Uma arquitetura de controle inteligente para múltiplos robôs

Gedson Faria

Orientadora:
Profª. Drª. Roseli Aparecida Francelin Romero

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação -USP, como parte dos requisitos para a obtenção do título de Doutor em Ciências - Área de Ciências de Computação.

São Carlos - SP
2006

Este documento foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza o padrão *Apalike2* do sistema bibTeX.

Aos meus pais, à minha irmã e ao Marcos.

Agradecimentos

À Deus pelas oportunidades que já tive e por mais uma realização.

Aos meus amados pais, Faria e Olga, que me apoiaram emocionalmente e financeiramente, se fazendo presentes em todos os momentos.

Ao Marcos Cintra, pela amizade e pelo apoio nos momentos mais difíceis.

À minha orientadora, Roseli Aparecida Francelin Romero, que sempre motivou a realização deste projeto, não deixando eu desanimasse quando desafios aumentavam.

À minha irmã Kélita, pelo carinho, apoio e compreensão.

Ao prof. Dr. Edson Prestes, que contribuiu com muitas idéias para este trabalho, incentivando a pesquisa com múltiplos robôs.

Aos amigos de longas datas Walter Nagai, Vanessa Nishi, Danielle Ruchkys, André Lourenço, Cláudia Izequi Nagai, Marcos Abreu, Miriam Abreu, Luciane Almeida e Liliane Almeida pela compreensão da minha ausência.

Aos amigos que conheci na USP, em especial, ao Enzo Seraphim, Patrícia Rufino e Katti Facelli pelos bons momentos que compartilhamos.

Aos meus companheiros de pesquisa em robótica, Rodrigo Bianchi, Marcos Quiles, Débora Medeiros, Andrew Barbosa e Rodrigo Calvo, pela amizade e incentivo.

Aos professores e funcionários do ICMC-USP, pelas muitas lições aprendidas.

À CAPES pelo apoio financeiro.

Resumo

O desenvolvimento de arquiteturas de controle para múltiplos robôs em ambientes dinâmicos tem sido tema de pesquisas na área de robótica. A complexidade deste tema varia de acordo com as necessidades exigidas da equipe de robôs. Em geral, espera-se que os robôs colaborem uns com os outros na execução de uma tarefa. Além disso, cada robô deve ser capaz de planejar trajetórias e replanejá-las em caso de situações inesperadas.

No presente trabalho, propomos uma Arquitetura de Controle Inteligente para múltiplos robôs denominada ACIn. Para esta finalidade, foram investigadas algumas técnicas utilizadas para o controle inteligente de robôs, tais como, Redes Neurais Artificiais, Campos Potenciais e Campos Potenciais baseados em Problema do Valor de Contorno (PVC). Tais técnicas, normalmente utilizadas para um único robô, foram adaptadas para tornar possível o controle de múltiplos robôs sob arquitetura ACIn. Uma outra contribuição deste trabalho refere-se ao aperfeiçoamento da técnica de Campos Potenciais baseada PVC denominada Campos Potenciais Localmente Orientados (CPLO). Este aperfeiçoamento foi proposto para suprir a deficiência das técnicas baseadas em PVC quando estas são aplicadas em ambientes com múltiplos robôs. Além disso, um Sistema Baseado em Regras (SBR) também foi proposto como parte integrante da arquitetura ACIn. O objetivo do SBR é caracterizar a funcionalidade de cada robô para uma determinada tarefa. Isto se faz necessário para que o comportamento dos integrantes da equipe de robôs não seja competitivo e sim colaborativo.

Por fim, através dos experimentos utilizando o simulador oficial de futebol de robôs da FIRA, observou-se que a arquitetura de controle inteligente (ACIn) implementada com a técnica de planejamento CPLO e SBR propostos, mostrou-se robusta e eficiente no controle de múltiplos robôs.

Abstract

In this work, an intelligent control architecture for multi-robots denominated ACIn was proposed. With this objective, some techniques considered intelligent were studied for the planning of trajectories, such as Artificial Neural Networks, Potential Fields and Potential Fields based on the boundary value problem (BVP). Such techniques, normally used for a single robot, were adapted to function with multi-robots inside the ACIn architecture. Another contribution of this work refers to the improvement of the Potential Fields based on the boundary value problem (BVP) technique. This improvement was proposed to supply the drawback of the BVP based techniques when they are applied to multi robots environments.

Besides, a Rule Based System (RBS) was also proposed as part of the ACIn architecture. The objective of the RBS is to characterize the functionality of each robot for a determined task. This is necessary for the behavior of the equip members not to be competitive, but collaborative.

Finally, it was observed through the experiments with the robot soccer simulated environment, that our intelligent control architecture (ACIn) proposal, integrating planning and RBS for the control of multi-robots was satisfactory.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1

Introdução

1.1 Contexto e Motivação

A robótica é um dos campos de pesquisa que tem alcançado grandes desenvolvimentos nos últimos anos. Inicialmente, os robôs foram utilizados para a automação de processos de produção industrial. Com o desenvolvimento tecnológico, os robôs começaram também ser utilizados para outros propósitos tais como: brinquedos e entretenimento, medicina e cirurgia, e realização de tarefas em ambientes perigosos (vulcões, espaciais, subaquáticos). Nesse contexto, surgiram os robôs móveis autônomos com a proposta de realizar uma variedade de tarefas mais complexas que seus antecessores, os robôs industriais.

A principal limitação na utilização de robôs móveis está em como controlá-los, ou seja, como criar programas capazes de operar estas máquinas complexas. A complexidade cresce ainda mais, quando se considera, não apenas um robô, mas uma equipe de robôs para realizar uma tarefa. Para tal, são necessárias técnicas que lhes permitam interagir de forma efetiva com o ambiente. A parte essencial desta interação é o módulo de planejamento de trajetórias que cada robô utiliza para decidir suas ações e encontrar seu caminho em um ambiente.

Contudo, o projeto de um sistema que seja capaz de gerenciar com sucesso tarefas do mundo real é uma tarefa desafiadora. Tal sistema deve possuir comportamentos inteligentes capazes de resolver diferentes problemas e utilizar suas próprias experiências de forma inteligente. Além disso, este sistema deve ser capaz de lidar com situações inesperadas reagindo instantaneamente aos estímulos recebidos. Neste sentido, várias abordagens de Arquiteturas de Controle robótico foram pesquisadas a fim de resolver o problema de se projetar uma Arquitetura de Controle de robôs móveis autônomos.

As arquiteturas de controle para um único robô têm por princípio resolver o problema do planejamento de trajetórias. De acordo com a forma com que este problema é tratado, estas arquiteturas podem ser classificas como: reativas, quando o sistema responde imediatamente aos sensores sem utilizar mapas ou regras [Arkin, 1989; Balch & Arkin, 1998; Brooks, 1986]; deliberativas, se as ações são tomadas após um processamento das regras existentes para cada situação [Carbonell et al., 1989]; ou híbrida, quando planeja ações sobre mapas do ambiente sem deixar de reagir imediatamente a estímulos inesperados [Arkin, 1987; Gat, 1992; Rosenblatt, 1995]. Além do problema de planejamento de trajetórias, uma arquitetura de controle de múltiplos robôs também deve especificar o que cada robô deve fazer para realizar uma missão. Por este motivo, uma arquitetura multi-robôs pode ser construída tendo como base as arquiteturas mono-robôs, sejam reativas [Balch & Arkin, 1998; Park et al., 2001], deliberativas [McClain, 2004] ou híbridas [Dias et al., 2004]. Algumas destas arquiteturas têm como base a arquitetura Esquema Motor, por ser uma arquitetura reativa e de baixo custo computacional.

Em [Balch & Arkin, 1998] foi utilizado um controlador baseado na arquitetura Esquema Motor para criar comportamentos de busca pela meta, desvio de obstáculo e manutenção da formação durante o deslocamento de um time de robôs. McClain [2004] desenvolveu um sistema chamado *Collective* composto de um time de robôs capaz de mapear um ambiente utilizando a arquitetura Blackboard, principalmente por esta arquitetura armazenar um modelo do mundo centralizado e oferecer dispositivos de comunicação entre vários robôs.

Uma outra forma de controlar os robôs não considera um controle central. Este tipo de controle é realizado de forma distribuída, passando mensagens entre os indivíduos da equipe de robôs. A comunicação entre os robôs é a parte essencial para a distribuição de tarefas neste tipo de abordagem. O controlador proposto por Dias et al. [2004], é um exemplo deste tipo de abordagem, no qual cada robô é autônomo quanto ao sensoriamento e navegação. A arquitetura provê a troca de informação e a distribuição de tarefas entre os membros da equipe. Em cada robô, existe um módulo de execução de tarefas inspirado na arquitetura DAMN no qual as tarefas podem ser locais ou distribuídas e o resultado das tarefas pode ser combinado por um árbitro. Um outro exemplo desta abordagem é o controlador para multi-robôs tolerante a falhas proposta proposto por Parker [1998]. Para esta proposta foi utilizada uma arquitetura híbrida com módulos para atribuição de tarefas, para comunicação entre os robôs e controle de comportamentos. A base deste controle utiliza os recursos de inibição e supressão presentes nas arquiteturas reativas como a Subsumption.

Os métodos para controle de múltiplos robôs também estão relacionados com o ambi-

ente de futebol de robôs, como pode ser visto nos trabalhos apresentados a seguir.

A estratégia de controle do time de futebol de robôs Guaraná consiste em utilizar uma máquina de estados para selecionar as ações de ataque e defesa. Após selecionada a ação, um processo iterativo é executado para selecionar uma rota sem colisões até sua meta [Costa & Pegoraro, 2000]. Na abordagem proposta por Park et al. [2001], o campo foi dividido quadrantes de ataque e defesa. Cada quadrante possui um único robô que planeja suas trajetórias utilizando a técnica de campos potenciais [Arkin, 1989]. Sendo que, para esta estratégia foi considerada a existência de áreas de intersecção entre os quadrantes e foi utilizado um algoritmo de aprendizado por reforço Sutton & Barto [1998] conhecido como Q-learning [Watkins, 1989] para selecionar qual robô deveria chutar a bola. Uma arquitetura chamada STP foi proposta por Browning et al. [2005], na qual três níveis de abstração foram considerados: (S) seqüência de movimentos para realizar uma tática; (T) o nível tático, no qual uma ação de alto nível é designada a um robô, e.g., defenda a linha, chute ao gol; (P) para sincronizar as táticas entre membros da equipe formando jogadas (*plays*). A seqüência dos movimentos do nível (S) é definida por uma máquina de estados [Lenser et al., 2002], no qual o estado depende da leitura dos sensores.

No presente trabalho, métodos de planejamento de trajetórias, normalmente aplicados a sistemas mono-robôs, como por exemplo, a técnica de Campos Potenciais [Arkin, 1989], baseada na arquitetura Esquema Motor e a técnica de Campos Potenciais Harmônicos [Connolly & Grupen, 1993], baseada na teoria do Problema de Valor de Contorno (PVC), são considerados para utilização em uma arquitetura multi-robôs.

Um aperfeiçoamento do método de planejamento de trajetórias através de PVC, denominado Campo Potencial Localmente Orientado (CPLO), é proposto neste trabalho como módulo de Planejamento de trajetórias em ambientes multi-robôs. Um módulo implementado por Sistema Baseado em Regras (SBR) também é proposto para definir as metas e restrições de funcionalidade de cada robô de uma equipe de múltiplos robôs. Por último um módulo Missão é proposto para definir a funcionalidade de cada unidade da equipe de robôs. Estes são os três principais módulos que irão compor Arquitetura de Controle Inteligente para múltiplos robôs denominada ACIn.

1.2 Objetivos

Este trabalho tem por objetivo principal propor uma arquitetura de controle capaz de gerenciar de forma inteligente um sistema robótico composto de múltiplos robôs. Para alcançar o objetivo principal, foram propostos:

- a investigação de Redes Neurais Artificiais como controlador de trajetórias para múltiplos robôs;
- o aperfeiçoamentos dos métodos de planejamento de trajetórias, Campos Potenciais e Campos Potenciais Harmônicos, para inclusão de múltiplos robôs;
- um aperfeiçoamento do método baseado em PVC para planejar trajetórias, o qual chamamos de Campos Potenciais Localmente Orientados;
- a inclusão na arquitetura ACIn de um SBR como controle estratégico, para cada método proposto, capaz de estabelecer os objetivos e restrições de comportamentos que cada robô deve seguir para realizar sua tarefa;
- a implementação de uma função que controle a velocidade das rodas dos robôs necessária para o ajuste de trajetória do robô;
- a avaliação do desempenho de cada arquitetura proposta composta da combinação Planejamento/Estratégia/Missão no controle de um time de futebol de robôs.

1.3 Organização da tese

Este trabalho está dividido em seis capítulos, sendo esta introdução o primeiro deles.

No Capítulo 2, são apresentados algumas arquiteturas de controle robótico que servem como base para a construção de arquiteturas multi-robôs.

No Capítulo 3, são apresentados algumas técnicas consideradas inteligentes que foram investigadas com o propósito de construir um planejador de trajetórias adequado para ambientes dinâmicos e com múltiplos robôs. Dentre estas técnicas estão as Redes Neurais Artificiais, Campos Potenciais e Campos Potenciais baseados em Problema do Valor de Contorno.

No Capítulo 4, é proposta a arquitetura de controle inteligente para múltiplos robôs - ACIn. Além disso, várias propostas utilizando diferentes técnicas para implementação desta arquitetura são apresentadas.

No Capítulo 5, os resultados experimentais, utilizando as técnicas de Redes Neurais Artificiais, Campos Potenciais e Campos Potenciais baseados em PVC, são apresentados e discutidos.

Finalmente, no Capítulo 6, são feitas algumas conclusões e considerações sobre as principais contribuições deste trabalho e também são apresentadas algumas sugestões de trabalhos futuros para os problemas que ficaram em aberto.

2

Arquiteturas de Controle

O projeto de um sistema de controle capaz de gerenciar com sucesso tarefas do mundo real é uma tarefa desafiadora. Tal sistema deveria possuir comportamentos inteligentes capazes de resolver diferentes problemas e utilizar suas próprias experiências de forma inteligente. Além disto, ele deve ser capaz de lidar com situações inesperadas reagindo instantaneamente a este estímulo. Neste sentido, várias abordagens de arquiteturas foram pesquisadas com o propósito se projetar um sistema de controle para robôs móveis autônomos.

Assim como existem várias abordagens, vários autores definem e classificam as arquiteturas de formas diferentes. Uma boa definição de arquitetura foi dada por Shaw & Garlan [1996]:

> *"A arquitetura define um sistema em termos de componentes computacionais e interações entre estes componentes."*

Entretanto várias outras definições podem ser encontradas na literatura, tais como:

> *"A arquitetura de um robô define como está organizada a tarefa de gerar ações a partir das percepções."* [Russell & Norvig, 1995]
>
> *"Arquitetura robótica é a disciplina dedicada ao projeto de robôs altamente específicos a partir de uma coleção de blocos de construção de software."* [Arkin, 1999]

Neste trabalho adotou-se a definição dada por Pettersson [1997] que compreende melhor a estrutura estudada:

"A arquitetura é um esboço ou uma abstração do sistema de controle enquanto que o sistema de controle é a realização da arquitetura."

As arquiteturas podem ser classificadas de várias formas diferentes, considerando sua representação, forma de armazenar conhecimento, etc. A forma mais comum de classificação pode ser observada na Figura 2.1. O lado esquerdo representa métodos que empregam raciocínio deliberativo e o lado direito representa controle reativo. Métodos que utilizam *raciocínio deliberativo* requerem relativamente um conhecimento completo do mundo e usam este conhecimento para predizer o resultado das ações, uma habilidade que capacita a otimização de desempenho relativo ao modelo do mundo. Já os métodos com *controle reativo* não necessitam de modelos do mundo. Eles simplesmente reagem às contingências do ambiente, fornecendo respostas imediatas mesmo para situações desconhecidas.

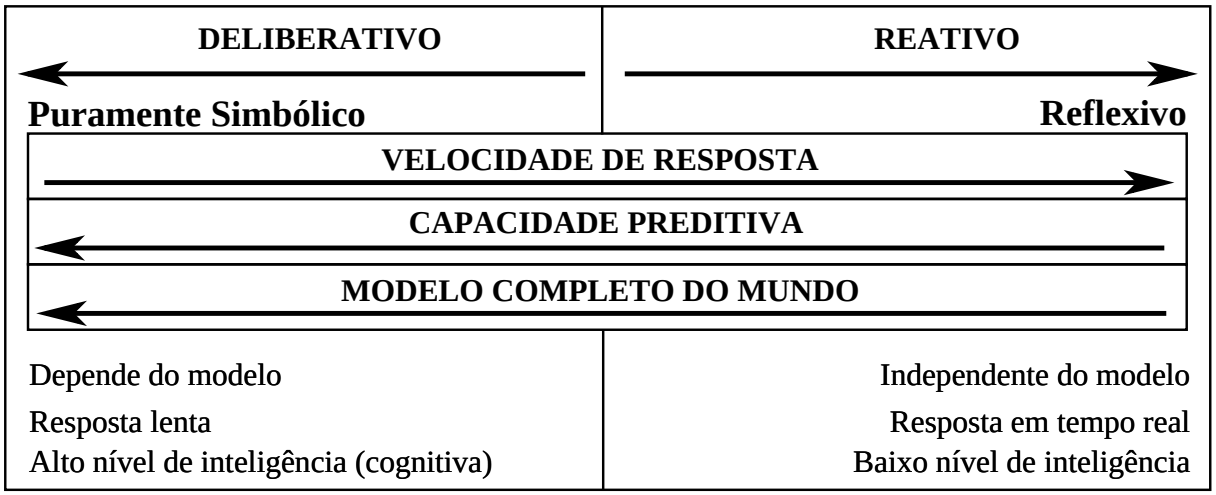


Figura 2.1: Representação simbólica dos sistemas de controle de robôs [Arkin, 1999].

Seguindo-se a representação mostrada na Figura 2.1, costuma-se classificar as arquiteturas de controle robótico em: reativas, deliberativas ou híbridas. A seguir, serão apresentadas algumas das arquiteturas mais conhecidas segundo esta classificação.

2.1 Arquiteturas Deliberativas

Uma estratégia para construção de arquiteturas de controle é baseada em técnicas tradicionais de Inteligência Artificial. Ou seja, utiliza-se a representação simbólica para representar metas, planos, conhecimento, etc. A estratégia de projeto desta arquitetura é freqüentemente chamada de SMPA (Sentir–Modelar–Planejar–Agir).

Observando-se o mundo biológico, pode-se perceber que não existe um controle pu-

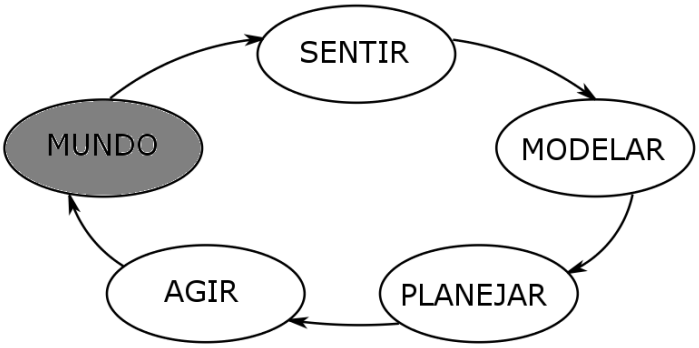


Figura 2.2: Modelo SMPA – Sentir-Modelar-Planejar-Agir.

ramente deliberativo. No entanto os modelos deliberativos são os preferidos quando o mundo pode ser modelado precisamente, as incertezas são restritas e existe a certeza de que o ambiente não mudará durante a execução. Isto fornece várias vantagens aos sistemas com controle deliberativo, como por exemplo:

- atuar com estratégias perceptuais e comportamentais que podem ser representadas como módulos configuráveis para varias missões e ambientes diferentes, adicionando assim versatilidade;
- o conhecimento do mundo, quando conhecido e estável, pode ser usado para configurar ou reconfigurar eficientemente estes comportamentos;
- modelos adquiridos dinamicamente podem ser usados para prevenir certos problemas nos quais métodos não representacionais estão sujeitos.

Abaixo estão descritas algumas particularidades das arquiteturas deliberativas mais conhecidas: SOAR, Blackboard, PRODIGY e NASREM.

2.1.1 SOAR

SOAR[1] é uma teoria, implementada como uma arquitetura cognitiva de propósito geral, para auxiliar a descrever e implementar componentes de inteligência básicos e funcionais, produzindo assim sistemas que demonstram um comportamento inteligente [Laird et al., 1987]. A arquitetura SOAR é classificada como deliberativa e seu aprendizado é realizado por meio de representações simbólicas do mundo.

Pesquisadores em todo mundo, seja do campo de Inteligência Artificial (IA) ou ciência cognitiva[Newell, 1990], utilizam o SOAR para uma variedade de tarefas, produzindo pes-

[1]SOAR sigla em inglês de *State, Operator And Result*

quisas destinadas à construção de sistemas inteligentes que interagem autonomamente em ambientes complexos que sejam habitados por outras entidades, incluindo outros agentes inteligentes ou agentes humanos.

Para proporcionar inteligência racional, a arquitetura SOAR utiliza todo o conhecimento disponível para cada tarefa que o sistema encontrar. Infelizmente, a complexidade de encontrar conhecimentos relevantes deixa esta meta fora de alcance, pois o crescimento da base de conhecimento e o pedido de tarefas mais diversas acabam por deixar o tempo de resposta do sistema limitado. O melhor que pode ser obtido atualmente é uma aproximação de um raciocínio completo. O projeto do SOAR pode ser visto como uma investigação desta aproximação. Abaixo estão os princípios que são a base do projeto SOAR e que guiam os esforços em aproximá-lo de um comportamento racional.

- O número de mecanismos arquiteturais deve ser minimizado. No SOAR existe uma estrutura simples para todas as tarefas e subtarefas (espaço de problemas), uma representação simples do conhecimento permanente (regras de produção), uma representação simples para conhecimento temporário (objetos com atributos e valores), um mecanismo simples para gerar metas (submetas automáticas) e um mecanismo simples de aprendizado (chunking)[2].
- Todas as decisões são tomadas através da combinação de conhecimentos relevantes em tempo real. No SOAR, todas as decisões são baseadas na interpretação dos dados atualizados obtidos dos sensores, no conteúdo da memória de trabalho criada por um problema resolvido *a priori* e em qualquer conhecimento relevante retirado do conhecimento permanente.

2.1.2 Blackboard

Arquiteturas Blackboard são construídas utilizando-se a idéia de módulos distribuídos (sensoriamento, atuação e raciocínio) comunicando-se através de uma memória comum. Esta bordagem torna o sistema modular e facilita o projeto paralelo dos módulos.

Carver & Lesser [1992] descreveram um modelo básico de uma arquitetura blackboard como sendo composto por três componentes principais: o *blackboard*, um conjunto de *fontes de conhecimento* (FCs) e um mecanismo de controle (Figura 2.3). O blackboard é uma base de dados global compartilhada por todos os FCs, contendo dados e hipóteses

[2]Chunking é essencialmente uma dedução, um mecanismo de aprendizado composicional. Uma nova associação é composta por uma parte *"se"* contendo o conhecimento a priori que contribuiu para a dedução e uma parte *"então"* contendo a dedução

(soluções parciais). O blackboard é estruturado como uma hierarquia (flexível) de níveis, nas quais classes de hipóteses são associadas com cada nível, sendo que as hipóteses são tipicamente ligadas a hipóteses de outros níveis.

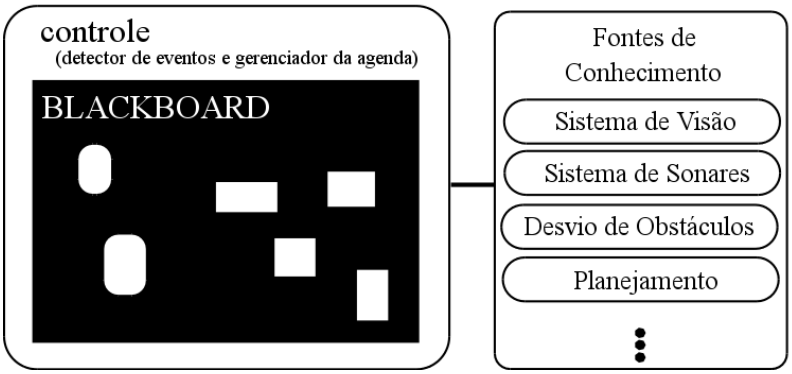


Figura 2.3: Arquitetura de controle Blackboard.

FCs são subsistemas como por exemplo: sistema de visão, sistema de sonar, sistema de desvio de obstáculos, sistema de planejamento. A chave principal da arquitetura blackboard é que toda comunicação entre subsistemas é realizada somente pelo blackboard. Os FCs examinam o estado do blackboard e criam novas hipóteses, ou modificam hipóteses existentes, quando for apropriado. Idealmente, os FCs devem ser independentes: suas execuções não devem depender explicitamente da execução de outros FCs e toda comunicação de informações entre os FCs deve ocorrer somente pela criação e/ou modificação das hipóteses no blackboard.

A arquitetura blackboard pode ser classificada como uma arquitetura deliberativa embora não utilize uma abordagem tradicional de produção de regras.

2.1.3 PRODIGY

O PRODIGY é uma arquitetura deliberativa baseada na representação simbólica do conhecimento, tendo sida inicialmente concebida por Carbonell et al. [1989] como sendo um sistema de IA para testar e desenvolver raciocínio sobre as regras de aprendizado de máquina e para resolução de problemas e planejamento. O PRODIGY consiste de um núcleo de um planejador de propósito geral, uma base de conhecimento e muitos módulos de aprendizagem projetados para reduzir o tempo de planejamento, melhorar a qualidades dos planos gerados e refinar o conhecimento sobre o domínio de planejamento. Na Figura 2.4 são mostrados os módulos desenvolvidos no PRODIGY segundo suas metas de aprendizado. O domínio de planejamento é especificado para o planejador do PRODIGY como um conjunto de *operadores*. Cada operador corresponde a uma ação de planejamento (atômica e generalizada) descrita em termos de seus efeitos e condições

necessárias. Um problema de planejamento em um domínio é apresentado ao PRODIGY como sendo uma configuração inicial do mundo e um conjunto de instruções para metas a serem alcançadas. A versão PRODIGY 4.0 utiliza um planejador não linear completo, que permite uma análise do raciocínio através de busca, com encadeamento para trás, sobre múltiplas metas e múltiplos operadores alternativos para atingir a meta. Os operadores podem ser organizados em diferentes níveis de abstração, sendo utilizados pelo PRODIGY 4.0 para planejar hierarquicamente.

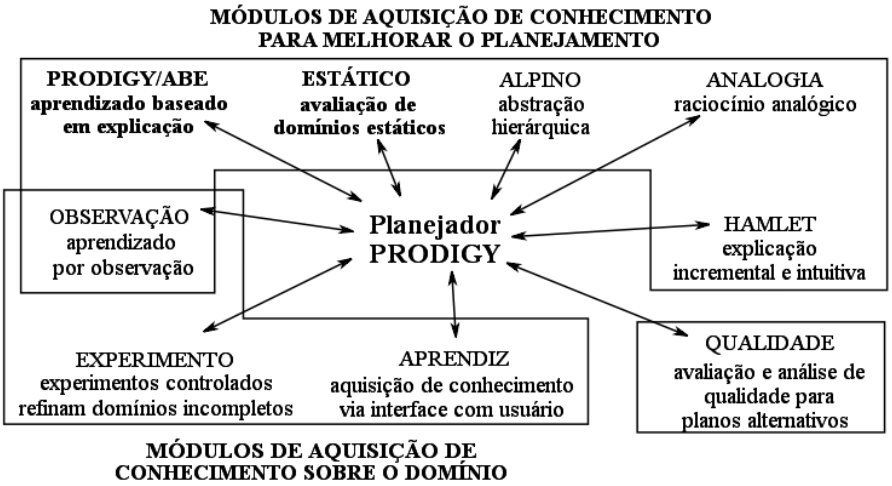


Figura 2.4: Módulos de aprendizado na arquitetura PRODIGY .

Abaixo estão descritas as funcionalidades dos módulos apresentados na Figura 2.4.

1. **Módulos de aquisição de conhecimento para melhorar o planejamento**

   - **ABE** – Aprendizado Baseado em Explicação [Minton, 1988] – é utilizado para aquisição de regras de controle a partir do caminho percorrido na resolução do problema. O ABE analisa o espaço de busca do algoritmo de planejamento e explica o raciocínio realizado nas derivações feitas durante a busca pela solução. As explicações resultantes são expressadas na forma de regras de controle.
   - **ESTÁTICO** é um método para aprender regras de controle através da análise das descrições do domínio antes de iniciar o planejamento.
   - **DINÂMICO** é uma técnica que combina o ABE e o ESTÁTICO para a aquisição de regras.
   - **ALPINO** é uma abstração do módulo de planejamento e aprendizado [Knoblock, 1994]. O ALPINO divide a descrição do domínio em múltiplos níveis de abstração. O algoritmo de planejamento utiliza estas abstrações para encontrar

um "esboço" de um plano para a solução do problema e então refina este esboço adicionando os detalhes necessários.

- **ANALOGIA** este módulo resolve problemas através de um motor de comparação derivacional utilizando-se do conhecimento sobre problemas similares anteriormente resolvidos.

2. **Módulos de aquisição de conhecimento sobre o domínio**

   - **EXPERIMENTO** é um módulo de aprendizado por experimentação para refinar a descrição do domínio caso suas propriedades não estejam completamente especificadas. Este módulo é utilizado quando o PRODIGY nota diferenças entre a descrição do domínio e o comportamento do domínio real.
   - **OBSERVAÇÃO** é um método que acumula conhecimento sobre planejamento no domínio através da observação de um agente especialista, do planejamento e a partir de sua própria prática. As observações de um agente consistem:
     (a) das seqüências de ações que estão sendo executadas;
     (b) dos estados nos quais cada ação é executada e,
     (c) do estado resultante a partir da execução de cada ação.
   - **APRENDIZ** é uma interface gráfica que permite ao usuário avaliar e guiar o planejamento do sistema e seu aprendizado. O sistema não só adquire conhecimento sobre o domínio através do usuário como também utiliza os conselhos do usuário para guiar a busca da solução do problema corrente.

3. **Aprendizado para melhorar a qualidade do plano**
   O aprendizado de conhecimentos para melhorar a qualidade do plano foi construído utilizando-se métricas de qualidade relacionadas ao custo de execução do plano. Este custo é expressado como uma função de avaliação linear acrescida sobre o custo dos operadores individuais. Duas estratégias foram implementadas para resolver este problema de aprendizagem:

   - **QUALIDADE** é um módulo no qual uma função de avaliação identifica quais dos muitos planos alternativos é o de melhor qualidade.
   - **HAMLET** adquire conhecimento de controle para guiar o PRODIGY na produção de planos. O HAMLET aprende através da exploração exaustiva do espaço de busca em problemas simples, explica, sem precisão, as condições de sucesso de qualidade e refina o seu conhecimento aprendido por experimentação.

2.1.4 NASREM

A arquitetura de controle telerobótico NASA/NBS Standard Reference Model (NASREM)[Albus et al., 1989] está organizada hierarquicamente em múltiplos níveis, como mostrado na Figura 2.5, de forma que diferentes transformações são realizadas em cada nível. No nível um, coordenações são transformadas e as saídas são executadas nos atuadores. No nível dois, dinâmicas mecânicas são computadas. No nível três, encontra-se obstáculos e desvia-se deles. No nível quatro, tarefas relacionadas a objetos são transformadas em movimentos para os atuadores. No nível cinco, as tarefas relacionadas a grupos de objetos são colocadas em seqüencia e escalonadas. No nível seis, objetos são agrupados em grupos, recursos e ferramentas são alocados para um serviço. Níveis mais altos também são possíveis e foram estudados e implementados no NBS AMRF.

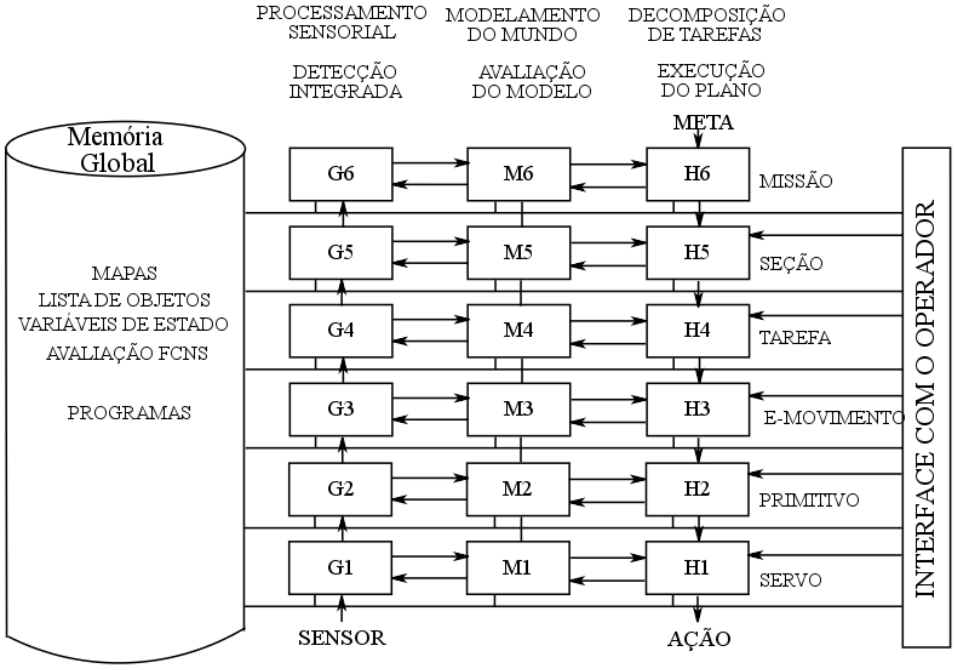


Figura 2.5: NASREM – Uma arquitetura hierárquica para controle telerobótico.

A arquitetura de controle robótico NASREM também é horizontal, particionada em três seções: decomposição de tarefas, modelamento do mundo e processamento sensorial. A decomposição de tarefas inclui o planejamento e o monitoramento das tarefas, as decisões guiadas por valor, o controle dos atuadores e a interface com o operador. O modelamento do mundo compreende modelos CAD[3] dos objetos e estruturas, mapas das área, lista de objetos com suas características e atributos, tabela das variáveis de estado – as quais descrevem o sistema e o ambiente. O processamento sensorial inclui

[3]Sigla do termo *Computer-Aided-Design*

o processamento de sinal, a detecção de padrões, o reconhecimento de características, o reconhecimento de objetos, os relacionamentos, as correlações e as diferenciações entre observação e expectativa.

2.2 Arquiteturas Reativas

Este paradigma foi introduzido por Brooks [1986] e é caracterizado por rejeitar, ou minimizar, as informações de estados internos, bem como as informações sobre um modelo do mundo. Os módulos de uma arquitetura reativa são comportamentos (em níveis) como evitar obstáculos, identificar objetos, explorar o ambiente, etc. Entretanto, funções como planejamento e aprendizado surgem a partir da interação dos módulos com o ambiente. Sua vantagem é que apenas um comportamento, em geral pequeno, é ativado por vez, e nenhum modelo do mundo é criado ou mantido, reduzindo assim o tempo de processamento.

As arquiteturas reativas são também chamadas de *arquiteturas baseadas em comportamentos* quando incluem comportamentos mais elaborados do que simplesmente uma reação a partir de um estímulo externo.

As arquiteturas puramente reativas podem ser criticadas por serem muito simples para produzir comportamentos complexos como aprendizado. No entanto, Brooks [1991] afirma que não há necessidade de representações simbólicas para produzir comportamentos eficientes, nem mesmo para produzir comportamentos inteligentes ou aprendizado.

O Subsumption e o esquema motor são exemplos de arquiteturas reativas, as quais são descritas abaixo.

2.2.1 Subsumption

Desenvolvida por Brooks [1986], a arquitetura Subsumption é um método baseado em comportamento puramente reativo. Ele afirma que o paradigma deliberativo SMPA prejudicou a construção de robôs reais, devido ao atraso causado pelo mapeamento do mundo em representações simbólicas de conhecimento.

O estilo da arquitetura Subsumption segue alguns dogmas, dentre eles:

- Comportamentos complexos não necessitam necessariamente ser produto de um sistema de controle complexo;
- "A inteligência está nos olhos do observador"[Brooks, 1991];
- "O mundo é o melhor modelo"[Brooks, 1991];

- O planejamento é apenas uma forma de evitar a compreensão do passo seguinte;
- O sistema deve ser robusto na presença de ruídos ou falha dos sensores;
- Os sistemas devem ser construídos incrementalmente;
- Sem representação. Sem calibração. Sem computadores complexos. Sem comunicação com pacotes enormes.

Nas arquiteturas Subsumption, os comportamentos para realizar tarefas são representados em níveis separados. Níveis individuais trabalham com objetivos individuais de forma concorrente e assíncrona. Não existe uma memória global, barramento ou clock, desta forma o projeto de cada nível de comportamento pode ser mapeado em seu próprio processador. Toda informação deve ser obtida diretamente dos sensores, pois a arquitetura não mantém modelos do mundo ou de sensores.

A coordenação dos comportamentos é fixada pela topologia hierárquica, na qual os baixos níveis não têm consciência dos níveis mais altos, fornecendo assim uma base sólida para um desenvolvimento incremental. Alguns mecanismos são utilizados para coordenar os comportamentos, fazendo com que somente o sinal de um único comportamento atinja os atuadores: **Inibição** (I) impede que um sinal transmitido por um comportamento alcance o atuador, **Supressão** (S) impede o sinal que está sendo transmitido e o substitui pelo sinal da mensagem de supressão e **Reiniciar** (R) restaura o comportamento ao estado original. Para entender melhor o funcionamento destes mecanismos, um exemplo da arquitetura Subsumption é mostrado na Figura 2.6.

O NÍVEL 1 é responsável por navegar sem considerar os obstáculos, sendo que:

- o módulo *motor* executa o comando recebido alterando direção e velocidade quando necessário ou parando o robô caso um mensagem “parar” seja recebida;
- o módulo *sonar* recebe os dados dos sensores e fornece um mapa das posições dos obstáculos em coordenadas polares;
- o módulo *repulsão* calcula um vetor força resultante a partir do mapa recebido; o módulo *andar* envia um comando para o *motor* se a força resultante for significativa; e
- o módulo *colisão* faz o robô parar caso o robô esteja em movimento e exista objetos imóveis a frente.

O NÍVEL 2 executa uma exploração evitando obstáculos, no qual o módulo *vagar* gera direções aleatórias a cada 10 segundos e o módulo *escapar* combina a direção recebida de

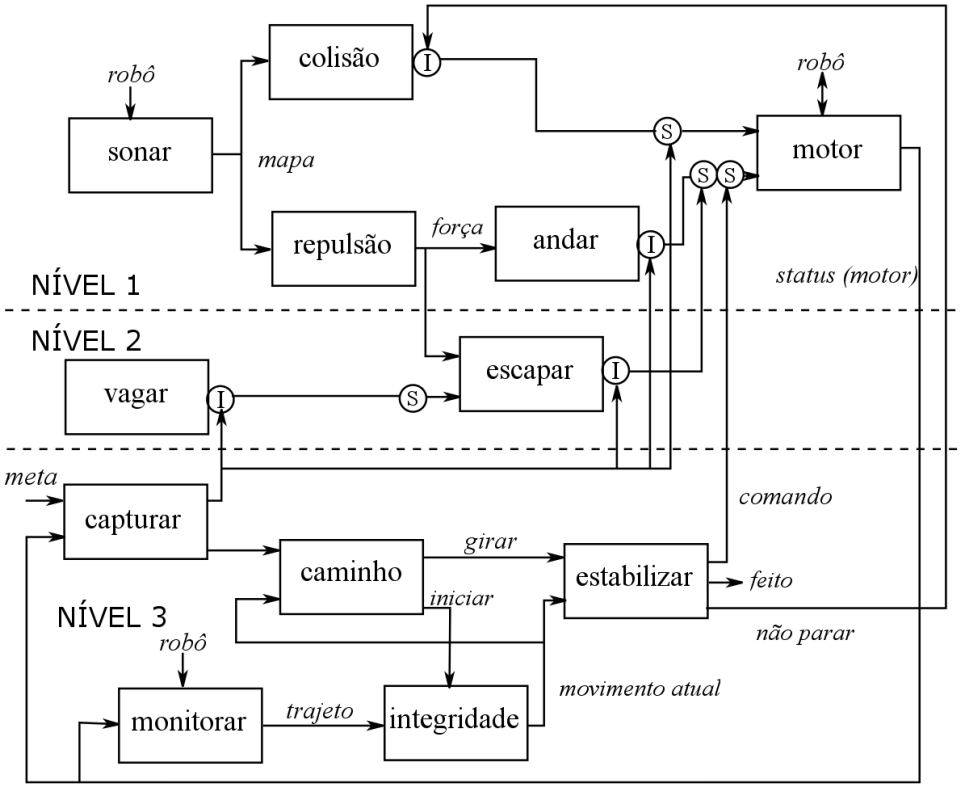


Figura 2.6: Arquitetura Subsumption: navegador para um robô móvel [Brooks, 1986].

vagar e *repulsão* para evitar colidir com objetos já detectados. Observe que ele suprime o módulo *andar*.

O NÍVEL 3 faz com que o robô chegue até um local desejado e possui os seguintes módulos:

- *capturar* força uma parada para garantir que este nível tenha o controle do *motor*, contudo o atributo meta recebido pelo *capturar* provém de uma computação externa que reconhece um caminho livre [Grimson, 1985] e traça uma linha até a meta;
- *planejar* calcula o ângulo e a distância até a meta;
- *estabilizar* permite uma aproximação refinada do alvo suprimindo qualquer outro comando, evitando inclusive uma parada indesejável;
- *monitorar* rastreia cada movimento, sendo capaz de descrever o caminho percorrido;
- *integridade* adiciona um monitoramento extra ao *planejar*, isto faz com que o robô pare e recolha novas informações sensoriais caso ele esteja fora do caminho livre detectado.

2.2.2 Esquema Motor

Arkin [1989] propôs uma decomposição de comportamentos em esquemas seguindo as idéias de Arbib & House [1987]. Cada esquema motor possui um esquema de percepção embutido que fornece a informação de percepção necessária. Algoritmos perceptuais fornecem as informações necessárias para o esquema no qual está inserido. Desta forma, esquemas perceptuais são definidos recursivamente, ou seja, um subesquema de percepção pode extrair pedaços de informações que serão subseqüentemente processados por outro esquema de percepção. A informação gerada em cada processo de percepção de baixo nível seria fundida em um esquema de interpretação de percepções de alto nível antes de passá-lo para o esquema motor correspondente a algum comportamento. Isto permite o uso de múltiplos sensores dentro de um contexto de comportamentos sensor/motor.

A saída de um esquema motor de navegação poderia ser, por exemplo, um vetor com a orientação e velocidade desejadas. Desta forma, o comportamento de navegação poderia ser obtido pela combinação dos esquemas: evitar obstáculos em movimento, evitar obstáculos estáticos, manter-se no caminho e mover-se para a meta. Neste exemplo, cada esquema pode ser implementado como um campo potencial, calculando um vetor força, composto de forças de repulsão aos obstáculos e uma força de atração pela meta. Assim, o comando de movimento seria dado através da superposição de todos os esquemas, ou seja, uma soma vetorial. Na Figura 2.7 o funcionamento dos relacionamentos entre os esquemas de percepção e ação (motor) pode ser observado.

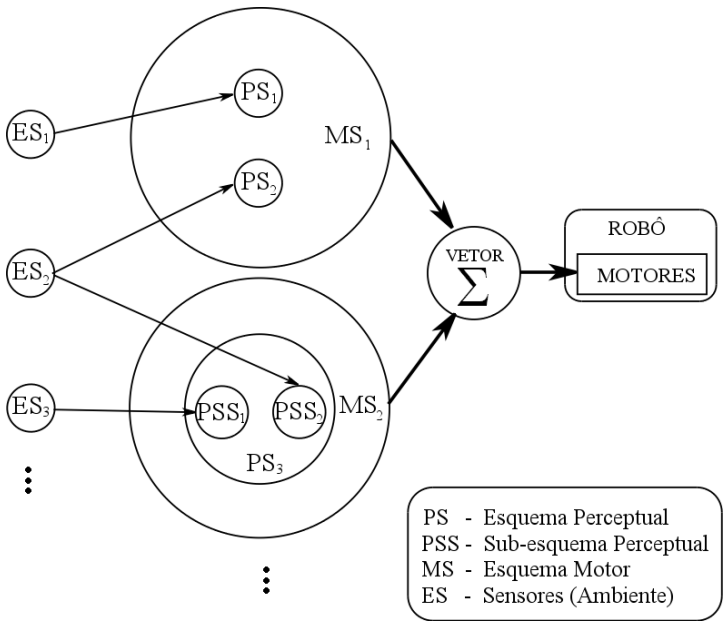


Figura 2.7: Relacionamento entre esquemas de percepção e ação.

2.3 Arquiteturas Híbridas

Embora o controle reativo lide bem com dados sensoriais imediatos e tenha um desempenho robusto em domínios dinâmicos e complexos, ele é menos efetivo em integrar conhecimentos do mundo. Além disso, a falta de um modelo preciso do mundo não permite que um robô, utilizando-se de uma abordagem reativa, consiga realizar a tarefa de localizar o robô relativo em um modelo de mundo [Arkin, 1999].

Contudo, uma arquitetura deliberativa pode não ser capaz de responder rapidamente e eficientemente às mudanças dinâmicas e não modeladas que ocorrem no mundo. Assim, um controle deliberativo pode acabar ficando restrito a nichos específicos.

Se um sistema puramente deliberativo tentar modelar e pré-planejar todas as eventualidades, corre-se o risco dele nunca terminar (*qualification problem*). Também não é seguro um sistema que faz suposições gerais sobre o mundo e que não reflete a sua natureza dinâmica. Neste contexto, surgem as arquiteturas híbridas que integram um componente deliberativo para planejamento e tomada de decisão e um componente reativo para manter a flexibilidade das respostas em tempo real.

> *"Nenhuma abordagem é completamente satisfatória isoladamente, mas ambas, reativa e deliberativa, devem ser levadas em conta para produzir um sistema flexível, robusto e inteligente".* [Arkin, 1999].

Através da habilidade de raciocínio sobre componentes de comportamento, as arquiteturas híbridas permitem a reconfiguração de sistemas de controle reativo baseado no conhecimento do mundo. Isto torna a reconfiguração dinâmica baseada em deliberação (raciocínio sobre modelos do mundo) uma importante adição para robôs de propósito geral.

Dentre as arquiteturas híbridas estão: AuRA, NOS, Atlantis, DAMN e TCA. Cada um destes exemplos são apresentados a seguir.

2.3.1 AuRA

A arquitetura AuRA[4] foi desenvolvida em meados dos anos 80 como uma abordagem híbrida para a navegação de robôs [Arkin, 1987]. O componente deliberativo é um planejador hierárquico baseado nas técnicas tradicionais de IA e o componente reativo é baseado na teoria de esquemas [Arbib & House, 1987; Arbib, 1972].

[4]Sigla em inglês de *Autonomous Robot Architecture*

Os componentes da arquitetura AuRA estão descritos esquematicamente na Figura 2.8.

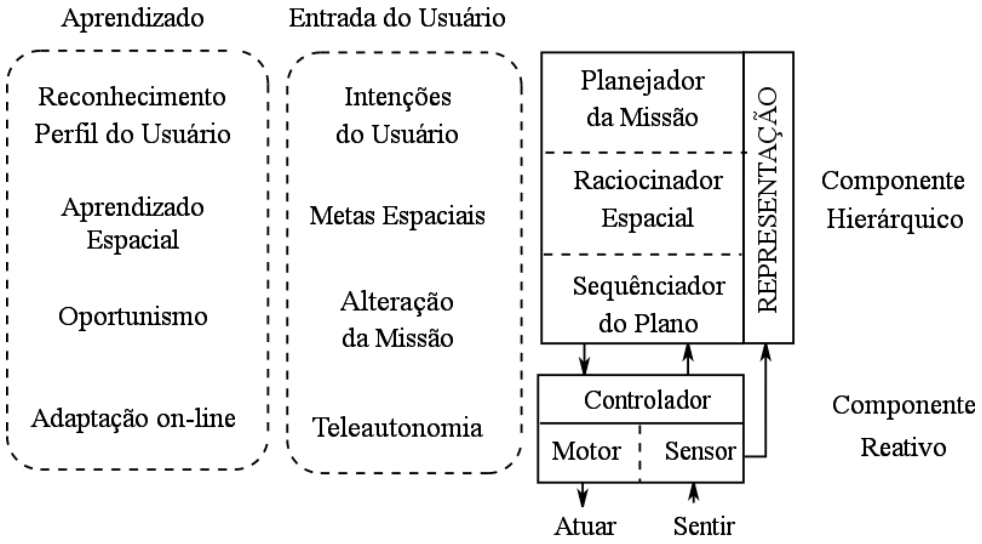


Figura 2.8: Arquitetura AuRA em uma representação de alto nível.

No nível mais alto da arquitetura AuRA está o *planejador da missão*, que é responsável por estabelecer as metas de alto nível e as restrições com as quais o robô deve operar. Nos sistemas baseados no AuRA este nível tem funcionado como uma interface entre o usuário e o sistema. O *raciocinador espacial*, definido inicialmente como navegador por Arkin [1987], utiliza conhecimento cartográfico armazenado em uma memória para criar os passos que o robô deve seguir para completar sua missão. O *seqüenciador do plano*, também referenciado como piloto nos primeiros trabalhos, traduz cada passo gerado pelo raciocinador espacial em um conjunto de comportamos motores a serem executados. Finalmente o conjunto de comportamentos (esquema motor), especificados e inicializados pelo seqüenciador do plano, é enviado para execução. Neste ponto cessa-se a deliberação e inicia-se o processo reativo.

O Controlador Esquema é responsável por controlar e monitorar os processos comportamentais em tempo de execução. Cada comportamento motor (ou esquema) está associado com um esquema de percepção capaz de fornecer o estímulo necessário para um comportamento específico. Cada esquema motor gera um vetor resposta de maneira análoga ao método de campos potenciais, como pode ser observado na Figura 2.7.

2.3.2 Atlantis

A arquitetura Atlantis [Gat, 1992], implementada na NASA em cooperação com Cal-Tech, introduziu o conceito de falha ciente na qual um robô torna-se consciente de suas inabilidades para completar a tarefa.

Atlantis é uma arquitetura híbrida de três níveis: um deliberador, um sequenciador e um controlador (Figura 2.9). Esta arquitetura foi projetada para agentes que operam em um mundo dinâmico. Os níveis diferentes permitem que o agente reaja de várias formas (deliberativo ou reativo), funcionando inclusive em ambientes imprevisíveis e ambientes hostis.

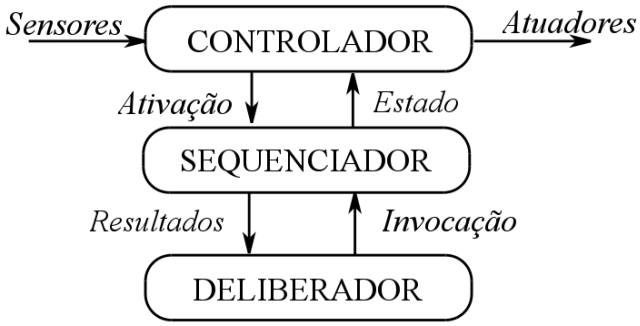


Figura 2.9: Arquitetura Atlantis

O nível *controlador* é responsável pelo controle dos sensores e atuadores, sendo bastante similar a arquitetura Subsumption. O nível *seqüenciador* é o centro de controle principal de um agente robótico com esta arquitetura, pois é o responsável por avisar ao nível de controle quando iniciar e terminar os comportamentos primitivos. O nível *deliberador* contém um modelo simbólico do mundo, não disponível aos outros níveis, e é o responsável por produzir um plano que realize uma determinada tarefa.

O aprendizado nesta arquitetura é limitado, pois todo aprendizado é realizado somente no nível deliberativo. Assim, novos conhecimentos terão seus efeitos notados somente nas capacidades de planejamento e predição sobre o modelo do mundo.

2.3.3 DAMN

O DAMN[5] é uma arquitetura distribuída destinada à navegação desenvolvida por Rosenblatt [1995] e projetada para que vários comportamentos possam ser facilmente adicionados ou removidos do sistema, dependendo da tarefa a ser realizada. Ela foi criada

[5]Sigla do termo *Distributed Architecture for Mobile Navigation*

para realizar a fusão centralizada de comandos provenientes de processos de tomada de decisão que rodam independentes e de forma distribuída. Tais características provêm reatividade e robustês aos sistemas baseados em comportamento sem sacrificar a coerência e racionalidade das arquiteturas centralizadas.

A organização da arquitetura DAMN, mostrada na Figura 2.10, consiste de um grupo de módulos de raciocínio distribuídos que enviam votos, para um árbitro central, a favor das ações que satisfazem seus objetivos e contra as ações que não os ajudam. O árbitro é o responsável por combinar os votos dos módulos e gerar ações que reflitam seus objetivos e prioridades.

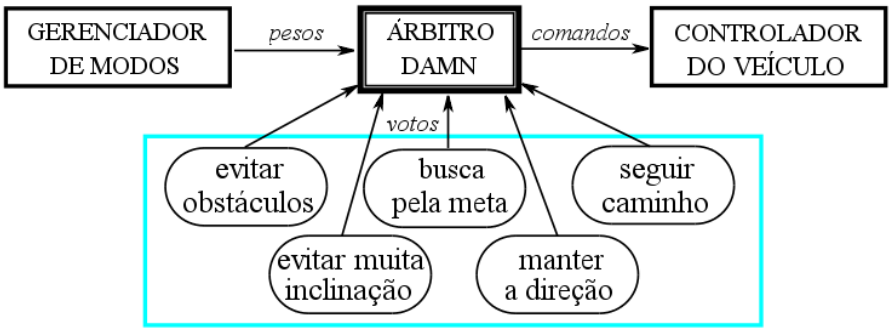


Figura 2.10: Arquitetura DAMN: Arbitro e Comportamentos.

Os comportamentos operam de forma assíncrona e em paralelo com os outros comportamentos. Seus votos são periodicamente combinados por um árbitro e o comando resultante é enviado ao controlador. Cada comportamento possui um peso que reflete sua prioridade relativa ao controle do veículo, contudo, são os votos de todos os comportamentos que determinam a próxima ação a ser executada. Além disso, um modo gerenciador também pode ser utilizado para variar os pesos durante o curso da missão baseando-se no conhecimento da relevância de cada comportamento em dada situação.

2.3.4 TCA

A Arquitetura de Controle de Tarefas (TCA[6]) [Simmons, 1992a,b] provê uma estrutura genérica para o controle distribuído de sistemas robóticos. Em essência, o TCA é um sistema de operação robótico de alto nível que integra um conjunto de mecanismos necessários à comunicação distribuída, decomposição de tarefas, gerenciamento dos dispositivos, monitoramento da execução e recuperação de erros.

Um sistema utilizando TCA consiste de um conjunto de módulos específicos para cada tarefa e um módulo de propósito geral utilizado para o controle central. Os módulos

[6]Sigla do termo *Task Control Architecture*

se comunicam uns com os outros (e com o controle central) através da passagem de mensagens. Na Figura 2.11 pode ser visto um exemplo de um sistema robótico com TCA.

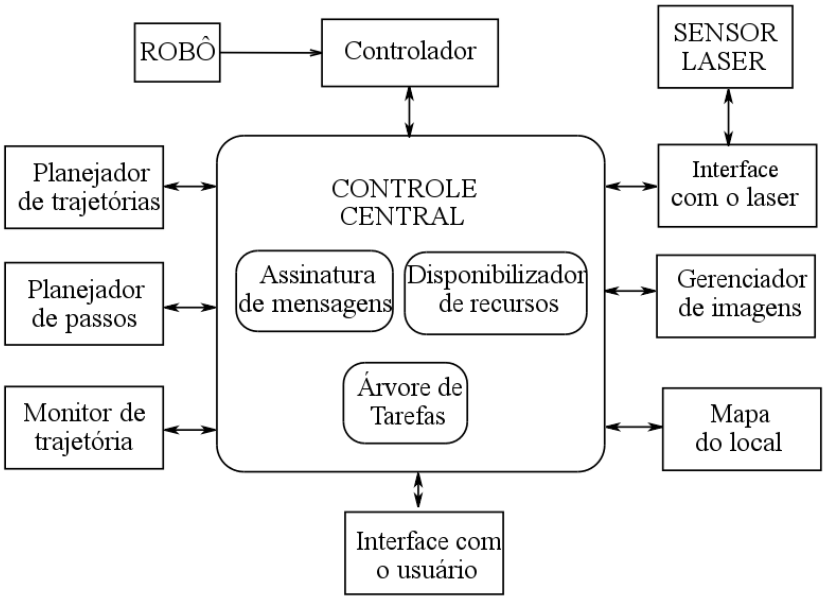


Figura 2.11: A arquitetura TCA.

A arquitetura TCA retém muitos conceitos da arquitetura Blackborad por possuir um controle centralizado da informação. No entanto, os dados necessários para resolver problemas são distribuídos entre os processos do sistema.

2.4 Considerações

Como foi visto, existem várias abordagens para implementação de uma arquitetura de controle, cada qual possuindo características que influem no comportamento final do robô móvel. A função principal de cada uma das arquiteturas apresentadas é disponibilizar um planejamento de trajetórias eficiente para atuação em ambientes dinâmicos. No próximo capítulo, serão apresentadas alguns técnicas inteligentes para obtenção de um sistemas de planejamento de trajetórias para a arquitetura ACIn.

3

Técnicas Inteligentes e de Planejamento

Neste capítulo será apresentada uma revisão sobre as técnicas que foram utilizadas neste trabalho com objetivo de obter um controlador eficiente para o planejamento de trajetórias de um ou mais robôs. O capítulo inicia com uma descrição das Redes Neurais com foco sobre as Redes Perceptons Multicamadas, em seguida dois modelos para geração de campos potenciais são apresentados: o modelo tradicional (reativo) e o modelo baseado nas soluções do Problema de Valor de Contorno (PVC).

3.1 Redes Neurais Artificiais

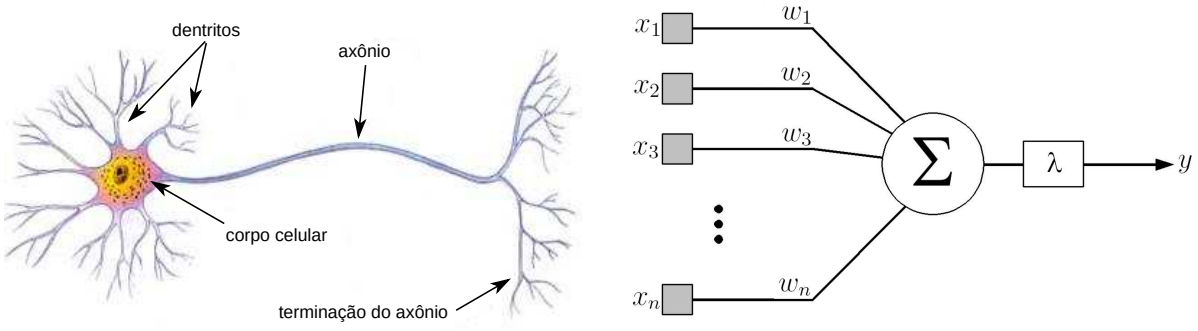
Uma Rede Neural Artificial (RNA) é um sistema computacional inspirado na estrutura, método de processamento e habilidade de aprendizado de um cérebro biológico [Cybenko, 1996]. As semelhanças estão no fato do conhecimento ser adquirido através de um processo de aprendizado e das ligações sinápticas excitatórias e inibitórias serem as responsáveis por armazenar o conhecimento. Nesta seção serão apresentados o modelo matemático do neurônio artificial, seu funcionamento e uma das RNAs mais populares, a rede Perceptron Multicamadas (MLP)[1].

3.1.1 O Neurônio Artificial

O primeiro modelo matemático do neurônio foi desenvolvido em 1943 pelos pesquisadores Warren McCulloch e Walter Pitts [McCulloch & Pitts, 1943]. Assim como o neurônio biológico mostrado na Figura 3.1(a), o neurônio conhecido como nó MCP (Figura 3.1(b)) é composto por diversas entradas (dendritos) ponderadas por pesos (simulando o comportamento das (sinapses), um somador (corpo celular) e uma saída (axônio). Na Figura 3.1(b),

[1]Multilayer Perceptron

λ representa o limiar de ativação do neurônio.



(a) (b)

Figura 3.1: (a) Neurônio biológico; (b) O neurônio artificial de McCulloch e Pitts.

Matematicamente, a saída do neurônio MCP se torna ativa quando:

$$\sum_{i=1}^{n}(x_i w_i) \geq \lambda \tag{3.1}$$

no qual n representa o número de entradas (dendritos), x as entradas, w os pesos associados as sinapses (quando é w positivo a sinapse é excitatória, quando negativo é inibitória) e λ representa o limiar de ativação, isto é, quando o somatório atingir o limiar λ, o neurônio se torna ativo. Matematicamente, a Equação 3.1 é o produto escalar (também conhecido como produto interno $x.w$) do vetor de pesos w com o vetor de entradas x.

Contudo, em sua definição original, o nó MCP foi proposto com pesos não-ajustáveis (fixos) [Braga et al., 2000]. Assim, alguns outros modelos artificiais foram propostos com o objetivo de eliminar tal limitação, como exemplo pode ser citado o *perceptron* e o Adaline. Em 1986, com o desenvolvimento do algoritmo de retropropagação pelos pesquisadores Rumelhart et al. [1986], foi definido um modelo de neurônio não-linear baseado no perceptron. Este neurônio, é atualmente o mais utilizado pela comunidade de redes neurais (Figura 3.2).

No neurônio não-linear representado pela Figura 3.2, x_i representa a entrada presente na sinapse i do neurônio, w_i o peso associado a sinapse i, v o somátório das entradas ponderadas pelos pesos acrescido do termo *bias* (Equação 3.2), $\varphi(.)$ a função de ativação não-linear do neurônio e y a sua saída $y = \varphi(v)$. As principais funções de ativação do neurônio serão apresentadas na Tabela 3.1.

$$v = (\sum_{i=1}^{n} x_i w_i) + bias \tag{3.2}$$

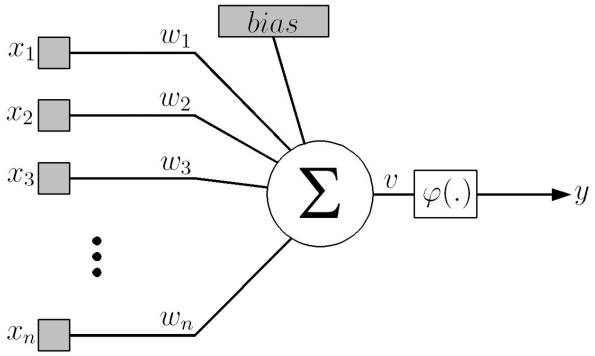
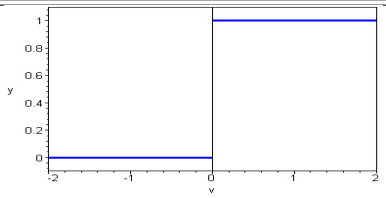
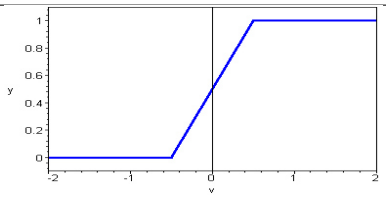
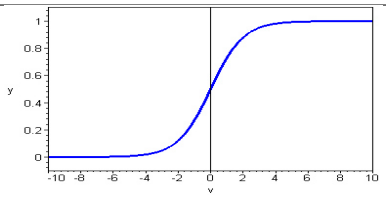
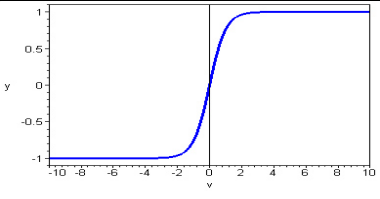


Figura 3.2: O neurônio não-linear

Tabela 3.1: Funções de Ativação do Neurônio

Tipo	**Função**	**Gráfico**
Limiar	$\varphi(v) = \begin{cases} 0 & \text{se} \quad v < 0 \\ 1 & \text{se} \quad v \geq 0 \end{cases}$	
Linear por Partes	$\varphi(v) = \begin{cases} 1 & \text{se} \quad v \geq \frac{1}{2} \\ v & \text{se} \quad -\frac{1}{2} < v < \frac{1}{2} \\ 0 & \text{se} \quad v \leq -\frac{1}{2} \end{cases}$	
Sigmóide Logística	$\varphi(v) = (1 + \exp(-a\ v))^{-1}$	
Tangente Hiperbólica	$\varphi(v) = \tanh(v)$	

O caráter dinâmico do aprendizado torna o uso de RNAs bastante atrativo, devido principalmente a capacidade destas de aprender, de aproximar funções, e de classificar padrões [Antsakalis, 1992].

A seguir será apresentado o modelo de Rede Neural que tem sido amplamente adotado

como sistema de apoio a tomada de decisão, chamado modelo Perceptron Multicamadas (MLP) [Haykin, 1994; Rosenblatt, 1963].

3.1.2 Redes Neurais MLP

As redes neurais MLP podem ser vistas como uma generalização das redes *perceptron* de uma única camada. Contudo, algumas diferenças podem ser destacadas:

- as MLPs utilizam funções de ativação não-lineares contínuas (diferenciaveis em todos os pontos) [Kovacs, 2002],
- podem resolver problemas não-linearmente separáveis [Haykin, 2001],
- por possuirem camadas ocultas, as MLPs podem mapear qualquer função matemática [Cybenko, 1988, 1989].

A arquitetura da rede MLP é apresentada na Figura 3.3, na qual é possível observar que a organização dos neurônios na rede é dada por camadas. A primeira camada (camada de entrada $l = 0$), é responsável por receber os sinais provenientes do ambiente (sensores, chaves, interrupções, etc.) e transmiti-lo à próxima camada ($l = 1$). Aquela não é propriamente uma camada da RNA, pois, não é constituída de neurônios como as demais camadas da rede (o sinal recebido pela camada $l = 0$ é o mesmo sinal transmitido à camada $l = 1$ ($y_j^0 = x_j$)). O fluxo de informação em uma rede MLP é propagado a partir da camada de entrada ($l = 0$), passando por todas as camadas ocultas até a camada de saída ($l = L$). Desta forma, as entradas de cada neurônio (N_i^l) das camadas ocultas e da camada de saída ($l = 1, 2, ..., L$) são as respectivas saídas (y_j) de todos os neurônios (N_j^{l-1}) da camada imediamente anterior ponderadas pelos respectivos pesos sinápticos (w_{ji}). A soma de todas as entradas (y_j) ponderadas pelos pesos (w_{ji}) é chamada de campo local induzido (v), representado pela equação 3.3.

$$v_j^l = (\sum_{i=1}^{m} w_{ji}^l y_i^{l-1}) + bias_j^l \tag{3.3}$$

no qual o termo *bias* representa a polarização do neurônio j.

Por sua vez, o sinal de saída do neurônio (N_i^l) é computado pela função de ativação $\varphi(.)$ (Equação 3.4).

$$y_j^l = \varphi(v_j^l) \tag{3.4}$$

no qual a função $\varphi(.)$ é representada por uma função do tipo sigmóide, comumente a função logística ou a função tangente hiperbólica é utilizada (Tabela 3.1).

O principal problema de uma rede MLP esteve justamente em como realizar o ajuste dos pesos w_{ji} das camadas ocultas da rede. Este problema esteve presente por toda a década de 70. Apenas em meados da década de 80, com o desenvolvimento do algoritmo de retropropagação do erro (*backpropagation*) pelos pesquisadores Rumelhart et al. [1986], foi possível realizar o treinamento de uma rede MLP.

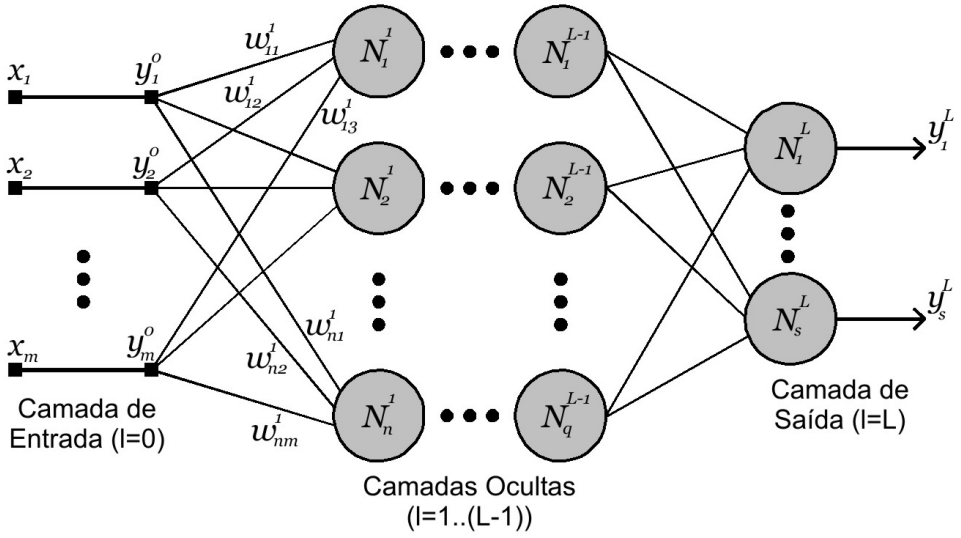


Figura 3.3: Arquitetura da rede neural MLP

3.1.3 Considerações

Apesar do algoritmo de retropropagação ser o mais utilizado, o treinamento através do mesmo introduz complexidades computacionais consideráveis devido à necessidade de se haver constantes cálculos de gradientes (diferenciações). Além disso, técnicas de otimização baseadas no gradiente, como o caso do método considerado, são limitadas pelo fato de realizarem uma busca pela menor solução local. Portanto, elas tipicamente alcançam a melhor solução na região do ponto em que começam a busca.

3.2 Campos Potenciais

Neste método, considera-se que existem forças atuando sobre um robô, sendo que obstáculos exercem forças repulsivas e a meta aplica uma força atrativa. A força resultante $\overrightarrow{F}$ é uma soma vetorial de uma força atrativa direcionada para a meta e forças repulsivas proveniente de obstáculos.

Por ser um método de baixo custo computacional, esta técnica tem sido muito utilizada para em planejamento de trajetórias de robôs móveis que necessitam navegar por ambientes desconhecidos e/ou dinâmicos evitando colisões com obstáculos. Vários pesquisadores têm apresentado propostas para melhorar a eficiência do método de campos potenciais.

Krogh [1984] aprimorou este conceito ao considerar que a velocidade do robô deve diminuir na vizinhança dos obstáculos para diminuir o risco de colisão. Thorpe [1984] aplicou o método de campos potenciais para planejamento *off-line* e Krogh & Thorpe [1986] sugeriram uma combinação do método para planejamento de caminhos locais e globais, esta nova proposta foi denominada por eles de Campos Potenciais Generalizado. Newman & Hogan [1987] introduziram a implementação de funções potenciais através da combinação de funções individuais de obstáculos com operadores lógicos.

Em todos os métodos citados foi assumido um modelo conhecido do mundo, com formas geométricas pré-definidas representando obstáculos ou o caminho do robô foi gerado *off-line*.

No entanto, Brooks [1986] e Arkin [1989] foram os pioneiros em utilizar a técnica de campos potenciais de forma reativa, ou seja, para cada novo passo do robô, uma nova força resultante deverá ser calculada para direcionar seu movimento. Esta característica reativa torna o método muito eficiente para robôs que realizam tarefas em ambientes desconhecidos e/ou com objetos dinâmicos. Neste modelo reativo, a cada instante de tempo alguns sensores captam o estado do ambiente e então uma força resultante é calculada para direcionar a trajetória do robô.

Brooks [1986] utilizou a técnica de campos potenciais como um controlador reflexivo. O seu controlador simplesmente reage executando ações diretamente ligadas as percepções, fornecendo assim respostas imediatas aos estímulos externos.

Um método similar, denominado Esquema Motor, foi utilizado por Arkin [1989] em seus robôs. Como visto na seção 2.2.2 (pág.16), este método consiste em ativar vários comportamentos simultâneos que produzem um vetor força (atração ou repulsão) como resposta. A direção a ser seguida pelo robô será o vetor resultante da soma de todos os vetores gerados.

Os métodos Campo de Força Virtual ou VFF[2] e Histograma de Campo Vetorial ou VFH[3] foram propostos por Borenstein & Koren [1989, 1991] e têm sido bastante utilizados

[2] *Virtual Force Field*
[3] *Vector Field Histogram*

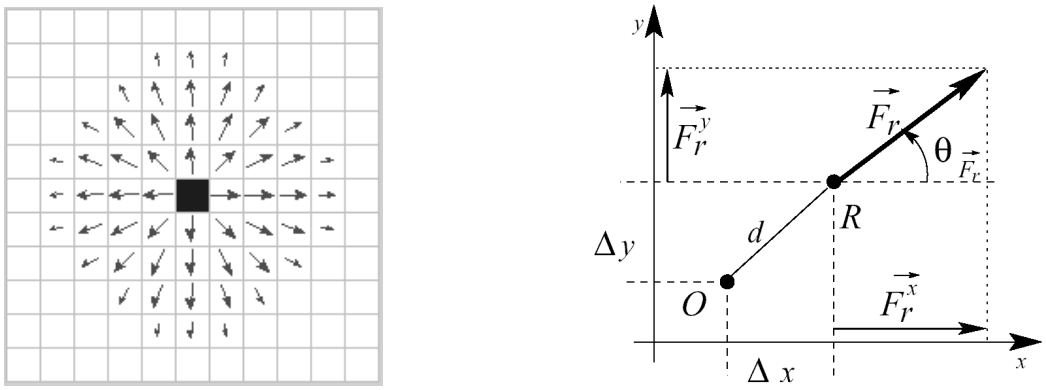
em pesquisas na área de navegação de robôs móveis. Estes métodos diferenciam-se dos demais métodos acima apresentados pela utilização de um mapa do ambiente inspirado na técnica de Grade de Ocupação[4] [Elfes, 1989] para representar o ambiente com seus obstáculos.

Algumas aplicações da utilização da técnica de campos potenciais aplicada ao planejamento de trajetórias para robôs móveis foram sugeridas por Faria & Romero [2000], Pacheco & Costa [2002] e Gomes & Campos [2001].

No método de campos potenciais considera-se que a meta é uma força de atração, cada obstáculo gera uma força de repulsão e a direção a ser seguida é dada por um vetor resultante da soma de todos os vetores força envolvidos. A seguir, será mostrado como realizar os cálculos das forças de repulsão, atração à meta e da força resultante, embora outros modelos de forças vetoriais possam ser criados para representar outros tipos de comportamentos.

3.2.1 Força de Repulsão

Na Figura 3.4(a) mostra-se que a força de repulsão decai ao se afastar de um obstáculo, pois a força de repulsão ($\overrightarrow{F_r}$) é um vetor cujo módulo é inversamente proporcional ao quadrado da distância (d) entre o robô (R) e objeto observado (O) (Figura 3.4(b)). O vetor



(a) (b)

Figura 3.4: (a) Força de repulsão na vizinhança de um obstáculo; (b) Força de repulsão do objeto O sobre um robô localizado em R.

$\overrightarrow{F_r}$ também pode ser representado por suas componentes: módulo e direção, apresentados na equação (3.5).

$$|\overrightarrow{F_r}| = \frac{Q}{d^2} \quad e \quad \theta_{\overrightarrow{F_r}} = \arctan(-\Delta y, -\Delta x) \tag{3.5}$$

[4] *Occupancy Grid*

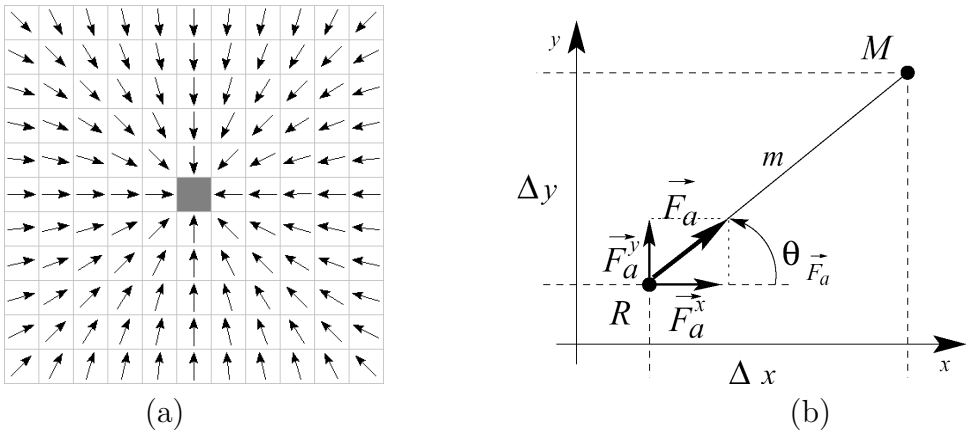
sendo que: Q representa um escalar constante de repulsão; arctan é uma função na qual se considera os sinais de Δx e Δy para fornecer o ângulo correto cuja tangente seja $(-\Delta y/-\Delta x)$; e os sinais negativos de Δx e Δy fornecem uma direção oposta ao objeto detectado (O).

Para calcular a força de repulsão para vários obstáculos, deve-se fazer o somatório dos vetores das forças de repulsão geradas, como mostrado na equação (3.6).

$$\overrightarrow{F_R} = \sum \overrightarrow{F_r} \tag{3.6}$$

3.2.2 Força de Atração

Na Figura 3.5(a) e Figura 3.5(b) mostra-se o campo potencial de atração, no qual deve-se considerar $|\overrightarrow{F_a}|$ constante para que o agente seja atraído pela meta mesmo estando distante da mesma (equação (3.7)).



(a) (b)

Figura 3.5: (a) Campo de atração constante; (b) Atração entre o robô R e a meta M.

$$|\overrightarrow{F_a}| = C \quad e \quad \theta_{\overrightarrow{F_a}} = \arctan(\Delta y, \Delta x) \tag{3.7}$$

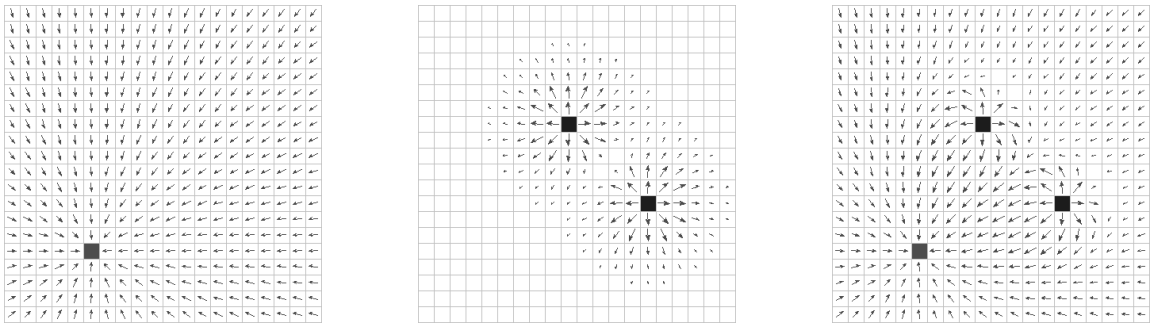
no qual C representa um escalar constante de atração. Note que Δx e Δy diferem da equação (3.5) por não estarem acompanhados do sinal negativo. Como resultado, a direção do vetor força de atração será na direção da meta.

3.2.3 Força Resultante

Ao realizar a soma dos vetores força de atração e força de repulsão obtém-se a força resultante dada pela equação (3.8).

$$\overrightarrow{F} = \overrightarrow{F_R} + \overrightarrow{F_a} \tag{3.8}$$

Para ilustrar a superposição dos campos de atração e repulsão, mostra-se na Figura 3.6(a) o campo de atração gerado pela meta, na Figura 3.6(b) o campo de repulsão gerado por dois obstáculos e na Figura 3.6(c) a soma dos campos de atração e de repulsão.



(a) (b) (c)

Figura 3.6: (a) potencial atrativo da meta, (b) potencial repulsivo de dois obstáculos, (c) soma destes dois campos.

3.2.4 Considerações

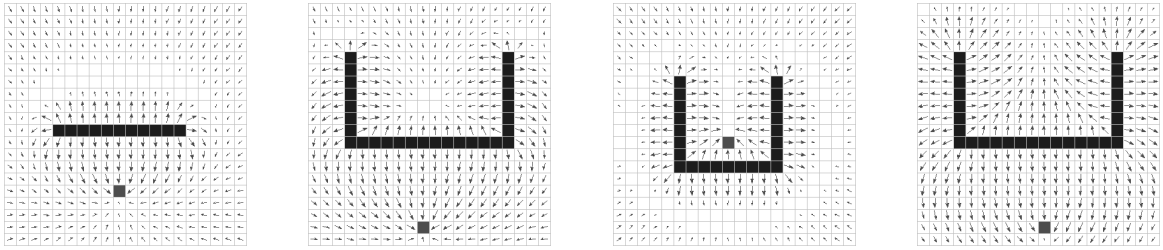
A aplicação da técnica de Campos Potenciais aplicada à navegação autônoma de robôs móveis pode apresentar algumas vantagens e desvantagens. Entre as vantagens destacam-se:

1. o custo computacional é bastante baixo se comparado a métodos no qual se utilizam mapas;

2. o sistema é reativo, ou seja, nenhuma modificação precisa ser feita para tratar os obstáculos dinâmicos pois o cálculo é refeito para cada nova posição do robô;

3. é um sistema modular no qual as forças podem ser calculadas em paralelo, o que facilita implementações em hardware.

Como desvantagem, o método não realiza exploração de ambientes, pois não guarda informações ou modelos do ambiente. Uma outra desvantagem se refere as forças de atração e de repulsão, pois estas podem se anular em determinados pontos gerando *mínimos locais*. Na Figura 3.7(a),(b) e (c) pode-se observar a presença de mínimos locais representados por células em branco que não contém um vetor indicando a direção a ser seguida.



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

Figura 3.7: Mínimos locais gerados pela anulação das forças de atração e repulsão.

O problema de mínimos locais também pode gerar armadilhas no ambiente. Na Figura 3.7(b) mostra-se uma destas armadilhas no qual um robô que se aproxime pela parte superior do campo pode ser direcionado para um local do qual não há saída. Já na Figura 3.7(c) o método não consegue gerar um caminho até a meta pois esta tem sua força de atração anulada pelas paredes que estão próximas.

Um outro problema do método é encontrar valores para as constantes de atração (K) e de repulsão (Q) de forma a maximizar a eficiência do robô para um determinado ambiente. Isto pode ser observado ao aumentar em 4 vezes o valor de (Q) para o ambiente apresentado na Figura 3.7(b) obtendo-se o campo mostrado na Figura 3.7(d). Pode ser notado que o problema da armadilha foi solucionado no entanto a meta teve seu campo anulado pelo campo gerado pelos obstáculos próximos a ela, e portanto, não existe um caminho que leve à meta.

3.3 Campos Potenciais Harmônicos

Os sistemas de planejamento que utilizam campos potenciais têm sido bastante utilizados, embora este método possua limitações, como visto na seção anterior, que possam diminuir a eficiência do sistema.

Estes problemas foram resolvidos, no trabalho desenvolvido por Connolly & Grupen [1993], com a utilização de funções harmônicas para o cálculo do campo potencial de ambientes nos quais as posições de paredes, objetos e metas sejam conhecidas. As funções harmônicas são soluções para a equação de Laplace, sendo que estas possuem propriedades muito úteis ao desenvolvimento de sistemas de controle robótico, tais como:

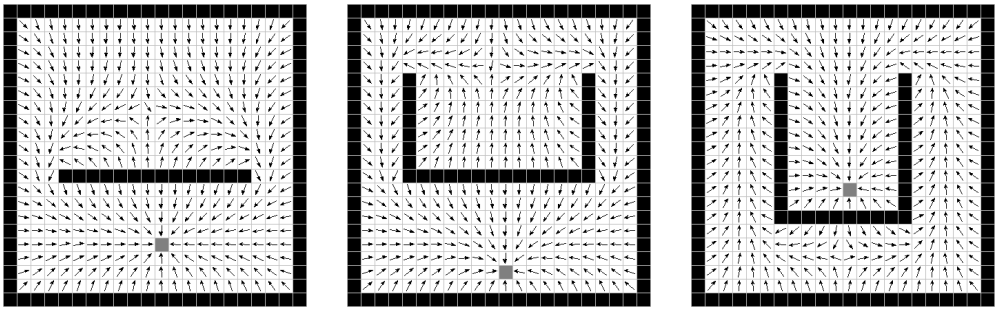
Integridade - o método fornece um sistema de planejamento *completo*, ou seja, é garantido encontrar um caminho livre entre dois pontos, caso ele exista.

Robustez - é capaz de lidar com a presença de obstáculos não conhecidos a priori.

Para o problema de planejamento de trajetória, o método de Campo Potenciais Harmônicos (CPH) pode ser aplicado sobre uma representação discreta do ambiente em forma de grade. As células desta grade que representam obstáculos são marcadas com potencial 1 e as metas com potencial zero, ou seja, utiliza-se as condições de contorno de Dirichlet. O potencial das células livres deve ser calculado continuamente (relaxados) através de uma discretização da equação de Laplace (3.9), ou seja, esta equação será utilizada para interpolar os valores dos potenciais para todas as células livres entre os obstáculos e a meta.

$$p_{i,j} = \frac{1}{4}(p_{i,j+1} + p_{i,j-1} + p_{i+1,j} + p_{i-1,j}) \tag{3.9}$$

As soluções da equação de Laplace não possuem mínimos/máximos locais [Courant & Hilbert, 1962], e assim, caso exista um caminho, o objetivo é sempre alcançado seguindo-se as linhas de força. Esta propriedade pode ser observada na Figura 3.8, na qual é mostrado CPH para três ambientes distintos idênticos aos ambientes da Figura 3.8 que utilizam os Campos Potenciais tradicionais.



(a) (b) (c)

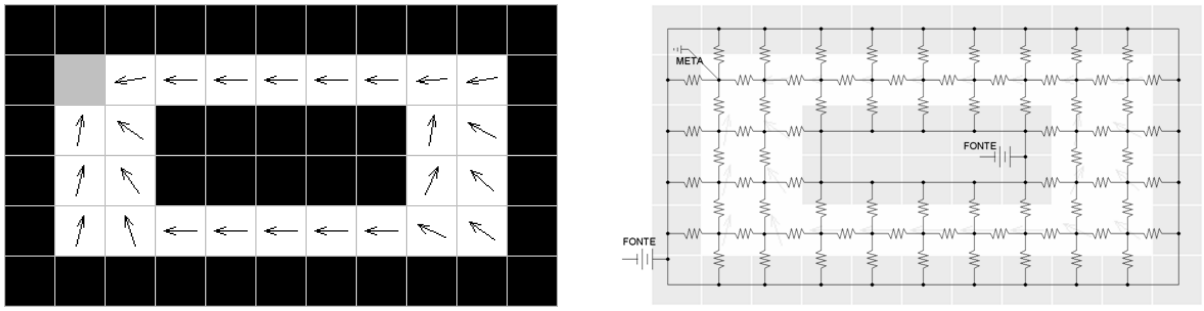
Figura 3.8: Exemplo de Campos Potenciais Harmônicos (CPH), o qual não apresenta mínimos locais.

As *linhas de força* são extraídas a partir do *gradiente descendente* do potencial que direciona o robô para sua meta desviando de obstáculos. Sendo assim, dada a posição de um robô em uma determinada célula (i, j), a direção a ser seguida Φ pode ser calculada como:

$$\Phi = \arctan(p_{i-1,j} - p_{i+1,j}, p_{i,j-1} - p_{i,j+1}) \tag{3.10}$$

no qual, Φ pertence ao intervalo $[-\pi, \pi]$.

Na Figura 3.9(a), pode-se observar um ambiente discretizado em (6x11) células, com células escuras marcando os obstáculos e a cinza representando a meta. Sabe-se também que o caminho gerado pelo gradiente descendente em um sistema a partir de um CPH é análogo ao caminho marcado pelo movimento da corrente elétrica em uma rede de resistores (Figura 3.9(b)) no qual os pontos com obstáculos recebem energia de uma fonte e as metas são drenos de energia. Os demais pontos, ou pontos livres, representam um exemplo discreto de uma função harmônica, ou seja, a voltagem de um nó livre é a média da voltagem dentre seus nós vizinhos.



(a) (b)

Figura 3.9: Representação do campo potencial harmônico. (a) em uma malha de potenciais; (b) rede de resistores

Esta implementação através de rede de resistores foi proposta por Tarassenko & Blake [1991] como sendo um método rápido para o planejamento de trajetórias de robôs móveis.

3.3.1 Funções Harmônicas

Uma função $p : \Omega \rightarrow R$, com $\Omega \subset R^n$, é definida como função harmônica caso satisfaça a equação de Laplace, ou seja:

$$\nabla^2 p = \sum_{i=1}^{n} \frac{\partial p^2}{\partial x_i^2} = 0 \tag{3.11}$$

Em um ambiente robótico, o robô move-se sobre um plano, ou seja, $\Omega \subset R^2$. Para este caso, a equação de Laplace seria representada pela equação 3.12, no qua x e y representam coordenadas espaciais no sistema de referência global.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0 \tag{3.12}$$

Para resolver esta equação, existem dois os métodos comumente utilizados na solução numérica de equações diferenciais parciais: o método de Gauss-Seidel (GS) e o método de Sucessiva Sobre-Relaxações (SOR). Estes métodos surgem a partir da discretização da

equação 3.12 por meio de diferenças centrais de segunda ordem usando séries de Taylor, como pode ser observado logo abaixo:

$$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\Delta x^2)} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\Delta y^2)} = 0 \tag{3.13}$$

no qual Δx e Δy representam o espaçamento utilizado para discretizar o eixo x e o eixo y, respectivamente.

A equação acima pode ser reescrita como:

$$p_{i+1,j} - 2p_{i,j} + p_{i-1,j} + \left(\frac{\Delta x}{\Delta y}\right)^2 (p_{i,j+1} - 2p_{i,j} + p_{i,j-1}) = 0 \tag{3.14}$$

Considerando-se um espaçamento regular na discretização de ambas as coordenadas, tem-se então $\frac{\Delta x}{\Delta y} = 1$. Sendo assim, a equação acima pode ser reescrita e obtém-se a forma discreta da equação de Laplace:

$$\begin{aligned} p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} = 0 \quad \text{ou} \\ p_{i,j} = \frac{1}{4}(p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1}) \end{aligned} \tag{3.15}$$

A equação 3.15 quando aplicada sobre uma grade consiste em resolver um sistema de equações lineares no qual as variáveis p são potenciais das células livres da grade. Métodos de relaxação como Jacobi, Gauss-Seidel e SOR podem ser utilizados para resolver este sistema numericamente. Para a equação de Laplace estes métodos consistem essencialmente em atualizar repetidamente os valores de cada posição livre da grade pelo valor da média de seus $2n$-vizinhos (vizinhança de Manhattam) em uma grade n dimensional.

3.3.2 Métodos de Relaxação

O método de Jacobi aplicado as equações de Laplace (equação 3.15) consiste na substituição simultânea do valor de todas as posições livres da grade pela média dos valores de seus vizinhos. A função de atualização de cada célula pode ser escrita como:

$$p_{i,j}^{t+1} = \frac{1}{4}(p_{i+1,j}^{t} + p_{i-1,j}^{t} + p_{i,j+1}^{t} + p_{i,j-1}^{t}) \tag{3.16}$$

no qual t é o número da iteração. O método de Jacobi normalmente necessita de muito mais atualizações para obter a convergência, quando comparado ao método de Gauss-Seidel ou SOR. Contudo, possui a vantagem de ser um método naturalmente paralelo e

quando utilizado em máquinas massivamente paralelas cada ponto $p_{i,j}$ pode ser calculado por uma unidade de processamento.

O método de Gauss-Seidel é similar ao método de Jacobi, mas diferencia-se pelo fato de possuir valores de vizinhos já atualizados no cálculo de uma posição. Este método pode ser implementado utilizando um array e percorrendo cada uma de suas posições e atualizando-a de acordo com a equação (3.15). Na Figura 3.10 mostra-se um array sendo atualizado da esquerda para direita e de cima para baixo. Ao atualizar o valor da posição (i, j) do array, a equação de atualização contará com metade dos vizinhos com valores já atualizados e a outra metade com valores ainda não atualizados, como descrito na equação abaixo:

$$p_{i,j}^{t+1} = \frac{1}{4}(p_{i+1,j}^{t} + p_{i-1,j}^{t+1} + p_{i,j+1}^{t} + p_{i,j-1}^{t+1}) \tag{3.17}$$

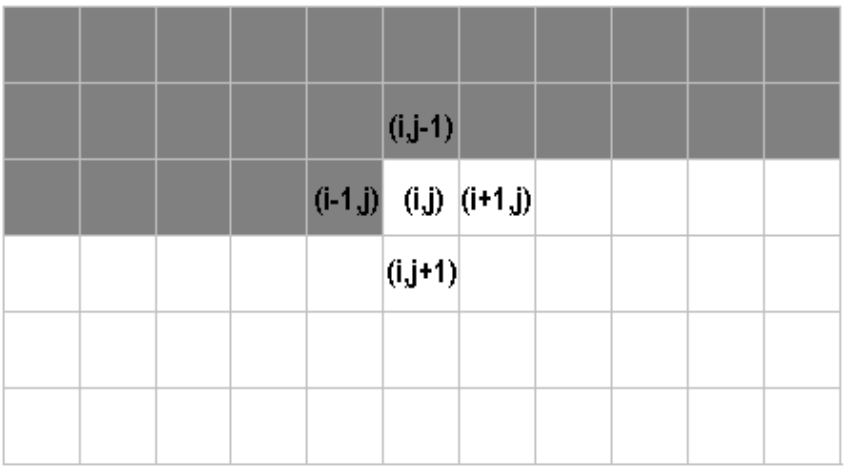


Figura 3.10: Atualização de um array utilizando o método de Gauss-Seidel. A área destacada representa os valores já atualizados $(t + 1)$.

Observe que ao atualizar uma posição por vez, a equação contará com metade dos vizinhos já atualizados e a outra metade ainda não. O cálculo seqüencial de cada valor torna este esquema o mais natural quando aplicado a um sistema monoprocessado.

A diferença entre p^t e p^{t+1} tende a diminuir entre as sucessivas aproximações. O método SOR consiste em aumentar esta pequena diferença de maneira que ele obtenha uma convergência mais rápida que os métodos Jacobi e Gauss-Seidel (GS). A base para o cálculo do SOR utiliza método GS, como pode ser visto abaixo:

$$\begin{aligned} p_{SOR}^{t+1} &= p_{GS}^{t+1} + \lambda(p_{GS}^{t+1} - p^t) \\ &= (1+\lambda)p_{GS}^{t+1} - \lambda p^t \quad \text{fazendo } w = (1+\lambda) \end{aligned}$$

$$
\begin{aligned}
&= w.p_{GS}^{t+1} - (w-1)p^t \\
&= w.p_{GS}^{t+1} - w.p^t + p^t \\
&= \frac{w}{4}(p_{i+1,j}^t + p_{i-1,j}^{t+1} + p_{i,j+1}^t + p_{i,j-1}^{t+1} - 4p_{i,j}^t) + p_{i,j}^t
\end{aligned} \tag{3.18}
$$

O valor $0 < w < 2$ é um fator de relaxação, sendo que Connolly & Grupen [1993] adotaram $w = 1.8$ para experimentos realizados e Press et al. [1992] propôs uma equação para calcular o valor ótimo para w, a partir do conhecimento a priori do número de linhas e de colunas da grade discreta do ambiente. Mais informações sobre constantes de aceleração podem ser encontrada no trabalho de Burden et al. [1978].

3.3.3 Além das Funções Harmônicas

A solução da equação de Laplace garante que a geração dos campos potenciais não irá apresentar mínimos locais, no entanto, este é apenas um caso particular de um conjunto de funções que também apresentam esta característica.

"Existe uma família de funções escalares geradas por um problema de valor de contorno que não apresenta mínimos locais"[Prestes, 2003], sendo esta família denotada por:

$$\nabla^2 p + F(\nabla p) = 0 \tag{3.19}$$

desde que satisfaça duas condições: $F(0) = 0$ e $F(\nabla p)$ deve ser uma função contínua.

Uma das sugestões feitas por Prestes [2003], e utilizada por Trevisan et al. [2006], considera a seguinte função linear para o novo termo $F(\nabla p)$:

$$F(\nabla p) = \epsilon \cdot \nabla p \cdot \mathbf{v} \tag{3.20}$$

no qual ϵ é um escalar e $\mathbf{v}$ é um vetor constante. Substituindo (3.20) em (3.19) obtém-se o problema de Sturm-Liouville [Kreider et al., 1966]:

$$\nabla^2 p + \epsilon \cdot \nabla p \cdot \mathbf{v} = 0 \tag{3.21}$$

A forma discreta da equação (3.20) é apresentada na equação (3.22 e utiliza-se o método de Gauss-Seidel para interpolar os valores dos potenciais de cada célula livre.

$$p_{i,j}^{t+1} = \frac{1}{4}(p_{i+1,j}^t + p_{i-1,j}^{t+1} + p_{i,j+1}^t + p_{i,j-1}^{t+1}) + \frac{\epsilon}{8}[(p_{i+1,j}^t - p_{i-1,j}^{t+1})\mathbf{v}x + (p_{i,j+1}^t - p_{i,j-1}^{t+1})\mathbf{v}y] \tag{3.22}$$

no qual, $\mathbf{v}x$ e $\mathbf{v}x$ são as componentes do vetor $\mathbf{v}$ nos eixos x e y, respectivamente.

Devido a influência do vetor $\mathbf{v}$ na orientação do campo potencial, esta abordagem utilizando a equação (3.22) será chamada por nós de *Campos Potenciais Orientados*(CPO). Na Figura 3.11 é mostrado o campo potencial gerado utilizando-se a equação (3.22) e a influência recebida pela direção do vetor $\mathbf{v}$. Deve ser observado que as coordenadas das imagens parte da esquerda para direita e de cima para baixo, por este motivo, um vetor $\mathbf{v}$ orientado a 45^o é descrito como $\mathbf{v} = (1, 1)$ e orientado a 90^o é descrito como $\mathbf{v} = (0, 1)$.

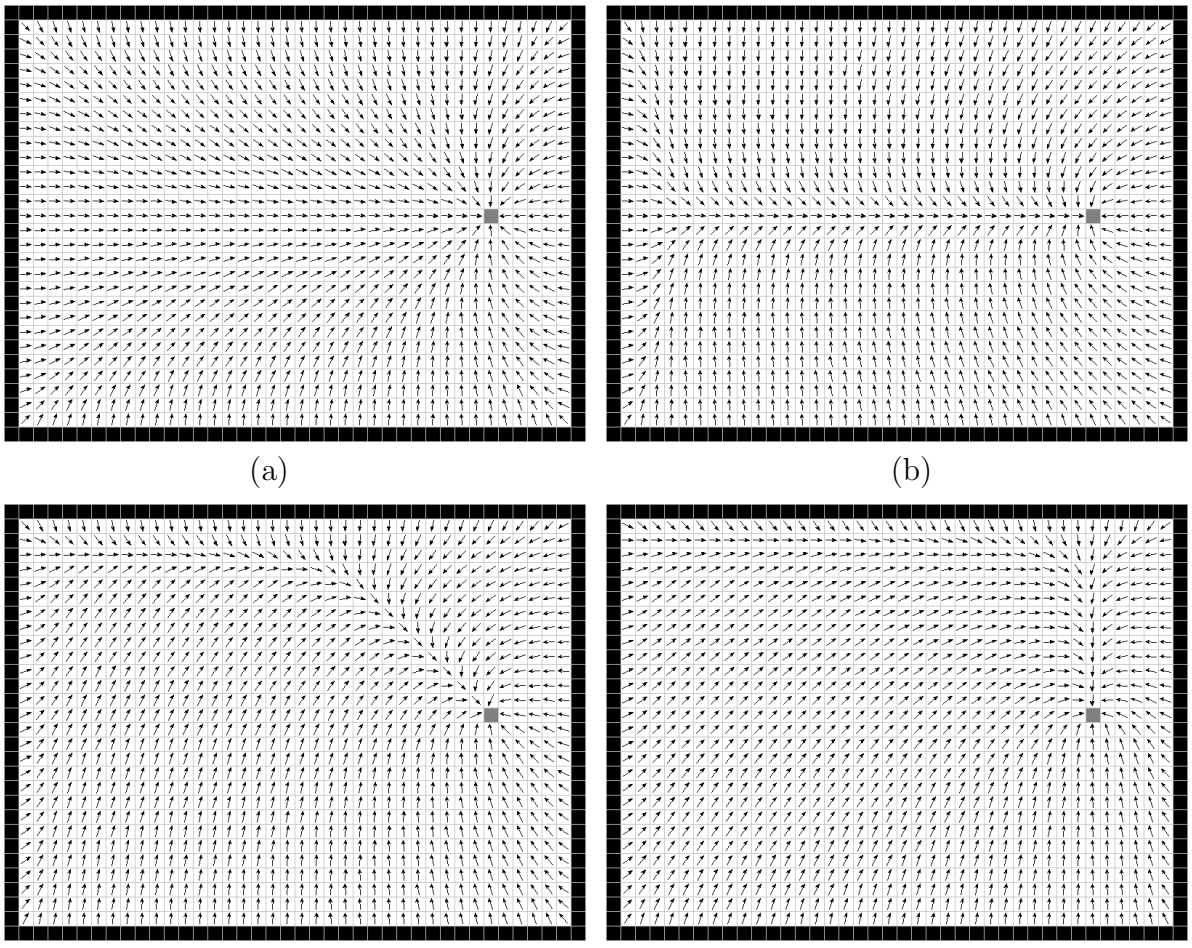


Figura 3.11: Influência do vetor $\mathbf{v}$ no campo potencial. (a) CPH sem influência de vetor; (b) CPO com $\mathbf{v} = (1, 0)$; (c) CPO com $\mathbf{v} = (1, 1)$; (d) CPO com $\mathbf{v} = (0, 1)$

Já nas Figuras 3.12(a) a (d), observa-se o comportamento que valores diferentes de ϵ podem provocar no campo potencial. A variável ϵ pode ser vista como a taxa de *influência* do vetor $\mathbf{v}$ sobre o campo potencial. Tal influência, distorce o tendência dos CPH em criar trajetórias distantes das paredes. Como pode ser observado na Figura 3.12, a taxa ϵ é capaz de modificar o campo de forma que as trajetórias fiquem mais próximas das paredes.

Na Figura 3.12(e) compara-se as trajetórias seguidas com $0 \leq \epsilon \leq 1.5$, e é possível verificar que, quanto maior o valor de ϵ, mais a trajetória se aproxima da direção do

vetor $\mathbf{v}$. Contudo, o valor de ϵ não pode crescer indefinidamente. Como mostrado na Figura3.12(e), o uso de $\epsilon = 4$ mostra um campo instável que não garante uma trajetória até a meta. Isto acontece pois a condições iniciais de contorno, na qual meta atrai e obstáculo repele, foram violadas.

Abaixo encontram-se os passos seguidos para discretizar a equação (3.21) obtendo-se a equação (3.22).

Sabe-se que as soluções numéricas de $\nabla^2 p$ e ∇p, realizada por meio de diferenças centrais de segunda ordem e de primeira ordem usando séries de Taylor são dadas por (3.24) e (3.23), respectivamente.

$$\nabla^2 p = p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} \quad \text{e} \tag{3.23}$$

$$\nabla p = \frac{p_{i+1,j} - p_{i-1,j}}{2} + \frac{p_{i,j+1} - p_{i,j-1}}{2} \tag{3.24}$$

Substituindo-se (3.23) e (3.24) na equação (3.21), obtém-se a forma discreta do problema de Sturm-Liouville (3.25), como pode ser visto abaixo:

$$\overbrace{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j}}^{\nabla^2 p} + \epsilon \overbrace{\left[\frac{(p_{i+1,j} - p_{i-1,j})\mathbf{v}x}{2} + \frac{(p_{i,j+1} - p_{i,j-1})\mathbf{v}y}{2} \right]}^{\epsilon \cdot \nabla p \cdot \mathbf{v}} = 0$$

$$4p_{i,j} = p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} + \frac{\epsilon}{2}[(p_{i+1,j} - p_{i-1,j})\mathbf{v}x + (p_{i,j+1} - p_{i,j-1})\mathbf{v}y]$$

$$p_{i,j} = \frac{1}{4}[p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1}] + \frac{\epsilon}{8}[(p_{i+1,j} - p_{i-1,j})\mathbf{v}x + (p_{i,j+1} - p_{i,j-1})\mathbf{v}y] \tag{3.25}$$

no qual, $\mathbf{v}x$ e $\mathbf{v}x$ são as componentes do vetor $\mathbf{v}$ nos eixos x e y, respectivamente.

3.3.4 Considerações

O método de cálculo de Campos Potenciais utilizando o Problema de Valor de Contorno, como CPH e CPO, têm demonstrado sua eficiência, tanto para o problema de planejamento de trajetórias como para exploração de ambientes. Assim como todos os métodos que utilizam modelos do mundo para sua implementação, o custo computacional deste método cresce proporcionalmente com o tamanho do ambiente mapeado. Contudo, vale salientar que este método se diferencia dos demais métodos de geração de campos potenciais por ser um método completo, ou seja, uma trajetória livre até a meta sempre será gerada caso ela exista.

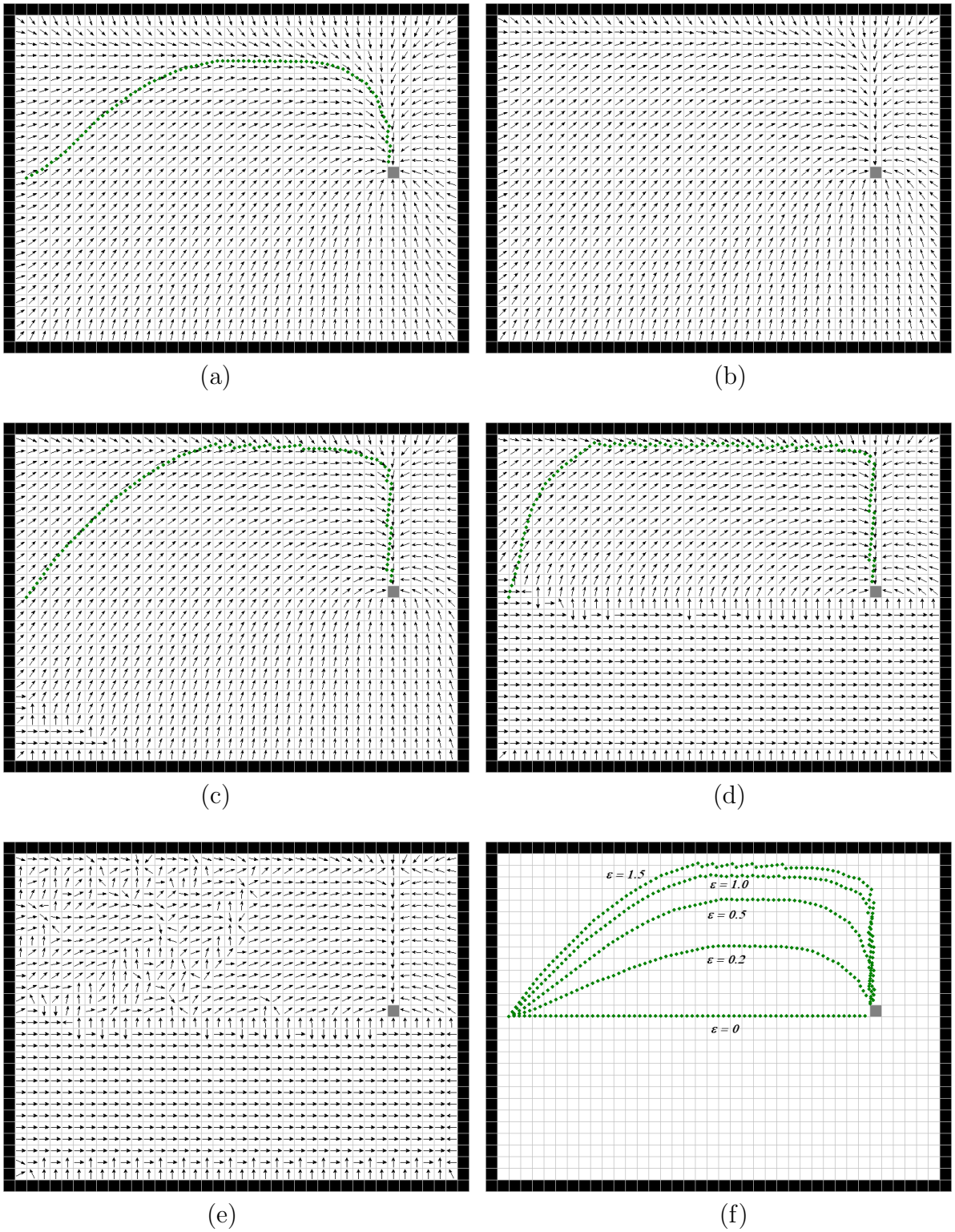


Figura 3.12: Várias taxas de influência ϵ em um campo potencial com $\mathbf{v} = (0, 1)$. (a) $\epsilon = 0.5$; (b) $\epsilon = 1.0$; (c) $\epsilon = 1.5$; (d) $\epsilon = 2.0$; (e) $\epsilon = 4.0$; (f) comparação entre as trajetórias seguidas de (a) a (d).

4

Arquitetura de Controle Inteligente – ACIn

Neste capítulo será proposta uma arquitetura de controle inteligente para múltiplos robôs juntamente com as diferentes técnicas propostas que a constituem.

A arquitetura proposta neste capítulo possui uma organização hierárquica, com componentes reativos e componentes deliberativos, sendo classificada como uma arquitetura de controle híbrida. Neste ponto, assemelha-se com as arquiteturas AuRA e Atlantis. Contudo, alguns métodos mais modernos foram estudados para compor cada nível desta hierarquia.

O ambiente no qual esta arquitetura será aplicada é o ambiente de futebol de robôs e, portanto, os exemplos fornecidos levam em consideração a função de cada integrante de um time de futebol de robôs como atacante, defesa e goleiro. No entanto, nada impede desta arquitetura ser utilizada para um conjunto de robôs com outras finalidades.

Como mostrado na Figura 4.1, a arquitetura ACIn possui quatro níveis bem diferenciados: Missão, Estratégia, Planejamento e Controle de Velocidade.

O nível mais alto, ou nível de Missão, estabelece a funcionalidade para cada um dos robôs. Neste nível, define-se quantos robôs do time ficarão na defesa e quantos no ataque. Nesta tese, considera-se uma missão fixa para cada robô, sendo que estes não mudam de missão durante o processo. Contudo, nada impede que seja utilizado um sistema deliberativo para replanejar a missão de cada robô.

O nível de Estratégia tem por finalidade estabelecer as metas e restrições para cada tipo de missão selecionada. Um exemplo de meta é estabelecer que o atacante deve chutar a bola ao gol, contudo possui a restrição de não chutar contra o próprio gol. Para implementação deste nível é proposto um Sistema Baseado em Regras (SBR) [Russell &

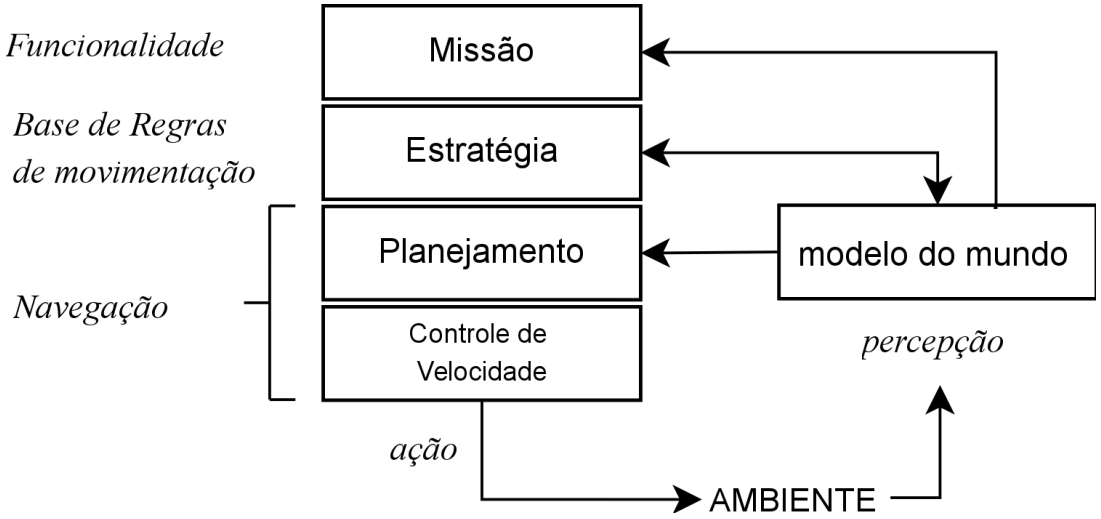


Figura 4.1: Estrutura da arquitetura ACIn

Norvig, 1995], composto por um conjunto de regras do tipo: SE-ENTÃO.

O nível de Planejamento possui métodos capazes de gerar a trajetória a ser seguida pelo robô de acordo com o nível estratégico. Este nível deve ser sensível às mudanças do ambiente e ter a capacidade de replanejar as trajetórias, caso necessário.

As trajetórias do nível de Planejamento são dadas por direções a cada instante de tempo. Entretanto, os robôs utilizados são controlados/direcionados alterando-se a velocidade de cada roda. Por este motivo, abaixo do nível Planejamento está o *Controle de Velocidade*, que transformará o ângulo da direção desejada em velocidade para cada roda.

O nível Missão distingue a funcionalidade de cada robô, como goleiro, ataque ou defesa. Contudo, nada impede que algum subsistema possa ser utilizado para mudar a funcionalidade de um integrante durante o jogo.

O foco deste trabalho foi dirigido aos níveis de Estratégia e Planejamento. Para esses níveis, foram investigadas a utilização das técnicas de Redes Neurais, Campos Potenciais e Campos Potenciais Harmônicos. Um aperfeiçoamento da técnica de Campos Potenciais Orientados, denominado por nós, de Campos Potenciais Localmente Orientados, é também proposto. A seguir serão detalhadas as várias propostas investigadas neste trabalho buscando definir uma Estratégia e Planejamento adequados à arquitetura ACIn. Destacam-se nas subseções seguintes duas contribuições importantes desta tese: a primeira se refere aos níveis constituintes da arquitetura ACIn, Planejamento e Estratégia, e a segunda se refere ao nível Controle de Velocidade.

4.1 Planejamento e Estratégia

Controlar um time de futebol de robôs é uma tarefa complexa, pois vários comportamentos podem ser exigidos para executar esta tarefa. Tais comportamentos estão relacionados ou ao time, ou a um robô específico. Ao considerar o comportamento do time, espera-se que o mesmo seja capaz de alternar entre um jogo mais ofensivo e um jogo mais na defensiva, e que seus componentes se ajudem ao invés de competirem entre si pela posse da bola. Um comportamento específico de um robô artilheiro é fazer gols, de um robô zagueiro é defender seu campo e de um goleiro é impedir que a bola entre em seu gol. O aspecto que deve ser observado é que existem comportamentos concorrentes que podem acionar ações divergentes, i.g., buscar pela bola mas desviar dos demais robôs; chutar a bola mas não fazer gol contra. Contudo, existem comportamentos concorrentes, i.g, direcionar o chute ao gol ou a um elemento de sua equipe (passe de bola).

Nesta seção são apresentados os métodos utilizados para implementar os níveis de Planejamento e Estratégia, responsáveis pelo comportamento de cada integrante do time de robôs.

4.1.1 Aprendizado via Rede Neural MLP

A primeira proposta para modelar os níveis de Planejamento e Estratégia foi considerar a utilização de uma única rede neural MLP como planejador de trajetória e base de regras de estratégia. Esperava-se que a rede distribuísse entre seus vários neurônios as tarefas de cada nível, pois uma das principais características das RNAs é o reconhecimento de padrões em um conjunto de dados.

O sucesso no aprendizado de uma MLP depende da estrutura (nº. de camadas e nº. nós) adotada e também da consistência dos dados coletados. Os dados foram coletados a partir dos movimentos desejados para um robô artilheiro, i.e, buscar pela bola, desviar dos demais robôs, chutar a bola ao gol e não fazer gol contra. Estes comportamentos foram mapeados na forma de uma base de dados e então uma MLP foi treinada para reconhecer o padrão dos movimentos, ou seja, a estratégia de jogo para o artilheiro.

4.1.2 Planejamento via Campos Potenciais

Em uma segunda abordagem, foi utilizado o método reativo de planejamento de trajetórias conhecido como campos potenciais para implementar o nível de Planejamento. No nível estratégico, regras foram definidas para cada funcionalidade de jogador. Estas regras determinam a posição do pontos atrativos específicos para cada robô de acordo com

a função estabelecida no nível Missão. Isto permite que um atacante possa ser atraído pela bola, enquanto o goleiro é atraído por algum ponto próximo ao gol que bloqueie a trajetória da bola.

Duas missões distintas foram definidas para serem utilizadas com Campos Potenciais: ataque e defesa.

Defesa

Um robô k pertencente à defesa deve ser alocado a uma linha de defesa l_k, como mostrado na Figura 4.2. Para definir as regras do nível Estratégico cria-se primeiramente uma linha imaginária r que passa pela posição da bola Pb e pelo centro do gol do próprio time Pg. Sabe-se que a função que define uma reta é dada por $f(x) = ax + b$, sendo a o coeficiente angular e b o coeficiente linear. Desta forma, para a reta r, de coeficientes a_r e b_r tem-se:

$$a\_r = \frac{Pb_y - Pg_y}{Pb_x - Pg_x} \quad \text{e} \tag{4.1}$$

$$b\_r = b_y - a\_r.Pb_x \tag{4.2}$$

sendo $Pb = (Pb_x, Pb_y)$ e $Pg = (Pg_x, Pg_y)$.

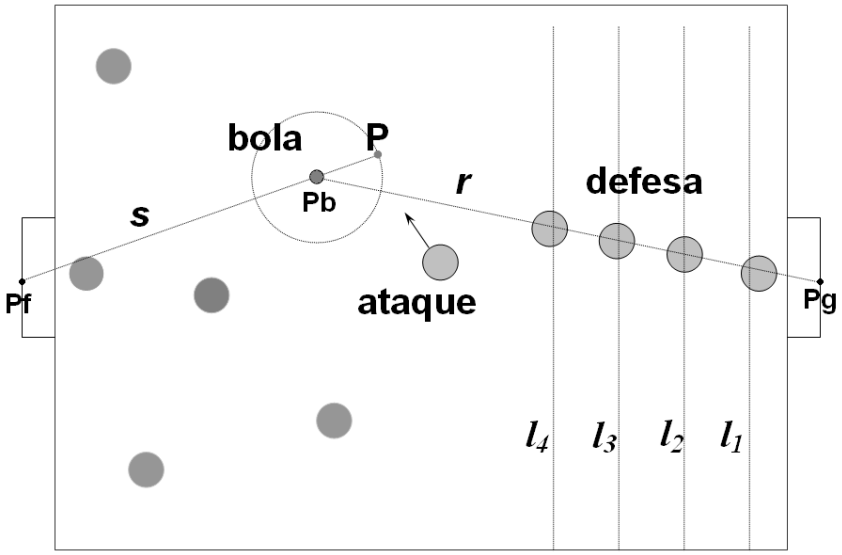


Figura 4.2: Comportamentos de ataque e defesa para Campos Potenciais

A regra para defesa define que o ponto de atração de um robô k é o ponto de intersecção entre as retas r e l_k. Considerando-se a reta l_k como sendo $x = C$, no qual C é uma constante, as coordenadas do ponto de intersecção entre r e l_k será dado por $(C, a\_r.C + b\_r)$.

Ataque

As regras de ataque foram definidas para obter-se o chute direcionado, pois isto favorece as chances de gol e evita os gols contra. A principal regra de ataque diz que o robô atacante é repelido pela bola e atraído para um ponto P próximo a bola, como pode ser visto na Figura 4.2. Para definir o ponto P, considere uma reta s que passa pelo centro do gol adversário e pela posição da bola e uma circunferência de raio q com o centro na posição da bola. O ponto P será o ponto de intersecção entre s e esta circunferência.

Os coeficientes a_s e b_s da reta s podem ser calculados de forma semelhante à reta r e são dados por:

$$a\_s = \frac{Pb_y - Pf_y}{Pb_x - Pf_x} \quad \text{e} \tag{4.3}$$

$$b\_s = Pb_y - a\_s.Pb_x \tag{4.4}$$

sendo $Pf = (Pf_x, Pf_y)$.

Assim, $P = (Pb_x + q/\sqrt{a\_s^2 + 1}, Pb_y + a\_s.q/\sqrt{a\_s^2 + 1})$.

Quando o robô estiver sobre o ponto P, ou pelo menos próximo a ele, o ponto P deixa de existir e a bola que antes repelia, passa a atrair. Esta mudança redireciona o robô e faz com que ele "chute" a bola ao gol.

No nível de Planejamento, as constantes de atração C e repulsão Q referentes ao método de Campos Potenciais devem ser ajustadas. Este ajuste é necessário para definir quão forte será a atração pela bola e quão forte será a força de repulsão gerada pelos outros robôs. Tome como exemplo um robô que está próximo a bola, e então uma força de repulsão é gerada. A intensidade desta força de repulsão pode aumentar quanto mais robôs forem se aproximando da mesma, e como visto na seção 3.2, a força de repulsão pode impedir que seja gerada uma trajetória até a meta. Para este exemplo, um robô atacante poderia ficar impedido de realizar seu objetivo de chutar a bola ao gol.

Para minimizar este efeito, as constantes de atração C e repulsão Q foram ajustadas empiricamente. Após vários testes exaustivos, definiu-se que os melhores valores seriam $C = 1$ e $Q = 20$.

Entretanto, para a regra do atacante, se a força de repulsão da bola for maior do que a atração do ponto P então o robô nunca conseguirá atingir P. Por este motivo, considerou-se que a bola repele menos que os outros objetos, ou seja, a bola possui uma constante de repulsão Qb menor do que Q. O valor de $Qb = 5$ também foi definido empiricamente,

ou seja, o valor da variável foi sendo ajustado a cada teste.

Bons resultados foram obtidos desta combinação entre Planejamento por Campos Potenciais e Estratégia com regras de deslocamento proposta por nós e podem ser encontradas no Capítulo 5 bem como em [Faria et al., 2004b].

4.1.3 *Planejamento via Campos Potenciais Harmônicos*

Como terceira proposta, o nível de Planejamento foi implementado utilizando-se Campos Potenciais Harmônicos. A implementação desta abordagem foi motivada pelas características dos CPH, pois, como visto na seção 3.3, CPH fornecem um controle de trajetórias mais suave e sem mínimos locais. De acordo com o que foi descrito no capítulo anterior, o método CPH utiliza um mapa discreto do ambiente sobre o qual calcula o potencial de cada célula. Um único mapa de trajetórias é gerado e todos os robôs consultam o mesmo mapa. Portanto, todos robôs irão seguir para a mesma meta. Para evitar que todos os robôs tenham o mesmo comportamento seguindo para o mesmo ponto, foi proposta e implementada por nós, no nível Estratégico, uma forma de delimitar as áreas de atuação de um robô de acordo com a missão estabelecida.

Para utilizar o método CPH, uma área retangular do ambiente foi representada por uma grade de 28 linhas por 34 colunas, como pode ser visto na Figura 4.3.

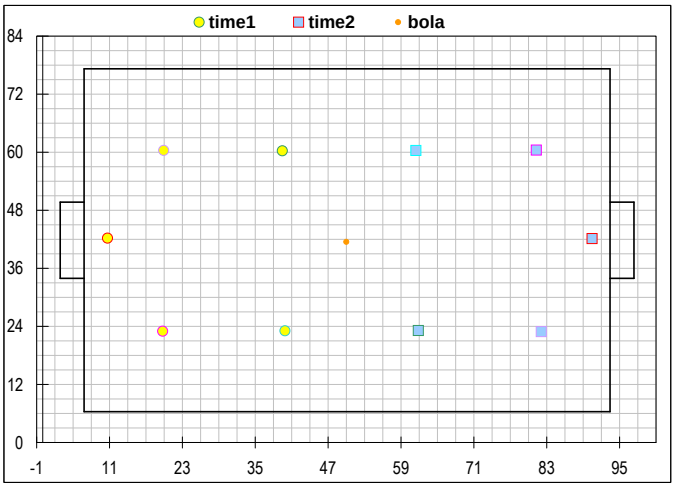


Figura 4.3: Ambiente do Futebol de Robôs discretizado uma grade de 28x34.

Os limites desta área foram definidos de tal forma que, após a discretização, as bordas que representam os limites dos campo ficassem com o mesmo número de células. Como o campo foi dividido em células de $3 \times 3pol$, os limites (x, y) do campo foram considerados

como: $-1 \leq x \leq 101$ e $0 \leq y \leq 84$.

Estabelecida as condições de contorno, no qual as bordas e os robôs são obstáculos e a bola é a meta, o método CPH é então aplicado obtendo-se o campo potencial mostrado na Figura 4.4.

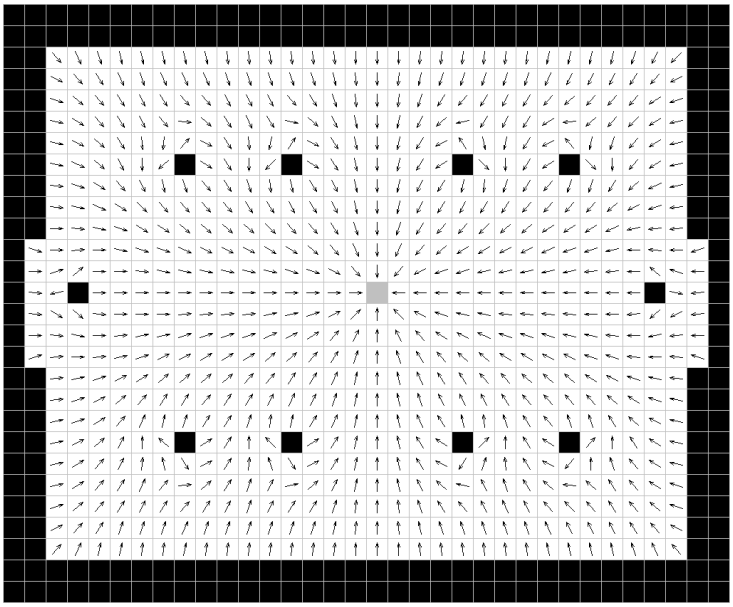


Figura 4.4: Fluxo utilizando-se Campo Potencial Harmônico.

Como pode ser visto na Figura 4.4, todas as trajetórias levam para a bola. Se for considerado mais de um robô sobre este campo, todos eles seguirão para o mesmo ponto, a bola. No entanto isto nem sempre é desejado, pois todos irão disputar pela posse da bola, além de favorecer chutes contra o próprio gol em determinadas situações.

Como visto nas seções anteriores, são desejados no mínimo dois comportamentos distintos: ataque e defesa. Sendo que ambos devem evitar fazer gols contra o próprio time. Assim, o nível de Estratégia foi proposto por nós utilizando o que chamamos de *Paredes Virtuais*. Estas paredes tem este nome pois existem somente na grade que representa o ambiente e são usadas para repelir certos movimentos ou guiar um robô por um caminho determinado.

Utilizando a abordagem de paredes virtuais para implementar o nível Estratégico, uma parede móvel foi colocada ao redor da bola com a finalidade de evitar gols contra. Além disso, para evitar que todos os robôs seguissem a bola ao mesmo tempo, fazendo um aglomerado ao redor da mesma, o campo foi delimitado em áreas de atuação para cada robô. Na Figura 4.5, são mostradas as paredes virtuais implementadas para restringir os movimentos indesejados.

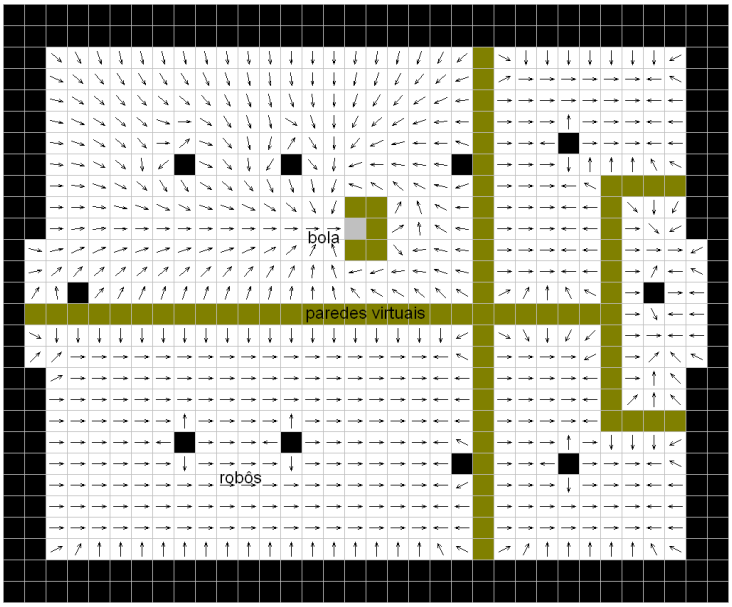


Figura 4.5: Estratégia de controle utilizando a técnica de paredes virtuais.

A abordagem de implementação do nível Estratégico por *paredes virtuais* apresentou algumas limitações, como:

1. a parede móvel ao redor da bola não é capaz de direcionar o chute ao gol do adversário. Como o campo foi discretizado, a tendência são chutes em ângulos retos;

2. os robôs acabam atravessando as paredes virtuais várias vezes, seja por causa da alta velocidade com que atuam ou por serem empurrados por outros robôs;

3. As áreas que não possuem meta tendem a marcar direções errôneas, isso acontece devido ao próprio método CPH, pois ele garante traçar uma trajetória livre até a meta caso ela exista.

Como solução para estas limitações, uma segunda estratégia para controle do time de robôs foi proposta e implementada utilizando Campos Potenciais Harmônicos. Esta estratégia continua com a utilização da parede móvel ao redor da bola, mas desta vez, sem delimitar o campo com paredes virtuais, pois como vimos estas impedem a criação correta do campo nas regiões isoladas. Desta vez, o nível Estratégia foi definido por regras que definem as regiões de atuação de cada robô. Tais regiões foram criadas para evitar que um robô ultrapasse os limites de sua área. Como pode ser visto na Figura 4.6, existem áreas nas quais mais de um robô pode mover-se criando possíveis áreas para a troca de bola entre os jogadores.

Esta segunda abordagem obteve sucesso na implementação, contudo o CPH direciona

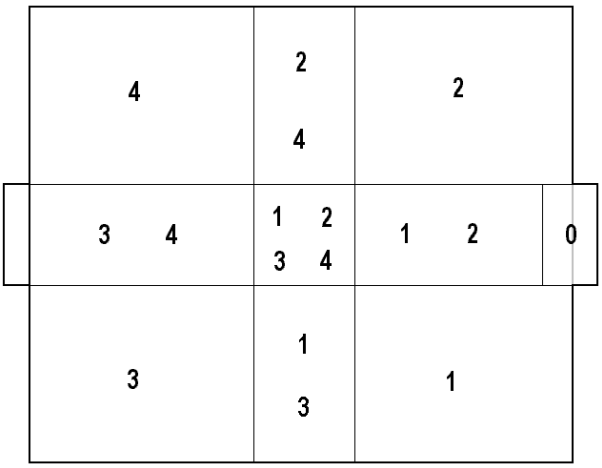


Figura 4.6: Estratégia com CPH utilizando limites de atuação. Cada robô pode atuar nas regiões nas quais aparecem seu número.

o movimento para a mesma meta e, portanto, os comportamentos de ataque e defesa não ficaram bem definidos no nível Estratégico [Faria et al., 2004a]. Um aperfeiçoamento do método de PVC foi proposto por nós como alternativa para as limitações, como é apresentado na próxima seção.

4.1.4 Planejamento via Campos Potenciais Localmente Orientados

Uma modificação do algoritmo de relaxação dos Campos Potenciais Orientados foi proposta e implementada neste trabalho, capaz de manusear diferentes comportamentos em um mesmo mapa discreto do ambiente. Esta proposta, denominada Campos Potenciais Localmente Orientados (CPLO), foi utilizada para implementar o nível de Planejamento e um Sistema Baseado em Regras foi implementado para estabelecer as metas e restrições de cada jogador no nível Estratégico.

Como visto na seção 3.3.3, foi sugerida uma família de funções que não são mais funções harmônicas, contudo continuam sendo soluções para o Problema de Valor de Contorno. Foi mostrado que o comportamento da função descrita pela equação (3.21) gera um campo orientado na direção de um vetor $\mathbf{v}$, ou seja, um Campo Potencial Orientado (CPO).

As mesmas limitações observadas no CPH também foram detectadas com o método CPO, ou seja, embora seja permitido que mais de um robô atue sobre a mesma grade, todos terão o mesmo comportamento de seguir o mesmo campo distorcido por $\mathbf{v}$.

A proposta apresentada nesta seção, utiliza uma única grade na qual muitos robôs possam seguir diferentes vetores de comportamento $\mathbf{v}$ que conduzam à meta por diferentes trajetórias. Este método foi obtido ao aplicar a equação (3.22) somente na vizinhança de

cada robô. Assume-se como vizinhança todas as células contidas em uma circunferência de raio fixo com centro na posição do robô.

Como vários robôs estão sendo considerados, vamos supor que cada robô k, que atua sobre a grade, possua um vetor de comportamento $\mathbf{v}_k$ e um parâmetro ϵ_k. Esta modificação sobre o método de relaxação foi denominada Campos Potenciais Localmente Orientados (CPLO), pois na vizinhança de cada robô o campo potencial pode ser modificado pela orientação do vetor de comportamento. O algoritmo de Gauss-Seidel que considera a proposta do CPLO está descrita no Algoritmo 1 que segue abaixo:

Algoritmo 1 Relaxação do CPLO

1: $\varphi \leftarrow$ raio de vizinhança do robô
2: **para toda** célula (i, j) da grade **faça**
3: encontre o robô k, dentre todos os robôs da grade, com a menor distância d_k entre ele e a célula (i, j)
4: **se** $d_k < \varphi$ **então** $\triangleright$ $p_{i,j} \in$ vizinhança de k
5: $p_{i,j} \leftarrow \frac{1}{4}[p_{i,j+1}+p_{i,j-1}+p_{i+1,j}+p_{i-1,j}]+\frac{\epsilon_k}{8}[(p_{i+1,j}-p_{i-1,j})\mathbf{v}x_k+(p_{i,j+1}-p_{i,j-1})\mathbf{v}y_k]$
6: **senão** $\triangleright$ $p_{i,j} \notin$ vizinhança de k
7: $p_{i,j} \leftarrow \frac{1}{4}[p_{i,j+1} + p_{i,j-1} + p_{i+1,j} + p_{i-1,j}]$
8: **fim se**
9: **fim para**

Nossa proposta para o controle de múltiplos robôs consta em utilizar o método CPLO para implementar o nível de Planejamento. Assim, todos os robôs farão consultas a grade para direcionar seus movimentos no campo. Para implementar o nível de Estratégia foram utilizadas regras para modificações dinâmicas de $\mathbf{v}_k$ e ϵ_k de acordo com a funcionalidade de cada robô.

Para o time de 5 robôs do simulador FIRA foram propostas algumas funcionalidades que são definidas no nível Missão. O Sistema Baseado em Regras proposto é constituído das seguintes funcionalidades - atacante, defesa, lateral e goleiro, definidas a seguir:

Atacante

Um campo que orientado por um vetor pode fornecer comportamentos distintos sobre um mesmo ambiente, alterando-se apenas os parâmetros $\mathbf{v}$ e ϵ de atualização da grade. Um campo potencial que esteja sempre apontando para o gol poderia implementar o comportamento de um robô atacante que com este método poderia direcionar seu chute ao gol. Na Figura 4.7 mostra-se o esquema do movimento desejado para um atacante para que este direcione seu chute ao gol.

Comparando-se a Figura 4.7 com a Figura 4.8, pode-se perceber que este movimento

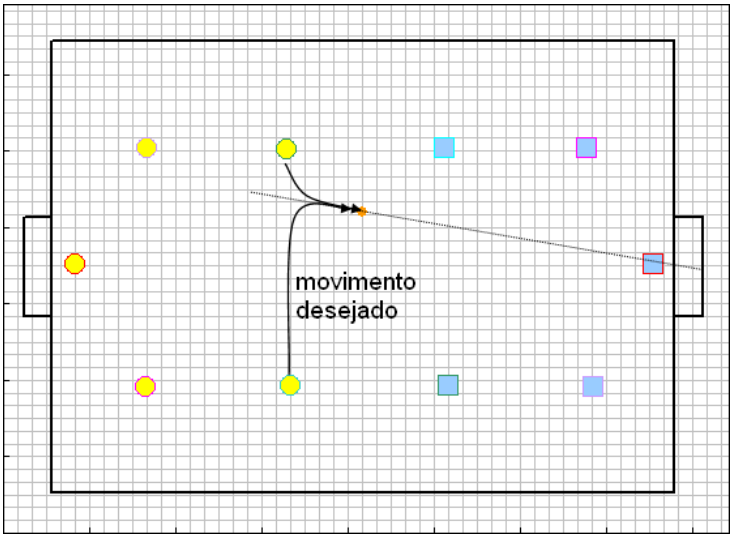


Figura 4.7: Movimento desejado para um robô atacante.

pode ser conseguido utilizando Campos Potenciais Orientados.

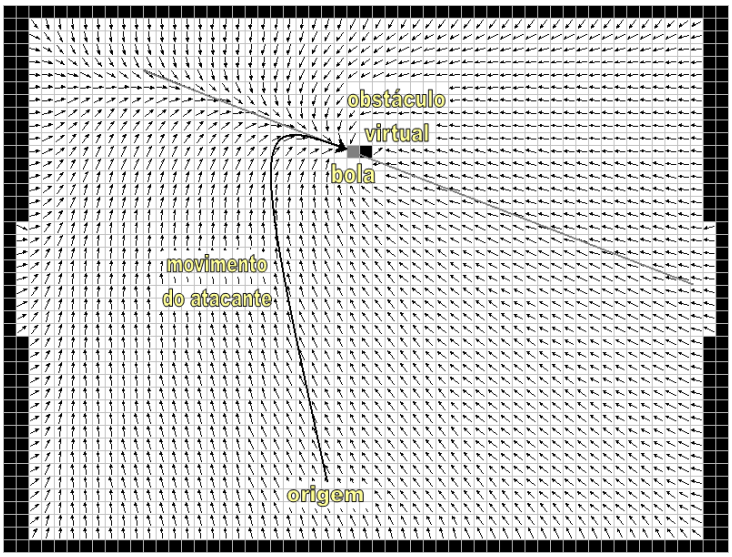


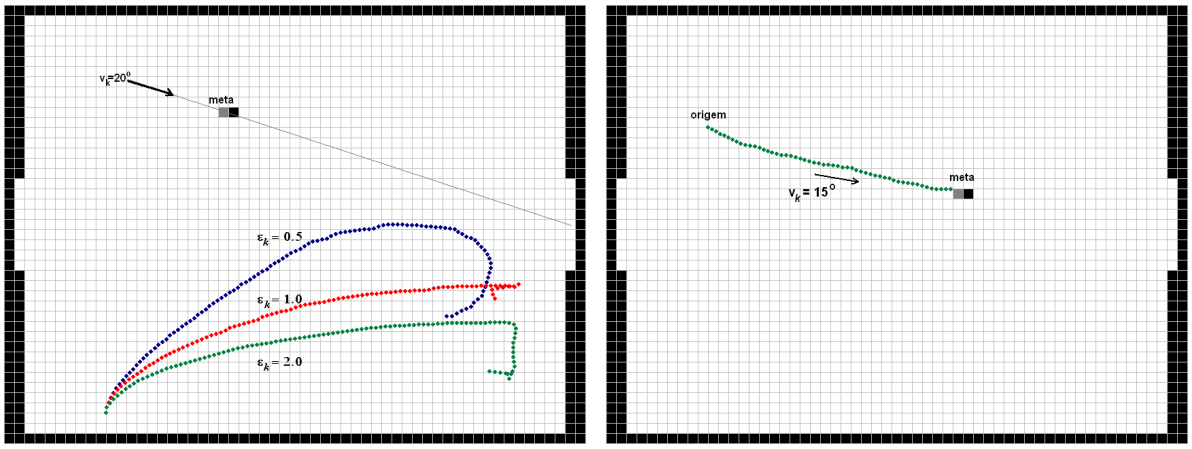
Figura 4.8: Comportamento gerado utilizando o método CPO com parede virtual.

Embora CPO direcione melhor o chute ao gol, o método sozinho não é capaz de evitar que o atacante chute contra o próprio gol. Como pode ser observado na Figura 4.8), foi utilizado novamente o recurso de *paredes virtuais* para colocar um obstáculo virtual junto à bola que previna chutes não desejados.

Esta implementação garante um bom atacante, mas assim como o método CPH, todos os robôs seguirão o mesmo campo potencial. Isto faria com que os 5 robôs atuassem com o comportamento de atacante e o pior, disputando a bola entre eles mesmos.

Um experimento semelhante foi realizado, mas desta vez utilizando o método CPLO

para gerar as trajetórias. Assim, o robô atacante A com $\mathbf{v}_A$ direcionado ao gol foi colocado sobre o campo. Contudo percebeu-se que o comportamento do robô não se assemelha ao da Figura 4.8. A trajetória seguida pelo robô foi descrita utilizando $\mathbf{v}_A = 20^o$ e pode ser observada na Figura 4.9(a). De cima para baixo, pode-se notar a diferença das trajetórias ao utilizar ϵ_A iguais a 0.5, 1.0 e 2.0, respectivamente. Como já mencionado anteriormente, o valor de ϵ é um valor de influência de $\mathbf{v}$ sobre o campo, portanto, quanto maior o valor de ϵ_A, mais a trajetória tenta se manter a 20^o, ou seja, indo em direção à parte inferior da figura. Contudo, caso o robô já esteja próximo ao ângulo desejado, a trajetória é mantida em $\mathbf{v}_A$. Na Figura 4.9(b), é mostrada a trajetória de um robô seguindo seu vetor $\mathbf{v}_A = 15^o$ com um $\epsilon_A = 2.0$ forte o suficiente para mantê-lo nesta trajetória.



(a) (b)

Figura 4.9: Atacante em um CPLO. (a) vetor $\mathbf{v}_A = 20^o$ direcionado para o gol. (b) vetor $\mathbf{v}_A = 15^o$ descrevendo a trajetória desejada.

Dois pontos importantes podem ser concluídos com os experimentos acima.

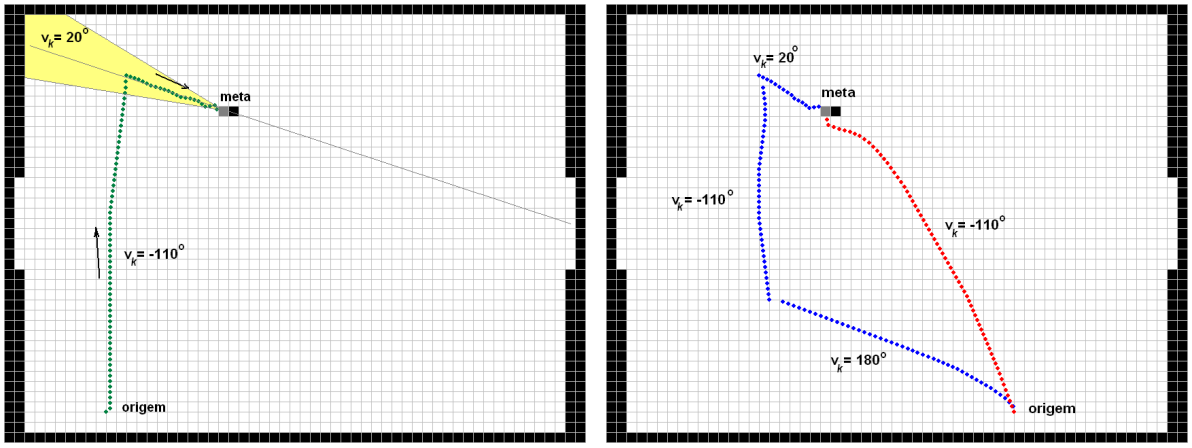
- Um valor alto para ϵ_A, como por exemplo, $\epsilon_A = 2.0$, força o robô a seguir o mais fielmente possível o vetor $\mathbf{v}_A$. Não será seguido fielmente, pois o campo sofre influência dos outros obstáculos e também da meta.

- Somente quando o robô já está alinhado com a direção bola-gol, é que ele descreve a trajetória desejada, definida por $\mathbf{v}_A$.

Utilizando-se da propriedade de $\mathbf{v}_A$ poder variar durante a execução do jogo, propõe-se que os dois comportamentos acima descritos sejam mesclados para produzir a trajetória desejada para o robô atacante. Como mostrado na Figura 4.10(a), o robô seguiu seu vetor $\mathbf{v}_A = -110$ até chegar em uma região na qual $\mathbf{v}_A$ deve direcionar o chute ao gol, que neste caso, é $\mathbf{v}_A = 20^o$. O ângulo adotado foi um pouco menor do que -90º para compensar a

repulsão causada pela parede da esquerda traçando uma trajetória bem próxima de -90º. Para o caso da bola estar na parte inferior do campo, a estratégia é similar a apresentada, mudando somente a orientação dos vetores.

Contudo, caso o robô esteja entre a bola e o gol do adversário, o chute pode não sair direcionado ao seguir $\mathbf{v}_A = -110$ como mostrado na trajetória mais a esquerda da Figura 4.10(b). Para evitar esta falha, mais uma regra foi adicionada. Esta nova regra estabelece que o robô só pode utilizar $\mathbf{v}_A$ no sentido vertical ou direcionado ao gol quando o robô estiver entre seu próprio gol e a bola. Para as outras posições será utilizado um $\mathbf{v}_A = 180$ para levar o robô a uma posição válida para o ataque. A trajetória da esquerda mostrada na Figura 4.10(b) foi descrita utilizando-se esta nova regra para o robô atacante.



(a) (b)

Figura 4.10: Atacante em um CPLO. (a) vetor $\mathbf{v}_k = 20^o$ direcionado para o gol. (b) vetor $\mathbf{v}_k = 15^o$ descrevendo a trajetória desejada.

O conjunto de regras definidas para controle do robô fazem parte da base de conhecimento necessária para ajustar as trajetórias do robô dentro de um CPLO. Estas regras foram agrupadas e são apresentadas no Algoritmo 2.

Defesa

O comportamento desejado para o robô de defesa é que este vigie uma área e impeça a passagem da bola do lado do oponente para o lado dos aliados. Nossa proposta para gerar um comportamento de defesa para o robô limita os movimentos deste a uma região do campo, na qual o robô deve mover-se para ficar alinhado horizontalmente com a bola. Como mostrado na Figura 4.11, a linha horizontal da posição da bola, H, atravessa a área de atuação do robô, e assim, o campo fica dividido em seis áreas, sendo que cada área possui um valor de $\mathbf{v}_D$ que direciona o robô para uma posição que bloqueia a passagem da bola. O robô da defesa move-se seguindo um CPLO e assume o valor de $\mathbf{v}_D$ da área

Algoritmo 2 Regras de ataque para o controle do robô

Seja (Ax, Ay) a posição de um robô atacante A, (bx, by) a posição da bola b, gol a posição do gol adversário e ξ a margem de erro aceitável entre o alinhamento robô-bola e bola-gol.

se $Ax > bx$ **então**
  $\mathbf{v}_A = 180^o$
senão se $|ang(A, b) - ang(b, gol)| < \xi$ **então** ▷ ang = ângulo
  $\mathbf{v}_A = ang(b, gol)$
senão se $Ay > by$ **então**
  $\mathbf{v}_A = 110^o$
senão
  $\mathbf{v}_A = -110^o$
fim se

na qual estiver incluso utilizando um $\epsilon_D = 2.0$.

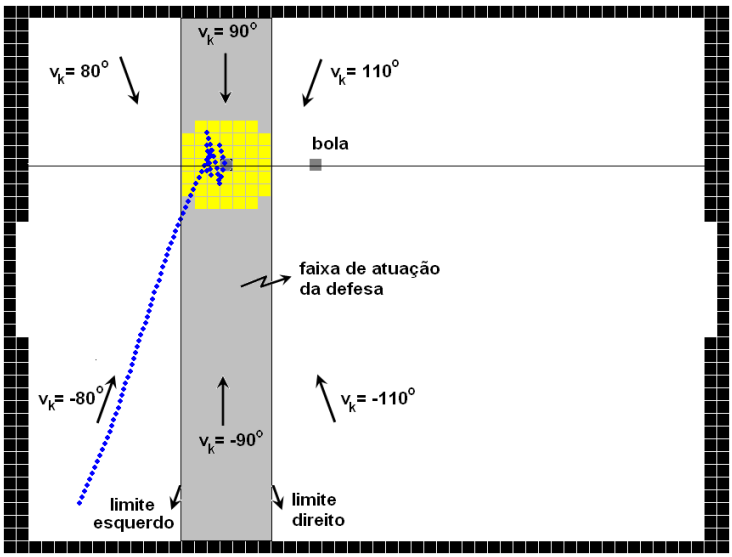


Figura 4.11: Comportamento de defesa gerado por Campos Potenciais Localmente Orientados com $\mathbf{v}_D$ definido dinamicamente pela área na qual o robô está inserido.

Como visto, o comportamento defensivo de um robô também necessita de regras para ajustar o valor do vetor $\mathbf{v}$. Portanto, a base de conhecimento será então acrescida das regras para defesa apresentadas no Algoritmo 3.

Lateral

Em um dos experimentos realizados, descobriu-se um comportamento bastante interessante que faz o robô seguir paredes ao direcionar o vetor de um robô perpendicularmente à parede desejada. Uma vez junto a parede, o robô ficará vagando de um lado ao ou-

Algoritmo 3 Regras de defesa para o controle do robô

Seja $D = (Dx, Dy)$ a posição de um robô defensor, $b = (bx, by)$ a posição da bola e LE_D, LD_D são os limites esquerdo e direito, respectivamente, da área de atuação de robô D.

se $Dy < by$ **então** ▷ robô acima da linha horizontal da posição da bola
  se $Dx < LE_D$ **então**
    $\mathbf{v}_D = 80^o$
  senão se $Dx > LD_D$ **então**
    $\mathbf{v}_D = 110^o$
  senão
    $\mathbf{v}_D = 90^o$
  fim se
senão ▷ robô abaixo da linha horizontal da posição da bola
  se $Dx < LE_D$ **então**
    $\mathbf{v}_D = -80^o$
  senão se $Dx > LD_D$ **então**
    $\mathbf{v}_D = -110^o$
  senão
    $\mathbf{v}_D = -90^o$
  fim se
fim se

tro sofrendo muito pouca influência de atração pela meta. Na Figura 4.12 é mostrado o comportamento de seguir paredes nos três robôs presentes no ambiente. A trajetória mostrada foi criada por robôs com $\mathbf{v}_1 = 180^o$, $\mathbf{v}_2 = -90^o$ e $\mathbf{v}_3 = 90^o$, todos eles com $\epsilon_k = 2.0 | k = [1, 2, 3]$.

O comportamento de seguir paredes foi modificado para fazer com que o robô permaneça nas laterais mas orientado verticalmente pela posição da meta (bola). Como pode ser observado na Figura 4.13, a posição da bola divide o campo verticalmente em duas partes, as quais atribuem ao robô um valor de $\mathbf{v}_k$ que faça o robô ir em direção a parede mas também se aproximar da linha vertical da posição da bola. O exemplo mostrado na Figura 4.13, mostra dois robôs, sendo um em cada lateral. O robô da parte superior alternará o seu vetor $\mathbf{v}_k$ entre -45º e -135º dependendo da sua posição em relação a bola, mantendo-se assim na parte superior. O robô da parte inferior segue o mesmo algoritmo, contudo, com valores de $\mathbf{v}_k$ diferentes.

O comportamento de seguir uma meta pela lateral do ambiente é descrito pelo conjunto de regras apresentado no Algoritmo 4:

• Goleiro

Inicialmente foi adotado um goleiro g com $\epsilon_g = 2.0$ and $\mathbf{v}_g = 180^o$, para manter o robô

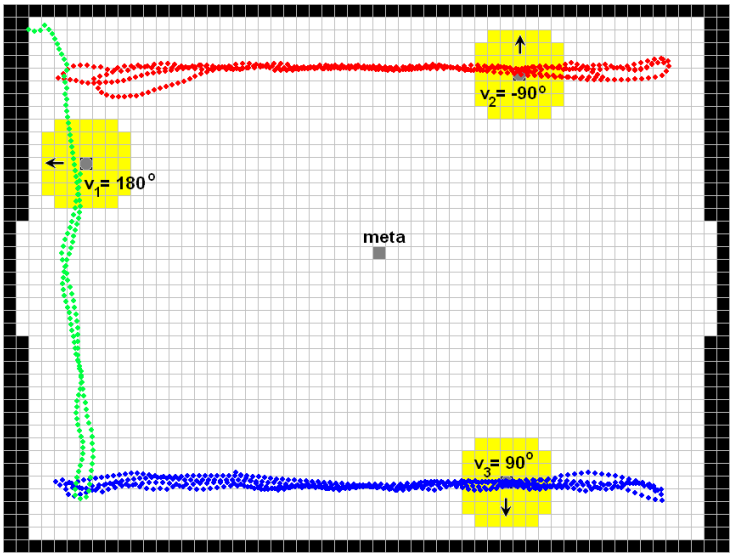


Figura 4.12: Comportamento de seguir paredes gerados utilizando-se CPLO.

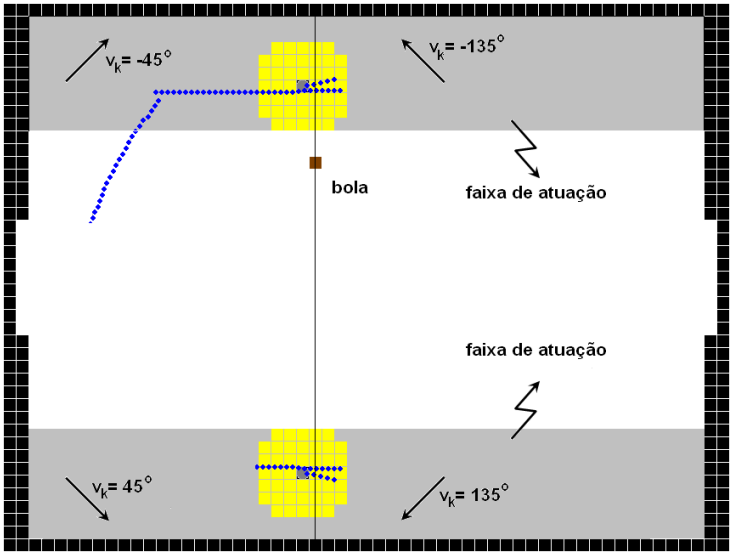


Figura 4.13: Comportamento de observador lateral com CPLO.

Algoritmo 4 Regras do comportamento lateral

Seja (Lx, Ly) a posição de um robô L e (bx, by) a posição da bola b

se L deve seguir a lateral superior **então**
  se $Lx < bx$ **então**
    $\mathbf{v}_L = -45^o$
  senão
    $\mathbf{v}_L = -135^o$
  fim se
senão ▷ L deve seguir a lateral inferior
  se $Lx < bx$ **então**
    $\mathbf{v}_L = 45^o$
  senão
    $\mathbf{v}_L = 135^o$
  fim se
fim se

próximo de seu gol. Entretanto, o comportamento adotado pelo robô não foi satisfatório, ou ele ignorava posição da bola devido ao valor de ϵ_g ser alto, ou abandonava seu posto de goleiro e perseguia a bola quando o valor de ϵ_g era baixo.

Notou-se que o comportamento de um goleiro é um comportamento de defesa que possui a restrição de não se afastar do gol. Para criar estes comportamentos, algumas regras foram criadas, mesclando o comportamento de defesa e de lateral além de adicionar limites superior e inferior para sua movimentação. A Figura 4.14 ilustra o comportamento do goleiro, que assim como no comportamento de defesa o robô deve mover-se para ficar alinhado horizontalmente com a bola. Pode-se notar também que somente dois valores para $\mathbf{v}_g$ são adotados, os quais direcionam o robô para a linha da posição da bola e contra os limites do fundo do campo, assemelhando-se ao comportamento *lateral*.

4.2 Controle de Velocidade

Um dos problemas para controlar o movimento do robô é o controle do hardware, ou seja, ajustar a velocidade de cada roda a fim de direcionar o robô para seu objetivo. Isto também ocorre ao utilizar o simulador FIRA, no qual cada robô possui as variáveis *velocityLeft* e *velocityRight*, nas quais ajusta-se a velocidade das rodas esquerda e direita, respectivamente.

O pacote de programação que acompanha o simulador inclui uma função que gira o robô para a posição desejada. Contudo, o robô realiza este giro sobre o próprio eixo e só então pode voltar a andar novamente. No entanto, deseja-se que o robô chegue a seu

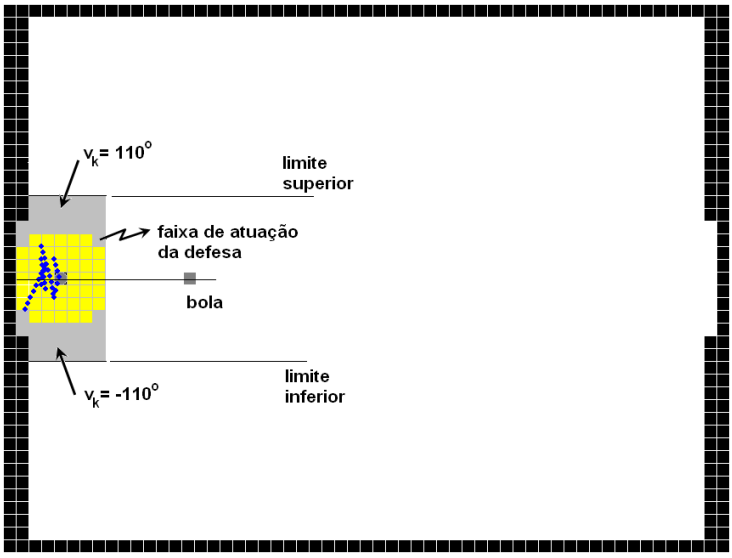


Figura 4.14: Comportamento do goleiro com CPLO.

objetivo o mais rápido possível, e isto torna necessário que o controle entre velocidade angular e velocidade linear estejam equilibradas de forma a utilizar a velocidade máxima (125 pol/s) sempre que possível.

Em nossa proposta de implementação do controle de velocidade, será considerado que o ângulo do robô Φ e de destino Θ pertençam ao intervalo [-180º,180º]. A diferença entre estes ângulos foi chamada de θ, e algumas considerações são feitas para obter-se $\theta \in [-180^o, 180^o]$, como segue:

$$aux = \Theta - \Phi \tag{4.5}$$

$$\theta = \begin{cases} aux - 360^o & se \quad aux > 180^o \\ aux + 360^o & se \quad aux < -180^o \\ aux & se \quad -180^o \leq aux \leq 180^o \end{cases} \tag{4.6}$$

Estes ajustes acontecem pois existem casos no qual seja melhor girar no sentido oposto, ou seja, quando $|\theta| > 180^o$. Como exemplo, pode ser visto na Figura 4.15 um suposto robô com $\Phi = -135^o$ que deve-se mover para o ângulo desejado $\Theta = 135^o$. A diferença utilizando (4.5) seria dada por $aux = 135 - (-135) = 270$. Como $270^o > 180^o$ então girar em sentido contrário seria uma melhor solução, ou seja, girar -90^o no sentido horário.

Nossa proposta para o controle da velocidade angular faz com que o robô gire sobre o próprio eixo para atingir a direção desejada. A velocidade de rotação é controlada por uma função linear

$$f(\theta) = V_{max} \cdot \theta / 180$$

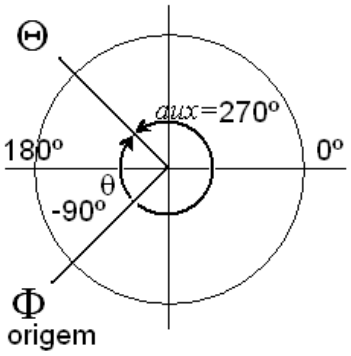


Figura 4.15: Exemplo de rotação de um robô. Duas opções: 270º em anti-horário ou -90º sem sentido horários.

que faz com que robô gire com velocidade máxima nas rodas, $V_{max} = 125$, quando $\theta = 180^o$ e não gire quando $\theta = 0^o$. Para fazer o robô girar sobre o próprio eixo, basta deixar as velocidades iguais, porém em sentido oposto, para ambas as rodas. Desta forma, a velocidade angular para cada roda é dada por:

$$\begin{aligned} velocityLeft &= f(\theta) \\ velocityRight &= -f(\theta) \end{aligned}$$

A velocidade linear, segue um comportamento inversamente proporcional ao valor da velocidade rotacional, pois deseja-se que o robô tenha a velocidade máxima para frente somente quando $\theta = 0^o$ e tenha velocidade linear zero, quando $\theta \geq L_\theta$. A constante L_θ foi estabelecida para indicar que o robô deve somente girar sobre o próprio eixo quando $\theta \geq L_\theta$. O limite fornecido por L_theta ajusta a suavidade do movimento do robô ao mudar de direção. Quanto menor L_theta mais brusco são os movimentos, ou seja, o robô pára, gira e somente então volta a ir para frente. A velocidade entre L_θ e 0^o foi descrita finida por nós como uma função linear. Assim, a velocidade linear é dada pela seguinte função:

$$g(\theta) = \begin{cases} -V_{max} \cdot |\theta|/L_\theta + V_{max} & se \quad |\theta| \leq L_\theta \\ 0 & se \quad |\theta| > L_\theta \end{cases}$$

A composição da velocidade angular e da velocidade linear resultará na velocidade de cada roda de um robô, ou seja,

$$\begin{aligned} velocityLeft &= f(\theta) + g(\theta) \\ velocityRight &= -f(\theta) + g(\theta). \end{aligned}$$

Uma outra idéia proposta é fazer com que o robô ande para trás quando a meta estiver

voltada para a parte de trás do robô assim evita-se que o robô faça giros maiores que 90º. Esta proposta não será abordada neste trabalho, pois, os giros rápidos que o robô faz para se direcionar muitas vezes funciona como um chute, ou seja, um comportamento desejado dentro do ambiente do futebol de robôs.

4.3 Considerações

Algumas propostas foram sugeridas para a implementação da arquitetura ACIn. Estas propostas, envolvem a utilização de métodos de planejamento de trajetórias combinados com um SBR capaz de modificar as trajetórias dependendo da funcionalidade de cada integrante da equipe de robôs. A eficiência de cada proposta foi comprovada através dos vários experimentos apresentados no capítulo seguinte.

5

Experimentos

O ambiente multi-robôs no qual a ACIn será avaliada é o ambiente de futebol de robôs e, portanto, será apresentado neste capítulo este ambiente, bem como o simulador utilizado para os testes. Em alguns experimentos, foram utilizadas ferramentas que implementam algoritmos de Inteligência Artificial, as quais serão descritas neste capítulo. Em seguida, serão apresentados os experimentos realizados bem como uma discussão sobre os resultados obtidos.

5.1 Ambiente de Teste

O futebol de robôs é uma iniciativa internacional voltada à pesquisa e educação, visando promover desenvolvimentos ligados às áreas de Inteligência Artificial e Robótica Inteligente Kitano et al. [1997a]. Uma das maiores razões do futebol de robôs ser o foco de estudo para muitos pesquisadores é fato deste ser um problema padrão e, portanto, a pesquisa pode ser claramente definida e acompanhada. Além disto, através da adoção deste problema pode-se fazer avaliações de várias teorias, algoritmos, arquiteturas e desempenhos. Tudo isto aliado a uma grande variedade de tecnologias que podem ser integradas e analisadas [Costa & Pegoraro, 2000; Kitano et al., 1997b].

O domínio de futebol de robôs é dinâmico e imprevisível, resultando num domínio bastante complexo, que exige o uso de sistemas com alto grau de autonomia atuando em tempo real. Diversos tópicos de pesquisa são oferecidos por este domínio, incluindo entre outros:

1. combinação entre as abordagens reativas e deliberativas;
2. reconhecimento, planejamento e raciocínio em tempo real;
3. completa integração entre percepção, ação e cognição em ambiente dinâmico;

4. definição de um conjunto de comportamentos robusto – cada agente deve ser capaz de realizar tanto ações individuais quanto colaborativas;
5. percepção em tempo real, robusta e confiável incluindo rastreamento de múltiplos objetos em movimento.

Em uma das categorias do futebol de robôs considera-se uma câmera para capturar a imagem de todo o campo e utilizando-se de recursos de processamento de imagem, um computador poderá fornecer as posições dos robôs pertencentes ao time, dos adversários, do gol e da bola. Um computador, chamado de técnico, pode utilizar algum sistema de controle que utilize as informações disponíveis para dar ordens de como os robôs de seu time devem agir (Figura 5.1).

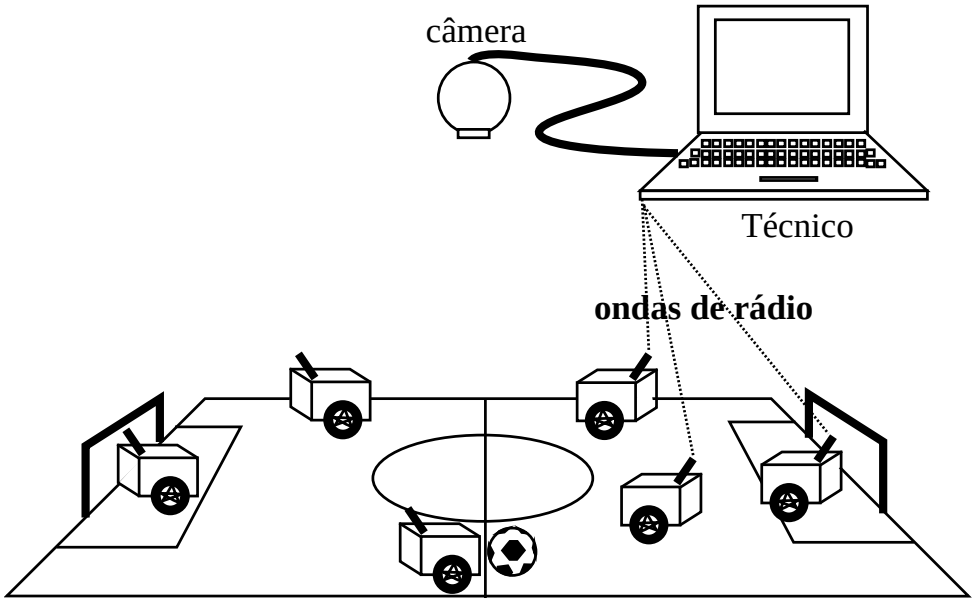


Figura 5.1: Futebol de robôs controlados por um computador central.

O computador central, além de atuar como técnico montando e executando a tática de jogo, atua como o sistema de navegação de cada robô, realizando o planejamento das trajetórias. Além disso, é o responsável pelos comandos de baixo nível como velocidade e controle de direção.

A arquitetura de controle inteligente ACIn é nossa proposta para atuar como sistema central realizando as tarefas citadas acima. Embora, a arquitetura ACIn seja utilizada inicialmente com futebol de robôs, a utilização de comportamentos como busca pela meta e desvio de obstáculos também podem ser utilizados com outros objetivos em sistemas robóticos. Desta forma, o futebol de robôs será utilizado como um primeiro passo para testar de eficiência da Arquitetura de Controle Inteligente ACIn.

Em virtude do ICMC ainda não possuir um time de futebol de robôs, o ambiente escolhido para testar o sistema multi-robôs foi um simulador oficial das competições disponibilizado pela Federação da Associação Internacional de Futebol de Robôs (FIRA).

Robot Soccer v1.5a é o simulador da liga média de futebol de robôs da FIRA, sendo cinco robôs para cada time. O simulador é constituído de um ambiente virtual que simula uma partida de Futebol de Robôs e as apresenta ao usuário de uma forma interativa. Ao instalar o simulador, um projeto implementado em linguagem C para ser compilado pelo Microsoft Visual C++. O usuário pode modificá-lo e implementar sua própria estratégia para o controle dos robôs tendo como base este código fonte. Embora compiladores C++ da Borland não sejam compatíveis, para quem não possui o Microsoft Visual Studio, uma solução é utilizar o Bloodshed Dev-C++, um compilador freeware, que com uns pequenos ajustes também compila os arquivos de controle de robôs para o simulador FIRA.

O simulador da FIRA fornece todos os dados referentes às medidas do campo em polegadas e de velocidade em pol/seg. Sendo assim, o campo de jogo do simulador mede 70pol de largura e 86pol de comprimento, como pode ser visto na Figura 5.2. As posições de cada jogador no campo também são fornecidas em polegadas pelo simulador. Para movimentar um robô basta altera a velocida de cada uma de suas rodas, sendo que a velocidade de cada roda pode variar de -125 a 125 (pol/s). O controle da velocidade de cada roda é independente uma da outra.

O ambiente de simulação faz chamadas a um procedimento a cada instante de tempo e passa como parâmetro as informações sobre o ambiente em uma variável (env). Através do env tem-se acesso a informações sobre os dois times (aliados e oponentes), além da posição corrente e a última posição da bola.

Para cada robô presente no campo, seja aliado ou oponente, é possível obter, por meio da variável (env), sua posição cartesiana no campo e o seu ângulo de rotação. Para os robôs aliados, tem-se o acesso para leitura e atualização da velocidade da roda direita e esquerda. A alteração da velocidade das rodas de um robô é a única forma que o simulador fornece para movimentar o time de robôs pelo campo. Portanto faz-se necessário a implementação de um sistema de controle de velocidade para facilitar o controle de trajetória dos robôs.

O simulador, a base do programa C++ e as regras de jogos podem ser baixados via `www.fira.net` na liga SimuroSot.

Após ter apresentado o ambiente de teste utilizado para realização dos experimentos, serão apresentados a seguir os resultados obtidos através da aplicação das técnicas já descritas no capítulo anterior.

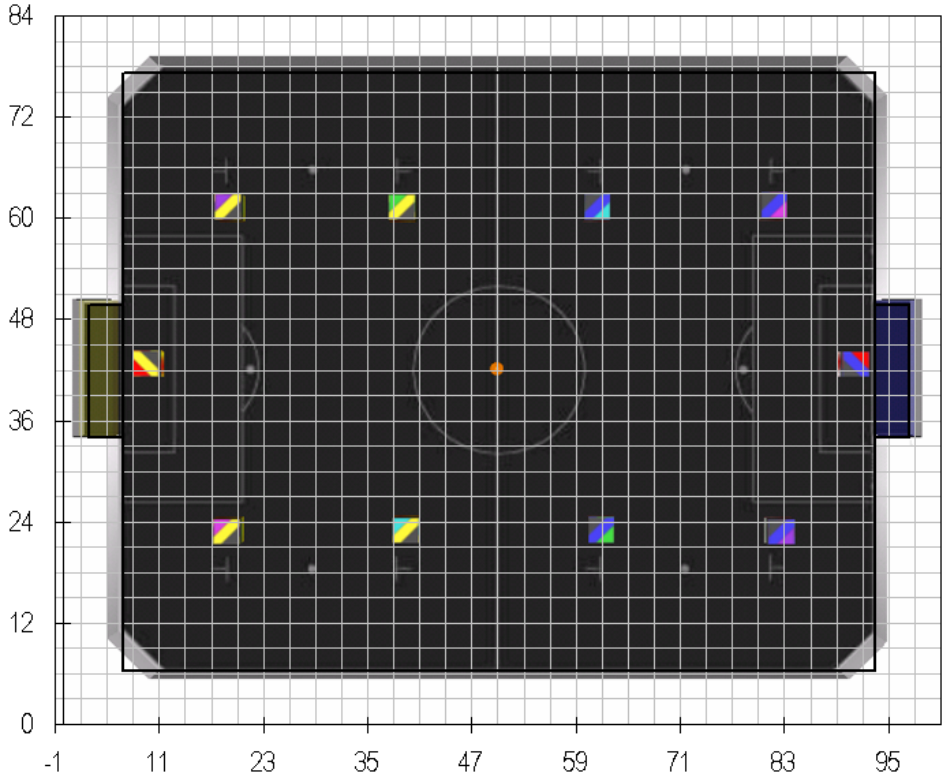


Figura 5.2: Simulador de futebol de robôs da FIRA.

5.2 Aprendizado via redes neurais MLP

Os dados coletados foram fornecidos pelo ambiente do simulador FIRA, ou seja, as posições e velocidade das rodas dos robôs aliados, posição dos robôs oponentes e posição da bola. O simulador espera que seja informado o movimento que cada robô deva fazer, ajustando para isso a velocidade das rodas direita e esquerda do robô. A partir destes dados, foram sugeridas algumas propostas para coleta de dados e pré-processamento dos dados. A seguir serão detalhadas estas propostas, assim como as estruturas de rede MLP escolhidas e os resultados obtidos de cada experimento.

5.2.1 Aprendizado da Velocidade dos Motores

A idéia de um robô artilheiro sendo controlado por um ser humano foi a primeira proposta para coletar os dados sobre comportamentos. Assim, a base de dados colhida corresponderia ao estilo de jogo adotado pelo tutor humano e a estratégia aprendida pela rede deveria representar o conhecimento utilizado pelo tutor durante o jogo.

Esta primeira idéia pareceu ser ideal, pois se o robô artilheiro pudesse decidir por si só qual seria a velocidade de cada um de seus motores, então todos os movimentos poderiam

ser realizados pela rede MLP, tendo então uma grande chance de sucesso do time.

A base de dados foi construída com exemplos contendo os seguintes dados: coordenadas (x, y) da bola, coordenadas (x, y) dos 5 robôs aliados (incluindo o artilheiro), coordenadas (x, y) dos 5 robôs adversários e velocidade dos motores de um robô artilheiro. Isto representa um total de 22 variáveis de entradas e 2 variáveis de saída para cada exemplo coletado. Com estes dados, uma rede neural MLP poderia aprender a transformar a informação destas 22 variáveis em velocidade das rodas direita e esquerda para um robô artilheiro.

A rede neural MLP foi projetada inicialmente com 22 nós de entrada e 2 de saída, como mostrado na Figura 5.3. Além desta arquitetura, também foram realizados experimentos utilizando nós nas camadas intermediárias, e valores diferentes para taxa de aprendizado. Embora a idéia inicial fosse muito animadora, todas as tentativas apresentaram erros de treinamento superiores a 30%.

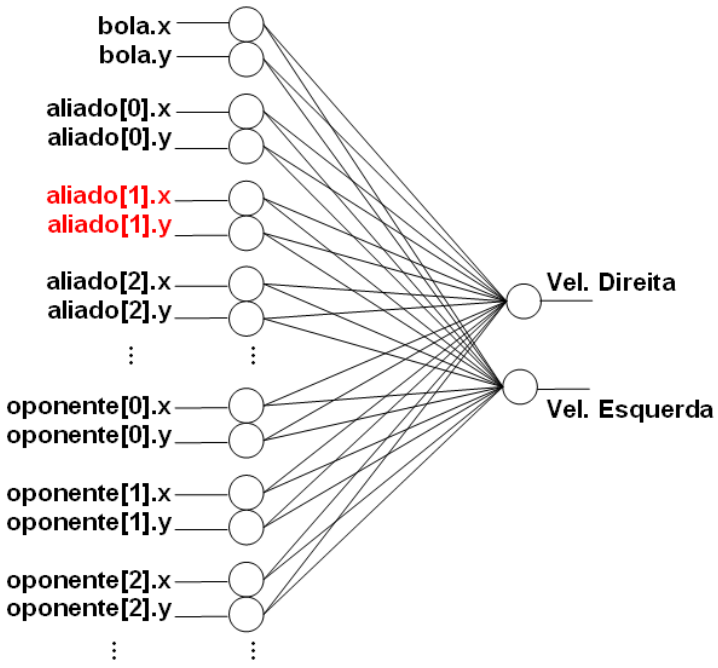


Figura 5.3: MLP proposta para o aprendizado da velocidade das rodas do artilheiro.

Supôs-se então que esta alta taxa de erro poderia ser devido ao número de nós da camada de entrada. Portanto, passou-se a utilizar apenas 2 robôs de cada time (um goleiro e um artilheiro). Com isso, reduziu-se o número de nós de entrada para 10, mesmo assim, não foi encontrada uma arquitetura para a rede MLP que se mostrasse eficiente.

5.2.2 *Aprendizado utilizando Saídas Binárias*

O problema de aprendizado para redes MLPs pode ser facilitado quando se trabalha com saída binárias, mas para isso, outro tipo de saída para rede teve que ser escolhida. Em vez de velocidade das rodas, optou-se por classes de direção provenientes da combinação de teclas pressionadas para controlar o robô atacante via teclado, ou seja, frente, frente-direita, frente-esquerda, trás, trás-direita e trás-esquerda.

O exemplo de jogo continua com 2 jogadores em cada time, e portanto, a base de dados foi montada com os seguintes dados: coordenadas da bola, coordenadas de 2 robôs aliados, coordenadas de 2 robôs adversários e a classe correspondente à combinação de teclas pressionadas do robô artilheiro. Esta base corresponde a uma rede MLP com 10 nós de entrada e 6 nós de saída, como mostrando na Figura 5.4.

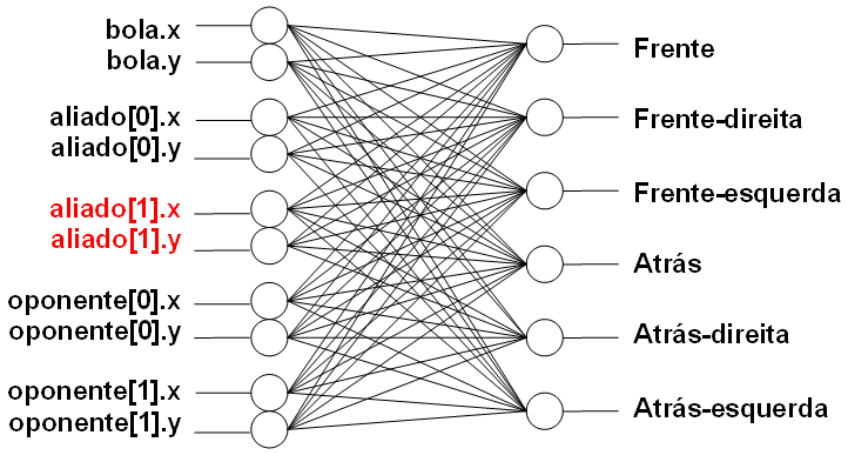


Figura 5.4: MLP com saída binária. Somente um nó será escolhido para direcionar o artilheiro aliado[1].

Para o treinamento desta rede MLP, foi coletado um conjunto com 5000 exemplos divididos em 3750(75%) exemplos para treinamento e 1250(25%) exemplos para teste. Após vários experimentos de treinamentos de diferentes arquiteturas de redes, chegou-se a uma rede MLP 10x80x80x6 [1] utilizando taxa de aprendizado $\alpha = 0.2$. Com esta rede obteve-se um *erro de treinamento* MSE = 0.010 e *erro de teste* MSE = 0.161. Em relação a estas medidas, deve ser notado que o erro de treinamento serve como medida de avaliação da quantidade de exemplos apresentados que não foram aprendidos e o erro de teste, ou erro futuro, é uma medida de avaliação da quantidade de exemplos nunca vistos anteriormente pela rede e que não foram classificados corretamente.

[1]Notação que descreve uma rede MLP com 4 camadas, sendo: 10 nós na camada de entrada, 80 nós na primeira camada intermediária, 80 na segunda camada intermediária intermediária e 6 na camada de saída.

Após as fases de treinamento e teste, a rede MLP foi convertida do formato JavaNNS para um arquivo em linguagem C por meio da ferramenta SNNS2C. A rede então passou a controlar um robô artilheiro do simulador FIRA, porém, sem sucesso nos movimentos realizados. O robô movia-se com trajetórias aparentemente aleatórias, sem direcionar sua trajetória para a bola e colidindo com os demais robôs ou com as paredes. Isto pode ter ocorrido, pois o tutor humano não possuía agilidade suficiente para controlar um robô tão rapidamente quanto os demais robôs que eram controlados via software do simulador FIRA.

5.2.3 Aprendizado utilizando saída Gaussiana

Para resolver o problema do controle manual, um planejador reativo baseado na técnica de Campos Potenciais Arkin [1989] (seção 3.2) foi implementado para controlar o robô artilheiro. O planejador considera que os robôs geram campos repulsivos e a bola gera um campo atrativo, e portanto, o robô consegue ir em direção a bola desviando dos demais robôs. Os comportamentos bem definidos, de busca pela bola e desvio de obstáculos, também serão úteis para avaliar o desempenho a rede MLP segundo estes comportamentos.

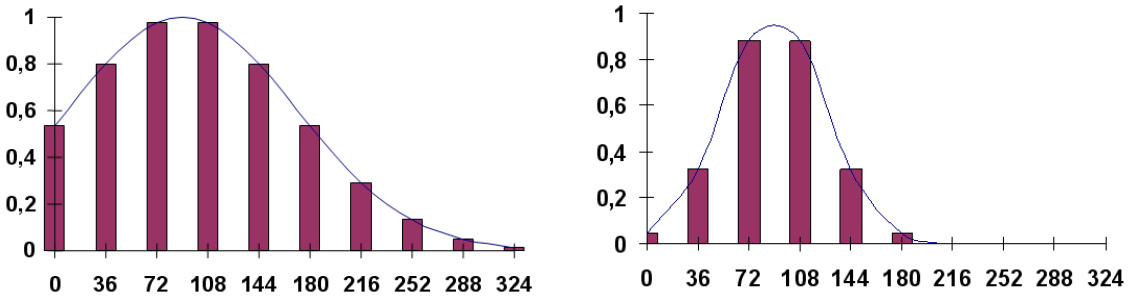
A próxima tentativa utilizou o ângulo de rotação do robô para determinar a estratégia adotada pelo robô. Através do ângulo de rotação do robô, determinava-se para qual direção deveria o robô seguir para cumprir seu objetivo, tanto de marcar gols quanto de evitar obstáculos. A base de dados colhida continha os seguintes dados: coordenadas da bola, coordenadas de 2 robôs aliados, coordenadas de 2 robôs adversários e o ângulo de rotação do robô artilheiro. Para esta base, propôs-se que o robô aprendesse qual direção ele deveria seguir para cada situação encontrada. Esta direção pode ser obtida no simulador FIRA através da variável ângulo de rotação do robô em um determinado momento.

A rede neural MLP não conseguiu aprender utilizando somente um neurônio na camada de saída para identificar o ângulo de rotação. Como já foi mencionado, o aprendizado de números reais pode ser difícil para a rede e como tentativa de solucionar este problema propôs-se a discretização do ângulo em n partes utilizando uma função gaussiana.

Para exemplificar, considere que deseja-se discretizar o ângulo de rotação $\Phi = 90^o$ em $n = 10$ partes. Ao particionar 360^o em 10 partes tem-se o seguinte conjunto de classes $C = \{0, 36, 72, 108, 144, 180, 216, 252, 288, 324\}$. Para cada $c \in C$, calcula-se a função gaussiana abaixo:

$$f(x) = e^{-}\frac{(\Phi/360 - c)^2}{fator} \tag{5.1}$$

Nos gráficos da Figura 5.5 são mostradas as gaussianas geradas a partir da equação (5.1) utilizando-se $\Phi = 90^o$ e $fator = 0.1$ e 0.02, respectivamente. Um ponto importante que deve ser observado é o fato da curvatura da gaussiana depender do valor da variável $fator$.



(a) (b)

Figura 5.5: Discretização de um ângulo $\Phi \in [0^o, 360^o]$, em 10 partes, pela função gaussina (5.1). (a) $fator = 0.1$ e (b) $fator = 0.02$; em ambas $\Phi = 90^o$.

A representação discreta da gaussiana para um determinado ângulo Φ consiste de um conjunto de n pares de pontos (x, y), com $y \in [0, 1]$ e $x \in C$. No ponto de ápice do gráfico, ou seja, o ponto no qual o valor de y apresenta o maior valor, o valor de x deve ser igual a Φ.

Em nossa proposta de discretização do ângulo de direção do robô atacante, foi utilizado um conjunto C com $n = 60$ elementos equidistantes de 6^o, i.e., $C = \{0, 6, 12, 18, ..., 354\}$. Desta forma, cada exemplo apresentado para rede consiste das mesmas 10 entradas e de 60 saídas cada qual representado um ponto da curva gaussiana para o ângulo da direção do robô atacante.

Após testar várias arquiteturas de redes MLP, obteve-se os melhores resultados com uma rede 10x60x60. Com esta rede obteve-se um erro de treinamento MSE=0.0050 e erro de teste MSE 0.07. Como já feito anteriormente, a rede treinada foi convertida para um arquivo em linguagem C utilizando a ferramenta SNNS2C. A rede então passou a controlar um robô artilheiro, porém, foram necessárias alterações no código de modo a incluir uma função que fizesse a conversão contrária, ou seja, convertesse novamente os pontos da função Gaussiana em um ângulo que seria então utilizado na estratégia do robô.

Embora tenha-se conseguido ótimos resultados de aprendizado com esta proposta de rede MLP, os movimentos do robô não ficaram dentro de nossas expectativas. O robô controlado pela rede não apresentou movimentos adequados, ou seja, desviar de obstáculos, evitar gols do time adversário e marcar gols.

5.2.4 Previsão da próxima posição (x, y)

Após os testes com as arquiteturas e base de dados anteriores, percebeu-se a necessidade de extrair os atributos relevantes da base de dados colhida. A partir destes dados poderia-se tentar explicar o motivo da rede ter falhado no aprendizado e ainda poderia mostrar quais variáveis selecionar para os próximos treinamentos. Por este motivo, a base foi colhida com todos os dados fornecidos pelo sistema, ou seja, as coordenadas da bola e de todos os robôs e os ângulos de rotação de todos os robôs.

Para saber quais dos atributos colhidos seriam os mais influentes na predição da próxima posição do robô atacante, foi utilizada a ferramenta de extração de conhecimento WEKA.

A Tabela 5.1 contém as informações obtidas por experimentos realizados com o módulo de seleção de atributos relevantes do WEKA com os seguintes parâmetros: *Attribute Evaluetor = CfsSubsetEval com locallyPredictive=false e missingSeperate=false; Search Method=BestFirst com direction=forward, searchTermination=5 e startSet={}* . As células contém valores entre [1,10] de acordo com seu grau de relevância. Nos casos de relevância zero, o símbolo "-" é utilizado.

Tabela 5.1: Atributos relevantes para a predição das variáveis (Φ, x, y) do robô atacante sendo (Φ, x, y) valores reais ou discretizados.

		Reais			**Discretos**		
Variáveis		Φ	x	y	Φ	x	y
Bola	x	-	10	-	-	10	10
	y	-	-	10	-	9	10
Goleiro da Casa	x	1	-	10	-	6	5
	y	7	-	10	2	2	10
	Φ	-	-	-	-	3	-
Parado1	x	2	-	-	-	10	2
	y	2	-	-	-	1	-
	Φ	-	10	10	-	-	5
Parado2	x	-	-	10	-	-	-
	y	-	-	-	-	1	-
	Φ	-	10	-	-	-	-
Atacante Guiado	x						
	y						
	Φ						
Parado3	x	5	-	-	-	-	2
	y	-	-	-	7	5	10
	Φ	-	-	10	4	-	9

		Reais			**Discretos**		
Variáveis		Φ	x	y	Φ	x	y
Goleiro Oponente	x	1	-	-	-	4	9
	y	2	-	10	2	10	2
	Φ	-	-	-	6	5	4
Parado4	x	8	-	-	-	-	-
	y	3	-	-	-	6	-
	Φ	-	-	-	10	2	-
Parado5	x	5	10	-	1	7	-
	y	-	10	-	5	-	1
	Φ	-	-	-	-	10	-
Parado6	x	8	-	-	-	8	1
	y	9	-	-	-	-	-
	Φ	-	-	-	-	10	9
Atacante Oponente	x	-	10	-	1	10	2
	y	2	-	10	8	-	10
	Φ	-	-	-	-	-	-

Para prever a posição de um atacante considerando valores reais para (Φ, x, y), observa-se na tabela que para prever o valor x (Coluna Reais / x), são importantes: a coordenada x da bola, o x do atacante oponente.

Os experimentos foram realizados tanto considerando os valores reais de (Φ, x, y) quanto para a base contendo estes valores discretizados. Φ foi discreditazado em 60 possíveis valores. (x, y) foi discretizado em uma grade na qual cada célula possui 3x3 polegadas, criando assim uma grade de 34x28 células sobre o campo.

Cada coluna intitulada com (Φ, x, y) representa a variável que está sendo predita e as células abaixo delas contém um número de 0 a 10 que indica quão importante o valor da variável na linha da célula é para a predição do valor da coluna em questão. Como exemplo, ao observar a coluna x de valores **Reais** da Tabela 5.1, podemos identificar que as seguintes variáveis foram classificadas com 100% de relevância para a predição da coordenada x do atacante guiado: bola x, Parado1 Φ, Parado2 Φ, Parado5 (x, y), Atacante Oponente x. Os valores selecionados consideram atributos de robôs que ficaram parados durante o jogo e isto pode indicar que utilizar valores reais pode não ser uma boa idéia.

Uma descoberta relevante está na tentativa de prever a direção Φ a ser seguida pelo atacante. O atributo atribuído como mais relevante, assim como no exemplo acima, pertence a robôs que não se mexem durante o jogo, isso pode ser observado tanto para os dados reais (Parado6 y) quanto discretizado (Parado4 Φ). Isto implica que a previsão da direção do robô atacante, como realizado na seção anterior, não é possível utilizando somente os dados analisados.

Neste ponto, a única opção que nos resta é a previsão da próxima coordenada (x, y) do robô atacante utilizando os valores discretizados. Pode-se observar na Tabela 5.1 que os valores altos estão concentrados nas coordenadas discretizadas (x, y) dos agentes que se moveram durante o jogo.

A partir da avaliação destes dados, concluíu-se que a direção Φ não é uma informação relevante e que as posições discretizadas (x, y) da bola e dos robôs que se movem durante o jogo são essenciais para estimar a próxima posição do robô atacante. Assim, uma nova base de dados foi coletada seguindo estas informações.

A nova base de dados colhida contém: as coordenadas cartesianas discretizadas da bola, do goleiro do time aliado e dos 2 robôs do time adversário. Teríamos, portanto, 2 saídas para a rede, as quais corresponderiam as coordenadas cartesianas discretizadas para qual o robô artilheiro deveria ir.

A arquitetura que melhor obteve resultados foi uma rede MLP 8x8X2, com erro de treinamento MSE = 0.0030 e erro de teste MSE = 0.05. Como pode ser visto, o índice de erros é bem menor do que dos experimentos anteriores e, portanto, valida os testes

realizados com o WEKA. Contudo, os testes da rede MLP controlando o robô atacante se mostraram ineficazes. Isto foi uma grande surpresa pois não era o resultado esperado.

5.2.5 Considerações

Devido a natureza dinâmica do futebol de robôs, no qual tanto robôs (obstáculos) quanto a bola (meta) estão em constante movimento, o desafio de criar um controlador autônomo se torna ainda maior. Infelizmente, com a base de dados criada, os resultados obtidos não foram os esperados. Acredita-se que um sistema para coleta de dados precisa ser desenvolvido para gerar dados que representem o comportamento desejado para um robô. Isto, certamente faz parte de trabalhos futuros.

5.3 Planejamento via Campos Potenciais

Nesta seção será apresentado alguns resultados obtidos utilizando-se a arquitetura ACIn sendo o nível Planejamento implementado pela técnica de Campos Potenciais proposta na seção 4.1.2. Primeiramente, serão mostrados os movimentos obtidos com gráficos mostrando a trajetória dos robôs, e em seguida uma tabela comparativa de jogos realizados entre um time com o ACIn e a estratégia padrão que acompanha o Simulador da FIRA.

Na Figura 5.6, pode-se observar os robôs da defesa se deslocando para o ponto de intersecção de suas linha de defesa com a reta r. As linhas de defesa não estão representadas nesta Figura por motivos de clareza, porém as mesmas linhas apresentadas na Figura 4.2 estão sendo consideradas. No comportamento do robô atacante, pode ser notado o direcionamento de sua trajetória para o ponto P, juntamente com o comportamento de desvio de obstáculos.

Na Figura 5.7, o comportamento de busca pelo ponto P é bem mais evidente e também é possível notar a mudança de direção ao atingir o ponto P, isto é, a bola passou a atrair o robô e o este a empurrou diretamente para o gol.

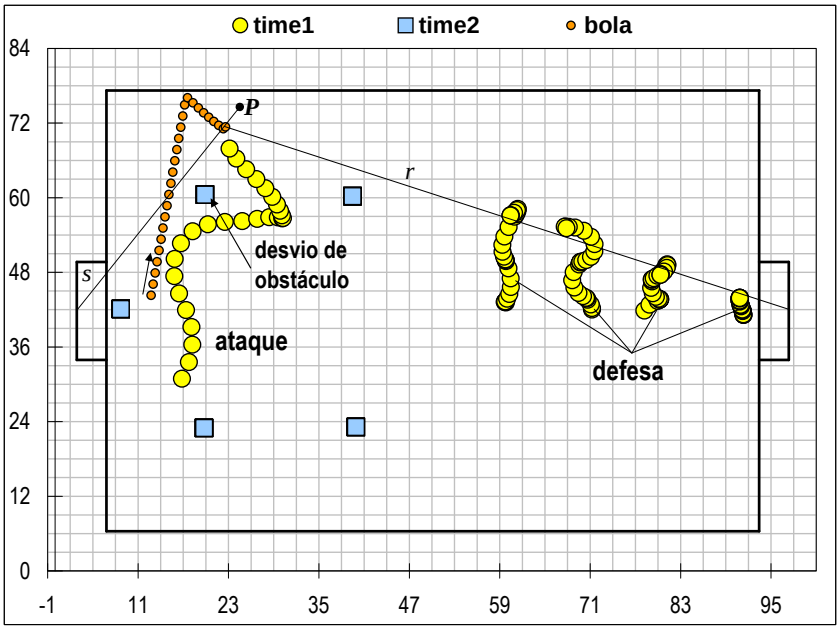


Figura 5.6: ACIn com CP: comportamentos de ataque e defesa

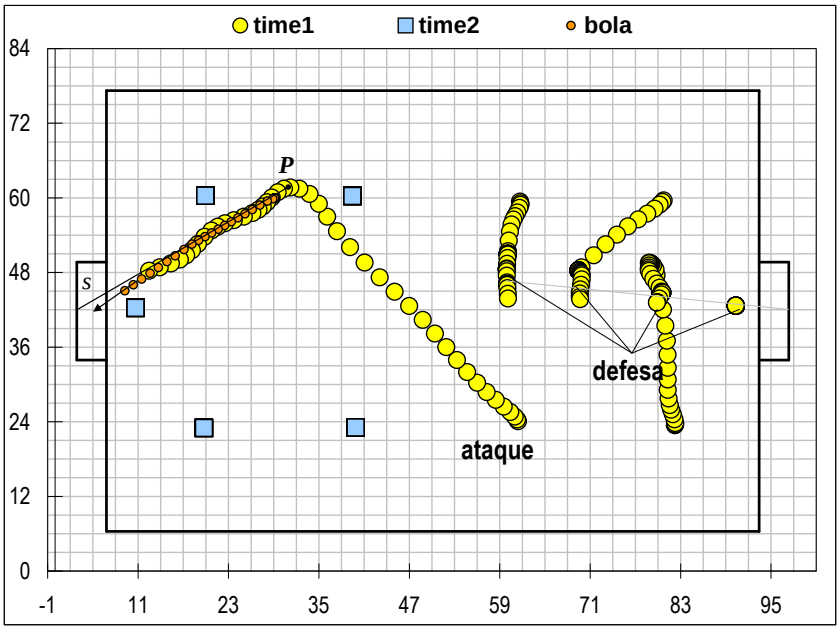


Figura 5.7: ACIn com CP: ataque com chute direcionado para o gol

Na Tabela 5.2 pode-se observar os resultados obtidos em dois minutos de jogo para times com diferentes estratégias. Pode-se observar também que para um time com dois atacantes, um atacante acaba repelindo o outro e assim acabam não realizando suas tarefas adequadamente. Isto provavelmente ocorre devido ao mínimos locais.

Tabela 5.2: Pontuação obtida utilizando estratégias diferentes.

Estratégia	**Pontos**
padrão do simulador X campo potencial com 1 atacante	1 X 3
padrão do simulador X campo potencial com 2 atacante	0 X 1
campo potencial com 1 atacante X campo potencial com 1 atacante	1 X 1
campo potencial com 1 atacante X campo potencial com 2 atacante*	3 X 0

* marcou 3 gols contra.

5.3.1 Considerações

Vários jogos com todos os robôs em movimento foram realizados para testar a arquitetura ACIn com CP. Todavia, também foram observados os comportamentos para um único robô em movimento, pois não é uma tarefa fácil analisar o comportamento de um robô em um ambiente dinâmico.

A partir destes testes, observou-se que:

1. Obteve-se sucesso com as regras utilizadas no nível Estratégia em conjunto com os Campos Potenciais;

2. O fato dos robôs estarem em movimento minimiza o problema de mínimos locais no qual força de atração e força de repulsão se anulam impedindo o movimento. Portanto, pode-se afirmar que o futebol de robôs é um ambiente que favorece o método de campos potenciais para o controle de trajetórias dos robôs jogadores;

3. O fato de ter mais que um atacante pode fazer com que estes acabem se repelindo mutuamente e evitando a bola;

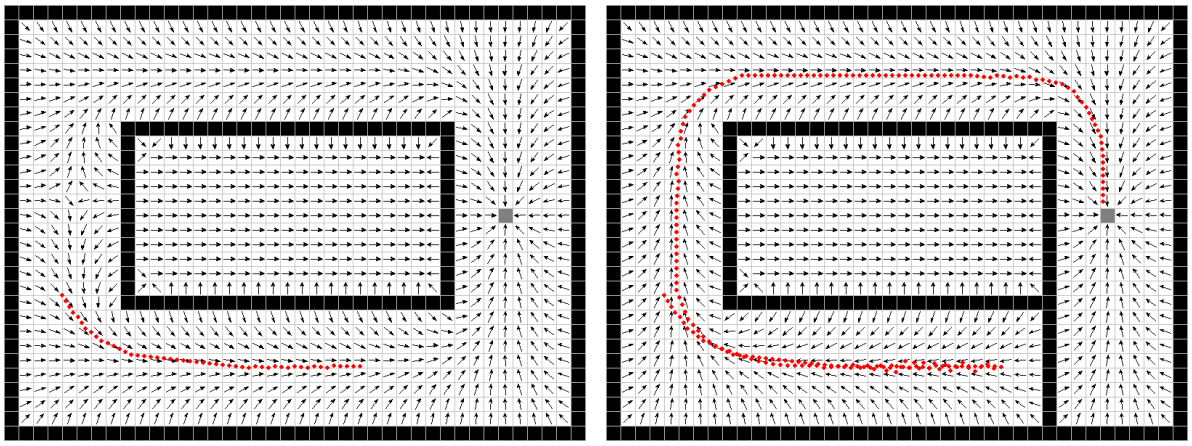
4. Os robôs movimentam-se rapidamente, a uma velocidade máxima de 317cm/s, mesmo assim, o método de campos potenciais mostrou-se bastante eficiente no planejamento de trajetórias.

O maior problema foi encontrar empiricamente os valores para as constantes de repulsão Q, Qb e atração C. Embora tenha-se ajustado estas constantes para uma certa distância, foi observado que ocorrem mais colisões quando existem obstáculos próximos posicionamos atrás do robô, ou seja, como se as forças de repulsão causadas por estes obstáculos empurrassem o robô fazendo-o colidir com obstáculos à sua frente.

5.4 Planejamento via CPH e CPO

O método de cálculo de Campos Potenciais utilizando o Problema de Valor de Contorno, como CPH e CPO, é eficiente para o problema de planejamento de trajetórias. A técnica CPH também mostrou-se eficiente para exploração de ambientes [Prestes, 2003]. No entanto, assim como todos os métodos que utilizam modelos do mundo para sua implementação, o custo computacional destes métodos cresce proporcionalmente com o tamanho do ambiente mapeado.

Dos experimentos realizados com ambas as técnicas, observou-se que para ambientes estáticos, o algoritmo de relaxação pode ser interrompido logo que o processo de interpolação tenha estabilizado. Ao considerar os ambientes dinâmicos, o processo de relaxamento deve ser contínuo para que o sistema seja capaz de replanejar trajetórias em função das mudanças do ambiente. Na Figura 5.8(a) pode ser observada a trajetória seguida por um robô até determinado ponto do ambiente, quando é percebido que o corredor está bloqueado. Na Figura 5.8(b) pode ser visto o novo caminho que foi seguido para chegar até a meta.



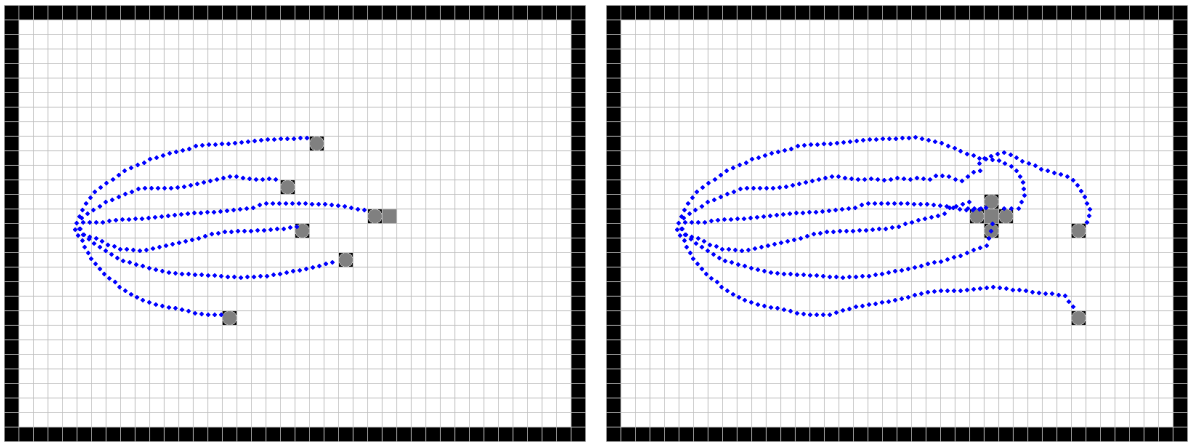
(a) (b)

Figura 5.8: Replanejamento de trajetórias com CPH. (a) o robô percorre o menor caminho até perceber que está bloqueado. (b) um caminho alternativo é gerado.

Contudo, vale salientar, que este método se diferencia dos demais métodos de geração de campos potenciais por ser um método completo, ou seja, uma trajetória livre até a meta sempre será gerada caso ela exista.

Uma desvantagem observada pode ser visualizada na Figura 5.9 na qual tem-se vários robôs sobre um mesmo mapa onde todos se dirigem para o mesmo ponto. Este comportamento competitivo não é adequado para uma equipe de robôs, pois no caso de futebol

de robôs, os robôs acabam competindo pela bola.



(a) (b)

Figura 5.9: Vários robobôs competindo pela meta em Campos Potenciais Harmônicos. (a) robôs repelindo uns aos outros (b) Meta boloqueada impedindo que dois companheiros se aproximem.

Na Figura 5.10 podem ser vistos algumas trajetórias seguidas utilizando a proposta de controle com CPO.

Na implementação do obstáculo virtual realizada, foi considerado que este obstáculo sempre ocuparia o vizinho direito da bola. Esta implementação se ajusta perfeitamente à situação da bola alinhada horizontalmente com gol, como pode ser visto na Figura 5.10(a). Contudo, isto pode prejudicar o direcionamento do chute, principalmente quando a bola estiver próxima as laterais do campo. Esta situação é mostrada na Figura 5.10(c) e (d), as quais mostram uma situação de chute ao gol e chute com desvio, respectivamente.

Contudo, o mesmo comportamento competitivo do CPH pode ser observado no CPO mostrado na Figura 5.11. Este comportamento elimina o comportamento do chute direcionado apresentado anteriormente. O método CPO possui um tipo de comportamento quando apenas um robô está na grade do ambiente, mas estes comportamentos são afetados quando na presença de mais de um robô sobre a mesma grade.

5.4.1 Considerações

As técnicas CPH e CPO, baseadas em PVC, embora sejam eficientes na condução de um robô para a meta, elas não apresentam características adequadas para uma equipe de robôs. Por este motivo, algumas modificações foram propostas e alguns resultados podem ser vistos a seguir.

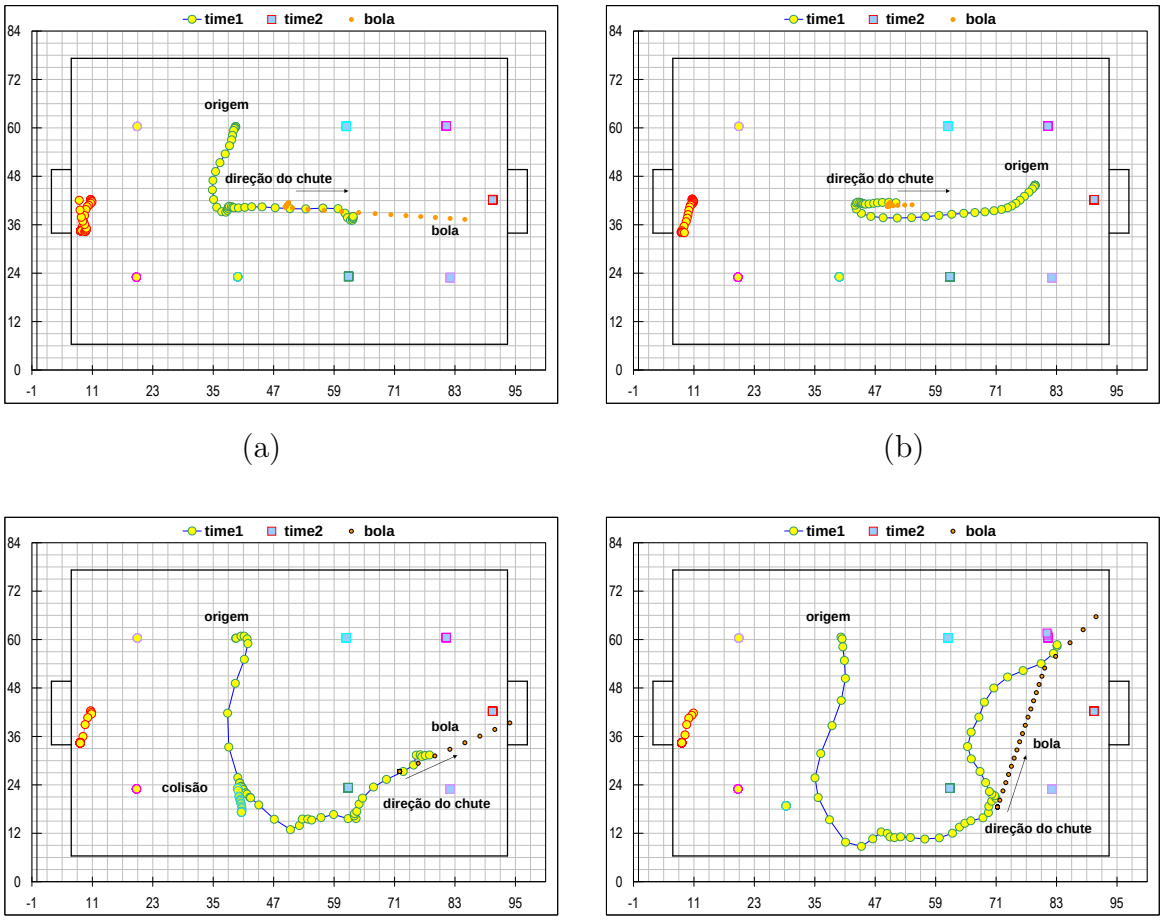


Figura 5.10: Comportamento de um único robô em um Campo Potencial Orientado. (a) chute direcionado ao gol. (b) atacante evitou chutar contra o próprio gol devido ao obstáculo virtual. (c) chute direcionado, mas houve uma colisão. (d) chute mal direcionado.

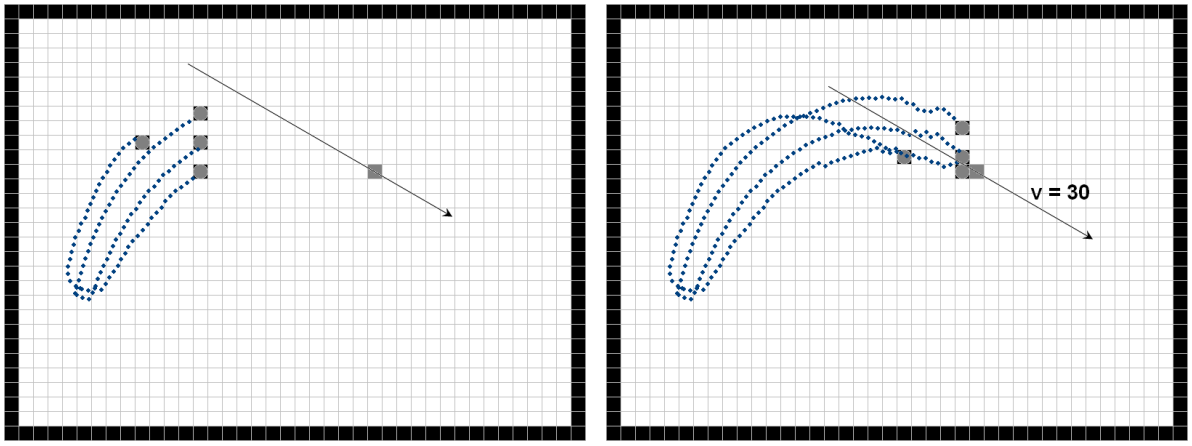


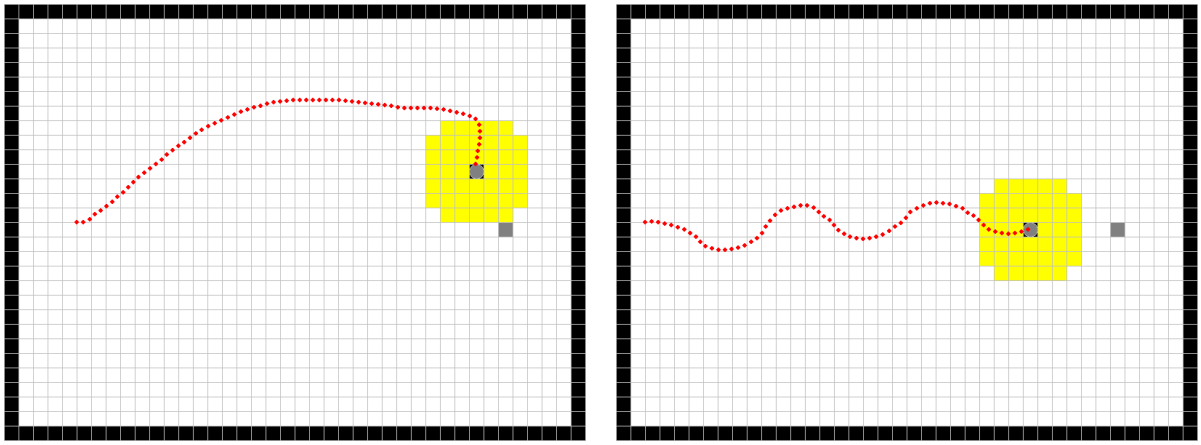
Figura 5.11: Vários robôs competindo pela meta em Campos Potenciais Orientados

5.5 Planejamento via CPLO

Os experimentos realizados com o método CPLO para o nível Planejamento são apresentados, primeiramente mostrando o comportamento de um único robô, em segundo lugar, para três robôs sobre um mesma grade correspondente a um determinado ambiente, e finalmente integrando um SBR, no nível Estratégia, aplicado a futebol de robôs.

5.5.1 Experimentos com um robô

Cada célula pertencente a vizinhança de um robô k será atualizada com seus respectivos parâmetros $\mathbf{v}_k$ e ϵ_k. Deve ser salientado, que ambos os parâmetros podem ser fixos ou variáveis, modificando seus valores durante a iteração do robô k com o ambiente. Isto implica que um robô k pode mudar seu comportamento durante a execução de uma tarefa. Na Figura 5.12(a) e (b), podem ser vistos exemplos de trajetórias seguindo parâmetros fixos e variáveis, respectivamente.



(a) (b)

Figura 5.12: Campo Potencial Localmente Orientado. (a) vetor fixo $\mathbf{v} = (0, 1)$; (b) vetor senoidal.

Observe que, na Figura 5.12, a área em destaque ao redor do robô correspondente às células da vizinhança do robô, ou seja, as células que estão sendo atualizadas utilizando o vetor comportamental $\mathbf{v}_k$. Em ambos os casos, (a) e (b), a vizinhança corresponde a uma circunferência com 3 células de raio e taxa de influência $\epsilon_k = 1$.

Como pode ser notado na Figura 5.12(a), a trajetória descrita pelo robô a não descreve o menor caminho até a meta. O robô descreve uma trajetória mais próxima de seu vetor comportamental, que neste caso é $\mathbf{v}_a = (0, 1)$.

Por outro lado, $\mathbf{v_k}$ pode ser variável, fazendo o robô seguir o comportamento de

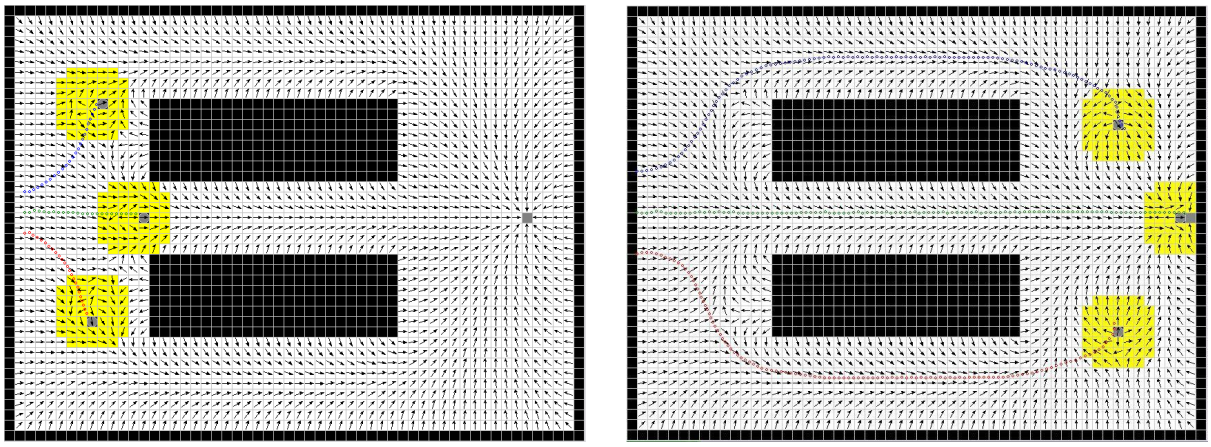
alguma função. No exemplo visto na Figura 5.12(b), o robô segue uma trajetória senoidal em direção a meta, pois $\mathbf{v}_k = (|cos(w * t)|, sin(w * t))$, no qual w é um ângulo de rotação fixo e t denota a iteração. Isto acontece pois, a direção do vetor é atualizada a cada fração de tempo.

A modificação proposta teve sucesso quando aplicada para um único robô na grade. Nos próximos experimentos serão relatados os resultados com múltiplos robôs.

5.5.2 *Experimentos com Múltiplos Robôs*

Para mostrar a eficiência da abordagem proposta, serão apresentados alguns resultados obtidos quando três robôs atuam sobre uma mesma grade do ambiente. Cada robô k tem seu próprio vetor $\mathbf{v}_k$ e parâmetro ϵ_k.

Pode ser notado na Figura 5.13 que o campo potencial das células que não pertencem as vizinhanças dos robôs continuam descrevendo o menor caminho até a meta. Contudo, cada robô é guiado em uma trajetória distinta até a meta, seguindo $\mathbf{v}_1 = (0, 1)$, $\mathbf{v}_2 = (0, 0)$ e $\mathbf{v}_3 = (0, -1)$, para os robôs $1, 2, 3$ de cima para baixo. Para os três robôs, foi adotado o mesmo valor $\epsilon_k = 1.0 | k \in [1, 2, 3]$.



(a) (b)

Figura 5.13: Robôs com vetores diferentes atuando sobre uma mesma grade. De cima para baixo, robôs com $\mathbf{v}_1 = (0, 1)$, $\mathbf{v}_2 = (0, 0)$ e $\mathbf{v}_3 = (0, -1)$ respectivamente, sendo $\epsilon_k = 1.0$ para todos os robôs.

5.5.3 *Estratégia para Futebol de Robôs*

Nesta seção são apresentados os resultados obtidos ao aplicar a arquitetura ACIn com o método CPLO no ambiente de futebol de robôs. A cada um dos 5 robôs do time foi atribuída uma missão, sendo, goleiro, lateral direita, lateral esquerda, defesa na linha 30

e atacante. Cada missão é descrita no nível Estratégia por um SBR, como descrito na seção 4.1.4. A seguir serão mostrados o trajeto percorrido para cada uma destas missões.

Na Figura 5.14 pode ser visto o comportamento de defesa e do goleiro, ambos tentam ficar alinhados horizontalmente com a bola para impedir que esta passe por eles.

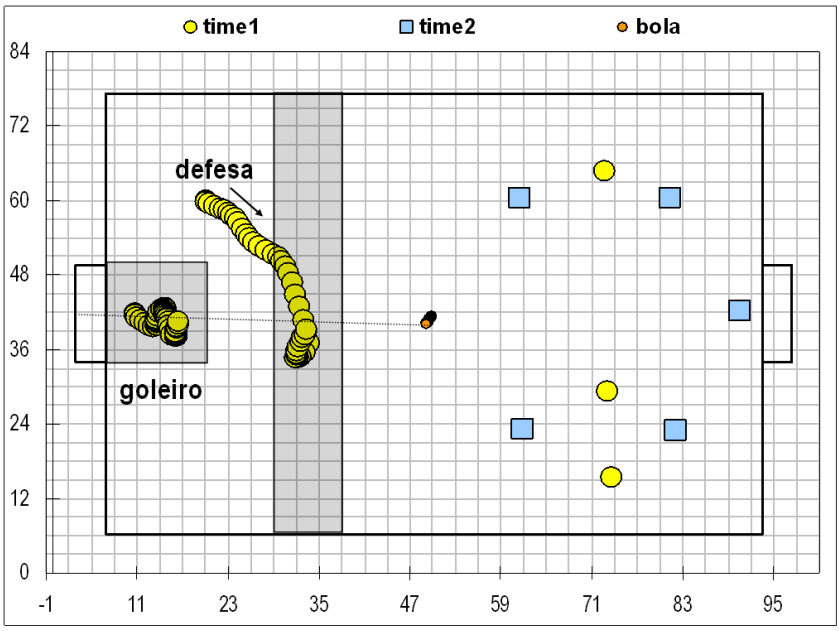


Figura 5.14: ACIn com CPLO: comportamentos de defesa e do goleiro

Um comportamento que pode ser bastante útil para auxiliar o atacante, é o comportamento de acompanhamento lateral. Este comportamento faz com que o robô acompanhe a parede e fique alinhado verticalmente com a bola. Na Figura 5.15, podem ser vistos dois robôs utilizando este comportamento, sendo um em cada lateral do campo.

O comportamento do atacante pode ser visto na Figura 5.16, no qual o robô segue uma trajetória para alinhar-se com a bola. Uma vez alinhado, o direcionamento do robô muda para que este possa realizar um chute direto ao gol.

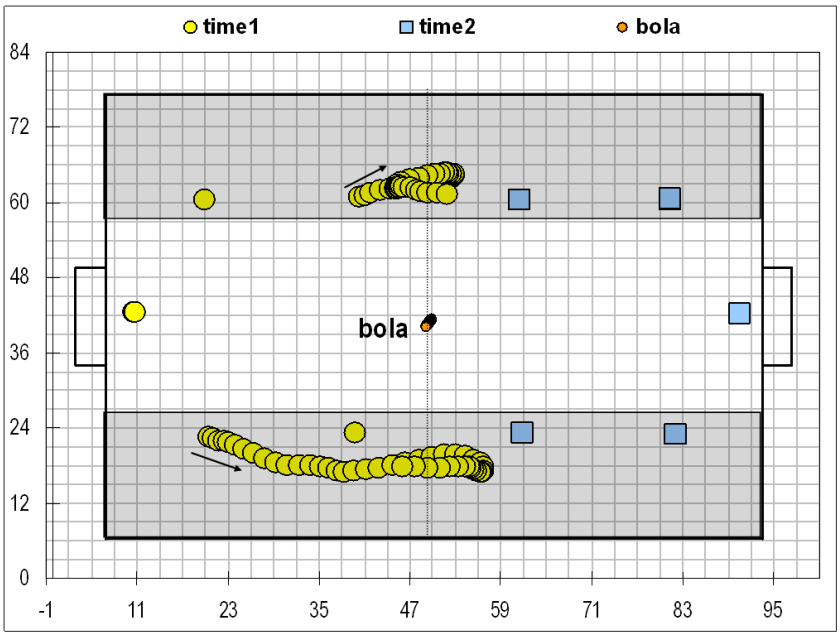


Figura 5.15: ACIn com CPLO: comportamento de acompanhamento lateral

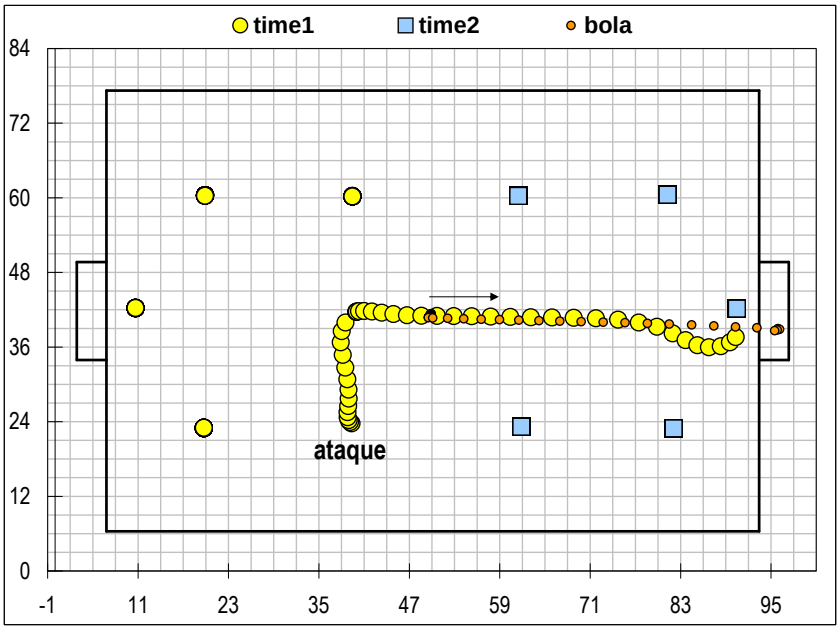


Figura 5.16: ACIn com CPLO: ataque com chute direcionado para o gol

Um total de 50 jogos foram realizados para observar o desempenho desta proposta na arquitetura ACIn. Como pode ser visto na Tabela 5.3, o time utilizando esta proposta superou as propostas anteriores em jogos com dois minutos de duração.

Estratégia	Pontos	Contras
CPLO x FIRA	5 x 1	0 x 2
CPLO x CP	3 x 1	0 x 0

Tabela 5.3: Resultados dos jogos entre CPLO, Campos Potenciais (CP) e Padrão do Simulador (FIRA)

5.5.4 Considerações

O método CPLO permite que as trajetórias possam variar de acordo com a funcionalidade de cada robô. O método ainda permite que as variáveis de ajuste de direção sejam modificadas durante a execução da tarefa. Estas características permitiram que fosse construído um SBR que definisse as diferentes funcionalidades para cada robô.

Como pode ser visto nos experimentos, vários podem ser colocados sobre uma grade do ambiente e apresentar comportamentos diferentes seguindo os seus vetores de direção. Contudo, o método é bastante sensível as condições de contorno, ou seja, algum obstáculo ou a meta próximo aos robôs podem interferir no comportamento determinado pelo vetor direcional.

6

Conclusão

6.1 Principais Considerações

Controle e planejamento de trajetórias de robôs móveis em ambientes dinâmicos tem sido um tema de pesquisa bastante explorado. Existem várias técnicas eficientes para controlar um único robô em um ambiente estático. No entanto, para ambientes dinâmicos existe a necessidade de um controle de trajetórias que seja capaz de replanejar o caminho em caso de situações imprevistas. A complexidade do problema cresce ainda mais quando são considerados mais de um robô em um mesmo ambiente, pois, além do planejamento de trajetórias existe a necessidade de coordenador dos integrantes da equipe de robôs para que estes não atrapalhem uns aos outros na realização das tarefas.

Nesta tese foi proposta a arquitetura ACIn, Arquitetura de Controle Inteligente, para o controle de múltiplos robôs em ambientes dinâmicos. A arquitetura proposta possui um organização hierárquica, com componentes reativos e componentes deliberativos, o que permite respostas rápidas às mudanças do ambiente e coordenação do comportamento de cada robô através de um Sistema Baseado em Regras. Por possuir ambas características, a arquitetura ACIn é classificada como uma arquitetura de controle híbrida.

O ambiente multi-robôs escolhido para validar a arquitetura ACIn foi o futebol de robôs, mas nada impede que esta arquitetura seja utilizada em outras aplicações. O futebol de robôs é um domínio dinâmico, imprevisível, no qual podem ser avaliadas combinações entre as abordagens reativas e deliberativas. Um grande desafio deste ambiente está no fato deste possuir não só obstáculos móveis, mas também uma meta móvel (a bola).

A arquitetura ACIn é composta de quatro níveis: missão, estratégia, planejamento e controle de velocidade. O nível missão foi considerado com funcionalidades fixas. Para

o nível estratégia foi proposto um Sistema Baseado em Regras capaz de modificar as trajetórias dependendo da funcionalidade de cada integrante da equipe de robôs. Para o nível de planejamento, foram investigadas algumas técnicas consideradas inteligentes, tais como, Redes Neurais Artificiais, Campos Potenciais e Campos Potenciais baseados em Problema do Valor de Contorno.

As Redes Neurais Artificiais do tipo Perceptron Multicamadas (MLP) foram investigadas como uma forma de unificar os níveis Planejamento e Estratégia em uma única estrutura. Esta proposta foi motivada pelo fato das redes MLP serem consideradas aproximadores universais, isto é, por possuírem a capacidade de aproximar por uma função não linear qualquer conjunto de dados. Todavia, com os dados coletados não foram obtidos os resultados desejados, mas espera-se que um sistema desenvolvido especificamente para coleta de dados possa representar o comportamento desejado para o time de robôs.

O método de Campos Potenciais considera forças atuando sobre um robô para direcionar o seu movimento. Por ser um método de baixo custo computacional, esta técnica tem sido muito utilizada para planejamento de trajetórias de robôs móveis, que necessitam navegar por ambientes desconhecidos e/ou dinâmicos evitando colisões com obstáculos. Este método foi utilizado como base para o nível Planejamento e um SBR foi proposto para tornar a arquitetura ACIn apta a controlar vários robôs. Esta proposta da ACIn foi utilizada para controlar um time de futebol de robôs, obtendo-se bons resultados: O desempenho em gols é melhor do que o time com a estratégia do Simulador; o problema de mínimos locais é minimizado, pois o ambiente de futebol possui robôs e bola em constante movimento. Contudo, em experimentos com mais de um atacante, observou-se que estes competem pela bola e acabam se atrapalhando. Entretanto, o maior problema foi encontrar empiricamente os valores para as constantes de repulsão Q, Qb e atração C.

Os métodos de cálculo de campos potenciais através do Problema de Valor de Contorno (PVC) tornaram-se alvo de nossa pesquisa, pois garantem encontrar um caminho livre entre dois pontos caso ele exista, além de ser capaz de lidar com a presença de obstáculos não conhecidos a priori. Duas técnicas de construção de campos potenciais através do PVC, exploradas no presente trabalho, foram: Campos Potenciais Harmônicos (CPH) e Campos Potenciais Orientados (CPO). Estas duas técnicas foram adaptadas para atuar no ambiente de futebol de robôs. Para que isto fosse possível, foram propostas algumas regras para compor o nível Estratégia, além da técnica *Paredes Virtuais*. As *Paredes Virtuais* mostraram-se bastante úteis para evitar que robôs chutassem a bola contra o próprio gol, contudo não foram eficientes para delimitar o espaço de atuação de cada robô. Isto ocorre pois não existe um caminho até a meta em áreas isoladas por por paredes virtuais. Observou-se também que os métodos CPH e CPO são eficientes para

um único robô, mas quando há mais de um sobre uma mesma grade do ambiente, surge o comportamento de competição entre os robôs.

Em virtude das vantagens dos métodos baseados em PVC, uma técnica denominada Campos Potenciais Localmente Orientados (CPLO) foi proposta para resolver o problema de competição entre os robôs, considerando-se uma única grade do ambiente. Neste método cada robô tem sua trajetória influenciada por um vetor direcional próprio. Assim, cada robô pode seguir por caminhos distintos, mesmo que partam do mesmo ponto de origem. Testes com CPLO foram realizados para avaliar o comportamento de um robô e de vários robôs sobre a mesma grade. O comportamento observado foi diferente tanto do CPH quanto do CPO, pois o comportamento de competição pôde ser diminuído ou até eliminado. Os resultados satisfatórios motivaram nossa proposta e implementação de um SBR, para o nível de Estratégia, necessário à arquitetura ACIn com CPLO no nível Planejamento. Testes da arquitetura ACIn com CPLO foram realizados no ambiente de futebol de robôs e os resultados foram comparados com os resultados das propostas anteriores da arquitetura ACIn. O resultado obtido foi bastante satisfatório, pois o método proposto atingiu um saldo de gols maior que os demais.

O problema de controle de múltiplos robôs é bastante amplo e vários métodos e arquiteturas têm sido propostos para este tipo de problema. Nesta tese, levantou-se alguns dos problemas envolvidos no controle de múltiplos robôs, sendo que alguns destes foram resolvidos utilizando-se a arquitetura ACIn e para alguns outros são apontadas sugestões para trabalhos futuros.

6.2 Principais Contribuições

As principais contribuições obtidas como resultado desta tese são:

- Adaptação de métodos anteriormente utilizados para um único robô para um ambiente de múltiplos robôs, particularmente para futebol de robôs. Uma rede neural do tipo MLP foi utilizada como planejamento de trajetórias. Foi utilizado a técnica de Campos Potenciais tradicional para gerar dados para treinamento deste modelo. Foram também adaptados para a aplicação de futebol de robôs os seguintes métodos: o método de Campos Potenciais tradicional combinado com um SBR, o método CPH com um SBR, o método CPO combinado com um SBR.
- Desenvolvimento de um método de controle de velocidade, para transformar ângulo de direção em velocidade para cada uma das rodas, em polegadas/s.
- Desenvolvimento de uma arquitetura híbrida, ACIn, para controlar múltiplos robôs

em ambiente dinâmico. Esta arquitetura foi testada no ambiente dinâmico de futebol de robôs e mostrou resultados satisfatórios.

- Novo método de controle de trajetórias de múltiplos robôs que é um aperfeiçoamento da técnica de Campos Potenciais Orientados. Este método permite que as trajetórias sejam variáveis a medida que imprevistos ocorrem numa dada direção e mesmo com esta liberdade de direção, continuem desviando de obstáculos e perseguindo a meta.

Os resultados obtidos nesta teste geraram publicações em congresso nacional[Faria et al., 2004b] e dois artigos em congressos internacionais [Faria et al., 2004a] e [Faria et al., 2006]. Um artigo a ser submetido à revista internacional está sendo elaborado.

6.3 Trabalhos Futuros

Nesta tese, foram investigadas técnicas os campos potenciais baseados em PVC que não possuem o problema de mínimos locais. Uma outra solução seria utilizar os métodos de campos potenciais propostos por Borenstein & Koren [1989, 1991] que visam diminuir o efeito deste problema.

Um dos maiores problemas foi encontrar valores para as constantes de repulsão Q, Qb e atração C. Propõe-se a utilização de um algoritmo de aprendizado como o Q-Learning [Watkins, 1989] ou Algoritmo Genético para ajustar os valores destas variáveis.

O aprendizado via RNA depende de uma base de dados consistente. Assim, um sistema para coleta de dados, que armazene as características da trajetória desejada para cada situação, pode colaborar de forma significativa para a construção de um sistema totalmente controlado via RNA.

As regras do nível de Estratégia são fixas. Como sugestão para implementação deste nível pode ser utilizado um Sistema Classificador [Holland, 1975, 1986; Wilson, 1994, 1995], que possa evoluir as regras de funcionalidade durante a interação dos robôs com o ambiente. Esta técnica está sendo investigada por nós, mas a forma de avaliação da aptidão de cada regra ainda está em estudo.

Todos os experimentos realizados nesta tese foram testados em um ambiente de simulação. Pretende-se testar as técnicas propostas em um time de futebol de robôs que está sendo desenvolvido no ICMC.

Referências Bibliográficas

Albus, J., Lumia, R., Fiala, J., & Wavering, A. (1989). *NASREM: The NASA/NBS Standard Reference Model for Telerobot Control System Architecture*. Proceedings of 20th International Symposium on Industrial Robots, Tokio, Japan.

Antsakalis, P. J. (1992). Neural networks in control systems. *IEEE Control Systems Magazine*, (pp. 8–10).

Arbib, M. & House, D. (1987). Depth and detours: An essay on visually guided behavior. In M. Arbib & A. Hanson (Eds.), *Vision, Brain, and Cooperative Computation.* (pp. 139–163). Cambridge: MIT Press.

Arbib, M. A. (1972). *The Metaphorical Brain: An Introduction to Cybernetics as Artificial Intelligence and Brain Theory*. Wiley.

Arkin, R. C. (1987). *Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-made Environments*. PhD thesis, University of Massachusetts, Departament of Computer and Information Science.

Arkin, R. C. (1989). Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 4(8), 92–112.

Arkin, R. C. (1999). *Behavior-Based Robots*. London, England: MIT Press.

Balch, T. & Arkin, R. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6), 926–939.

Borenstein, J. & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), 1179–1187.

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

Braga, A. P., Ludermir, T. B., & Carvalho, A. C. P. L. F. (2000). *Redes Neurais Artificiais - Teoria e Aplicações*. LTC.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), 14–23.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence Journal*, 47, 139–159.

Browning, B., Bruce, J., Bowling, M., & Veloso, M. (2005). Stp: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering*, 219, 33–52.

Burden, R. L., Faires, J. D., & Reynolds, A. C. (1978). *Numerial Analysis*. Prindle, Weber and Schimidt.

Carbonell, J. G., Knoblock, C. A., & Minton, S. (1989). *PRODIGY: An Integrated Architecture for Planning and Learning*. Technical Report CMU-CS-89-189, CMU.

Carver, N. & Lesser, V. (1992). *The Evolution of Blackboard Control Architecture*. Techinical report, CMPSCI.

Connolly, C. I. & Grupen, R. A. (1993). On the application of harmonic functions to robotics. *Journal of Robotic Systems*, 10, 931–946.

Costa, A. H. R. & Pegoraro, R. (2000). Construindo robôs autônomos para partidas de futebol: O time Guaraná. *SBA Controle e Automação*, 11(3), 141–149.

Courant, R. & Hilbert, D. (1962). *Methods of mathematical physics*, volume 2. New York: John Wiley and Sons.

Cybenko, G. (1988). *Continous Valued Neural Network With Two Hidden Layers are Sufficient*. Technical report, Department of Computer Science - Tufts University.

Cybenko, G. (1989). Approximation by superpositions of a sigmoid function. *Mathematics of Control - Signal and Systems*, 2, 303–314.

Cybenko, G. (1996). Neural networks in computational science and engineering. *IEEE Computational Science & Engineering*, (pp. 36–43).

Dias, M. B., Zlot, R., Zinck, M., Gonzalez, J. P., & Stentz, A. (2004). A versatile implementation of the traderbots approach for multirobot coordination. In *the 8th International Conference on Intelligent Autonomous Systems (IAS-8)*.

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

Faria, G., Prestes, E., Idiart, M. A. P., & Romero, R. A. F. (2006). Multi robot system based on boundary value problems. In *International Conference on Intelligent Robots and Systems (IROS)*. (to appear).

Faria, G. & Romero, R. A. F. (2000). Incorporating fuzzy logic to reinforcement learning. In *Proceedings of the 9th IEEE International Conference on Fuzzy Systems*, volume 1 (pp. 847–851).

Faria, G., Romero, R. A. F., Prestes, E., & Idiart, M. A. P. (2004a). Comparing harmonic functions and potential fields in the trajectory control of mobile robots. In *IEEE Conference on Robotics, Automation and Mechatronics*, volume 1 (pp. 762–767). Singapore.

Faria, G., Teizen, L., & Romero, R. A. F. (2004b). Controle de trajetórias utilizando campos potenciais aplicado ao futebol de robôs. In *Anais do II Encontro de Robótica Inteligente* Salvador-BA: SBC.

Gat, E. (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. *AAAI*, (pp. 809–815).

Gomes, M. R. S. & Campos, M. F. M. (2001). Desvio de obstáculos para micro-robôs móveis utilizando campo potencial. In *Anais do V Simpósio Brasileiro de Automação Inteligente (SBAI)* Canela/RS.

Grimson, W. E. L. (1985). Ieee transactions on pattern analysis and machine intelligence. *PAMI–7*, (pp. 17–34).

Haykin, S. (1994). *Neural Networks: A Compreensive Foundation*. IEEE Computer Society Press.

Haykin, S. (2001). *Redes Neurais: Princípios e prática*. Bookman, 2º edition.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press. Republished by the MIT press, 1992.

Holland, J. H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Mitchell, Michalski, & Carbonell (Eds.), *Machine Learning, an Artificial Intelligence Approach.*, volume 2 chapter 20, (pp. 593–623). Morgan Kaufmann.

Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H., & Osawa, H. (1997a). Robocup: A challenge problem for ai. *AI Magazine*, 1(18), 73–85.

Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., & Asada, M. (1997b). The robocup synthetic agent challenge,97. In *International Joint Conference on Artificial Intelligence (IJCAI97)*.

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2), 243–302.

Kovacs, Z. L. (2002). *Redes Neurais Artificiais: Fundamentos e Aplicações*. Livraria da Física, 3.ed. edition.

Kreider, D. L., Ostberg, D. R., Kwller, R. G., & Perkins, F. W. (1966). *An Introduction to Linear Analysis*. Reading, Massachusetts, USA: Addison-Wesley Publishing Co.

Krogh, B. H. (1984). A generalized potential field approach to obstacle avoidance. In *International Robotics Research Conference*. Bethlehem, Pennsylvania.

Krogh, B. H. & Thorpe, C. E. (1986). Integrated path planning and dynamic steering control. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation* (pp. 1664–1669). San Francisco, California.

Laird, J., Newell, A., & Rosenbloom, P. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1), 1–64.

Lenser, S., Bruce, J., & Veloso, M. (2002). A modular hierarchical behavior-based architecture. In A. Birk, S. Coradeschi, & S. Tadokoro (Eds.), *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Berlin: Springer Verlag.

McClain, J. T. (2004). A behavior-based blackboard architecture for multi-robot control. Master's thesis, The University of Georgia.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideias immanente in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.

Minton, S. (1988). *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Boston, MA: Kluwer Academic Publishers.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newman, W. S. & Hogan, N. (1987). High speed robot control and obstacle avoidance. In *Proceedings of the 1987 IEEE International* (pp. 14–24). Raleigh, North Carolina.

Pacheco, R. N. & Costa, A. H. R. (2002). Navegação de robôs móveis utilizando o método de campos potenciais. In M. T. S. Sakude & C. de A. Castro Cesar (Eds.), *Workshop de Computação WORKCOMP'2002* (pp. 125–130). ITA – São José dos Campos/SP: SBC.

Park, K.-H., Kim, Y.-J., & Kim, J.-H. (2001). Modified uni-vector field navigation and modular q-learning for soccer robots. In *Proceedings of the 32nd International Symposium on Robotics - ISR* (pp. 19–21).

Parker, L. (1998). Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 220–240.

Pettersson, L. (1997). *Control System Architectures for Autonomous Agents – A Survey Study.* Technical report, Department of Machine Design, KTH.

Press, W., Teukolsky, S., Vetterling, W., & Flannery, B. (1992). *Numerical Recipes in C: The art of Scientific Computing.* Cambridge University Press, 2nd edition.

Prestes, E. (2003). *Navegação Exploratória Baseada em Problemas de Valores de Contorno.* PhD thesis, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS.

Rosenblatt, F. (1963). *Principles of Neurodynamics.* New York: Spartan.

Rosenblatt, J. K. (1995). DAMN: A distributed architecture for mobile navigation. In *Proc. of the AAAI Spring Symp. on Lessons Learned from Implememted Software Architectures for Physical Agents.* Stanford, CA: AAAI Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. McClelland (Eds.), *Parallel Distributed Processing*, volume I + II: MIT Press.

Russell, S. & Norvig, P. (1995). *Artificial Intelligence – A Modern Approach.* Prentice Hall International.

Shaw, M. & Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline.* Prentice Hall.

Simmons, R. G. (1992a). Concurent planning and executino for autonomous robots. *IEEE Control Systems Magazine*, 1(12), 46–50.

Simmons, R. G. (1992b). Monitoring and error recovery for autonomous walking. In *Proceedings of the 1992 lEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2 (pp. 1407–1412). Raleigh, USA.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tarassenko, L. & Blake, A. (1991). Analogue computation of collision-free paths. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation* (pp. 540–545).

Thorpe, C. (1984). *Path Relaxation: Path Planning for a Mobile Robot*. Technical Report CMU-RI-TR-84-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Trevisan, M., Idiart, M. A. P., Prestes, E., & Engel, P. M. (2006). Exploratory navigation based on dynamic boundary value problems. *Jounal of Intelligent and Robotic Systems*. (to appear).

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1), 1–18. http://prediction-dynamics.com/.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2), 149–175. http://prediction-dynamics.com/.