de comportamentos.

13.2 Aprendizado por Reforço em Cenários Multi-Agentes

A autora e seu grupo iniciaram o uso de aprendizado por reforço – AR – em cenários com mais de um robô com um trabalho de uso do algoritmo Q-learning aplicado individualmente a cada robô de um time, porém com retorno de reforço global (COSTA; VELOSO, 2000).

Os testes foram realizados num simulador (campo discretizado 10x10), com um time composto por dois robôs homogêneos, estrutura de controle descentralizada, cooperação ativa e comunicação explícita, porém controlada. Os robôs eram dispostos aleatoriamente no campo e deveriam atingir um alvo comum, desconhecido por ambos.

A intenção era que aprendessem a dividir tarefas (por exemplo, áreas de busca no campo) e que aprendessem o momento apropriado de comunicar uns cons os outros – por exemplo, no momento em que um deles encontrasse o alvo, para direcionar a ida do outro ao alvo e finalizar a tarefa conjunta.

Os experimentos apresentaram bons resultados, mas os robôs demoravam excessivamente para aprender a tarefa. Assim, esforços da autora e seu grupo foram direcionados para formas de agilizar os algoritmos de AR, utilizando abordagens

distribuídas de AR agregadas ao uso de heurísticas (veja Seção 11.4), abordagens com agregação de estados (veja Seção 12.4.2) e generalização da experiência, tanto em cenários com um único agente (veja Seção 12.4.1) quanto em cenários multi-agentes, descrito a seguir.

13.3 Generalização da Experiência em Cenários Multi-Agentes

O uso de generalização da experiência (veja Seção 12.4.1) no aprendizado *on-line* e concorrente de políticas em cenários multi-agentes, utilizando algoritmos de AR, foi investigado pela autora e seu grupo.

Agentes aprendendo concorrentemente implicam num cenário não estacionário, uma vez que o reforço recebido por um agente (ao executar uma ação em um estado) depende do comportamento dos outros agentes.

Cenários não estacionários podem ser vistos como um jogo de dois jogadores, onde um agente e o outro jogador (que representa o ambiente e os outros agentes) selecionam ações de um conjunto de ações possíveis no estado corrente. Estas ações definem o próximo estado possível.

Um algoritmo AR que pode ser aplicado em tais cenários é o algoritmo Minimax-Q (LITTMAN, 1994), que é conhecido por garantir convergência ao equilíbrio, no limite. Entretanto, encontrar políticas ótimas de controle usando qualquer algoritmo AR (incluindo o (Minimax-Q) pode consumir muito tempo.

A autora e seu grupo têm investigado o uso de generalização de experiência para aumentar a taxa de convergência de algoritmos AR, investigação esta que resultou na proposta de um novo algoritmo, Minimax-QS, qe incorpora a generalização de experiência no algoritmo Minimax-Q.

Este novo algoritmo tem prova de convergência ao equilíbrio no limite, sob condições adequadas. A prova desta convergência se encontra no Anexo B deste texto.

13.3.1 Aprendizado Multi-Agentes

Neste item é primeiramente descrito o Jogo de Markov – MG (*Markov Game*), o qual pode ser visto como uma extensão dos Processos de Decisão de Markov – MDP – para múltiplos agentes. A seguir é apresentado o algoritmo Minimax-Q, que soluciona MGs. Minimax-Q é baseado no Q-learning (veja Seção 8.3.1) e no algoritmo Minimax, conhecido algoritmo usado para busca de soluções em jogos (RUSSELL; NORVIG, 1995).

Jogos de Markov

Considere n agentes interagindo no ambiente via percepção e ação. Em cada passo de interação, cada agente i sensoria o estado corrente s_t do ambiente e escolhe uma ação a_i para executar. O conjunto de ações $a_1, \ldots, a_n$ altera o estado s_t do ambiente e um escalar de sinal de reforço r_i (uma recompensa ou penalização) é fornecido para cada agente i, indicando a adequação do estado resultante.

Um MG é formalmente representado pela ênupla:

$$\langle n, S, A_1, A_2, \ldots, A_n, r_1, r_2, \ldots, r_n, P \rangle$$

(veja (LITTMAN, 1994)), onde: n é o número de agentes; S é o conjunto de estados; $A_1, \ldots, A_n$ é uma coleção de conjuntos A_i das possíveis ações do agente i; $r_i : S \times A_1 \ldots \times A_n \rightarrow \mathcal{R}$ é uma função escalar de reforço para o ith agente, $P : S \times A_1 \ldots \times A_n \rightarrow \Pi(S)$ é a função de transição de estado, onde um membro de $\Pi(S)$ é a distribuição de probabilidade sobre S. $P(s_{t+1}|s_t, a_1, \ldots, a_n)$ representa a probabilidade de transitar do estado s_t para s_{t+1} quando os n agentes executam respectivamente as ações $a_1, \ldots, a_n$ no estado s_t.

Minimax-Q

Considere uma especialização dos MGs onde dois agentes executam ações alternadamente, num jogo de soma zero. Seja A o conjunto de possíveis ações do jogador **A** e $\bar{A}$ o conjunto de ações do jogador oponente **B**. $r_{s_t,a_t,\bar{a}_t}$ é o reforço imediato recebido por **A** por executar a ação $a_t \in A$ no estado $s_t \in S$ quando seu oponente **B** executa a ação $\bar{a}_t \in \bar{A}$.

O objetivo de **A** é aprender uma política ótima de ações que maximize a soma cumulativa esperada dos reforços descontados (veja Seção 8.3). O aprendizado

desta política é muito difícil, uma vez que depende das ações executadas pelo oponente. A solução para este problema é avaliar cada política com respeito à estratégia do oponente, considerando a estratégia que gera a pior conseqüência para a política em análise. Esta idéia é o âmago do algoritmo Minimax-Q, conhecido por garantir convergência ao equilíbrio no limite (SZEPESVÁRI; LITTMAN, 1996).

Para políticas de ação determinísticas, o valor ótimo do estado $s_t \in S$ em um MG é:

$$V^*(s_t) = \max_{a \in A} \min_{\bar{a} \in \bar{A}} Q(s_t, a, \bar{a}) \tag{13.2}$$

e a regra do aprendizado Minimax-Q é:

$$\begin{aligned} Q_{t+1}(s_t, a_t, \bar{a}_t) &= Q_t(s_t, a_t, \bar{a}_t) + \\ &\alpha_t[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s_t, a_t, \bar{a}_t)] \end{aligned} \tag{13.3}$$

onde: s_t é o estado corrente, a_t é a ação executada por **A** em s_t, $\bar{a}_t$ é a ação executada por **B** em s_t, $Q_{t+1}(s_t, a_t, \bar{a}_t)$ é o reforço descontado esperado por executar a ação a_t quando **B** executa $\bar{a}_t$ no estado s_t, e continua a seguir a política ótima daí em diante, $r_{s_t,a_t,\bar{a}_t}$ é o reforço recebido por **A**, s_{t+1} é o próximo estado, $\hat{V}_t(s_{t+1})$ é a estimativa corrente do valor ótimo descontado esperado $V^*(s_{t+1})$, γ é o fator de desconto, α_t é a taxa de aprendizagem.

13.3.2 Generalizando o Minimax-Q

A autora e seu grupo propuseram o uso de generalização da experiência (veja Seção 12.4.1) no algoritmo Minimax-Q, resultando num novo algoritmo denominado Minimax-QS.

Formalmente, no Minimax-QS para MGs alternados, no tempo t:

1. **A** e **B** observam o estado corrente s_t.
2. **A** seleciona uma ação $a_t \in A$ e a executa.
3. **B** seleciona uma ação $\bar{a}_t \in \bar{A}$ e a executa.
4. **A** e **B** observam o novo estado s_{t+1} e recebem o reforço $r_{s_t,a_t,\bar{a}_t}$.
5. Os valores Q para toda tripla estado-ação **A**-ação **B**, dada por $(s, a, \bar{a})$, são atualizados de acordo com:

$$Q_{t+1}(s, a, \bar{a}) = Q_t(s, a, \bar{a}) +$$

$$\alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s, a, \bar{a})] \quad (13.4)$$

6. Repetir os passos acima até que um critério de parada seja atingido.

Onde $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$ é a *função de espalhamento* ($0 \leq \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) \leq 1$).

O algoritmo Minimax-Q padrão corresponde à Equação 13.4 com:

$$\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = \delta(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$$

onde $\delta(.,.)$ é a função delta de Kronecker (isto é, $\delta(u, v) = 1$ se $u = v$, e $\delta(u, v) = 0$ no caso contrário).

O uso de $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$ torna possível a redução do tempo de aprendizado do algoritmo Minimax-Q, uma vez que a conseqüência de escolher a ação a_t quando o oponente escolhe $\bar{a}_t$ no estado s_t pode ser espalhada para todas as outras triplas similares $(s, a, \bar{a})$.

Nos experimentos considerou-se um mecanismo que produz a generalização no espaço de estados S, isto é $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = g_t(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$, onde $g_t(s_t, s)$ é a função de similaridade entre estados. Esta função foi definida como $g_t(s_t, s) = \tau^d$, onde τ é uma constante e d é uma medida de similaridade – distância – entre o estado corrente s_t e um estado similar s.

13.3.3 Experimentos no Domínio de Futebol de Robôs

Nos experimentos realizados foi utilizado um simulador proposto por Littman (1994), que se refere a um jogo de soma zero entre dois jogadores, numa grade 4x5. Os jogadores sempre ocupam células distintas na grade. A configuração inicial do campo consiste nos jogadores **A** e **B** nas posições mostradas na Figura 13.3, e a posse de bola é dada aleatoriamente a **A** ou **B** (na figura, o agente **A** tem posse da bola).

Em cada passo de tempo, os jogadores podem se movimentar escolhendo uma de 5 ações: N, S, E, W e Parado. Quando um jogador tenta se mover para uma célula ocupada por outro jogador, o movimento falha e o segundo jogador fica com a posse da bola. Quando um jogador tenta executar uma ação que o levaria

para fora do campo, o movimento não ocorre. Quando o jogador com a posse da bola alcança o gol (à esquerda para **A** e direita para **B**), ele marca um gol e todos devem retornar à configuração inicial.

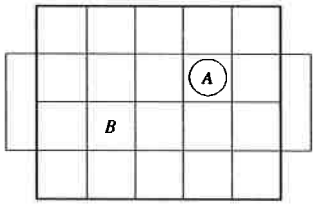


Figura 13.3: O simulador de futebol, na configuração inicial (A com a bola).

Várias funções de similaridade podem ser exploradas no domínio de futebol de robôs. A eficácia do uso do mecanismo de espalhamento proposto é demonstrada considerando uma função de similaridade baseada em regras muito simples derivadas de configurações particulares do campo.

Similaridade entre configurações foi definida como uma função do número de ações necessário para mover o oponente (jogador **B**) da posição onde a experiência real ocorreu para a posição definida na configuração similar considerada. Formalmente esta medida de similaridade mencionada corresponde à função de espalhamento $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = g_t(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$ (veja a Equação 13.4) com $g_t(s_t, s) = \tau^d$, onde d é o número mínimo de ações para levar o oponente de s_t para s e τ ou é uma constante (em alguns experimentos, $\tau = 0.7$) ou é uma função de decaimento linear com o número de iterações (sendo $\tau = 0.7$ o valor inicial máximo).

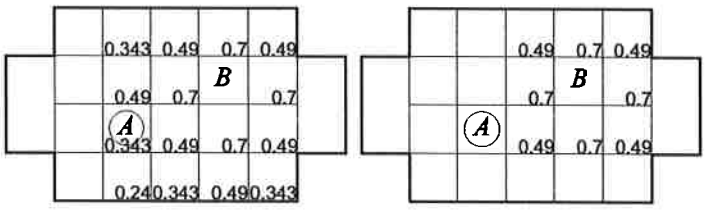
A experiência $\langle s_t, a_A, a_B, s_{t+1}, r_t \rangle$ no tempo t somente foi espalhada para os estados s definidos para uma vizinhança de $\mathbf{B}_{\mathbf{s_t}}$ (veja a Figura 13.4).

Nos experimentos, esta vizinhança é medida pela métrica d_{ch} para duas posições $\mathbf{B}_{\mathbf{s_t}} = (x_{B_{s_t}}, y_{B_{s_t}})$, onde a experiência real ocorreu, e os estados similares definidos pelo jogador **B** em $\mathbf{B}_{\mathbf{s}} = (x_{B_s}, y_{B_s})$:

$$d_{ch}(\mathbf{B}_{\mathbf{s_t}}, \mathbf{B}_{\mathbf{s}}) = \max |x_{B_{s_t}} - x_{B_s}|, |y_{B_{s_t}} - y_{B_s}| \tag{13.5}$$

Dado um valor d_{ch}, uma área de espalhamento é definida, onde cada célula nesta área representa a posição de **B** no estado similar considerado. A Figura 13.4 mostra o valor de espalhamento τ^d para todas as células incluídas na área

definida por $d_{ch} = 1$ (Figura 13.4(a)) e $d_{ch} = 2$ (Figura 13.4(b)).



(a) $d_{ch} = 1$ (8 vizinhos de $\mathbf{B}_{s_t}$) e (b) $d_{ch} = 2$ (24 vizinhos de $\mathbf{B}_{s_t}$).

Figura 13.4: Área de espalhamento para a configuração da Figura 13.3.

Foram realizados 9 experimentos diferentes no simulador de futebol, três utilizando uma função constante de espalhamento (Minimax-QS, *spreading constant*), três usando uma função de decaimento linear de espalhamento (Minimax-QS, *spreading vanishes* e três usando a implementação tradicional do algoritmo Minimax-Q (Minimax-Q).

O jogador aprendiz **A** foi treinado contra um oponente aleatório **B** (Figuras 13.5 e 13.6) e contra um oponente Minimax-Q (Figure 13.7).

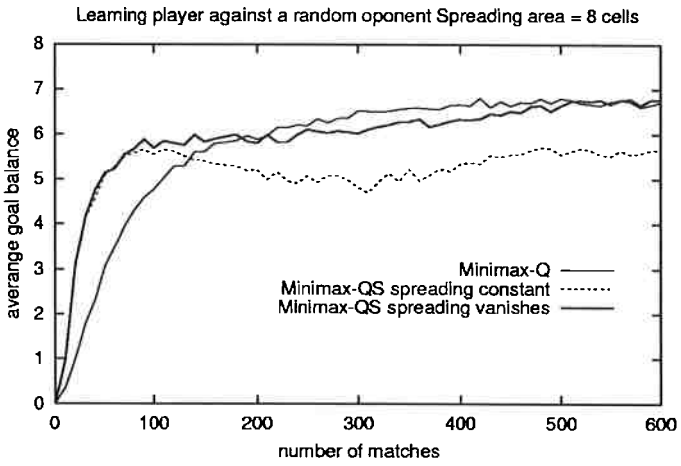
Os parâmetros usados foram $\gamma = 0.9$, valor inicial de $\alpha = 1.0$ (máximo) e decaimento linear para 0 na iteração 150000 — isto é, torna-se 0 aproximadamente no jogo 500, valor inicial da tabela-$Q = 0$, taxa de exploração aleatória $= 0,2$, e taxa de execução aleatória de ação $= 0.2$, representando o não determinismo na execução da ação.

Para cada algoritmo foram executadas 100 sessões, cada uma com 600 jogos. Cada jogo é composto por 10 partidas e cada partida termina com a marcação de um gol (pelo jogador ou pelo oponente) ou ao atingir um número pré-definido de iterações.

Foram computadas as médias dos saldos de gols marcados pelo aprendiz dos jogos 1 a 600, nas 100 sessões. O saldo de gols resultante do aprendiz **A** sobre o oponente **B**, para cada experimento, é mostrado nas Figuras 13.5, 13.6, e 13.7.

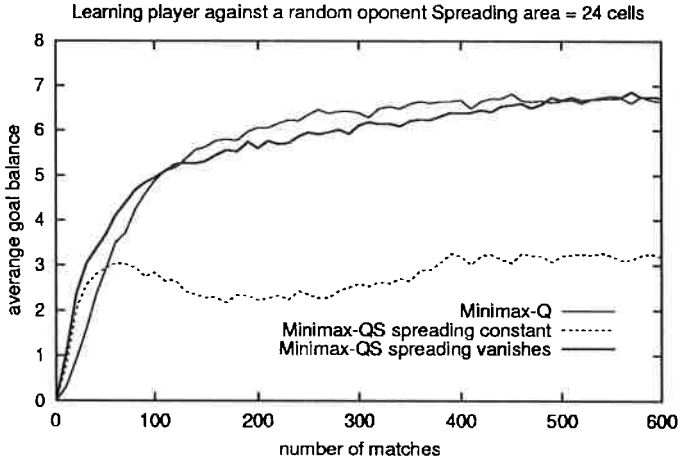
Os resultados mostram um efetivo aumento no saldo de gols para o aprendiz no início do aprendizado, quando a função de espalhamento foi usada, demonstrando a eficácia da técnica proposta.

Um fato interessante que pode ser observado é que, se a função de espalha-



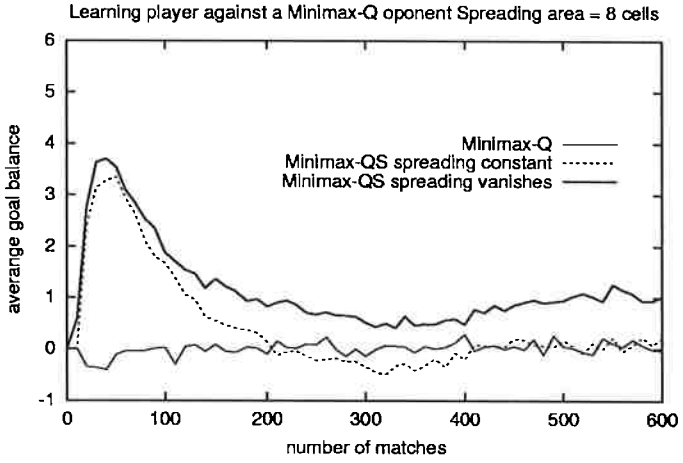
O eixo X representa o número de jogos e o eixo Y, o saldo de gol do aprendiz sobre o oponente. Os valores de espalhamento são mostrados na Figura 13.4(a).

Figura 13.5: Resultados dos jogos usando Minimax-Q (linha fina) e Minimax-QS, com decaimento linear (linha grossa) e função de espalhamento constante (linha tracejada).



O eixo X representa o número de jogos e o eixo Y, o saldo de gol do aprendiz sobre o oponente. Os valores de espalhamento são mostrados na Figura 13.4(b).

Figura 13.6: Resultados dos jogos usando Minimax-Q (linha fina) e Minimax-QS, com decaimento linear (linha grossa) e função de espalhamento constante (linha tracejada).



O eixo X representa o número de jogos e o eixo Y, o saldo de gol do aprendiz sobre o oponente. Os valores de espalhamento são mostrados na Figura 13.4(a).

Figura 13.7: Resultados dos jogos usando Minimax-Q (linha fina) e Minimax-QS, com decaimento linear (linha grossa) e função de espalhamento constante (linha tracejada).

mento desaparece após alguns jogos — por volta de 100 jogos, nos experimentos mostrados — Minimax-QS converge aos valores de Minimax-Q. A prova teórica mostrada no Anexo B corroboram esta evidência.

A autora e seu grupo pretendem continuar as pesquisas nesta linha, utilizando generalização de experiência em algoritmos AR híbridos e, mais especificamente, em avaliações das abordagens em domínios reais de aplicação e no uso de adaptação automática da função de espalhamento no domínio da tarefa.

13.4 Discussões

Os desenvolvimentos da autora e seu grupo na área de times de robôs ainda estão em fase embrionária, apesar do sucesso adquirido com os times FUTEPOLI e textscGuaraná de futebol de robôs.

Na verdade, aplicações reais, em tempo real, de times de robôs móveis, com capacidades de organização, cooperação explícita entre os robôs e regras sociais conhecidas por eles, é ainda um grande desafio.

Investimentos devem ser feitos na pesquisa e desenvolvimento de arquiteturas apropriadas para a atuação em times cooperativos, de forma eficiente.

Além disso, o uso de diversos níveis de controle, tanto no agente, quanto na sociedade, tornam-se necessários. A exploração de técnicas de aprendizado autônomo nestes controles também é uma área bastante promissora.

14 Conclusões

Nesta tese um espectro concreto de temas relevantes para a Robótica Móvel foi abordado e as realizações da autora e seu grupo foram descritas, com enfoque em visão computacional, navegação, controle e atuação dos robôs em times – todos estes temas permeados com capacidades de aprendizado e adaptação.

Algumas concepções importantes têm sido firmadas nos últimos anos para o projeto e desenvolvimento de robôs móveis inteligentes.

Uma primeira concepção é a de que hardware e software são, não somente inseparáveis, mas também dependentes um do outro no projeto de robôs móveis inteligentes. Os sinais dos sensores (hardware) formam a base do processo de controle (software) que, por sua vez, comanda os atuadores (hardware) na execução de uma tarefa. Esta concepção conduz a pesquisas com robôs *reais*, imersos no seu ambiente de execução da tarefa (*"embedded, situated agents"*). Isto porque os efeitos imprevisíveis das interações do robô com o ambiente – e do próprio ambiente – dificilmente são modeláveis em simuladores ou em modelos matemáticos simples.

Outra importante concepção é a de que o sucesso na atuação de um robô é fortemente baseado nas suas capacidades perceptuais. Visão é uma modalidade extremamente poderosa e deve ser integrada no projeto de robôs móveis. Assim, um desafio que se impõe é a exploração de novas formas de organizar a visão e integrá-la a outras capacidades cognitivas dos robôs, de forma que os mesmos possam usá-la, em tempo real.

Uma terceira concepção igualmente importante é a necessidade do uso de estruturas de controle *distribuídas*, compostas por comportamentos simples ou agentes, interagindo entre si para gerar o comportamento global do robô. Tais controles tendem a ser menos instáveis e mais flexíveis que os controles mo-

nolíticos. Assim, a combinação e uso concorrente, de forma *eficiente*, de diversos mecanismos e de múltiplas fontes de informação mostram-se como um grande desafio, que certamente conduzirão a uma solução adequada para o projeto de robôs móveis inteligentes, pois permitem que sejam preservadas as vantagens de cada princípio e minimizadas suas desvantagens.

Finalmente, uma última concepção importante diz respeito à autonomia do robô. Pesquisas e desenvolvimentos na integração de técnicas robustas e eficientes de adaptação e aprendizado autônomo nos robôs mostram-se como desafios a serem vencidos para alcançar a meta de robôs móveis inteligentes.

Assim, abraçando o paradigma multi-agentes, integrando visão computacional e aprendizado autônomo ao robô e concentrando pesquisas em robôs reais, operando nos ambientes de execução de suas tarefas, certamente conduzirão a novos avanços no sentido de obter robôs robustos, confiáveis, autônomos e que tenham sucesso na execução de suas tarefas.

Enquanto as pesquisas na área da Robótica Inteligente têm conduzido a um espectro muito amplo de robôs com competências físicas, de locomoção e percepção bastante diversas, a sociedade tem se direcionado à incorporação dos robôs no dia-a-dia das pessoas, desde tarefas voltadas à diversão e entretenimento até a assistência a idosos, enfermos, debilitados ou com deficiências físicas ou mentais. Embora inicialmente os robôs tenham sido desenvolvidos para aplicações repetitivas ou que envolvessem perigo ou condições insalubres para humanos, hoje os mesmos estão sendo dirigidos à condição de brinquedos inteligentes e assistentes pessoais, tornando-os mais e mais presentes em nossas vidas.

Anexo A – Teorema de Probabilidade Total e Regra de Bayes

A.1 Teorema de Probabilidade Total

Seja A um evento pertencente ao espaço amostral Ω e o conjunto $E = E_1, \ldots, E_n$ uma partição desse espaço, isso é, conjunto de eventos tais que $E_i \cap E_j = \emptyset$, para $i \neq j$ e, a união de todos os E_i é igual a Ω, então o teorema da probabilidade total diz que:

$$P(A) = \sum_{i=1}^{n} P(A|E_i) \cdot P(E_i) \tag{A.1}$$

A.2 Regra de Bayes

O teorema de Bayes atesta que:

$$P(E_k|A) = \frac{P(A|E_k)P(E_k)}{P(A)} \tag{A.2}$$

Aplicando a lei da probabilidade total na Equação A.2, vem:

$$P(E_k|A) = \frac{P(A|E_k)P(E_k)}{\sum_{E \in \Omega} P(A|E_i)P(E_i)} \tag{A.3}$$

A importância desse teorema é que ele permite calcular a probabilidade de um evento posterior à ocorrência de outro evento, dados probabilidades que podem ser previamente calculadas.

Anexo B – Convergência do Algoritmo Minimax-QS

Este Anexo mostra a prova de convergência do algoritmo Minimax-QS (RIBEIRO; PEGORARO; COSTA, 2002), proposto pela autora e seu grupo (veja Seção 13.3). A proposição a ser provada nesta Anexo é a seguinte:

> Considere que as condições de convergência do Minimax-Q para a função de valor da ação Q esteja satisfeita. Então os valores gerados pelo Minimax-QS convergem para Q com probabilidade um se a convergência de $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) - \delta(s, s_t)\delta(a, a_t)\delta(\bar{a}, \bar{a}_t) \rightarrow 0$ for da ordem $O(\alpha_t)$.

O que esta proposição diz é que, se o mecanismo de espalhamento desaparece no mínimo tão rapidamente quanto a taxa de aprendizado α_t, então Minimax-QS converge para os valores gerados pelo Minimax-Q.

B.1 Aproximações Simultâneas em AR

Encontrar uma função ótima usando algoritmos AR implica em duas aproximações ocorrendo simultaneamente: uma que aproxima o operador de Programação Dinâmica e a outra, que usa o resultado desta aproximação para estimar a função de custo.

Para ilustrar, considere o algoritmo Minimax-Q para Jogos de Markov alternados. A equação:

$$Q_{t+1}(s_t, a_t, \bar{a}_t) = Q_t(s_t, a_t, \bar{a}_t) + \alpha_t[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s_t, a_t, \bar{a}_t)] \quad \text{(B.1)}$$

aproxima o operador:

$$(\overset{mq}{T}Q)(s_t, a_t, \bar{a}_t) = r_{s_t,a_t,\bar{a}_t} + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t, \bar{a}_t)V^*(s_{t+1}) \qquad \text{(B.2)}$$

aplicado a $Q_t(s_t, a_t, \bar{a}_t)$. A aproximação resultante é utilizada para aproximar os custos ótimos $V^*(s_{t+1})$ via $\hat{V}_t(s_{t+1}) = \max_a \min_{\bar{a}} Q_t(s_{t+1}, a, \bar{a})$.

Aproximações Estocásticas

Seja $B(S)$ o conjunto de funções de custo sobre S e $T : B(S) \mapsto B(S)$ um mapeamento arbitrário de contração com ponto fixo V^*.

Para os algoritmos de Programação Dinâmica em geral, um mapeamento de contração T (ou T^q) é aplicado diretamente, para aproximar sucessivamente V^* (ou Q). Nos métodos AR, tal operador não existe e o agente deve usar sua própria experiência para aproximá-lo.

Considere a seqüência de operadores $T_t : (B(S) \times B(S)) \mapsto B(S)$ e defina $U_{t+1} = T_t U_t V$ onde V and U_0 são funções de custo arbitrárias. Diz-se que T_t aproxima T em V com probabilidade um uniformemente sobre S, se U_t convergir a TV uniformemente sobre S.

> (SZEPESVÁRI; LITTMAN, 1996) Seja T um mapeamento arbitrário com ponto fixo V^*, e T_t aproxima T em V^* com probabilidade um uniformemente sobre S. Seja V_0 uma função de custo arbitrária e defina $V_{t+1} = T_t V_t V_t$. Se existirem funções $0 \leq F_t((x)) \leq 1$ e $0 \leq G_t((x)) \leq 1$ que satisfaçam as condições abaixo com probabilidade um, então V_t converge a V^* com probabilidade um uniformemente sobre S.
>
> 1. Para cada U_1, $U_2 \in B(S)$ e $s \in S$,
>
> $$|(\underset{t}{T}U_1V^*)(s) - (\underset{t}{T}U_2V^*)(s)| \leq G_t(s)|U_1(s) - U_2(s)| \qquad \text{(B.3)}$$
>
> 2. Para cada U, $V \in B(S)$ e s, $s' \in S$,
>
> $$|(\underset{t}{T}UV^*)(s) - (\underset{t}{T}UV)(s)| \leq F_t(s) \sup_{s'} |V^*(s') - V(s')| \qquad \text{(B.4)}$$

3. Para todo $k > 0$, o produto $\prod_{t=k}^{n} G_t(s)$ converge a zero uniformemente em s conforme n aumenta;
4. Existe $0 \leq \beta \leq 1$ tal que para todo s e grande o suficiente t, $F_t(s) \leq \beta(1 - G_t(s))$.

A prova e detalhamentos da aplicabilidade deste teorema podem ser encontrados em (LITTMAN; SZEPESVÁRI, 1996) e (SZEPESVÁRI; LITTMAN, 1996).

B.2 Uma Prova de Convergência

A prova envolve três passos. Primerio, um espaço apropriado e operadores T e T_t precisam ser definidos. Depois, as condições 1 – 4 do Teorema B.1 devem ser verificadas. Finalmente, deve-se verificar se T_t aproxima T com probabilidade 1.

Considere a equação de atualização do Minimax-QS. O sobre-escrito s indica que o espalhamento foi usado para atualizar os valores:

$$\begin{aligned} Q^s_{t+1}(s,a,\bar{a}) = Q^s_t(s,a,\bar{a}) + \\ \alpha_t \sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})[r_{s_t,a_t,\bar{a}_t} + \\ \gamma \hat{V}_t(s_{t+1}) - Q^s_t(s,a,\bar{a})] \end{aligned} \tag{B.5}$$

Considere que a função σ_t seja tal que $\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})$ converge uniformemente a $\delta(s_t,s)\delta(a_t,a)\delta(\bar{a}_t,\bar{a})$. Isto significa que a regra de atualização do Minimax-QS aproxima mais e mais da equação padrão do Minimax-Q, conforme o aprendizado evolui.

O operador T é o operador Minimax-Q,

$$(\overset{mq}{T} Q)(s,a,\bar{a}) = r_{s,a,\bar{a}} + \gamma \sum_{s_{t+1}} P(s_{t+1}|s,a,\bar{a}) V^*(s_{t+1}) \tag{B.6}$$

e T_t é

$$\begin{aligned} (\overset{mqs}{T}_t UV)(s,a,\bar{a}) = U(s,a,\bar{a}) + \\ \alpha_t \sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})[r_{s_t,a_t,\bar{a}_t} + \\ \gamma \max_u \min_{\bar{u}} V(s_{t+1},u,\bar{u}) - U(s,a,\bar{a})] \end{aligned} \tag{B.7}$$

onde $U, V : S \times A \times \bar{A} \to \Re$. A equação de atualização do Minimax-QS pode então

ser escrita como $Q^s_{t+1} = T^{mqs}_t Q^s_t Q^s_t$, e se todas as condições do Teorema B.1 forem satisfeitas, então este processo converge ao ponto fixo de T^{mq} — isto é, convergem aos valores Q do algoritmo Minimax-Q. Escolhendo:

$$G_t(s, a, \bar{a}) = 1 - \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) \tag{B.8}$$

e

$$F_t(s, a) = \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) \tag{B.9}$$

que respectivamente satisfazem as primeiras duas condições do Teorema B.1 (inigualdades B.3 e B.4).

A condição 3 implicaria $\prod_{t=k}^{\infty}(1 - \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})) = 0$. Considere então este resultado: *if* $0 < m_t < 1$ *então* $\prod_{t=k}^{\infty}(1 - m_t) = 0$ *sse* $\sum_{t=k}^{\infty} m_t = \infty$. No nosso caso, isso significa que é necessário que: $\sum_{t=k}^{\infty} \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = \infty$. No entanto, *considerando* $\sum_{t=k}^{\infty} \alpha_t \delta(\mathbf{z}, x_t)\delta(u, a_t)\delta(\bar{u}, \bar{a}_t) = \infty$ (obrigatório para garantir convergência do Minimax-Q – veja (LITTMAN; SZEPESVÁRI, 1996)) e $\delta(\mathbf{z}, x_t)\delta(u, a_t)\delta(\bar{u}, \bar{a}_t) \leq C\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$ para algum $0 < C \leq \infty$, então $C \sum_{t=k}^{\infty} \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) \geq \sum_{t=k}^{\infty} \alpha_t \delta(\mathbf{z}, x_t)\delta(u, a_t)\delta(\bar{u}, \bar{a}_t) \geq \infty$, satisfazendo a condição 3.

A condição 4 é satisfeita pois $F_t(s, a) = 1 - G_t(s, a)$. Finalmente, é necessário mostrar que T^{mqs}_t aproxima T^{mq} em Q. Sabe-se que o Minimax-Q converge para Q (e então seu correspondente operador T^{mq}_t aproxima T^{mq} em Q), basta mostrar que, para um $\hat{V}(s_{t+1})$ fixo,

$$\begin{aligned} Q^s_{t+1}(s, a, \bar{a}) = Q^s_t(s, a, \bar{a}) + \\ \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})[r_{s_t, a_t, \bar{a}_t} + \\ \gamma \hat{V}(s_{t+1}) - Q^s_t(s, a, \bar{a})] \end{aligned} \tag{B.10}$$

e

$$\begin{aligned} Q_{t+1}(s_t, a_t, \bar{a}_t) = Q_t(s_t, a_t, \bar{a}_t) + \\ \alpha_t[r_{s_t, a_t, \bar{a}_t} + \gamma \hat{V}(s_{t+1}) - Q_t(s_t, a_t, \bar{a}_t)] \end{aligned} \tag{B.11}$$

convergem para o mesmo valor. Particularmente, mostrando isso iria implicar que T^{mqs}_t aproximate T^{mq}_t para o ponto fixo $V^*(s_{t+1})$. Como T^{mq}_t aproxima T^{mq} neste ponto, isto significa que T^{mqs}_t também aproxima T^{mq} em Q.

A prova pode ser isolada para pares estado-ação. Fixando $(s, a, \bar{a})$ e sendo Q_t^s, Q_t, σ_t, *etc.* os valores de $Q_t^s(s, a, \bar{a})$, $Q_t(s, a, \bar{a})$, $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$, *etc.*. Se $\alpha_t = 0$ então nem Q_t^s nem Q_t mudam, então pode-se considerar $\alpha_t > 0$ para todo t.

Considere agora que $\sigma_t - \delta_t = O(\alpha_t)$, *i.e.*, existe uma função B_t tal que $\sigma_t - \delta_t = B_t\alpha_t$. Esta consideração significa que a função de espalhamento σ_t deve convergir para δ_t pelo menos tão rapidamente quanto α_t converge a zero.

Com algumas manipulações nas equações B.10 vem:

$$Q_{t+1}^s = Q_t^s + \alpha_t[r_t + \gamma\hat{V}(s_{t+1}) - Q_t^s] + \alpha_t^2 B_t[r_t + \gamma\hat{V}(s_{t+1}) - Q_t^s] \quad \text{(B.12)}$$

Pode-se ver a equação acima como uma pequena perturbação na atualização do Minimax-Q, onde o termo aditivo da perturbação $\alpha_t^2 B_t[r_t + \gamma\hat{V}(s_{t+1}) - Q_t^s]$ pode ser desprezado conforme α_t diminui.

Assim, as equações B.10 e B.11 convergem para o mesmo valor $\hat{V}(s_{t+1})$, e portanto o operador T_t^{mqs} do Minimax-QS aproxima o operador T^{mq} do Minimax-Q em Q. Assim, T_t^{mqs} converge para Q.

Referências Bibliográficas

ACTIVMEDIA ROBOTICS. *Saphira's Manual.* Menlo Park, CA, 2001. Version 8.0a.

ALBUS, J. S. The nist real-time control system (RCS): An approach to intelligent systems research. *Journal of Experimental and Theoretical Artificial Intelligence*, v. 9, p. 157–174, 1997.

ALOIMONOS, Y. What I have learned. *CVGIP: Image Undrestanding*, v. 60, n. 1, p. 74–85, 1994.

ARKIN, R. C. Path planning for a vision-based autonomous robot. In: *Procs. of the SPIE Conf. on Mobile Robots.* Cambridge, MA: SPIE Press, 1986. p. 240–249.

ARKIN, R. C. *Behavior-Based Robotics.* Cambridge, MA: MIT Press, 1999.

BARRA, R. J. G.; DOMENECCI, R. P. *Construção de mapas globais de ambiente para robôs móveis usando sonares.* 2002. Projeto de Formatura do Curso de Engenharia Elétrica, modalidade Computação, Escola Politécnica da Universidade de São Paulo.

BARRA, R. J. G.; DOMENECCI, R. P.; COSTA, A. H. R. Construção de mapas globais de ambiente para robôs móveis usando sonares. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 56–65.

BIANCHI, R. A. C. *Uma Arquitetura de Controle Distribuída para um Sistema de Visão Computacional Propositada.* Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 1998. Mestrado em Engenharia Elétrica.

BIANCHI, R. A. C.; COSTA, A. H. R. A distributed control architecture for a purposive computer vision system. In: *IEEE International Joint Symposia on Intelligence and Systems, 2nd. Symposium on Intelligence in Automation and Robotics – IAR'96.* Rockville, Maryland, USA: IEEE Computer Society Press, 1996. p. 288–294.

BIANCHI, R. A. C.; COSTA, A. H. R. A purposive computer vision system: a multi-agent approach. In: *Workshop on Cybernetic Vision, 2nd.* Los Alamitos, CA: IEEE Computer Society Press, 1997. p. 225–230.

BIANCHI, R. A. C.; COSTA, A. H. R. Uma arquitetura de controle distribuída para um sistema de visão computacional propositada. In: *I Encontro Nacional de Inteligência Artificial - ENIA'97, XVII Congresso da Sociedade Brasileira de Computação.* Brasília, DF: SBC, 1997. p. 65–70. A. H. R. Costa antes: A. H. R. C. Rillo.

BIANCHI, R. A. C.; COSTA, A. H. R. Uma arquitetura de controle para sistemas complexos de visão computacional. In: *3o. Simpósio Brasileiro de Automação Inteligente - 3o. SBAI.* Vitória, ES: SBA, 1997. p. 94–99. A. H. R. Costa antes: A. H. R. C. Rillo.

BIANCHI, R. A. C.; COSTA, A. H. R. O sistema de visão computacional do time FUTEPOLI de futebol de robôs. In: *Congresso Brasileiro de Automática - CBA'2000.* Florianóplis, Brazil: SBA, 2000. p. 2156–2161. ISBN 85-901664-1-4.

BIANCHI, R. A. C.; COSTA, A. H. R. Comparing distributed reinforcement learning approaches to learn agent coordination. In: *Advances in AI: 8th. Ibero-American Conference on AI - IBERAMIA'2002.* Berlin, Heidelberg: Springer Verlag, 2002. (Lecture Notes in Artificial Intelligence), p. 575–584.

BIANCHI, R. A. C.; COSTA, A. H. R. Comparing distributed reinforcement learning approaches to learn agent coordination. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 87–93.

BIANCHI, R. A. C.; COSTA, A. H. R. Implementing computer vision algorithms in hardware: an FPGA/VHDL-based vision system for mobile robot. In: BIRK, A.; CORADESCHI, S.; TADOKORO, S. (Ed.). *RoboCup-01: Robot Soccer World Cup V.* Berlin, Heidelberg: Springer Verlag, 2002. (Lecture Notes in Artificial Intelligence, v. 2377), p. 281–286.

BIANCHI, R. A. C.; COSTA, A. H. R. ANT-VIBRA: a swarm intelligence approach to learn task coordination. In: *Advances in AI: 16th. Brazilian Symposium on AI - SBIA'2002.* Berlin, Heidelberg: Springer Verlag, 2002. (Lecture Notes in Artificial Intelligence), p. 195–204.

BIANCHI, R. A. C.; SIMÕES, A. S.; COSTA, A. H. R. Comportamentos reativos para seguir pistas em um robô móvel guiado por visão. In: *Simpósio Brasileiro de Automação Inteligente - SBAI'2001.* Canela: SBA, 2001. Anais.

BONVENTI Jr., W.; COSTA, A. H. R. Classificação de pixels de imagens coloridas digitais por lógica nebulosa. In: RIBEIRO, C. H. C.; SAKUDE, M. T. S. (Ed.). *Workshop de Computação - WORKCOMP'2000.* Instituto Tecnológico de Aeronáutica - ITA, São José dos Campos, SP: ITA, 2000. p. 67–72.

BONVENTI Jr., W.; COSTA, A. H. R. Comparação entre métodos de definição de conjuntos nebulosos de cores para classificação de pixels. In: BARROS, L. N.; Cesar Jr., R. M.; COZMAN, F. G.; COSTA, A. H. R. (Ed.). *International Joint*

Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, I WAICV - Workshop on Artificial Intelligence and Computer Vision. Atibaia, SP, Brazil: SBC, 2000. p. 105–110. ISBN 85-901664-1-4.

BONVENTI Jr., W.; COSTA, A. H. R. Classificação por agrupamento nebuloso de padrões de cores em imagens. In: SAKUDE, M. T. S.; A. CASTRO CESAR, C. de (Ed.). *Workshop de Computação – WORKCOMP'2002.* Instituto Tecnológico de Aeronáutica – ITA, São José dos Campos, SP: ITA, 2002. p. 47–55.

BORENSTEIN, J.; EVERETT, B.; FENG, L. *Navigating Mobile Robots: Systems and Tecnhiques.* Wellesley, MA: A K Peters Limited, 1996.

BOUGUET, J.-Y. *Camera Calibration Toolbox for MatLab.* 2003. Disponível em: `http://vision.caltech.edu/bouguetj/calib_doc/index.html`. Acesso em: 06/02/03.

BRAITENBERG, V. *Vehicles: Experiments in Synthetic Psychology.* Cambridge, MA: MIT Press, 1984.

BRAMBILA, A. P.; COSTA, A. H. R. Navegação de robôs móveis utilizando a arquitetura subsumption. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 11–13.

BROOKS, R. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, v. 1, p. 1–10, 1986.

BROOKS, R. New approaches to robotics. *Science*, v. 253, p. 1227–1232, September 1991.

BRUCE, J.; BALCH, T.; VELOSO, M. Fast and inexpensive color image segmentation for interactive robots. In: *Proceedings of the IEEE/RSJ Internation Conference on Intelligent Robots and Systems – IROS'00.* Japan: IEEE Computer Society Press, 2000. v. 3, p. 2061–2066.

BUGARD, W.; FOX, D.; HENNIG, D. Fast grid-based position tracking for mobile robots. In: *21st German Conference on Artificial Intelligence.* [S.l.: s.n.], 1997.

BUGARD, W.; FOX, D.; HENNIG, D.; SCHMIDT, T. Estimating the absolute position of a mobile robot using position probability grids. In: *Thirteen National Conference on Artificial Intelligence.* [S.l.: s.n.], 1996.

BURGARD, W.; CREMERS, A. B.; FOX, D.; HÄHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W.; THRUN, S. Experiences with an interactive museum tour-guide robot. In: *NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE.* [S.l.: s.n.], 1998. p. 98–139.

BURGARD, W.; CREMERS, A. B.; FOX, D.; HÄHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W.; THRUN, S. The interactive museum tour-guide robot. In: *NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE.* [S.l.: s.n.], 1998. p. 11–18.

CAMUS, T. Real-time quantized optical flow. *Journal of Real-Time Imaging*, v. 3, p. 71–86, 1997.

CAMUS, T.; COOMBS, D.; HERMAN, M.; HONG, T. *Real-time Single-Workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence.* 2002. Disponível em: `http://mitpress.mit.edu/e-journals/videre/001/articles/Camus/diverge-avoid.pdf`. Acesso em 28/09/2002.

CENTRO de qualidade em horticultura – CEAGESP. Programa brasileiro para melhoria dos padrões comerciais e embalagens de hortifrutigranjeiros – Classificação de laranja: programa de adesão voluntária. junho 2000.

CHOSET, H.; NAGATANI, K. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, v. 17, n. 2, p. 125–137, 2001.

CLARKE, T. A.; FRYER, J. G. The development of camera calibration methods and models. *Photogrammetric Record*, XVI, n. 91, April 1998.

COSTA, A. H. R. *Sistema de Visão Binária para Peças Parcialmente Oclusas.* Dissertação (Mestrado) — Faculdade de Engenharia Industrial da Fundação de Ciências Aplicadas de São Bernardo do Campo, São Paulo, 1989. Dissertação de Mestrado em Engenharia, 137 p.

COSTA, A. H. R. Um sistema de visão binária para reconhecimento de peças isoladas e parcialmente oclusas. In: *III Simpósio Brasileiro de Computação Gráfica e Processamento de Images – III SIBGRAPI.* Gramado, RS: SBC, 1990. p. 236–245. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Um sistema de visão computacional para reconhecimento de peças industriais. In: *V Congresso Nacional de Automação Industrial – IV CONAI.* São Paulo, SP: Sucesu-SP, 1990. p. 220–229. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Grouping-based recognition system. In: LARSON, R. M.; NASR, H. N. (Ed.). *SPIE'91 – Advances in Intelligent Robotic Systems - Model-based vision development and tools conference.* Boston, Massachusetts, USA: SPIE Press, 1992. v. 1609, p. 274–282. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Model-based 3d object recognition from 2d. In: *4th Portuguese Conference on Pattern Recognition - RECPAD 92.* Coimbra, Portugal: [s.n.], 1992. p. 12–13. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Reconhecimento tridimensional a partir de uma única imagem bidimensional de intensidade luminosa. In: *Congresso Nacional de Automação Industrial - CONAI'92.* São Paulo, SP: Sucesu-SP, 1992. p. 263–270. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Sistema de reconhecimento de peças industriais baseado em agrupamento de características. In: *9o. Congresso Brasileiro de Automática - 9o. CBA.* UFES - Vitória, ES: SBA, 1992. p. 234–239. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. 3d object recognition using a decision hierarchy. In: TESCHER, A. G. (Ed.). *SPIE'1992 International Symposium on Optical Applied Science and Engineering - Applications of digital image processing XV.* Lockheed Palo Alto Research Lab., Saratoga, CA, USA: SPIE Press, 1993. v. 1771, p. 225–233. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. Construção automática de uma hierarquia de estratégias para reconhecimento tridimensional de objetos poliédricos. In: *VII Simpósio Brasileiro de Computação Gráfica e Processamento de Images - SIBGRAPI'94.* Curitiba, PR: SBC, 1994. p. 85–92. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. RECTRI*: um sistema de reconhecimento tridimensional a partir de uma única imagem de intensidade luminosa.* Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 1994. Tese de Doutorado em Engenharia Elétrica, 180 p., A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R. RECTRI: um sistema de reconhecimento 3d baseado em uma hierarquia de estratégias. In: *2º Simpósio Brasileiro de Automação Inteligente - 2o. SBAI.* CEFET, Curitiba, PR: SBA, 1995. p. 1–6. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R.; BARROS, L. N.; BIANCHI, R. A. C. Integrating purposive vision with deliberative and reactive planning: engineering support for robotic applications. *Journal of the Brazilian Computer Society, Special Issue on Robotics*, v. 4, n. 3, p. 52–60, April 1998. M. F. M. Campos and A. Elfes, eds., ISSN 0104-6500.

COSTA, A. H. R.; BIANCHI, R. A. C. L-VIBRA: Learning in the VIBRA architecture. In: MONARD, M. C.; SICHMAN, J. S. (Ed.). *Advances in AI: International Joint Conference, Proceedings, 7th. Ibero-American Conference on AI; 15th. Brazilian Symposium on AI; IBERAMIA-SBIA'2000.* Berlin, Heidelberg: Springer Verlag, 2000. (Lecture Notes in Artificial Intelligence, v. 1952), p. 280–289.

COSTA, A. H. R.; BIANCHI, R. A. C.; Moreira Jr., B. S.; FERRAZ, F. C. Integrando visão e comportamento: uma aplicação de reconstrução propositiva. In: *XI Congresso Brasileiro de Automática - CBA'96.* São Paulo: SBA, 1996. p. 573–578. A. H. R. Costa antes: A. H. R. C. Rillo.

COSTA, A. H. R.; PEGORARO, R. Construindo robôs autônomos para partidas de futebol: o time GUARANÁ. *Controle & Automação, SBA*, v. 11, n. 2, p. 141–149, 2000.

COSTA, A. H. R.; PEGORARO, R.; STOLFI, G.; SICHMAN, J. S.; PAIT, F. M.; Ferasoli Filho, H. GUARANÁ robot-soccer team: some architectural issues. In: *FIRA Robot World Cup France'98 Proceedings, 29th. Federation of International Robot Soccer Association.* Paris, França: Fira, 1999. p. 43–49.

COSTA, A. H. R.; PEGORARO, R.; STOLFI, G.; SICHMAN, J. S.; PAIT, F. M.; Ferasoli Filho, H. GUARANÁ robot-soccer team: some architectural issues. In: *IV Simpósio Brasileiro de Automação Inteligente - 4o. SBAI.* São Paulo, SP: SBA, 1999. p. 457–461.

COSTA, A. H. R.; VELOSO, M. Multi-agent learning of when to communicate. In: BARROS, L. N.; Cesar Jr., R. M.; COZMAN, F. G.; COSTA, A. H. R. (Ed.). *International Joint Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning.* Atibaia, SP, Brazil: SBC, 2000. p. 169–174. ISBN 85-901664-1-4.

CRITES, R. H.; BARTO, A. G. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, v. 33, p. 235–262, 1998.

DAVISON, A. J.; MURRAY, D. W. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 865–880, 2002.

DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 2, p. 237–267, 2002.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, 1997.

ELFES, A. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, v. 3, n. 3, p. 249–265, 1987.

FAUGERAS, O. *Three-Dimensional Computer Vision - A Geometric ViewPoint.* Cambridge, MA: MIT Press, 1987.

FOX, D. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation.* Tese (Doutorado) — University of Bonn, 1998.

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

GAT, E. *Reliable Goal-Directed Reative Control of Autonomous Mobile Robots.* Tese (Doutorado) — Virginia Polytecchnic Institute and State University, Blacksburgh, EUA, 1991.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing.* Boston, MA: Addison Wesley Publishing Company, 1992.

GREINER, R.; ISUKAPALLI, R. Learning to select useful landmarks. In: *Proceedings of the Twelfh National Conference on Artificial Intelligence - AAAI'94.* Seattle, Washington: [s.n.], 1994.

HARVILLE, M. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In: *Proceedings of the 7th European Conference on Computer Vision, ICCV.* Copenhagen, Denmark: [s.n.], 2002.

HAYKIN, S. *Neural Networks - A Comprehensive Foundation.* Upper Saddle River, NJ: Prentice-Hall, 1999.

HEIKKILÃ, J.; SILVÉN, O. A four-step camera calibration procedure with implicit image correction. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition - CVPR'97.* Puerto Rico: IEEE Computer Society Press, 1997. p. 1106–1112.

JOLION, J. M. Computer vision methodologies. *CVGIP: Image Understanding*, v. 59, n. 1, p. 53–71, 1994.

KAELBLING, L.; CASSANDRA, A.; LITTMAN, M. L. Acting optimally in partially observable domains. In: *12th National Conference on Artificial Intelligence.* [S.l.: s.n.], 1994.

KAELBLING, L. P.; LITTMAN, M. L. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, 1996.

KAWAI, J. M.; COSTA, A. H. R. Localização markoviana para robôs móveis utilizando sonares. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 66–71.

KITANO, H.; MINORO, A.; KUNIYOSHI, Y.; NODA, I.; OSAWA, E. Robocup: The robot world cup initiative. In: *Procs. of the Workshop on Entertainment and AI/Alife, IJCAI.* Montreal, Canada: IJCAI Press, 1995.

KONDO, N.; AHMAD, U.; MONTA, M.; MURASE, H. Machine vision based quality evaluation of iyolan orange fruit using neural networks. *Computers and eletronics in agriculture*, v. 29, p. 135–147, 2000.

KONOLIGE, K. Improved occupancy grids for map building. *Autonomous Robots*, v. 4, p. 351–367, April 1997.

KROGH, B. *A Generalized Potential Field Approach to Obstacle Avoidance Control.* Michigan, EUA, 1984.

KRÖSE, B. J. A.; VLASSIS, N.; BUNSCHOTEN, R.; MOTOMURA, Y. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, v. 19, p. 381–391, 2001.

LAPLANTE, P. A.; STOYENKO, A. D. *Real Time Imaging: Theory, Techniques, and Applications.* Los Alamitos, CA: IEEE Computer Society Press, 1997.

LITTMAN, M. L. Markov games as a framework for multi-agent reinforcement learning. In: *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94).* [S.l.]: Morgan Kaufmann, 1994. p. 157–163.

LITTMAN, M. L.; SZEPESVÁRI, C. A generalized reinforcement learning model: Convergence and applications. In: *Procs. of the Thirteenth International Conf. on Machine Learning (ICML'96).* [S.l.: s.n.], 1996. p. 310–318.

LYONS, D.; HENDRIKS, A. Planning for reactive robot behavior. In: *Procs. of the IEEE Int. Conf. on Robotics and Automation.* Nice, França: IEEE Computer Society Press, 1992. p. 2675–2680.

MAIA Jr., L. C.; COSTA, A. H. R. Fluxo ótico para o monitoramento inteligente de plataformas de trens. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 33–39.

MARR, D. *Vision.* San Francisco: Freeman, 1982.

MARTINEZ, V. V. V. A. O.; COSTA, A. H. R. Reconhecimento de marcos visuais como cenas para localização de robôs móveis. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 47–55.

MATARIC, M. Minimizing complexity in controlling a mobile robot population. In: *Procs. of the IEEE International Conference on Robotics and Automation.* Nice, France: IEEE Computer Society Press, 1992. p. 830–835.

MEYSTEL, A. Knowledge based nested hierarchical control. In: SARIDIS, G. (Ed.). *Advances in Automation and Robotics.* Greenwich, CT, EUA: JAI Press, 1990. v. 2, p. 63–152.

MITCHELL, T. *Machine Learning.* New York: McGraw Hill, 1997.

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

NAKANO, K. Application of neural networks to the color grading of apples. *Computers and eletronics in agriculture*, v. 18, p. 105–116, 1997.

NEHMZOW, U. *Mobile Robotics: A Practical Introduction.* Berlin, Heidelberg: Springer-Verlag, 2000.

NOURBAKHSH, I.; POWERS, R.; BIRCHFIELD, S. Dervish: an office-navigating robot. *AI Magazine*, v. 2, n. 16, p. 53–60, 1995.

PACHECO, R. N.; COSTA, A. H. R. Navegação de robôs móveis utilizando o método de campos potenciais. In: SAKUDE, M. T. S.; A. CASTRO CESAR, C. de (Ed.). *Workshop de Computação – WORKCOMP'2002.* Instituto Tecnológico de Aeronáutica – ITA, São José dos Campos, SP: ITA, 2002. p. 125–130.

PACHECO, R. N.; COSTA, A. H. R. Uma arquitetura reativa para navegação de robôs móveis utilizando visão computacional e sonares. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 1–10.

PARKER, L. E. Alliance: an architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, v. 14, n. 2, p. 2200–2240, 1998.

PEGORARO, R. *Agilizando aprendizagem por reforço em robótica móvel através do uso de conhecimento sobre o domínio.* Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2001. Tese de Doutorado em Engenharia Elétrica, 106 p.

PEGORARO, R.; COSTA, A. H. R. Agilizando aprendizagem por reforço através da utilização de similaridades entre estados. In: BARROS, L. N.; Cesar Jr., R. M.; COZMAN, F. G.; COSTA, A. H. R. (Ed.). *International Joint Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning.* Atibaia, SP, Brazil: SBC, 2000. p. 175–184. ISBN 85-901664-1-4.

PEGORARO, R.; COSTA, A. H. R.; RIBEIRO, C. H. C. Experience generalization for multi-agent reinforcement learning. In: PRUGNER, N.; LIMA, P. E. M. C.; COSTA, C. D. (Ed.). *ProTeM-CC – Projects Evaluation Workshop: International Cooperation.* Brasília, DF, 2001. p. 68–84. Ch. 1.

PEGORARO, R.; COSTA, A. H. R.; RIBEIRO, C. H. C. Experience generalization for multi-agent reinforcement learning. In: *SCCS'2001 – XXI International Conference of the Chilean Computer Science Society.* Los Amitos, CA: IEEE Computer Society Press, 2001. p. 233–239.

PROESMAN, M.; GOOL, L. V.; PAWELS, E.; OOSTERLINCK, A. Determination of optical flow and its discontinuities using non-linear diffusion. In: *3rd European Conference on Computer Vision, ECCV'94.* [S.l.: s.n.], 1994. v. 2, p. 295–304.

RIBEIRO, C. H. C. Reinforcement learning agents. *Artificial Intelligence Review*, v. 17, p. 223–250, 2002.

RIBEIRO, C. H. C.; COSTA, A. H. R.; ROMERO, R. A. F. Robôs móveis inteligentes: princípios e técnicas. In: MARTINS, A. T.; BORGES, D. L. (Ed.). *I Jornada de Atualização em Inteligência Artificial - JAIA. Anais do XXI*

Congresso da Sociedade Brasileira de Computação. Fortaleza, 2001. v. 3, p. 257–306.

RIBEIRO, C. H. C.; PEGORARO, R.; COSTA, A. H. R. Experience generalization for concurrent reinforcement learners: the minimax-qs algorithm. In: CASTELFRANCHI, C.; JOHNSON, W. L. (Ed.). *International Joint Conference on Autonomous Agents and Multi-Agent Systems AAMAS'2002.* Bologna, Italy: [s.n.], 2002. p. 1239–1245. ISBN 1-58113-480-0. The Association for Computing Machiney.

RIBEIRO, C. H. C.; SZEPESVÁRI, C. Q-learning combined with spreading: Convergence and results. In: *Procs. of the ISRF-IEE International Conf. on Intelligent and Cognitive Systems – Neural Networks Symposium.* [S.l.: s.n.], 1996. p. 32–36.

RILLO, M.; COSTA, A. H. R.; CIRELLI, S.; ZILINSKAS, W. Controlador lógico programável por grafcet. In: *I Congresso Nacional de Automação Industrial – I CONAI.* São Paulo, SP: Sucesu-SP, 1983. p. 246–252. A. H. R. Costa antes: A. H. R. C. Rillo.

RILLO, M.; COSTA, A. H. R.; COSTA, L. A. R. A célula de montagem do lsi. In: *9o. Congresso Brasileiro de Automática – 9o. CBA.* UFES – Vitória, ES: SBA, 1992. p. 952 – 957. A. H. R. Costa antes: A. H. R. C. Rillo.

RILLO, M.; COSTA, A. H. R.; COSTA, L. A. R. Lsi assembly cell. In: *Proc. of 7th IFAC/IFIP/IFORS/IMACS/ISPE Symposium on Information Control Problems in Manufacturing Technology (INCOM 92).* Toronto, Canada: IFAC, 1992. p. 25–28.

ROMANO, V. F. *Robótica Industrial: aplicação na indústria de manufatura e de processos.* São Paulo: Editora Edgard Blücher Ltda, 2002. ISBN 85-212-0315-2.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach.* Upper Saddle River, NJ: Prentice Hall, 1995. (Artificial Intelligence).

SCÁRDUA, L. A.; COSTA, A. H. R.; CRUZ, J. J. da. Aprendizado de comportamento por reforço do ambiente. In: *IV Simpósio Brasileiro de Automação Inteligente – SBAI'99.* São Paulo: SBA, 1999. p. 171–176.

SCÁRDUA, L. A.; COSTA, A. H. R.; CRUZ, J. J. da. Learning to behave by environment reinforcement. In: *IJCAI'99 - The Third International Workshop on RoboCup.* Stockholm, Sweden: IJCAI Press, 1999. p. 181–186.

SCÁRDUA, L. A.; COSTA, A. H. R.; CRUZ, J. J. da. Learning to behave by environment reinforcement. In: VELOSO, M.; PAGELLO, E.; KITANO, H. (Ed.). *RoboCup-99: Robot Soccer World Cup III.* Berlin, Heidelberg: Springer Verlag, 2000. (Lecture Notes in Artificial Intelligence, v. 1856), p. 439–449.

SCÁRDUA, L. A.; CRUZ, J. J. da; COSTA, A. H. R. Controle ótimo de descarregadores de navios utilizando aprendizado por reforço. In: *Congresso Brasileiro de Automática – CBA'2000.* Florianópolis, SC: SBA, 2000. p. 104–109.

SCÁRDUA, L. A.; CRUZ, J. J. da; COSTA, A. H. R. Controle Ótimo de navios utilizando aprendizado por reforço. *Controle & Automação, SBA*, 2003. Aceito para publicação em setembro de 2002. /no prelo/.

SCÁRDUA, L. A.; CRUZ, J. J. da; COSTA, A. H. R. Optimal anti-swing control of ship unloaders using reinforcement learning. *Journal Artificial Intelligence in Engineering, renomeado Advanced Engineering Informatics em 2002. Elsevier Science*, 2003. /no prelo/.

SEIXAS, M. O.; COSTA, A. H. R. Um sistema para determinação de ocupação de plataformas rodoviárias. In: COSTA, A. H. R.; RIBEIRO, C. H. C. (Ed.). *1o. Workshop do Projeto AACROM.* Escola Politécnica da USP, São Paulo, SP: [s.n.], 2002. p. 40–46.

SIMMONS, R.; KOENIG, S. Probabilistic robot navigation in partially observable environments. In: *International Joint Conference on Artificial Intelligence – IJCAI'95.* Montreal, Quebec, Canadá: IJCAI Press, 1995.

SIMÕES, A. S. *Segmentação de Imagens por Classificação de Cores: uma Abordagem Neural.* Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2000. Mestrado em Engenharia Elétrica.

SIMÕES, A. S.; COSTA, A. H. R. Classificação de cores por redes neurais usando a representação HSV. In: MONARD, M. C.; SICHMAN, J. S. (Ed.). *International Joint Conference IBERAMIA'2000 and SBIA'2000, Open Discussion Track Proceedings.* Atibaia, SP: SBC, 2000. p. 314–323.

SIMÕES, A. S.; COSTA, A. H. R. Segmentação de imagens por classificação de cores: uma abordagem neural para a representação RGB. In: RIBEIRO, C. H. C.; SAKUDE, M. T. S. (Ed.). *Workshop de Computação – WORKCOMP'2000.* Instituto Tecnológico de Aeronáutica – ITA, São José dos Campos, SP: ITA, 2000. p. 25–31.

SIMÕES, A. S.; COSTA, A. H. R. Using neural color classification in robotic soccer domain. In: BARROS, L. N.; Cesar Jr., R. M.; COZMAN, F. G.; COSTA, A. H. R. (Ed.). *International Joint Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning.* Atibaia, SP, Brazil: SBC, 2000. p. 208–213. ISBN 85-901664-1-4.

SIMÕES, A. S.; COSTA, A. H. R. Classificação de cores por redes neurais artificiais: um estudo do uso de diferentes sistemas de representação de cores no futebol de robôs móveis autônomos. In: *Encontro Nacional de Inteligência Artificial – ENIA'2001 – Anais do XXI Congresso da Sociedade Brasileira de Computação.* Fortaleza, CE: SBC, 2001. v. 1, p. 182.

SIMÕES, A. S.; COSTA, A. H. R. Utilizando processos gaussianos para a segmentação de imagens monocromáticas. In: *Anais do IX SEMINCO – Seminário de Computação da Universidade Regional de Blumenau – FURB.* Blumenau, SC: [s.n.], 2002. p. 177–188.

SIMÕES, A. S.; COSTA, A. H. R. Color- and shape-based machine vision for orange sorting. Computers and Electronics in Agriculture. Elsevier Science. Submetido em janeiro de 2003. /submetido/. 2003.

SIMÕES, A. S.; COSTA, A. H. R.; ANDRADE, M. T. C. Utilizando um classificador fuzzy para seleção visual de laranjas. In: RIBEIRO, C. H. C.; SAKUDE, M. T. S. (Ed.). *Workshop de Computação - WORKCOMP'2001.* Instituto Tecnológico de Aeronáutica - ITA, São José dos Campos, SP: ITA, 2001. p. 113–117.

SIMÕES, A. S.; COSTA, A. H. R.; HIRAKAWA, A. R.; SARAIVA, A. M. Applying neural networks to automated visual fruit sorting. In: ZAZUETA, F.; XIN, J. (Ed.). *World Congress of Computers in Agriculture and Natural Resources.* Iguaçu Falls, Brazil: American Society of Agricultural Engineers ASAE Publication 701P0301, Michigan, USA, 2002. p. 1–7. ISBN 1-892769-22-0.

ANDEEN, G. (Ed.). *Robot Design Handbook.* New York: McGraw-Hill, 1988.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

SZEPESVÁRI, C.; LITTMAN, M. L. *Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms.* Brown University, Providence, Rhode Island 02912, 1996. CS-96-11.

TARR, M. J.; BLACK, M. J. A computational and evolutionary perspective on the role of representation in vision. *CVGIP: Image Understanding*, v. 60, n. 1, p. 65–73, 1994.

TARR, M. J.; BLACK, M. J. Reconstruction and purpose. *CVGIP: Image Understanding*, v. 60, n. 1, p. 113–118, 1994.

TESAURO, G. Temporal difference learning and td-gammon. *Communications of the ACM*, 1995.

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

THRUN, S. Probabilistics algorithms in robotics. *Artificial Intelligence Magazine*, v. 21, n. 4, p. 93–109, 2000.

THRUN, S.; BENNEWITZ, W.; BURGARD, W.; CREMERS, A. B.; DELLAERT, F.; FOX, D.; HÄHNEL, D.; ROSENBERG, C.; SCHULTE, J.; SCHULZ, D. MINERVA: A second-generation museum tour-guide robot. In: *International Conference on Robotics and Automation.* [S.l.: s.n.], 1999.

THRUN, S.; BURGARD, W.; FOX, D. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, v. 31, p. 29–53, 1998.

TSAI, R. Y. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, RA-3, n. 4, p. 323–344, 1987.

TSITSIKLIS, J.; ROY, B. . V. Feature-based methods for large scale dynamic programming. *Machine Learning*, 1996.

ZHANG, J.; SOKHANSANJ, S.; WU, S.; FRANG, R.; YANG, W. A trainable grading system for tobacco leaves. *Computers and eletronics in agriculture*, v. 16, p. 231–244, 1997.

ZHANG, Z. Flexible camera calibration by viewing a plane from unknown orientations. In: *Proceedings of the IEEE International Conference on Computer Vision - ICCV'99*. [S.l.]: IEEE Computer Society Press, 1999. p. 666–673.

ZHANG, Z.; FAUGERAS, O.; DERICHE, R. An effective technique for calibrating a binocular stereo through projective reconstruction using both a calibration object and the environment. *Journal of Computer Vision Research - VIDERE*, v. 1, 1997.

ZIVOTÁ, B.; FLUSSER, J. Landmark recognition using invariant features. *Pattern Recognition Letters*, v. 20, p. 541–547, 1999.

Apêndice I – Plataformas de Hardware

Neste Apêndice são descritas as principais características dos robôs, câmeras e placas digitalizadoras utilizadas nas pesquisas e desenvolvimentos realizadas pela autora e seu grupo.

I.1 Robôs

Nesta Seção são apresentados rapidamente os robôs do LTI – Laboratório de Técnicas Inteligentes – e os manipuladores da Célula Flexível de Montagem da EPUSP, utilizados nos desenvolvimentos realizados.

I.1.1 Robô Pioneer 2DX

O robô Pioneer 2DX, da ActivMedia Robotics (http://www.activmedia.com), é utilizado nos trabalhos do grupo da autora. O grupo atualmente possui dois destes robôs (Figura I.1).

O robô Pioneer possui 8 sonares, dispostos na sua parte frontal, abrangendo 180 graus, que são capazes de detectar objetos entre 10 cm e 5m de distância em relação aos sonares. Testes indicaram que objetos de diâmetro inferior a 5 cm dificilmente são detectados. A direção de dois sonares consecutivos (com exceção dos laterais) difere em um ângulo de 20 graus.

Os odômetros presentes no robô possuem, cada um, 9850 marcas por rotação da roda. Outros parâmetros do robô são:
Dimensão: 44 x 38 cm;
Peso: 9 kg;
Diâmetro das rodas: 19 cm;

Carga máxima suportada: 23 kg;
Velocidade máxima: 1,6 m/s;
Velocidade máxima de giro: 300 graus/s.

O ambiente de programação utilizado é o Saphira 8.0, com linguagem orientada a objetos escritos em C++. ARIA (*ActivMedia Robotics Interface for Aplication*) é um software com objetos que permitem um controle mais eficiente do robô (ACTIVMEDIA ROBOTICS, 2001).

Há dois tipos de interface para controlar o robô. Uma delas comunica-se diretamente com o robô (*Direct Motion Commands*) e permite comandos de navegação, tais como velocidade, rotação e translação. Uma outra forma trata-se de um controle indireto do robô (*Behavioral Control*), onde ações (ou comportamentos) são implementadas como objetos, que atuam na arquitetura híbrida interna de controle do robô. A primeira interface é a usada nos experimentos e estudos realizados, para melhor domínio dos algoritmos desenvolvidos.

O robô é movimentado por duas rodas controladas independentemente por um motor cada. A energia do robô é fornecida por baterias internas.

É utilizado um notebook Toshiba Satellite 4025CDT para executar o processamento embarcado nos testes com o robô real. A Figura I.1 mostra o robô utilizado nos experimentos.

I.1.2 Times Futepoli e Guaraná

Nas investigações e experimentos realizados com time de futebol, os robôs utilizados foram construídos pelo grupo da autora. Os robôs dos times FUTEPOLI e GUARANÁ seguiram o mesmo projeto.

Os robôs usam pequenos motores DC Mabushi retirados de máquinas de calcular eletromecânicas. Cada um destes motores apresentam torque de 20 gf.pol com 5V. Um estágio único de redução é usado em cada montagem motor/roda, com redução de 5:1. Em cada roda existe um reticulado reflexivo estampado, de onde um sensor infravermelho detecta informações de rotação que realimentam a unidade de controle, permitindo a regulagem da velocidade.

Os robôs podem desenvolver velocidade máxima de 1 m/s. A unidade de

O arranjo estéreo, juntamente com o notebook responsável pelo processamento embarcado, encontram-se acoplados ao robô.

Figura I.1: Robôs Pioneer 2DX utilizados no LTI.

controle é composta por um microcontrolador (Atmel 89C5021), executando em 14,75 Hz de clock. O microcontrolador controla a tensão nos motores através de duas pontes Hs completas, ligadas aos pinos E/S, permitindo o controle de velocidade e direção por PWM (*Pulse Width Modulation*).

O receptor dos robôs é adaptado de um receptor de rádio controle remoto de servo FUTABA. O sinal recebido é enviado ao microcontrolador, que é responsável pela recuperação do sincronismo e pela demodulação do FSK Manchester.

O corpo dos robôs foi construído com placas de fibra-epoxi de vidro e canos de plástico PVC de 1/2 polegada. A potência elétrica é fornecida por 4 pilhas de NiCd, fornecendo 800 mA.h. em 4,8V. Cada robô possui um tampo de cobre, que atua como uma antena receptora. Devido à relativa alta potência de transmissão (100 mW) e aos receptores altamente sensíveis, a faixa de comunicação do rádio alcança mais de 50 metros. As Figuras I.2 e I.3 mostram fotos dos robôs.

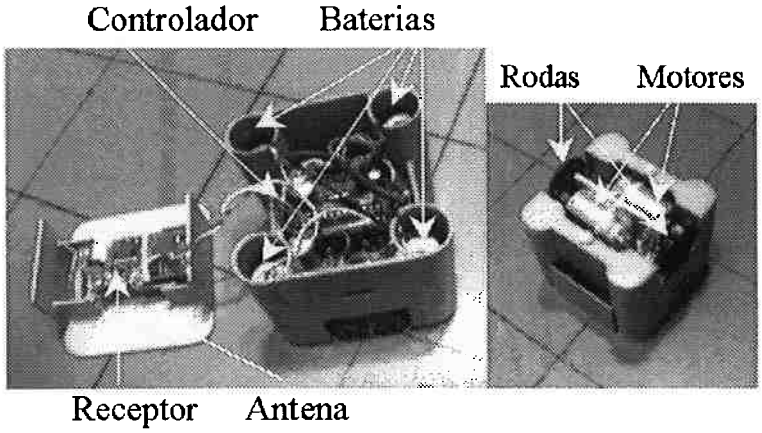


Figura I.2: Componentes do robô do time GUARANÁ.

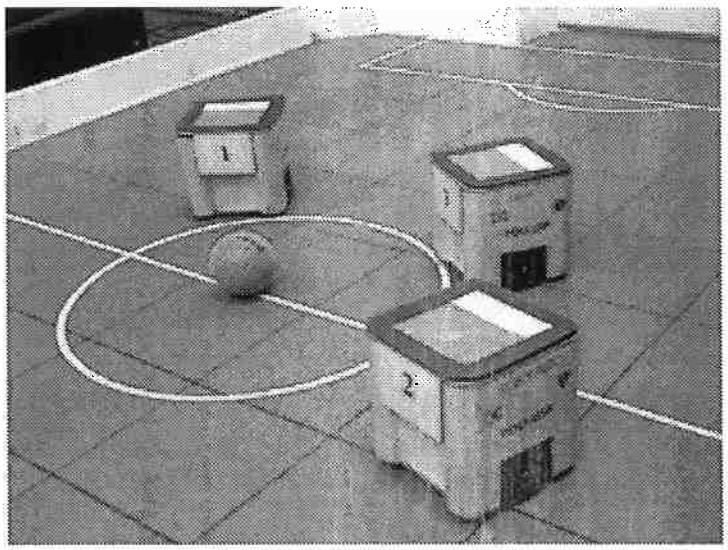


Figura I.3: Os três robôs do time GUARANÁ.

I.1.3 Célula Flexível de Montagem da EPUSP

Nas investigações cujas aplicações foram no domínio de montagem, a plataforma utilizada foi a Célula Flexível de Montagem da Escola Politécnica da USP. Na Figura I.4 pode-se ver os dois manipuladores em uma tarefa de manipulação de objetos.

Essa célula começou a ser construída em 1989, num projeto conjunto com o "Institut fuer Prozessrechentechnik und Robotik" (IPR) da Universidade de Karlsruhe, Alemanha, e atualmente é composta por: três estações de trabalho Sun Sparc, uma das quais com uma placa digitalizadora SunVideo, que permite a captura de imagens de 640 x 480 pixels com 24 bits de cores; uma câmera CCD

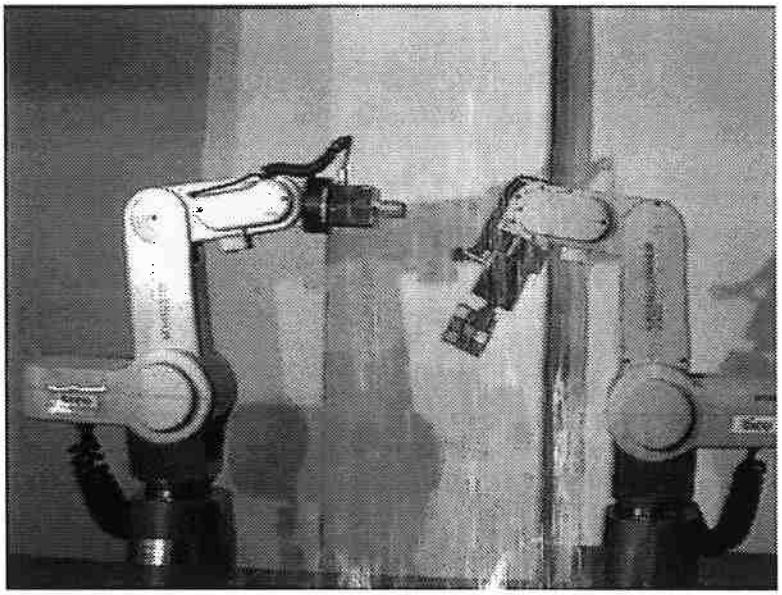


Figura I.4: Vista parcial da Célula de Montagem da EPUSP com dois manipuladores.

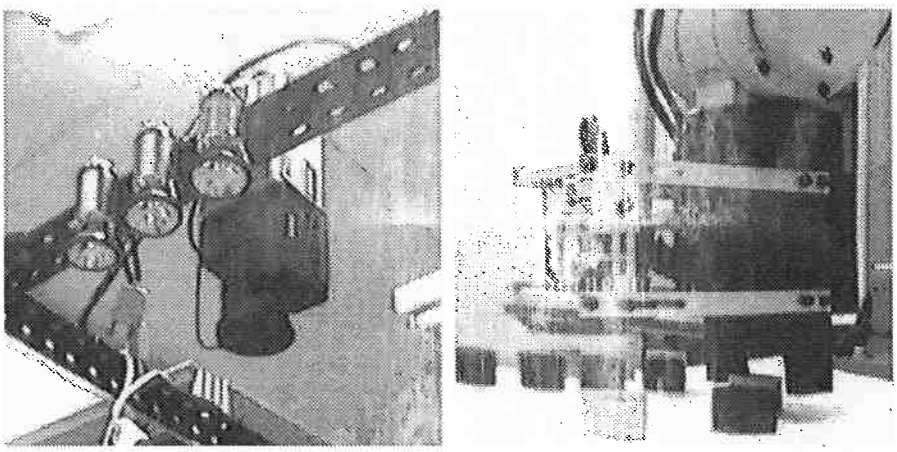


Figura I.5: Detalhes da câmera superior e da garra do manipulador com a micro câmera.

marca JVC e 3 microcâmeras Sony; dois manipuladores robóticos Mitsubishi articulados verticalmente, de 5 graus de liberdade, cada um controlado por um microcomputador padrão IBM-PC; um microcomputador padrão IBM-PC dotado de uma placa de aquisição de imagens Imaging que permite a captura de imagens no formato RGB. A Figura I.5 mostra detalhes da câmera usada para visão global do sistema e da garra do manipulador, onde se encontra uma microcâmera e da câmera usada para visão global do sistema.

Todos os computadores que formam a célula estão interligados por uma rede

ethernet local de 100Mb/s. A Célula de Montagem é utilizada como plataforma de desenvolvimento de diversos sistemas. Finalmente, ela possui uma esteira rolante que permite a simulação de peças chegando a esteira. A Figura I.6 mostra os dois manipuladores com a esteira ao fundo.

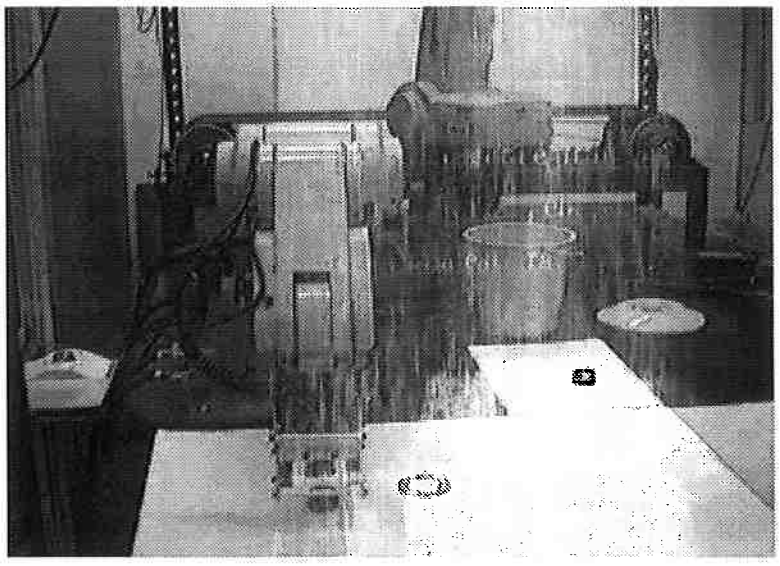


Figura I.6: Vista parcial da Célula de Montagem da EPUSP com manipuladores e esteira.

I.2 Câmeras e Placas Digitalizadoras

As câmeras usadas pelo grupo de Visão Computacional e Robótica Inteligente do Laboratório de Técnicas Inteligentes (Figuras I.7 e I.8) são:

Hitachi KP-D50 Color Digital O LTI possui uma câmera deste tipo, com as seguintes características: 1/2 polegada IT Microlens CCD, tamanho da célula de 8.4(H) x 9.8(V)(μm), saídas em Vídeo Composto, S-Video e RGB, 470 linhas de TV, pixels em 811(H) x 494(V) (NTSC), iluminação mínima de 2.0 lux (F1.2), relação sinal/ruído de 50 db, alimentação de 12 VDC, 40 mA max, dimensão de 64 x 55 x 122 (mm), 400 g de peso e sincronismo e controle externo via protocolo RS-232C.

Creative PC-CAM 300 O LTI possui duas câmeras deste tipo, com as seguintes características: lentes *focus-free* 1 metro para o infinito, taxa de aquisição de 30 quadros por segundo com resolução de 352x288 e 15 quadros

por segundo em resolução de 640x480, conexão por porta USB ou Hub USB com alimentação própria, capacidade própria de armazenamento de até 30 segundos de vídeo.

Creative Labs CT6840 USB VIDEO BLASTER WebCAM III O LTI possui uma unidade desta câmera, que apresenta as seguintes características: lentes de foco ajustável de 3 polegadas para o infinito, sensor CMOS colorido de 640x480 (CIF), controle automático de exposição e equilíbrio de luz, botão *Snapshot* para fotos instantâneas, Interface USB 1.1.

Kodo KCZ21N Color O LTI se encontra com duas câmeras deste tipo, emprestadas da TRENDS Engenharia e Tecnologia SC Ltda, com as seguintes características: sensor de imagem de 1/4 polegada super-HAD color CCD, pixels em 768(H) x 493(V) (NTSC), tamanho de célula de 4.75(H) x 5.55(V)(μm), resolução de 470 linhas de TV, iluminação mínima de 3.0 lux (F1.6,50IRE), relação sinal/ruído de 46 db (AGC off), alimentação de 12 VDC, 40 mA max, saídas em Vídeo Composto e S-Video, dimensão de 55 x 46 x 83 (mm), peso de 180 g, taxa de zoom x21 (óptico) e x8 (digital).

A autora e seu grupo utilizam as seguintes placas digitalizadoras de imagens:

PixelView 2 placas PCI do tipo PV-Bt-8x8, com interface Video for Windows e Video for Linux.

Matrox Meteor 1 placa PCI, com entradas para sinal S-Video, RGB e Vídeo Composto, com ajuste independente de ganho e *off-set* para os canais RGB.

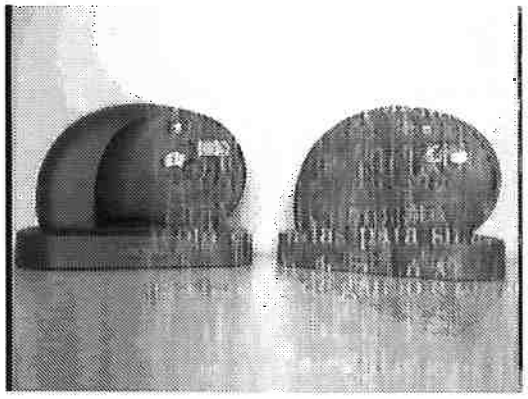


Figura I.7: Câmeras *Creative* do LTI.

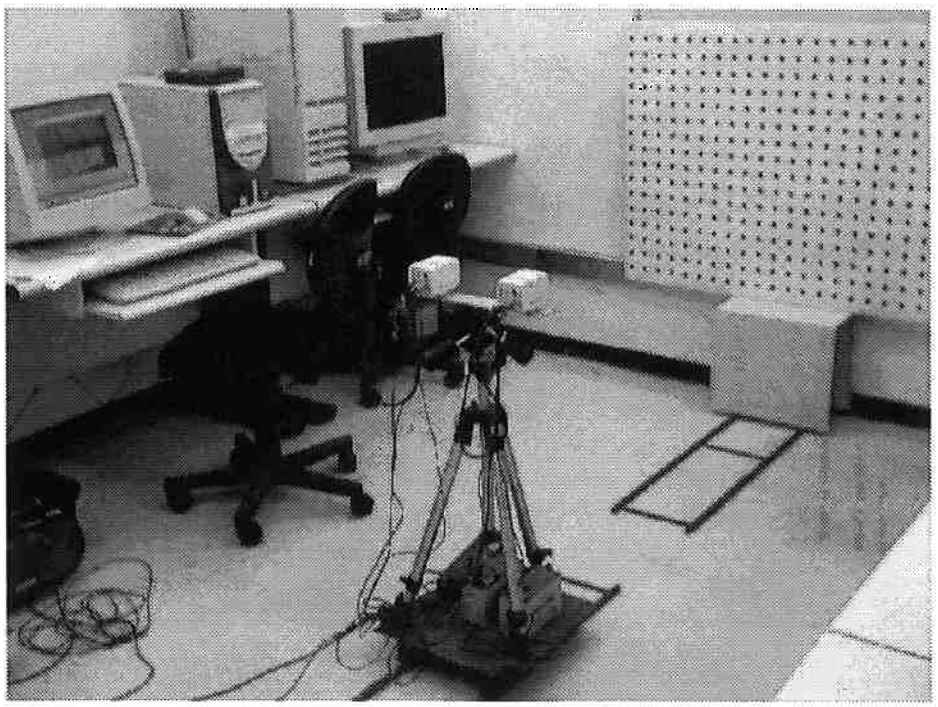
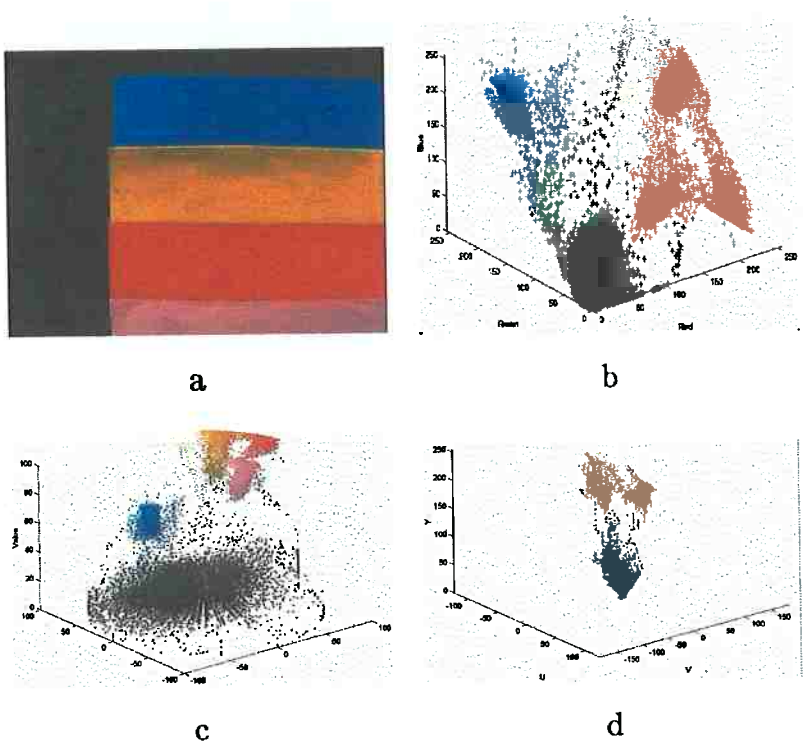


Figura I.8: O arranjo estéreo binocular.

Apêndice II – Figuras Coloridas

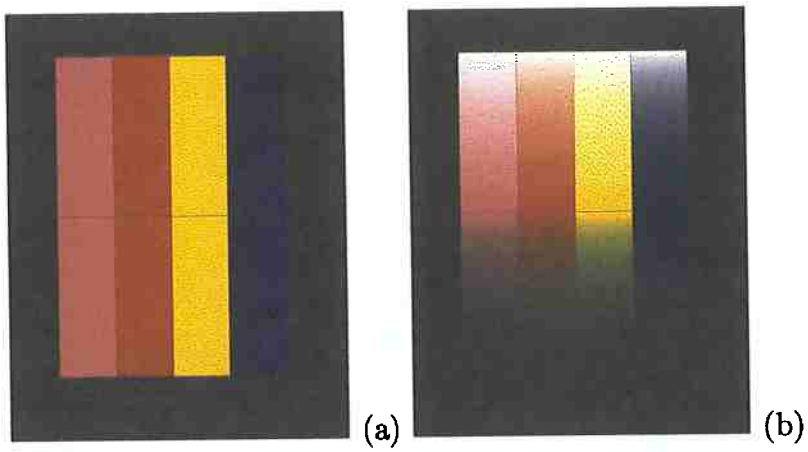
Neste Apêndice são mostradas algumas figuras do corpo do texto que são coloridas e a visualização de suas cores facilita o entendimento.



a b c d

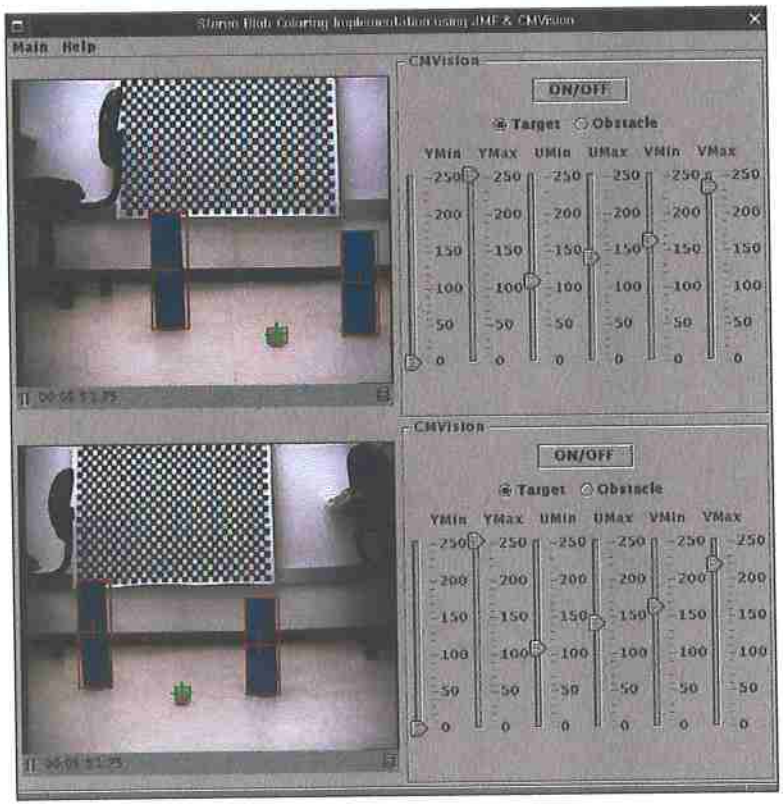
Exemplo de (a) uma imagem representada no (b) espaço RGB de cores, (c) espaço HSV e (c) espaço YUV.

Figura II.1: Figura 10.1 em cores.



Imagens utilizadas nos experimentos: (a) paleta contínua e (b) paleta degradê.

Figura II.2: Figura 10.2 em cores.



Interface do sistema de visão do Pioneer. O programa desenvolvido, resultante da integração das técnicas de segmentação por cores, determinação dos centróides e estereoscopia. Dois obstáculos (cilindros) e um alvo (bolinha) são segmentados nas imagens das duas câmeras. Graças ao deslocamento dos centróides de uma imagem para a outra, é possível recuperar a informação tridimensional dos objetos. Os controles deslizantes ao lado das imagens têm a função de definir os limiares, no espaço de cores YUV, para a segmentação.

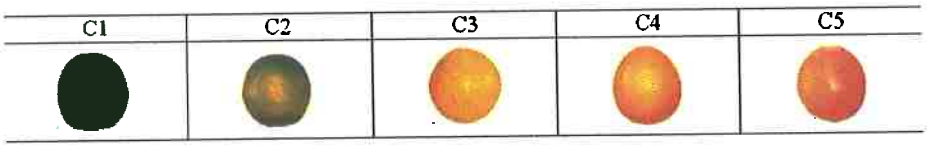
Figura II.3: Figura 10.11 em cores.

Imagem do campo de futebol de robôs utilizada para testes.

Figura II.4: Figura 10.13 em cores.

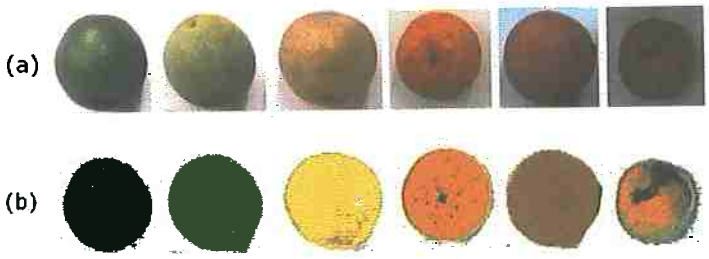
Resultados do uso de MLPs: (a) RGB; (b) HSV e (c) YUV.

Figura II.5: Figura 10.14 em cores.



Padrão brasileiro para classificação de laranjas segundo sua coloração.

Figura II.6: Figura 10.15 em cores.



a) Classes C1 a C5 e rejeitada; b) Resultado do MLP.

Figura II.7: Figura 10.19 em cores.

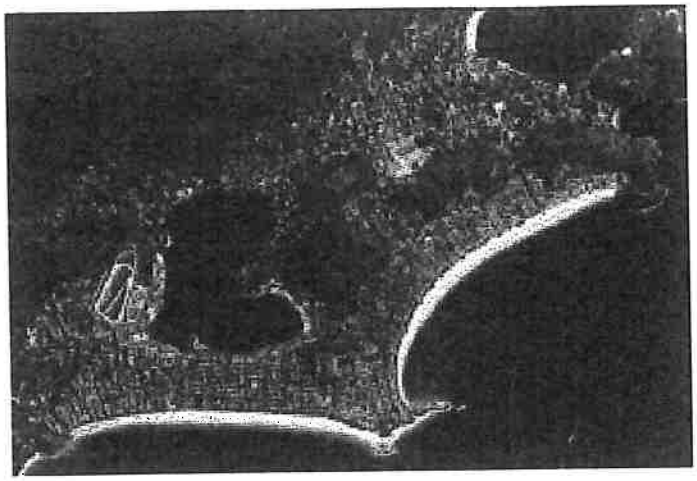


Imagem de satélite do Rio de Janeiro.

Figura II.8: Figura 10.22 em cores.

Cesta com pimentões verdes e vermelhos.

Figura II.9: Figura 10.24 em cores.