UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CLAUDIO JOSE BIAZUS

Desenvolvimento de uma Arquitetura Híbrida e Distribuída para Sistemas Multiagentes e sua Aplicação no Futebol de Robôs

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Mauro Roisenberg, Dr. - Orientador

Florianópolis, abril de 2008

Desenvolvimento de uma Arquitetura Híbrida e Distribuída para Sistemas Multiagentes e sua Aplicação no Futebol de Robôs

CLAUDIO JOSE BIAZUS

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração *Sistemas de Conhecimento* e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Mário Antônio Ribeiro Dantas, Dr. - Coordenador

Banca Examinadora

Prof. Mauro Roisenberg, Dr. - Orientador

Prof. Guilherme Bittencourt, Dr.

Prof. Jovelino Falqueto, Dr.

Prof. Olinto José Varela Furtado, Dr.

Profa. Sílvia Modesto Nassar, Dra.

"Futebol é simples. No entanto, é duro jogar futebol simples."
Johan Cruijff

A todos aqueles que tiveram disponibilidade e dedicação
para ler este trabalho, dando suas contribuições e opiniões
e também àquelas pessoas que terão a curiosidade e a
vontade de saber o que está escrito neste trabalho...

Agradecimentos

Registro o meu agradecimento primeiramente a Deus, por Ele sempre me dar forças para continuar, e também:

Ao Programa de Pós-Graduação em Ciência da Computação e à Universidade Federal de Santa Catarina, pela infra-estrutura e organização que viabilizaram o desenvolvimento deste trabalho.

Em especial ao meu orientador, Prof. Dr. Mauro Roisenberg, por ter me escolhido para fazer parte de sua equipe, pelos seus ensinamentos, seu apoio e dedicação, pelo prazer e entusiasmo em ensinar.

Aos professores e membros do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina, em especial, à professora Dra. Sílvia Modesto Nassar, pelo seu amor, carinho e dedicação, com que sempre me atendeu.

Ao professor Dr. Guilherme Bittencourt, por ter cedido os robôs *EyeBot*, tornando possível a realização dos experimentos.

Aos meus pais (José e Terezinha), os responsáveis por eu estar aqui, pela compreensão deles e pelos ensinamentos em me mostrar os caminhos corretos.

Aos meus irmãos (Neori e Maria do Carmo), que sempre me apoiaram, não medindo esforços e compreensão em todos os momentos difíceis pelos quais passamos juntos.

Aos meus colegas de mestrado pela convivência, amizade, discussões e pelos diversos diálogos que foram abordados a respeito dos inúmeros assuntos e temas durante o tempo que passamos juntos.

A todos aqueles que de uma forma ou de outra contribuíram.

Publicações

BIAZUS, C. J. and ROISENBERG, M. The Development of a Hybrid, Distributed Architecture for Multiagent Systems and its Application in Robot Soccer. IEEE World Congress on Computational Intelligence. Hong Kong, 2008.

BIAZUS, C. J. et. al. Artigo de Descrição da Estratégia da Equipe UFSC-Team na liga Small-Size (F-180) da RoboCup. In: Simpósio Brasileiro de Automação Inteligente, Florianópolis. Competição Brasileira de Robótica, 2007.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de siglas

3D – Tridimensional

CCD – *Charge-Coupled Device*

FIRA – *Federation of International Robosoccer Association*

FPS – *Frames per Second - fps*

LCD – *Large Graphics Display*

PSD – *Position Sensitive Detection*

PWM – *Pulse Width Modulation*

RGB – *Red, Green, Blue*

RoboCup – *Robot Soccer World Cup*

Resumo

Dentre as principais dificuldades encontradas para a construção de sistemas multiagentes em que existe a disponibilidade de um sistema de visão global, como é o caso de algumas categorias de futebol de robôs, pode-se destacar: a necessidade de resposta em tempo real para identificação dos objetos em cena, conhecimento do ambiente, distribuição das competências de controle entre os comportamentos reativos a cargo de cada agente e os comportamentos deliberativo e estratégico, a cargo da entidade central. Este trabalho descreve a implementação de uma arquitetura híbrida reativa x deliberativa e distribuída para controle de sistemas multiagentes equipados com sistema de visão global e dotados de sensores e visão local, e sua aplicação em ambientes de futebol de robôs. A arquitetura híbrida proposta é composta pela integração de modelos de arquitetura deliberativa e reativa. Esta arquitetura é distribuída em duas partes. A primeira parte é implementada em uma entidade central e possui os níveis: estratégico e de ação. Por outro lado, a segunda parte da arquitetura é implementada diretamente nos robôs *EyeBot* (embarcada), e possui os níveis: de comportamento e de execução. Assim, este novo modelo de arquitetura proposta distribui as competências de forma que tarefas relacionadas aos níveis estratégicos e de ação, tais como reconhecimento do ambiente, dos agentes que fazem parte da equipe, da equipe adversária e bola foi realizado por um sistema de processamento de imagens em uma entidade central. Por outro lado, informações de natureza reativa, tais como controlar a bola, vagar pelo ambiente, desviar de obstáculos são realizados por um sistema de processamento embarcado.

Abstract

Many difficulties must be faced during the development of multi-agent systems equipped with global vision, as is the case of some robotic soccer leagues. We can emphasize the real time constraints for scene objects recognition, the environment knowledge acquisition, and the distribution and allocation of control competencies between the agents' reactive behavior repertoire and the strategic and deliberative behavior of the central control entity. This work describes the implementation of a distributed and reactive x deliberative hybrid architecture to control multi-agents system with on-board sensors and cameras where a global vision device is present. We also describe its application in the robot soccer environment. The proposed hybrid architecture is compounded by the integration of the reactive and deliberative models and distributed in two parts. The fist one is implemented in a central control entity and possesses the strategic and action level. The second part is implemented in the physical mobile agents, the EyeBots, and possesses the behavior and execution level. Thus, the new architecture model proposed here distributes competencies in such way that strategic and action related tasks, as scene, team players and adversary players recognition and ball location is supplied by the central control entity with the global vision system. On the other hand, reactive actions as ball control, wandering and obstacle avoidance are implemented in the embedded control system.

Capítulo 1

Introdução

1.1 Motivação

O desenvolvimento da robótica móvel se ocupa do estudo e da construção de robôs móveis inteligentes que possam perceber o ambiente, contendo uma alta quantidade de ruídos e, ao mesmo tempo, possam atuar neste ambiente desconhecido de maneira eficiente. Para que sejam atendidos estes e outros quesitos, tais como, tempo de reação adequado e segurança, em robótica móvel, a multidisciplinaridade se fez necessária. A aplicação de recursos mecânicos, elétrico-eletrônicos e de computação é necessária para que possam contribuir no desenvolvimento e melhorias desta área [61].

A inteligência artificial aplicada em sistemas autônomos é uma área que se desenvolveu nos últimos anos [10], [11] e [14]. Tornar robôs móveis inteligentes e que se assemelham aos humanos é um sonho de muitos pesquisadores e está se tornando cada vez mais uma realidade. Gerações futuras de robôs móveis inteligentes estão sendo aguardadas, com melhores sistemas de mobilidade para atuarem em ambientes dinâmicos e com flexibilidade na tomada de decisões. Sendo estes robôs aplicados no processo de reconhecimento e realização de tarefas, para detectar e avaliar objetos que estão inseridos no ambiente e, até mesmo, na reação diante de situações não previstas ou pouco prováveis de acontecer [58] e [68].

Novas tecnologias de hardware surgiram e contribuíram para o con-

siderável progresso dos robôs móveis inteligentes. Dentre estas tecnologias pode-se destacar: desempenho computacional, capacidade de sensores, aplicação de câmeras, sonar, laser, infravermelhos. A autonomia que é oferecida aos robôs móveis inteligentes, no processo de locomoção assim como o aprendizado de novas tarefas, é atribuída ao software [58]. Construir sistemas que sejam eficientes, tem se tornado um grande desafio [2] e [78].

Durante a década de 90, surgiram duas organizações científicas, a RoboCup *(Robot Soccer World Cup)* www.robocup.org [64] e a FIRA *(Federation of International Robosoccer Association)* www.fira.net [27], cujo objetivo é promover e desenvolver internacionalmente pesquisas na área de Inteligência Artificial aplicada à robótica móvel inteligente. Para atingir seus objetivos, tanto a RoboCup quanto a FIRA, escolheram o jogo de futebol como a aplicação primária. Estas entidades passaram a organizar campeonatos mundiais de futebol de robôs uma vez que, para formar uma equipe de robôs móveis inteligentes, é necessário o desenvolvimento de diversas tecnologias, tais como: projeto de robôs móveis inteligentes, controle inteligente, processamento de imagens, colaboração multiagentes, raciocínio e mobilidade em tempo real, aquisição e realização de estratégias de jogo [13], [30] e [71].

Muitas dessas tecnologias têm sido desenvolvidas de maneira fortemente acoplada com o aparato sensório [61]. No caso de haver um dispositivo de visão global, tal como uma câmera, posicionada sobre o campo de jogo, as estratégias de implementação utilizadas remetem ao desenvolvimento de robôs móveis inteligentes com baixo grau de autonomia e que têm no "técnico", um sistema de controle centralizado e de caráter deliberativo, localizado junto ao dispositivo de visão global, a sua principal fonte de estratégia e controle. Este técnico gera comandos que dirigem os robôs móveis inteligentes em suas tarefas. Em configurações que não contam com o sistema de visão global, todo o aparato sensório e "decisório", está nos próprios robôs móveis inteligentes, que contam então com câmeras de visão local, sensores de proximidade e sensores de toque [16] e [17]. Neste caso, as equipes normalmente não têm um "técnico", os robôs tendem a ser totalmente autônomos. Os comportamentos e estratégias normalmente têm caráter fortemente reativo e são implementados diretamente nos robôs móveis inteligentes

e, a cada momento, um comportamento pode emergir sem que haja interferência externa [6], [7], [15] e [75].

Obviamente, quando consideradas de maneira isolada, cada uma dessas estratégias apresenta vantagens e desvantagens no atendimento das restrições de tempo de resposta, capacidade de processamento dos sinais sensóreos e de geração de estratégias [73].

Em um modelo tradicional de arquitetura de controle centralizada para futebol de robôs, todo o processamento das informações é realizado em uma entidade central [62]. Neste modelo, os robôs móveis inteligentes atuam no ambiente de acordo com informações enviadas por este sistema. Esta arquitetura tem como vantagem a facilidade para reconhecimento do ambiente e definição de estratégias, porém, em certos casos, não oferece um poder reativo em tempo real [71].

Já em um modelo de arquitetura local ou embarcada, todo o controle, comportamento e ações do agente encontram-se no sistema, que está concentrado no próprio robô e, portanto, ele é bastante eficiente para desvio de obstáculos e condução da bola, porém, sofre com as restrições de capacidade de memória, processamento e visão incompleta do ambiente. Neste caso, a única forma de obter informações globais do ambiente é através da cooperação entre robôs [73].

Uma tendência tem sido aproveitar o que as estratégias apresentam de melhor, levando ao desenvolvimento de arquiteturas híbridas (reativa x deliberativa) e distribuídas nas quais as tarefas de controle de caráter mais deliberativo, tais como a geração de estratégias, criação e manipulação do modelo do ambiente e controle de navegação global dos robôs sejam feitas por uma entidade de controle central, enquanto que tarefas de caráter mais reativo e que envolvem a necessidade de um tempo de resposta restrito, tais como o desvio de obstáculos, a condução da bola e a decisão do momento de chute sejam feitas localmente e de maneira autônoma para cada robô da equipe [22] e [53].

Uma grande dificuldade no desenvolvimento deste tipo de arquitetura é a definição precisa de que competências colocar no controle central e de que competências colocar nos robôs móveis inteligentes locais de forma a minimizar o tamanho e a quantidade de mensagens de comunicação trocadas entre a entidade central e os robôs móveis

inteligentes; permitir que os robôs móveis inteligentes ainda possam operar no ambiente de maneira satisfatória, mesmo no caso da entidade central não estar disponível; e poder atender às restrições presentes em aplicações robóticas reais, tais como a necessidade das informações serem tratadas em tempo real, sempre considerando a influência de ruído que está presente em cada informação, bem como informações incompletas, pouco precisas, extraídas do ambiente dinâmico e com iluminação não uniforme [58] e [59].

1.2 Objetivos

1.2.1 Objetivo geral

Propor um modelo de arquitetura híbrida e distribuída que seja considerado eficaz para sistemas multiagentes aplicado ao futebol de robôs.

1.2.2 Objetivos específicos

Para se chegar ao objetivo geral deste trabalho, é necessária a realização dos seguintes objetivos específicos:

- Identificar quais são os pontos fortes e fracos encontrados nos modelos de arquiteturas vistos no levantamento bibliográfico;

- Propor um modelo de arquitetura híbrida e distribuída para sistemas multiagentes e sua aplicação no futebol de robôs;

- Implementar o modelo da arquitetura híbrida e distribuída para sistemas multiagentes e aplicar no domínio do futebol de robôs na categoria *Small Size Robot League (F-180)* da RoboCup;

- Avaliar o modelo da arquitetura híbrida distribuída proposta.

1.3 Domínio da aplicação

Atualmente pesquisas relacionadas ao jogo de futebol de robôs estão sendo realizadas em diversas instituições tanto nacionais quanto internacionais, apresentando resultados significativos tanto no processo de compreensão dos sistemas robóticos quanto em sistemas de cooperação multiagentes.

1.3.1 Características do Cenário

O cenário utilizado para testes, na liga *Small-Size* (F-180) da *RoboCup*, é composto de um campo plano, de cor verde com dimensões de 4,9m de comprimento x 3,4m de largura [64]. Para capturar as imagens do ambiente, é utilizado um sistema de visão global e também uma câmera digital colorida acoplada a cada robô *EyeBot*. A bola utilizada é de golfe, de cor laranja.

1.3.2 Características do *EyeBot*

O modelo de robô *SoccerBot Plus (EyeBot)*, conforme pode ser observado na Figura 1.1, apresenta as seguintes características: microcontrolador MC68332 Motorola de 32 *bits* com 25 *MHz*, 1 *MB* de memória *RAM*, 512 *Kb* de memória *Flash-ROM, display* de *LCD* com resolução de 128x64 *pixels* para apresentação de gráficos com baixa resolução, portas paralelas e seriais, entradas e saídas digitais e analógicas.

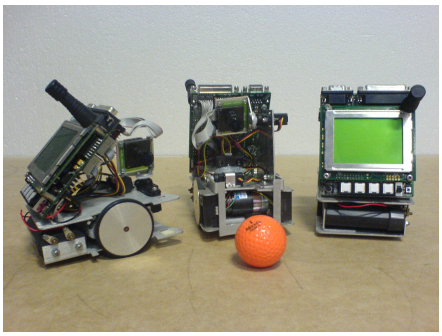

Figura 1.1: Robô *EyeBot*.

Este modelo de robôs apresenta também dois motores de passo, dois servo motores, dois *encoders* acoplados a cada uma das rodas, três sensores de aproximação infravermelho, bateria recarregável com indicador de nível, câmera digital colorida de 24 *bits* com resolução de 80x60 *pixels*, comunicação via rádio com velocidade 9.600 *bps*, atuando em um freqüência de 433 *MHz*, protocolo de tolerância à falha e configuração automática de rede, mecanismo de chute posicionado na parte da frente [26].

1.4 Estrutura deste trabalho

A estrutura deste trabalho está dividida em capítulos, sendo que os conteúdos vão sendo abordados na tentativa de conduzir de forma clara e concisa para o entendimento dos assuntos necessários para que se possa transmitir ao leitor a proposta desta dissertação. A seguir, descreve-se de forma sucinta cada capítulo bem como o conteúdo abordado em cada um deles.

O segundo capítulo, **Fundamentação Teórica**, fornece os pré-requisitos necessários para o desenvolvimento deste trabalho, incluindo os modelos com as características das arquiteturas existentes propostas por diversos autores, aplicados ao futebol de robôs e as principais idéias que abordam este assunto.

O terceiro capítulo, **Modelo de Arquitetura Proposta**, expõe as principais características que compõem o modelo de arquitetura híbrida para sistemas multiagentes e sua aplicação no futebol de robôs proposto por este trabalho. Este capítulo descreve o funcionamento do modelo de arquitetura proposta, bem como justificativas e decisões tomadas no processo que envolve o modelo de arquitetura adotada.

O quarto capítulo, **Implementação da Arquitetura**, apresenta de forma generalizada, os principais recursos e técnicas utilizados no processo de desenvolvimento desta aplicação, bem como os passos seguidos para a obtenção dos resultados.

O quinto capítulo, **Resultados**, apresenta os resultados obtidos ao final desta aplicação.

Finalmente, o último capítulo, **Conclusão**, apresenta as conclusões pessoais, e sugere alguns trabalhos futuros nesta linha de pesquisa.

Capítulo 2

Fundamentação Teórica

2.1 Introdução

Um agente pode ser definido como sendo qualquer entidade que percebe o ambiente através de sensores e atua neste ambiente através de atuadores [65]. Neste trabalho, quando nos referimos a um agente, estamos nos referindo especificamente ao robô.

Franklin [28] define agentes sobre dois aspectos: primeiro em relação à autonomia, que diz respeito às características que devem existir no agente para poder agir no ambiente por si só, tendo como objetivo alcançar as metas pré-definidas de acordo com os estados internos, e segundo, em relação à habilidade, que torna o agente capaz de exercer um controle sobre suas próprias ações sem a interferência de um supervisor.

Segundo Arleo [5], um modelo ideal de agente autônomo deve possuir um sistema autônomo de navegação para adaptar-se ao ambiente em que está inserido, para que possa modificar suas ações diante de situações imprevistas. O método clássico e pré-definido é aquele modelo em que o agente interage com o ambiente externo. O problema surge a partir do momento em que se percebe que o mundo externo é muito complexo [35]. Desta forma, a partir de um sistema desenvolvido e pré-definido, o agente autônomo só será capaz de atuar em um ambiente altamente controlado [56].

Há muitas maneiras de endereçar o problema relacionado à localização.

Uma das formas de aproximação muito comum é a de ignorar erros relacionados a sua posição [53]. Este método tem a vantagem de ser simples, porém uma desvantagem considerada importante é que este método não pode ser considerado um método global de planejamento. Para superar este problema, pode-se inserir um processo de correção de posição aplicando técnicas que utilizam reconhecimento do ambiente, como sensores a laser e mapas que indicam a posição correta [38]. Normalmente os sensores que são utilizados para indicar a posição do agente são sensores de distância, tais como laser, infravermelho e sonar [52]. O problema é que os dados fornecidos ao agente autônomo, geralmente sofrem a influência de ruídos; mesmo assim podem ser utilizados em ambientes altamente restritos [32]. Este problema pode ser superado com a utilização de pequenos mapas que indicam a posição exata em que o agente se encontra no ambiente [33]. No entanto, o processo de construção do conjunto de mapas geralmente exige um alto poder de precisão [53].

Entre os desafios a serem abordados e os problemas a serem estudados, Shen [72] afirma que agentes autônomos devem ser capazes de reconhecer o ambiente em que estão inseridos em tempo real, navegar em um ambiente dinâmico, ser capazes de reconhecer ou rastrear um objeto que está em movimento e cooperar com outros agentes que estão inseridos no mesmo ambiente. Para que todos estes objetivos sejam atingidos, é necessário que os robôs sejam eficientes, autônomos, cooperativos, com capacidade de aprendizado, raciocínio e planejamento, sendo que tudo isso deve ser realizado em tempo real [29], [68] e [81].

O objetivo de jogar futebol com agentes autônomos inteligentes foi proposto por diversos pesquisadores [40], [42], [43] e [69], tendo como princípio criar um novo desafio em longo prazo para a Inteligência Artificial. De acordo com [8], [9] e [68], o futebol de robôs foi escolhido por apresentar um excelente campo de teste para a evolução de várias teorias, algoritmos e arquiteturas de agentes autônomos. Na perspectiva de Sahota [66], um dos objetivos da Inteligência Artificial é o processo de construção de robôs totalmente autônomos. Tais agentes devem perceber, interagir e atuar em ambientes dinâmicos e incertos [35]. Assim é preciso desenvolver teorias e ferramentas que possam auxiliar nesta meta [36] e [37]. A formação de uma equipe de jogadores de

futebol de robôs envolve algo mais do que apenas recursos e técnicas de Inteligência Artificial. Conforme [44], podem ser citados os seguintes recursos utilizados para a formação de um time de futebol de robôs: dispositivos de hardware especializados no controle de sensores e atuadores, dispositivos mecatrônicos, além de recursos de software tais como, sistemas que auxiliam no controle do agente, na interpretação do ambiente, sistemas de fusão sensorial (combinação de vários sensores), visão computacional, redes neurais, computação evolutiva e multiagentes [21], [54] e [79]. A Figura 2.1 mostra a equipe de agentes autônomos, *CMUnited97*, que competiu na *RoboCup97* [82].

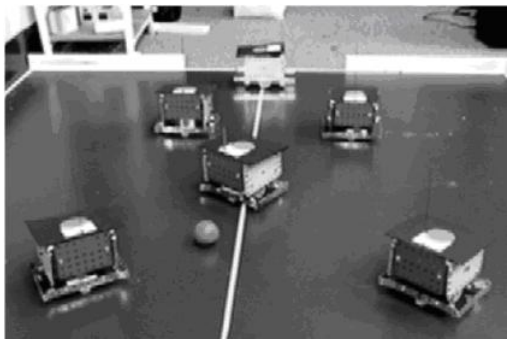

Figura 2.1: *CMUnited97* que competiu na *RoboCup97*. Extraído de [82].

A organização dos campeonatos mundiais de futebol de robôs é de responsabilidade das organizações científicas, a RoboCup e a FIRA. Estas organizações coordenam as entidades que promovem os eventos. Atualmente existem diversas modalidades de jogos para futebol de robôs, que envolvem variações no número de competidores, tamanho e, até mesmo, a capacidade computacional de cada jogador. Dentre as modalidades existentes da FIRA, podem-se citar: *HuroCup, KheperaSot, MiroSot, NaroSot, RoboSot, SimuroSot*.

HuroCup - *Humanoid Robot World Cup Soccer Tournament*: Esta categoria é composta por um único robô bípede. O tamanho máximo permitido para o robô é de 150cm, com peso máximo de 30kg. As dimensões do campo são variáveis entre 340 ~430cm x 250~350cm. O robô pode ser totalmente autônomo ou controlado através de um controle remoto. A Figura 2.2 apresenta o modelo do robô para a categoria *HuroCup*

[27].

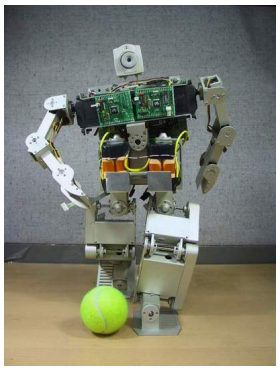

Figura 2.2: Modelo de robô categoria *HuroCup*. Extraído de [27].

KheperaSot - *Khepera Robot Soccer Tournament*: Para esta categoria, uma partida de futebol de robôs é jogada por duas equipes, sendo que cada equipe é composta por um único robô. Os robôs são autônomos e possuem seu próprio sistema de visão. As dimensões do campo são de 130cm de comprimento x 90cm de largura. A bola usada é uma bola de tênis de cor amarela. A Figura 2.3 apresenta o modelo do robô para a categoria *KheperaSot* [27].

Figura 2.3: Modelo de robô categoria *KheperaSot*. Extraído de [27].

MiroSot - *Micro Robot World Cup Soccer Tournament*: esta modalidade é considerada a de mais fácil implementação, uma partida é jogada por duas equipes composta de três robôs, sendo que um robô de cada equipe deve ser o goleiro. Esta

modalidade é organizada com robôs menores e de mais baixo custo. Fisicamente a estrutura desta modalidade é composta de um campo plano para realização do jogo, com dimensões de 150cm de comprimento x 130cm de largura, sendo que cada time possui uma câmera de CCD (*Charge-Coupled Device*), para captura das imagens, um computador para processamento das imagens e transmissão dos dados. O tamanho de cada robô está limitado em 7,5cm de comprimento x 7,5cm de largura x 7,5cm de altura e são compostos por dois motores controlados por um microprocessador, com bateria própria e sistema de comunicação sem fio. A bola usada é uma bola de golfe de cor laranja. A Figura 2.4 mostra o modelo de robô categoria *MiroSot* [27].

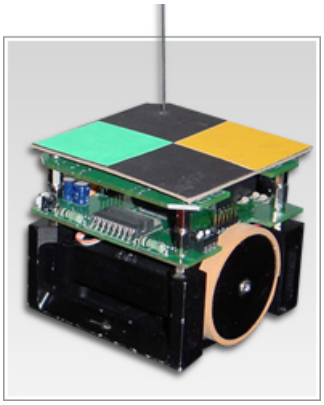

Figura 2.4: Modelo de robô categoria *MiroSot*. Extraído de [27].

NaroSot - *Nano Robot World Cup Soccer Tournament*: para esta modalidade de competição, as equipes são formadas por cinco competidores, sendo que um robô de cada equipe deve ser obrigatoriamente o goleiro. As dimensões dos robôs estão limitadas a 4cm de comprimento x 4cm de largura x 5,5cm de altura. A altura da antena não está sendo considerada. As dimensões do campo são de 130cm de comprimento x 90cm de largura. A bola usada, é uma bola laranjada de pingue-pongue. A Figura 2.5 mostra o modelo de robô categoria *NaroSot* [27].

RoboSot - *Autonomous Robot-soccer Tournament*: nesta categoria, as equipes podem ser formadas por um ou até três robôs. As dimensões de cada robô são 20cm de comprimento x 20cm de largura, sem limite de altura, e as dimensões do campo

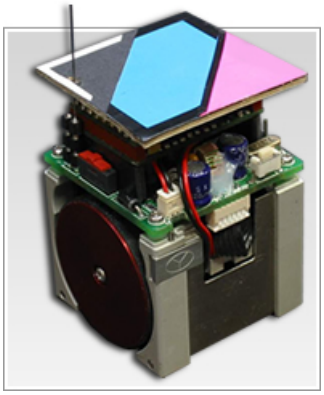

Figura 2.5: Modelo de robô categoria *NaroSot*. Extraído de [27].

são de 260cm de comprimento x 220cm de largura. A bola utilizada é uma bola de tênis amarela com verde claro. A Figura 2.6 mostra o modelo de robô categoria *RoboSot* [27].

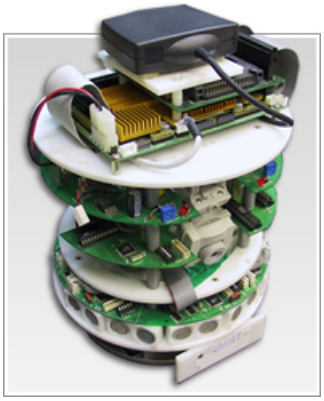

Figura 2.6: Modelo de robô categoria *RoboSot*. Extraído de [27].

SimuroSot - *Simulated Robot Soccer Tournament*: esta categoria é simulada através de uma entidade central que possui o ambiente de jogo, sendo apresentado através de uma tela gráfica 3D. Ao mesmo tempo, a entidade central roda os programas clientes, responsáveis pelas estratégias do jogo. Cada equipe possui sua própria estratégia de jogo. Uma partida pode ser jogada com um número mínimo de cinco ou máximo de onze jogadores. A Figura 2.7 mostra a tela gráfica da categoria *SimuroSot* [27].

Já na modalidade da RoboCup, podem-se citar: *Four Legged League,*

Figura 2.7: Tela gráfica da categoria *SimuroSot*. Extraído de [27].

Humanoid League, Middle Size League, Simulation League, Small Size Robot League (F-180).

Four Legged League: para esta categoria, as equipes são compostas de até quatro robôs, sendo que um robô deve ser obrigatoriamente o goleiro. As equipes devem utilizar robôs Sony AIBO. Os robôs são totalmente autônomos. As dimensões do campo são de 6m x 4m. A Figura 2.8 mostra o modelo de robô categoria *Four Legged League* [64].

Figura 2.8: Modelo de robô categoria *Four Legged League*. Extraído de [77].

Humanoid League: para a categoria de robôs *humanoid* da *RoboCup*, os robôs são divididos em duas categorias: *KidSize* com altura entre 30cm e 60cm e *Teen-*

Size com altura entre 80cm e 130cm. Cada equipe é formada por um único robô. Além de jogar futebol, existem outras competições para esta modalidade, tais como: cobrança de pênalti e corrida dinâmica [64].

Middle Size League: para esta modalidade, as equipes são compostas por robôs totalmente autônomos. Não é permitida a intervenção humana, exceto na inserção e na retirada dos robôs do campo. Os robôs possuem sensores *on-board* para detectar obstáculos e sistema de visão para reconhecimento de cores [64].

Simulation League: a categoria de simulação tem por objetivo oferecer aos pesquisadores o desenvolvimento de algoritmos com estratégicas de jogo diferenciadas. Cada equipe é composta por onze jogadores robóticos para jogar futebol, sendo que o sistema de simulação apresenta os agentes em um ambiente 3D [64].

Small Size Robot League (F-180): nesta categoria, as equipes podem ser formadas por um ou até cinco robôs, sendo que um robô deve ser obrigatoriamente o goleiro. As dimensões do campo são de 4,9m de comprimento x 3,4m de largura, de cor verde, podendo ser de feltro ou carpete. A bola utilizada é uma bola padrão de golfe, de cor laranja, aproximadamente 46g de massa com aproximadamente 43mm de diâmetro. A Figura 2.9 mostra um modelo de robô categoria F-180 [64].

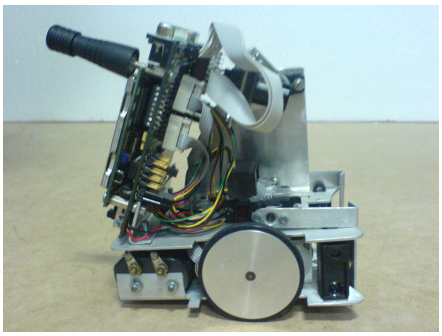

Figura 2.9: Modelo de robô categoria F-180.

O desenvolvimento de um projeto, com o intuito de jogar futebol envolvendo uma equipe de robótica móvel, é considerado uma aplicação prática do uso de agentes autônomos cooperando entre si, com recursos de Inteligência Artificial e visão

computacional para a realização de uma tarefa computacional considerada complexa [67]. O grau de cooperação que deve existir entre estes agentes, no meio em que estão inseridos com o objetivo de realizar determinada tarefa, está relacionado à quantidade de informações que estão sendo percebidas por um determinado agente durante a realização e execução de uma partida de futebol. Portanto, deve-se desenvolver sistemas que sejam capazes de obter estas informações de modo que elas possam ser tratadas e fornecidas para cada agente de maneira eficiente e precisa, no contexto em que estão sendo consideradas. Tendo em mente que o domínio em que os agentes autônomos estão inseridos, estas características são consideradas complexas, pois envolvem a necessidade de serem tratadas em tempo real, sempre considerando o grau de incerteza presente em cada informação, bem como a existência de ruídos e de informações incompletas extraídas de um ambiente dinâmico [58] e [59].

Robôs móveis inteligentes utilizam os três principais paradigmas para a navegação móvel. Controlar de forma eficiente o problema dos robôs móveis inteligentes sempre foi um desafio para os pesquisadores. Historicamente pode-se definir as estratégias em três fases distintas, representadas por três paradigmas de controle: o hierárquico ou deliberativo, o reativo e híbrido deliberativo/reativo. Estes paradigmas podem ser descritos através de três princípios robóticos: Perceber, Planejar e Atuar [53]. Perceber corresponde a informações aceitas pelos sensores do robô e à transformação dessas informações em saídas úteis para outras funções. Planejar corresponde à função que utiliza a saída do princípio Perceber ou o conhecimento adquirido para criar comportamentos ou tarefas para serem executadas pelo princípio Atuar. Atuar corresponde às mensagens de comando que os atuadores do robô executam [49].

2.2 Paradigma hierárquico ou deliberativo

O paradigma hierárquico ou deliberativo é considerado como sendo o paradigma mais antigo, 1967 - 1990 [53]. Antes do surgimento deste paradigma, os robôs atuavam no ambiente de maneira *top-down* com relação ao planejamento. Neste paradigma, os robôs utilizam os três princípios robóticos: Perceber, Planejar e Atuar para

navegação. Portanto o robô sente o mundo, planeja a próxima ação e executa a ação planejada. Para cada execução, é necessário que o robô planeje o próximo movimento. Uma outra característica do paradigma hierárquico ou deliberativo é que todos os sentidos que correspondem às informações tendem a ser reunidas em um único modelo do mundo global. Isto significa dizer, que uma única representação é utilizada pelo princípio planejar e corresponde a todas as ações que serão executadas na navegação. O processo de construção de um único modelo genérico para representar o mundo global pode ser considerado como sendo uma representação não adequada desse mundo ou pouco precisa com relação ao problema que se está pretendendo modelar [51]. A Figura 2.10 representa o paradigma hierárquico ou deliberativo.

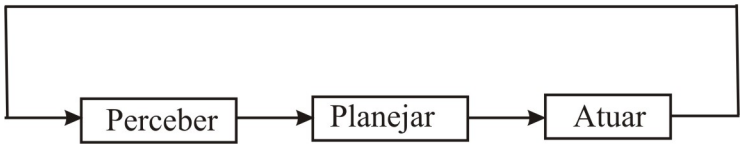

Figura 2.10: Paradigma híbrido hierárquico ou deliberativo. Adaptado de [53].

Dessa forma, o paradigma hierárquico ou deliberativo pode ser imaginado como sendo uma passagem de fluxo de eventos, ignorando evidências biológicas de perceber informações que podem ser inseridas diretamente em uma nova ação, isto é, as entradas que correspondem às informações, perdem o sentido [53].

2.3 Paradigma reativo

O paradigma reativo é uma evolução do paradigma hierárquico ou deliberativo e tornou-se um dos responsáveis pelo desenvolvimento da robótica móvel. Este paradigma foi utilizado em grande escala nas décadas de 80 e 90 [53]. Ele ainda permanece em uso, só que em menor escala. A partir de 1992, a tendência do paradigma reativo foi migrar em direção à construção de arquiteturas híbridas [50].

Com a utilização do paradigma reativo, tornou-se possível o processo de construção de robôs móveis inteligentes diante de duas tendências: A primeira constitui-

se em um movimento popular formado por pesquisadores de Inteligência Artificial, tendo como objetivo direcionar suas pesquisas para as áreas de biologia e psicologia com o objetivo de investigar e ao mesmo tempo explorar exemplos de inteligência. A outra tendência constitui-se diante da diminuição de forma acelerada dos custos envolvendo hardwares de computadores e, ao mesmo tempo, maior poder de capacidade de armazenamento. Como resultado considerado expressivo para a época, pode ser citado o processo de construção de robôs que procuravam simular comportamentos de insetos [45].

O paradigma reativo foi responsável por lançar o princípio primitivo Planejar junto com a saída. Ele constitui-se no mecanismo de Perceber-Atuar em forma de uma organização. Enquanto que o paradigma hierárquico assume que o resultado que corresponde à saída para uma ação sempre deriva do resultado produzido por um plano, o paradigma reativo assume que o resultado que corresponde à saída de uma ação sempre será a saída do próprio sensor [70] e [53]. A Figura 2.11 representa o paradigma reativo.

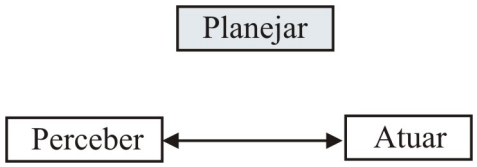

Figura 2.11: Paradigma reativo. Adaptado de [53].

Para o caso onde o processo Perceber-Atuar está conectado diretamente com a saída da ação a ser executada, pode-se dizer que o paradigma reativo está em pleno funcionamento. Para os robôs móveis inteligentes, são apresentados inúmeros exemplos condicionados ao comportamento Perceber-Atuar [52]. Estes pares de comportamentos são processos considerados concorrentes, responsáveis pela coleta de informações, que estão presentes no ambiente e, ao mesmo tempo, escolhem a melhor ação para que seja aceita, independente de outros processos que estejam em execução [50]. Um comportamento em particular pode significar a direção que o agente deve seguir. Por exemplo: mova-se para frente 2 metros (Atuar corresponde à direção dos motores), para que possa ser atingida uma meta (Perceber a meta), enquanto um outro comportamento informa que

os motores devem girar 90^0 no sentido horário para que possa evitar uma colisão com um outro agente que se encontra parado em frente (Perceber obstáculos). O robô tem a capacidade de combinar comportamentos, mesmo que em um determinado instante ele esteja temporariamente fora de seu curso. Este recurso é necessário para que haja a possibilidade de ser bem sucedido na tarefa que a ele foi designada. No caso do comportamento acima citado, o robô deve girar 45^0 para evitar uma colisão. Percebe-se que em nenhum momento este robô recebeu um comando direto no sentido de que deveria girar 45^o. O mecanismo que envolve o processo de atuação dos robôs móveis inteligentes emerge da combinação de dois comportamentos [51].

Dessa forma, o paradigma reativo é responsável por produzir resultados e demonstrações através de modelos de robôs que procuram simular insetos. Através do processo de demonstração é possível perceber a não necessidade do princípio Planejar com a construção de robôs móveis inteligentes. O processo que envolve o estímulo-resposta como forma de treinamento explica como acontece o processo de aprendizado, conforme modelo de Pavlov [84]. O paradigma reativo apresenta muitas propriedades que são consideradas desejáveis. Dentre as que se pode destacar está a capacidade de execução de forma rápida e sem a necessidade de planejamento [53].

2.4 Paradigma híbrido deliberativo/reativo

O paradigma híbrido deliberativo/reativo permanece como sendo uma das áreas que ainda está em fase de pesquisas [53]. Nesse paradigma, o agente primeiramente realiza planos (deliberados) que consistem em melhor decompor uma tarefa em subtarefas que, ao mesmo tempo, pode-se chamar de "missão planejar", que são os componentes considerados essenciais e adequados para a realização de cada subtarefa [3]. Deste modo, cada comportamento inicia seu processo de execução como sendo um paradigma reativo. Esta forma de organização é chamada de Planejar, Perceber-Atuar. Isto significa dizer que o processo Planejar está pronto em um único passo; já os processos Perceber-Atuar ficam prontos ao mesmo tempo. O mecanismo que consiste em perceber a organização no paradigma híbrido deliberativo/reativo é também ao mesmo tempo uma

mistura dos paradigmas reativo e hierárquico [70]. A Figura 2.12 representa o paradigma híbrido deliberativo/reativo.

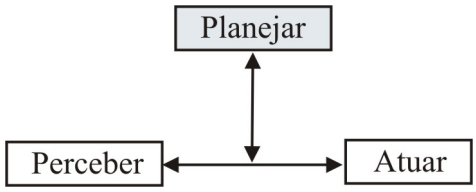

Figura 2.12: Paradigma híbrido deliberativo/reativo. Adaptado de [53].

Os sensores são responsáveis pelo processo de captura de informações para um comportamento traçar rota e necessitam do processo que envolve perceber o que está acontecendo no ambiente. Ao mesmo tempo, precisam disponibilizar estas informações para o processo Planejar, que é responsável pela construção de uma determinada tarefa orientada a partir de um modelo do mundo global [3].

O processo Planejar pode estar monitorando a entrada de informações mesmo que ele esteja oculto, isto é, está sempre pronto para uma possível necessidade de um novo comportamento. Um exemplo desse comportamento seria identificar um obstáculo que foi inserido por uma pessoa no ambiente em que se encontra o robô móvel inteligente. Dessa forma, o processo Planejar faz atualização computacional sempre que necessário, isto é, o planejamento deliberativo pode atualizar-se a qualquer momento. Diferentemente, os comportamentos reativos freqüentemente executam o processo de atualização em intervalos de tempo pré-estabelecidos e constantes. [53].

2.5 Descrição de algumas arquiteturas relevantes

Nesta seção, apresenta-se diversos exemplos de implementações de arquiteturas que seguem o paradigma reativo e híbrido. A justificativa pela escolha em descrever em detalhes apenas quatro modelos de arquitetura (a *subsumption*, a *SOTY II*, a *LOGUE* e a *fuzzy*), é porque eles apresentam aspectos considerados relevantes que influenciaram no modelo de arquitetura que se está propondo. A partir da descrição destes qua-

tro modelos, pretende-se buscar idéias e aspectos inspiradores no processo de construção da arquitetura a ser proposta neste trabalho. Assim, monstram-se as principais características das arquiteturas escolhidas, analisando suas vantagens e desvantagens.

A primeira arquitetura a ser exposta é o modelo proposto por Rodney Brooks [18], sendo chamado de arquitetura de *subsumption*. O segundo modelo foi sugerido por Hyun-Sik Shim [73], denominado de arquitetura de controle híbrida do SOTY II. O terceiro é o chamado LOGUE: uma arquitetura para transmissão de ação de comportamento e tarefa entre múltiplos robôs autônomos, criado por Wang [83]. O quarto, desenvolvido por Vadakkepat [80], foi o modelo de evolução de comportamentos *fuzzy* para sistema multi-robótico.

2.5.1 Arquitetura de *Subsumption*

2.5.1.1 Introdução

Na década de oitenta, surgiu um grupo de pesquisadores, cujo pesquisador de maior destaque foi Rodney Brooks [18]. Parte de sua influência está relacionada ao grande número de publicações, envolvendo a arquitetura de *Subsumption* como uma implementação do paradigma reativo [53]. Rodney Brooks é pesquisador do *Artificial Intelligence Laboratory of Massachussets Institute of Technology (MIT)*. Esse grupo de pesquisadores acreditava que o nível de inteligência dos seres humanos apresentava-se em um nível de complexidade muito elevado e ao mesmo tempo pouco se sabia de que forma decompor esta inteligência para que pudesse ser compreendida corretamente [18].

De acordo com Maes [46], o grupo de pesquisadores do MIT propôs uma arquitetura totalmente diferente da arquitetura tradicional, voltada para agentes autônomos inteligentes, permitindo solucionar o problema de planejamento de uma outra forma, envolvendo a modelagem do sistema de controle dos agentes autônomos móveis [4], [18] e [48] A Figura 2.15 mostra esta arquitetura.

2.5.1.2 Percepção - Ação

O ciclo de percepção-ação é fundamental para a navegação de agentes autônomos inteligentes [18]. Uma interpretação simples do ambiente pode provocar uma ação a ser executada pelo agente neste ambiente. Portanto, a percepção do agente com relação ao mundo é modificada e a nova percepção então é utilizada na próxima ação a ser executada [53].

O princípio da percepção-ação implementado na Arquitetura de *Subsumption*, descarta a representação do conhecimento como sendo uma base fundamental dos sistemas artificiais inteligentes. Este princípio é baseado na iniciativa de que a inteligência é abordada de forma incremental. O modelo é construído com camadas superpostas, isto é, uma camada qualquer produz suas próprias atividades, ou comportamentos, de forma completa e específica e então a representação que é dada ao conhecimento desaparece. Em outras palavras, pode-se dizer que o sistema desenvolvido percebe modificações que são inseridas no ambiente e reage de forma adequada e imediata a esta percepção [20] e [49].

2.5.1.3 Unidades funcionais

O sistema de controle para robôs autônomos móveis deve ser completo ao ponto de ser capaz de realizar tarefas consideradas complexas do mundo real [18]. Uma forma de desenvolver este sistema de controle para robôs autônomos móveis é decompor o sistema em uma série de unidades funcionais. A Figura 2.13 apresenta o modelo da decomposição tradicional de um sistema de controle em unidades funcionais.

Neste modelo, cada unidade funcional está estritamente ligado à unidade anterior e ao nível da unidade posterior, de modo que este sistema tem que ser projetado de maneira completa e precisa, haja vista que um nível não possui a característica de ser completo e ao mesmo tempo ser capaz de realizar atividades de forma independente. Havendo a necessidade de incrementar uma nova unidade funcional, é necessário que todo o projeto seja alterado [19].

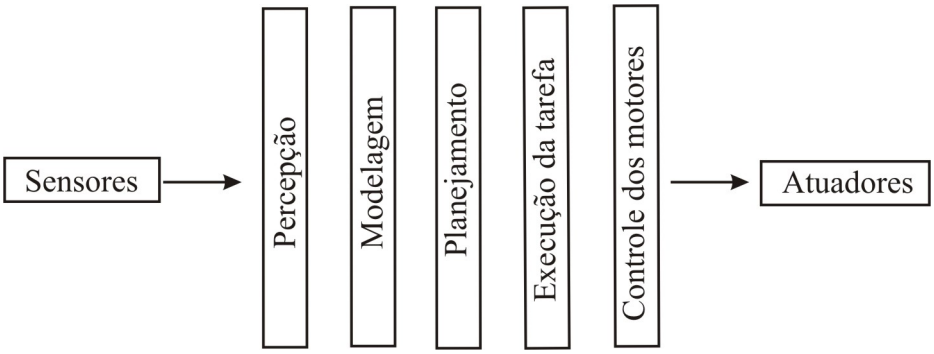

Figura 2.13: Modelo da decomposição tradicional de um sistema de controle em unidades funcionais. Adaptado de [19].

2.5.1.4 Níveis de atividades

A forma como a arquitetura de *subsumption* é representada difere notavelmente do modelo de unidade funcional, que consiste em dividir o controle do sistema em níveis funcionais: percepção, modelagem, planejamento, execução da tarefa e controle dos motores. Ao invés disso, a arquitetura de *subsumption* consiste em organizar o sistema de controle para os robôs autônomos móveis em forma de camadas, sendo que cada camada representa uma tarefa qualquer ou um comportamento completo. A Figura 2.14 apresenta a decomposição de um sistema de controle baseado em camadas de atividades.

Neste novo formato de divisão, a utilização de camadas de atividades permite acrescentar um novo comportamento no exato momento em que ele se faz necessário para a aplicação, ou seja, o processo de inserção de uma nova camada consiste apenas em interfacear esta camada com o sistema. O princípio deste modelo de arquitetura consiste em construir um sistema autônomo mesmo que seja simples, mas que seja ao mesmo tempo completo e, após a construção, testá-lo no ambiente real [19]. A Figura 2.15 representa o modelo da arquitetura de *subsumption*.

A arquitetura de *subsumption* caracteriza-se pelo fato de que a saída da camada que se encontra em mais alto nível sobrescreve a saída da camada que se encontra no nível inferior. Este fato é representado na Figura 2.15 pelo "S" no círculo. Assim, em um determinado momento considerado adequado, os sensores responsáveis

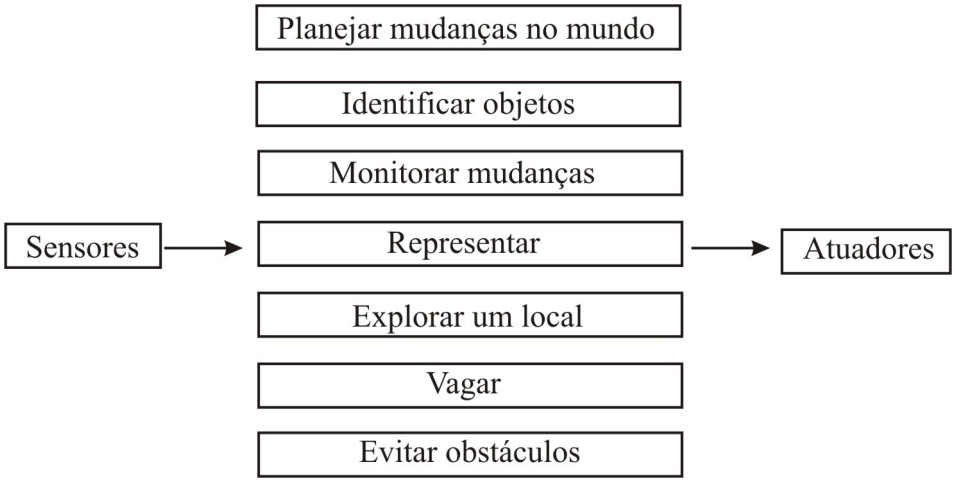

Figura 2.14: Divisão do sistema de controle baseado em camadas de atividades. Adaptado de [19].

pela camada indicam uma situação que é considerada propícia para sua atuação. Diante desta característica, a camada que se encontra acima, suprime o fluxo de ações da camada imediatamente inferior. O sistema desenvolvido deve apresentar como característica o fato de ser completo e ao mesmo tempo independente de cada nível e, mesmo assim, as camadas inferiores continuarão formando um sistema que simultaneamente deve ser funcional e completo [63].

Na metodologia de decompor o sistema a ser desenvolvido em camadas de comportamentos, pode-se inferir os seguintes argumentos [18]:

1- Todas as atividades que são consideradas simples e que se encontram nos níveis inferiores da arquitetura podem apresentar características de reações rápidas. Isto ocorre em virtude de que estas camadas procuram representar o mundo através da simplicidade. Este sistema utiliza o princípio de percepção-ação. Isto significa dizer que o monitoramento do ambiente é realizado através de sensores de forma freqüente e a partir deste monitoramento, o sistema aplica uma atualização de maneira constante;

2 - Com a aplicação de múltiplas atividades e sendo que estas atividades se encontram em paralelo, há uma menor possibilidade de ocorrer um colapso provocado por uma mudança qualquer no sistema. O que pode ocorrer em um determinado instante

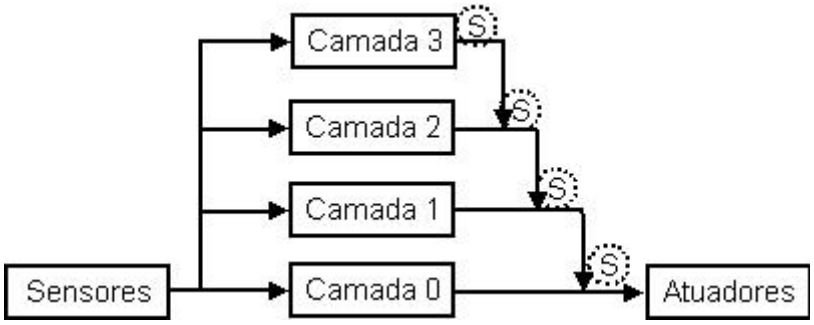

Figura 2.15: Arquitetura de *subsumption*. Adaptado de [19].

é a incapacidade de execução de algumas camadas do sistema;

3 - Cada camada de atividade de comportamentos pode ser imaginada como sendo uma camada que apresenta seu próprio mundo ou um modelo deste mundo. A principal atividade desta camada é a utilização do mundo como sendo seu próprio modelo, e de forma contínua, adaptar cada objetivo às condições do ambiente atual;

4 - A função do agente autônomo inteligente está implícita em suas camadas.

2.5.2 Arquitetura de controle híbrida do SOTY II

A arquitetura proposta por Shim [73] tem por objetivo auxiliar na construção de uma arquitetura de controle híbrida para um robô jogador de futebol [41] e [74]. Esta arquitetura apresenta níveis estratégicos e de execução. Estes níveis são: atribuição, ação, comportamento e execução. A Figura 2.16 representa a arquitetura de controle híbrida do SOTY II.

A arquitetura proposta é uma combinação dos níveis hierárquicos e de comportamentos. Esta combinação satisfaz comportamentos e ações, específicas para um sistema de futebol de robôs. O nível de execução é responsável pelo processo que comanda os atuadores, isto é, este nível é quem indica para o robô como ele deve atuar no ambiente. O nível hierárquico foi desenvolvido com base nos níveis de comportamentos, tais como: comportamento de movimentos que podem ser encontrados em um ambi-

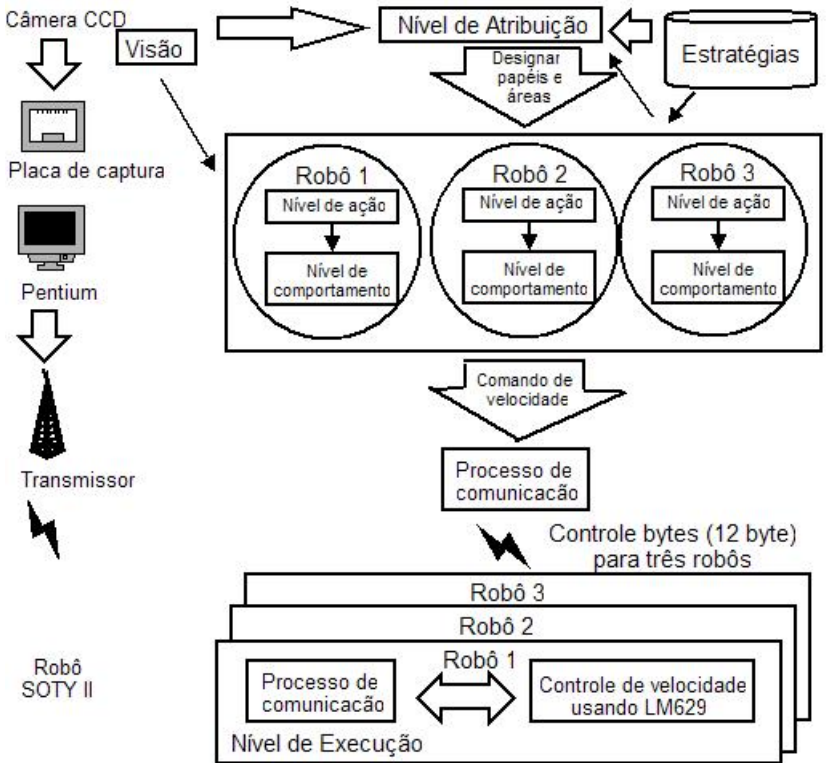

Figura 2.16: Arquitetura de Controle Híbrida do SOTY II. Adaptado de [73].

ente dinâmico. Por exemplo: desviar de obstáculos. Já o nível de ação foi desenvolvido com o objetivo de jogar futebol. Uma partida de futebol pode ser jogada sem o nível de atribuição (o nível de atribuição consiste no papel que será designado ao jogador), mas é necessário que sejam implementadas no nível de ação todas as estratégias que serão utilizadas durante a realização do jogo. Como exemplo pode-se citar a definição da posição em que o agente deve atuar, defesa ou ataque. Vale lembrar que devido a problemas que possam surgir no transcorrer da partida associados com a zona de defesa (indefinição da zona de defesa), geralmente associados ao mau funcionamento dos robôs (defeitos do próprio sistema), ou até mesmo em situações onde robôs ficam bloqueados por um robô da equipe oposta, justifica o processo de implementação do nível de atribuição. O nível de atribuição é responsável pelo processo de distribuição dos papéis dos agentes, sendo esta definição tomada de acordo com a situação em que o agente se encontra no campo.

O desenvolvimento do sistema de futebol de robôs do SOTY II é baseado no modelo de implementação do controle centralizado. Assim, apenas o nível de execução foi implementado nos robôs. Os demais níveis da arquitetura foram implementados em uma entidade central. A entidade central é a responsável pelo processamento das informações que são capturadas com o auxílio de um sistema de visão global e ao mesmo tempo executa as atribuições. Estas atribuições são realizadas pelos algoritmos que pertencem aos níveis de ação e de comportamento. Todos os comandos responsáveis pelo deslocamento e velocidade do robô são enviados aos robôs através de um módulo de comunicação via rádio.

Embora os níveis de atribuição, ação e comportamento sejam implementados na entidade central, estes níveis são ao mesmo tempo distribuídos ao longo do algoritmo desenvolvido. Isto equivale a dizer que cada robô apresenta sua própria função, ou seja, níveis de ação e comportamento foram implementados em algoritmos diferentes, utilizando variáveis de controle de forma diferenciada, para que cada algoritmo pudesse satisfazer o papel exercido a cada agente. Como exemplo, a diferenciação dos movimentos exercidos através de ações por um robô goleiro e um robô artilheiro. As variáveis responsáveis pelo controle são postas de forma diferenciada de acordo com os papéis que foram designados para cada robô.

Mesmo que a estratégia de jogo seja excelente, se os movimentos executados pelo robô são lentos, o sistema pode se tornar inútil. As ações do robô, tais como movimentos e comportamentos responsáveis pelo desvio dos obstáculos que possam surgir são fatores que devem ser considerados importantes na construção de um sistema voltado para jogar futebol de robôs [35].

O nível mais baixo da arquitetura de controle, o nível de execução de controle da velocidade dos robôs, deve apresentar um tempo menor para que possa apresentar uma reação imediata. Por outro lado, para níveis mais elevados, este tempo de reação pode se tornar mais longo. O tempo de reação responsável pelo nível de ação é determinado pelas informações que são atualizadas como o auxílio do sistema de visão [73].

Este modelo de arquitetura, apesar de oferecer diversos benefícios, tais

como, exploração completa do ambiente e decisão estratégica em nível global, deixa a desejar diante de um poder de reação imediato, pois, apenas o nível de execução foi implementado nos robôs. Desta forma apresenta um alto grau de incerteza de acordo com a dinâmica do jogo, visto que todas as ações de execução são planejadas e enviadas pela entidade central para que sejam executadas nos robôs. Assim, este modelo de arquitetura pode falhar no processo de identificação da posição exata em que o robô se encontra no ambiente. Uma melhor distribuição dos níveis entre a entidade central e os robôs pode melhorar este modelo de arquitetura.

2.5.3 LOGUE: uma arquitetura para transmissão de ação de comportamento e tarefa entre múltiplos robôs autônomos

A arquitetura, proposta por Wang [83], tem por objetivo auxiliar na comunicação da arquitetura de controle para melhorar a capacidade e a flexibilidade de múltiplos sistemas de robôs autônomos na execução de tarefas complexas e lidar com situações imprevisíveis. Para executar tarefas utilizando múltiplos sistemas de robôs autônomos, é preciso que cada robô possua informações precisas a respeito da tarefa a ser realizada. Além do mais, é desejável que o sistema de robôs autônomos apresente características de flexibilidade diante da realização de tarefas, que combinem a organização de ações de acordo com informações a respeito do processo de realização da tarefa [60].

Uma solução para este caso, é construir e incorporar todas as informações necessárias para a realização de tarefas e possíveis decisões que possam ser tomadas diante de diferentes situações pelo agente. Por outro lado, também é possível desenvolver um sistema de captura de informações para os robôs a respeito da tarefa e a ser executada em tempo-real. Estas informações podem ser adquiridas de outros agentes que já estejam engajados na realização desta tarefa, ou até mesmo da entidade central, através de uma rede que contenha uma base de dados das diferentes ações e comportamentos a serem relacionados para a tarefa a ser executada. Este sistema apresenta uma enorme vantagem em relação aos demais, pois permite ao agente autônomo realizar novas tarefas e até mesmo enfrentar novos desafios que possam surgir no ambiente. Este sistema de controle

permite o processo de construção a partir de um estilo incremental. Assim, o sistema de controle de um agente autônomo pode ser pequeno, necessitando apenas de informações com características e funções de controle básicas. Um sistema de controle considerado pequeno é realmente útil para agentes autônomos inteligentes, especialmente a partir do momento em que o número de agentes aumenta.

A arquitetura proposta permite a troca de tarefas e comportamentos envolvendo informações que possam auxiliar o agente autônomo a ser bem sucedido na realização da tarefa para a qual ele foi designado. Além disso, este modelo mantém um pequeno sistema de controle a bordo do robô que pode receber a tarefa e informações de comportamentos de outros agentes ou da entidade central. A Figura 2.17 apresenta a arquitetura do sistema LOGUE.

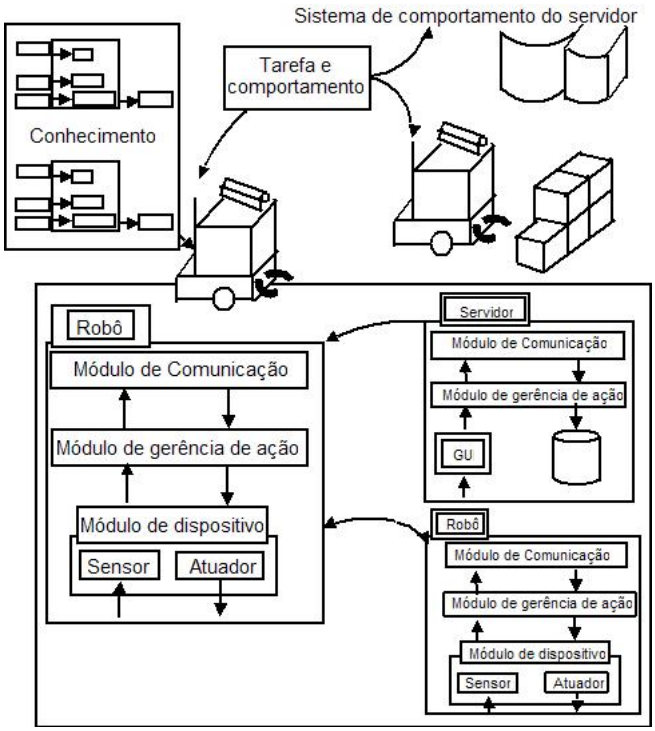

Figura 2.17: Arquitetura extraída do sistema LOGUE. Adaptado de [83].

O sistema LOGUE pode rodar tanto em robôs autônomos quanto em sistemas que gerenciam comportamentos. Para este sistema, três componentes são in-

cluídos: o módulo de comunicação, módulo de gerência de ação e módulo de dispositivo. O módulo de comunicação é responsável pelo envio dos pacotes de mensagens aos robôs. O módulo de gerência de ação inclui a gerência de comportamentos e a gerência de tarefas. Este módulo constrói ações que são transferidas pela rede, verifica sua capacidade, armazena e roda a ação da tarefa, como sendo componentes chaves do sistema. O módulo de dispositivo foi projetado com o objetivo de fornecer uma interface comum a todos os dispositivos nativos do robô, tais como sensores e atuadores. A Figura 2.18 apresenta uma visão interna do sistema LOGUE.

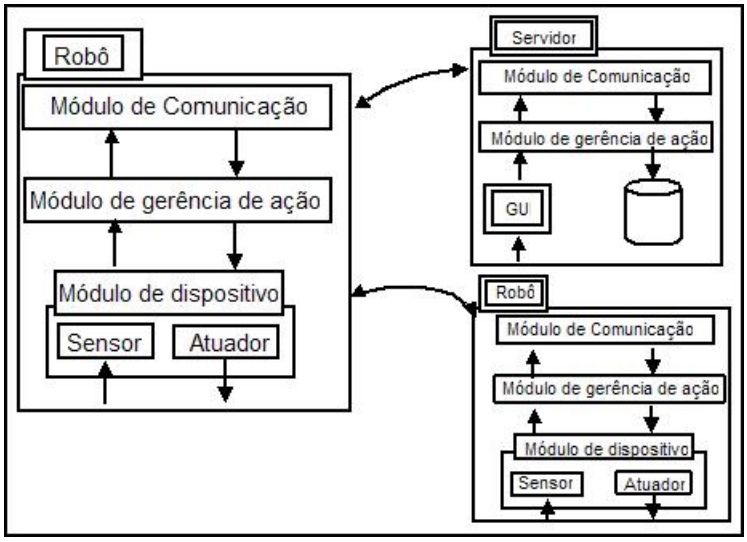

Figura 2.18: Visão interna parcial do sistema LOGUE. Adaptado de [83].

O módulo de comunicação é responsável pelo processo de transmissão e recepção de ações entre robôs e a entidade central. Este sistema roda em uma máquina virtual Java com ação e interface remotas, enviando argumentos da máquina virtual local para os robôs. Os resultados obtidos são organizados na máquina remota e são enviados à máquina virtual local. Ao mesmo tempo, ações que incluem alguns argumentos ou resultados não são invocadas remotamente e sim copiadas para a máquina virtual remota. Isto ocorre em função de que a invocação deve ser realizada na mesma máquina virtual. Assim, garante-se a consistência de controle da base de comportamento e ação de comportamento ou ação de tarefa, que deve ser transferida à máquina virtual do robô

atual.

No módulo de gerência de ação, todos os objetos de elementos de comportamento são administrados por um gerente de comportamento. Do mesmo modo, todas as ações de tarefas são administradas por um gerente de tarefas. O gerente de comportamento armazena em uma lista todos os objetos de elementos de comportamento. Neste objeto, um método chamado requer um comportamento enviando uma mensagem de pedido a outros robôs ou para a entidade central através do módulo de comunicação. Então, o gerente de comportamento informa ao remetente que o objeto do elemento de comportamento está disponível para o robô que solicitou. Mas se o gerente de comportamento verificar que o objeto do elemento de comportamento não existe, então é enviado uma mensagem ao solicitante, informando que este comportamento não está disponível.

O módulo de dispositivos foi projetado com dois propósitos. O primeiro é fornecer interface, mesmo que de forma abstrata, entre sensores e atuadores; o segundo mantem informações do estado interior do robô, tais como: posição, orientação e direção da câmera.

Este modelo de arquitetura apresenta a vantagem da cooperação multiagente através da troca de informações para a realização de novas tarefas. Por outro lado, esta arquitetura apresenta como uma das principais desvantagens a complexidade do sistema na definição e controle de seu conjunto de ações, visto que com o aumento do número de robôs atuando no mesmo ambiente, maior será o grau de complexidade do sistema, e sendo que o jogo de futebol de robôs é composto por diversos robôs, atuando ao mesmo tempo, o processo de definição e deliberação das ações exige um alto poder computacional.

2.5.4 Evolução de comportamentos *fuzzy* para sistema multi-robótico

A arquitetura, proposta por Vadakkpat [80], apresenta uma estrutura hierárquica de papéis e comportamentos. A base de comportamento deste modelo é composta de regras consideradas complexas e que ao mesmo tempo deverão ser decompostas em novas regras para os robôs através do módulo de comportamento e dos diferentes

níveis de complexidade. As ações consideradas primitivas são armazenadas primeiro, tendo como base a complexidade dos comportamentos. Papéis e comportamentos de maior complexidade são colocados nos níveis mais alto da hierarquia. Os comportamentos considerados de mais alto nível são decompostos em comportamentos simples, modulares e acessíveis. A Figura 2.19 apresenta o *layout* da arquitetura de comportamento do sistema multi-robótico.

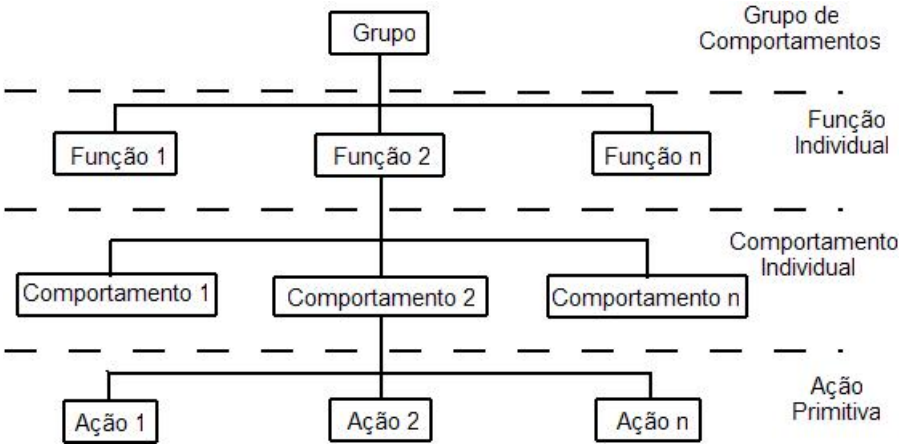

Figura 2.19: *Layout* da arquitetura de comportamentos do sistema multi-robótico. Adaptado de [80].

Na arquitetura, o comportamento de grupo, que está no topo da hierarquia, representa a estratégia que será adotada pela equipe. O comportamento de grupo é incorporado através dos efeitos coletivos que representam as atividades individuais de cada agente. Cada agente pode exibir diversos modos de comportamento, dependendo do papel que a ele foi designado pelo segundo nível. Em intervalos de tempo, o controle lógico *fuzzy* que se encontra no nível superior, designa papéis específicos para cada agente. Para que o agente possa cumprir o papel que a ele foi designado, é necessário executar comportamentos, ou seja, tarefas específicas que serão definidas pelo terceiro nível. Alguns comportamentos são reativos e podem ser observados na natureza através de estímulos provocados pelo ambiente. No entanto, outros comportamentos são guiados por objetivos e estratégias. Comportamentos considerados individuais são decompostos

futuramente em ações primitivas e encontram-se no nível inferior da hierarquia. Mesmo que apresentem características consideradas fáceis de serem realizadas ou ações primitivas, representam componentes que derivam de outros, que eram considerados complexos.

A lógica *fuzzy* foi aplicada neste sistema para realizar vários comportamentos em diferentes níveis da hierarquia. Há requisitos diferentes que ao mesmo tempo estão associados com o controle lógico *fuzzy* em diferentes níveis. Os controles responsáveis pelas ações primitivas devem apresentar movimentos considerados exatos para o agente. Já sensores *fuzzy*, que estão posicionados em níveis mais altos da hierarquia, são necessários para indicar o processo de construção de decisão e coordenação de comportamento.

O sistema desenvolvido de coordenação de comportamento utiliza dois mecanismos *fuzzy* na arquitetura. O primeiro mecanismo consiste da coordenação *fuzzy*, baseada nas regras de Connel [25]. Já o segundo mecanismo de atividade e contribuição de ação é conforme Abreu [1].

O processo de coordenação utiliza uma base de regras *fuzzy* que consiste em um processo de avaliação das condições do ambiente e, a partir deste conjunto de regras, escolhe o comportamento que é considerado conveniente para servir ao objetivo. A base de regras é o que determina a decisão que deverá ser tomada para que seja possível alcançar a coordenação do comportamento. Este método adota uma decisão de aproximação de cima para baixo e freqüentemente é usado na designação de papéis, coordenação de comportamento para papéis e coordenação de ação para comportamentos deliberativos. No entanto, a atividade e o método de contribuição de ação utilizam um mecanismo de aproximação de baixo para cima em seu processo de construção de comportamento.

O motor de inferência do sistema *fuzzy*, composto por sub-comportamentos, é o que fornece um par de atividades do tipo valor-ação. O valor-ação corresponde à ação sugerida, e o valor de cada atividade significa o grau de contribuição do sub-comportamento para cada comportamento do plano superior. Um algoritmo em forma de árbitro, localizado no plano superior, é responsável por revisar os valores que foram propostos pelos níveis inferiores e, ao mesmo tempo, calcular a saída de cada com-

portamento, semelhante ao método que calcula o centro da média. Na maioria dos casos, a atividade e o método de contribuição de ação são usados no processo que envolve a fusão de ação e seleção dentro de cada comportamento reativo.

Na arquitetura de comportamento proposta, a lógica *fuzzy* é aplicada na realização de ações primitivas, coordenação de comportamento, construção de regras e designação de papéis. Este projeto de controle lógico *fuzzy* contou com trabalhos relacionados e experimentos, embora diversos ensaios tenham sido realizados. Mudanças, tais como configuração do sistema, variações de características provocadas pelo ambiente ou características do cenário podem ser previstas pelo sistema de controle lógico *fuzzy*. Modificações das informações do conjunto de sensores *fuzzy* e inserção de métodos evolutivos é o que se pretende obter através do processo de evolução deste sistema. O mecanismo evolutivo melhora o desempenho do sistema e reduz a carga de trabalho humana.

Com a utilização de algoritmos genéticos em sistemas de robôs móveis inteligentes, tem sido possível apresentar um melhor desempenho no comportamento e na aquisição de novos conhecimentos [31] e [34]. Por outro lado, o processo de evolução do conjunto composto pelas regras *fuzzy* é considerado essencial diante de duas aproximações: melhora no conjunto da base de regras e através das funções de aproximação [12], [39], [55] e [76].

A formação de uma função de pertinência é normalmente caracterizada através de um conjunto de parâmetros. Por exemplo, são necessários três parâmetros para uma função de pertinência triangular. O ponto de pico decide qual deve ser a posição da função de pertinência no universo de discurso, enquanto que à esquerda e à direita, bases de regras definem seu alcance. A Figura 2.20 apresenta as funções de pertinência da evolução da arquitetura *fuzzy* de comportamento.

A utilização de algoritmos genéticos aperfeiçoa a função de pertinência através do desenvolvimento do conjunto associativo de parâmetros. O conjunto de regras que compõe a base *fuzzy* para ações primitivas é em sua maioria composto por regras simples e claras. Ao mesmo tempo, uma melhoria neste conjunto de regras é limitada. A evolução de funções pertinências concentra-se em sua maioria nos níveis inferiores da arquitetura. A base de regras é o núcleo do sistema *fuzzy*. Uma regra *fuzzy* é composta

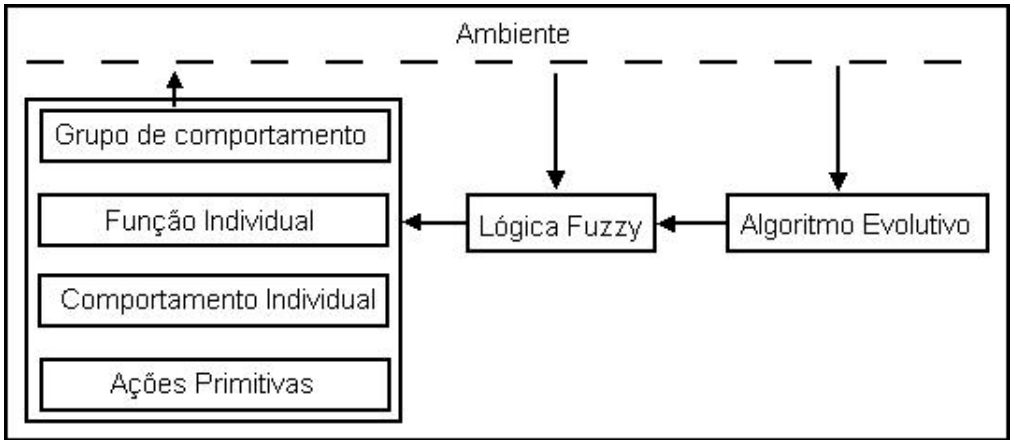

Figura 2.20: Evolução da arquitetura *fuzzy* de comportamento. Adaptado de [80].

de antecedentes, conseqüentes e relações *fuzzy*. Esta composição de partes de uma regra *fuzzy* é ao mesmo tempo conveniente para a evolução, sendo a escolha feita a partir dos requisitos e características do problema que está em questão. De acordo com a arquitetura *fuzzy* de comportamento, a otimização da base de regras é aplicada diante dos comportamentos que se encontram em mais alto nível, tais como, designação do papel que será exercido pelo agente.

Visto que a lógica *fuzzy* é aplicada através da estrutura de comportamento, e considerando uma evolução dos sensores *fuzzy*, o sistema de um modo geral também evolui. Assim, o princípio da evolução está focalizado nesses comportamentos *fuzzy*, que passam a ser considerados como recursos fundamentais no desempenho do sistema. É especificamente neste ponto onde existe um grande leque de opções para melhoria do sistema. Este sistema foi desenvolvido através de um simulador de futebol de robôs. A Figura 2.21 apresenta a arquitetura de comportamento do sistema de futebol de robôs.

A importância do ambiente de simulação é fundamental para validar e verificar o processo de execução dos algoritmos genéticos do sistema. No entanto, esta aplicação em um ambiente do mundo real torna-se complexa mesmo contendo um número pequeno de agentes, o processo de evolução é considerado demorado, sem mencionar o

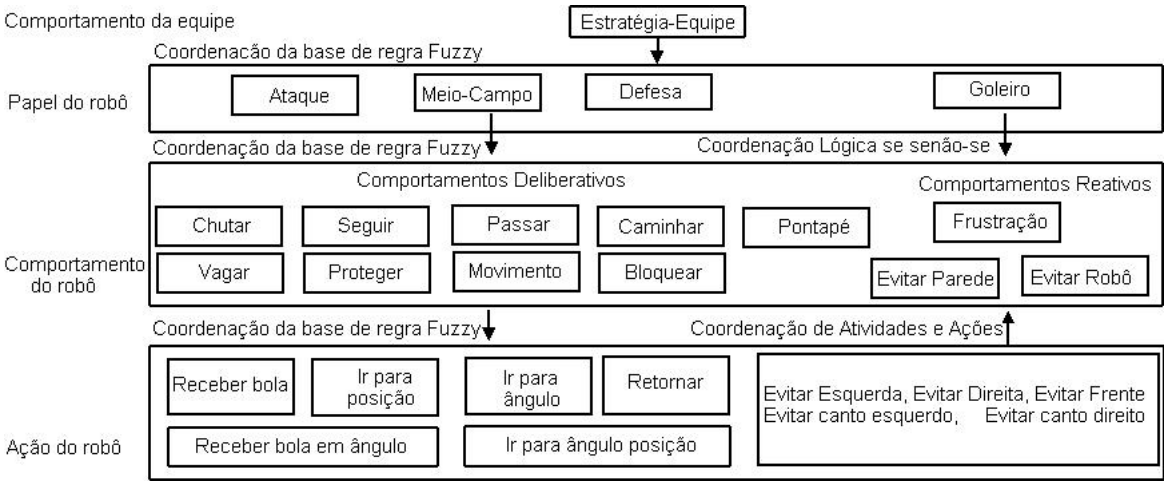

Figura 2.21: Arquitetura de comportamento do sistema de futebol de robôs. Adaptado de [80].

esforço computacional e limitações relacionadas à capacidade da bateria de cada robô.

O simulador desenvolvido pela equipe utilizou a ferramenta de desenvolvimento C++, usando uma biblioteca de visualização *OpenGL*. O ambiente de simulação permitiu modelar diversas condições do mundo real, tais como, cooperação entre agentes, interação com a bola e os limites do campo [80].

Este modelo de arquitetura, apesar de ser um modelo para a categoria de futebol de robôs simulado, e sendo que não é o caso do modelo de arquitetura que se vai propor, mesmo assim permite observar e implantar a forma de definição do conjunto de papéis que serão distribuídos entre os robôs. Esta arquitetura proposta apresenta, como principal vantagem, a definição das atribuições de cada robô no ambiente.

O modelo de arquitetura usado por [80] foi implementado em um simulador 3D e aqui a arquitetura proposta será implementada em hardware. Extraiu-se deste trabalho a forma de divisão dos papéis que são atribuídos aos agentes: goleiro, defensor e atacante.

Capítulo 3

Modelo da Arquitetura Proposta

3.1 Introdução

Neste capítulo são apresentadas as principais características que fazem parte do modelo da arquitetura híbrida para sistemas multiagentes, com aplicação ao futebol de robôs, proposta nesta dissertação. Além de descrever o modelo da arquitetura (ver Figura 3.1), também aborda-se cada etapa do processo de construção da arquitetura proposta separadamente.

3.2 Justificativas para uma arquitetura híbrida e distribuída

A vantagem em construir uma arquitetura híbrida e distribuída consiste no melhor aproveitamento dos recursos de hardware e software disponíveis, pois oferece maior poder de precisão com relação ao controle dos robôs durante a dinâmica do jogo.

Com relação aos recursos de hardware, pode-se destacar: a) utilização do conjunto de sensores que informa o posicionamento preciso do robô; b) oferta de um poder reativo imediato no instante em que é necessário desviar de obstáculos de forma precisa; c) precisão com relação ao controle dos atuadores. O controle, sendo realizado no próprio robô através do sistema embarcado de controle, faz com que a atualização do

vetor de velocidade, que corresponde à velocidade angular e linear, seja preciso no deslocamento do robô; precisão com relação ao reconhecimento e posicionamento da bola, utilizando a câmera embarcada. Com a utilização deste recurso, evitam-se problemas relacionados ao período de atraso decorrente do processamento da imagem capturada pelo sistema global e a transmissão da mensagem de comando ao robô indicando o instante futuro do ponto alvo, conforme [23]. A utilização da câmera embarcada indica a posição precisa em que a bola se encontra, mesmo estando em movimento. No instante em que o robô está próximo da bola, e sendo que cabe ao robô conduzi-la até à meta da equipe adversária, é necessário um controle preciso do robô e imediato com relação ao ponto onde a bola se encontra. Dessa forma, a utilização da câmera embarcada evita problemas desta natureza.

Por outro lado, dentre os recursos de software que se pode destacar estão: distribuição dos comportamentos através de camadas adicionais. Comportamentos de natureza reativa, tais como desviar de obstáculos, precisão no direcionamento do robô em relação à posição da bola, momento de chegada do robô na bola, controle da bola e acionamento do mecanismo de chute são implementados diretamente no robô. Dessa forma, evita-se a necessidade de processamento destas informações pela entidade central que, simultaneamente, apresenta uma margem de erro durante a dinâmica do jogo com relação ao posicionamento preciso dos robôs na ordem de 12%, conforme [15]. No entanto, decisões de ordem estratégica, que consistem no posicionamento dos robôs no ambiente, são realizadas pela entidade central, pois esta apresenta um sistema de captura global que oferece uma visão completa do ambiente. Dessa forma, é possível planejar jogadas futuras para os robôs, que se encontram no meio de campo ou no ataque, e evitar a realização de jogadas futuras pela equipe adversária, através do posicionamento em pontos estratégicos pelos robôs que fazem parte da defesa e pelo goleiro [24].

Para que sejam atendidas as especificações do sistema a ser desenvolvido, a arquitetura híbrida responsável pelo controle dos robôs é projetada com base na arquitetura clássica de Brooks, baseada em camadas hierárquicas [18], e pela arquitetura comportamental proposta por [73].

A principal razão pela qual este modelo de arquitetura pode ser apli-

cado ao futebol de robôs está na maneira com que combinam-se recursos de baixo desempenho computacional, sendo que é preciso obter o melhor desempenho possível para os robôs, no menor tempo possível em função da dinâmica de jogo. Dentre os recursos de hardware disponíveis, podem-se destacar: a câmera CCD com placa de aquisição de imagens com uma taxa de 29.97 *fps*; câmera embarcada com processamento local e um conjunto de sensores com necessidade de processamento imediato, sendo estes recursos responsáveis pelo processo de captura das informações do ambiente. Dessa forma, a integração destes recursos para capturar informações do ambiente e o imediato processamento destas informações em pontos distribuídos torna o sistema mais rápido, visto que não têm a necessidade de transmissão destas informações para uma entidade externa de processamento. Assim, evita-se a necessidade de utilizar o sistema de comunicação para enviar as informações para serem processadas e obter como resposta as mensagens de comandos para os atuadores dos robôs, tornando o sistema com um menor desempenho computacional e poder de precisão. Isto justifica a necessidade e a aplicação desta arquitetura. Portanto, uma arquitetura híbrida distribuída pode ser aplicada ao futebol de robôs para que seja possível obter um melhor desempenho, por parte dos robôs que estão atuando em um ambiente dinâmico.

3.3 Modelo da arquitetura híbrida e distribuída para sistemas multiagentes

Em um modelo de arquitetura de controle centralizado para futebol de robôs, todo o processamento das informações é realizado em uma entidade central que possui um sistema de visão global, responsável pela captura das imagens, pré-processamento destas informações e envio de comandos através do controle de velocidade para cada robô. Neste modelo o cérebro do sistema está concentrado no sistema de visão. Os robôs atuam no ambiente de acordo com as informações enviadas por este sistema.

Já em um modelo de arquitetura embarcada, cada robô é suficientemente autônomo ao ponto de poder tomar suas próprias decisões diante de informações

que foram coletadas pelos seus próprios sensores. Neste modelo de arquitetura, todo o controle, comportamento e ações do agente encontram-se no próprio robô. A única forma de obter informações de um meio colaborativo é através da cooperação entre agentes.

Com base nos modelos de arquiteturas que foram propostos e apresentados no capítulo anterior pelos diversos pesquisadores [73], [80] e [83], propõe-se um modelo que integre tanto o modelo de arquitetura remota sem informação quanto o modelo de arquitetura embarcada.

Neste novo modelo, pretende-se integrar o que cada modelo de arquitetura apresenta de melhor. Isto significa dizer que se propõe um sistema distribuído onde parte das informações, tais como, reconhecimento do ambiente, dos agentes que fazem parte da equipe, da equipe adversária e bola é realizado por um sistema de processamento de imagens em uma entidade central. Por outro lado, informações de natureza reativa, tais como, controlar a bola, vagar pelo ambiente, desviar de obstáculos e decisão de chute são realizados por um sistema de processamento embarcado.

O sistema embarcado recebe informações através do conjunto de sensores e da câmera, processa as informações capturadas no próprio sistema embarcado do robô e imediatamente envia comandos para seus atuadores. A vantagem em construir um modelo de arquitetura distribuída consiste no fato de que os recursos disponíveis são mais bem aproveitados. Pois o uso de forma independente, tanto da arquitetura embarcada quanto da arquitetura centralizada, não oferece a melhor solução para o problema. Dessa forma, é necessário que haja cooperação e negociação entre os sistemas, deliberativos e reativos. Assim, agentes individuais podem ignorar comandos vindos do sistema deliberativo quando o sistema reativo oferece uma melhor solução para o problema, por exemplo, quando o sistema de visão local indicar a posição em que a bola se encontra. Por outro lado, os agentes também podem ignorar comandos do sistema reativo quando o sistema de visão local não indicar a presença da bola e o sistema deliberativo indicar que o sentido de deslocamento do agente deve ser um sentido oposto ao deslocamento atual do robô. Diante de tudo isso, o novo modelo procura atender as reais necessidades dos objetivos, que é construir um modelo de arquitetura híbrida para sistemas multiagentes que possa ser aplicado ao futebol de robôs. A Figura 3.1 apresenta o modelo da arquitetura híbrida

e distribuída para sistemas multiagentes.

Como se pode observar, a arquitetura apresentada está dividida em duas partes (na divisão não é considerado o módulo de comunicação). A primeira parte da arquitetura é composta por um módulo, que é implementado em uma entidade central. Este módulo possui os níveis: estratégico e de ação. Por outro lado, a segunda parte da arquitetura é composta pelo módulo do agente que é implementado nos robôs (embarcado). Este módulo contém os níveis de comportamento e de execução. Além disso, tem-se um módulo de comunicação, que é responsável por ligar as duas partes da arquitetura.

3.4 Módulo da entidade central

O módulo da entidade central é composto por uma câmera e os níveis estratégico e de ação. O nível estratégico é composto por quatro módulos: sistema de exploração do ambiente, sistema de definição de estratégia, sistema de definição de trajetória e sistema de controle de navegação. Já o nível de ação corresponde à ação, que é atribuída aos robôs de forma independente. No módulo da entidade, é considerada a importância hierárquica da mesma forma como está sendo aplicado no módulo do agente. No entanto, é necessário que haja informações suficientes nos sistemas de exploração do ambiente, estratégico, de trajetória e de navegação para que possa ser definida qual deve ser a ação que é executada pelo agente.

3.4.1 Sistema de visão

O sistema de visão é composto por uma câmera que está conectada à entidade central. As imagens capturadas correspondem à única fonte de entrada de informações que são enviadas para o sistema de visão, alocado nesta entidade. A câmera é responsável pelo processo de captura de quadros que representam as imagens do ambiente onde estão inseridos os robôs que fazem parte da equipe, os robôs que fazem parte da equipe adversária e a bola. Estas imagens em forma de quadros são capturadas e processadas de maneira seqüencial, permitindo identificar cada objeto que está inserido no

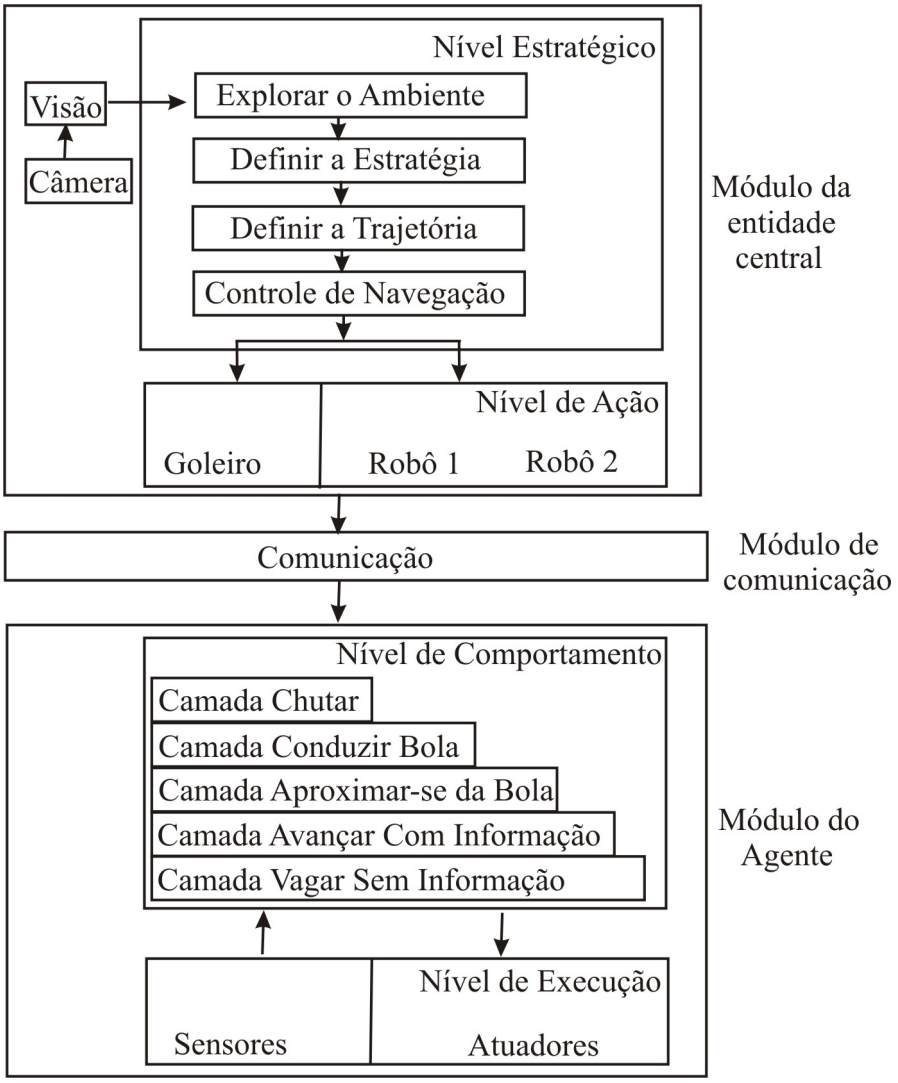

Figura 3.1: Arquitetura híbrida e distribuída para sistemas multiagentes.

ambiente, bem como a posição em que se encontra o objeto e a direção de cada robô que faz parte da própria equipe e da equipe adversária.

A entidade central é responsável pelo processamento das imagens que são capturadas com o auxílio da câmera e extrai de cada quadro todas as informações necessárias para o bom desempenho da equipe. É através das informações que estão presentes em cada imagem capturada que o sistema toma decisões. Estas decisões têm como objetivo auxiliar o sistema de navegação de cada agente na estratégia que será adotada pela equipe.

3.4.2 Sistema de exploração do ambiente

O sistema de exploração do ambiente é responsável pelo processo de identificação dos objetos, isto é, bola e jogadores, que estão inseridos no ambiente. Este sistema deve indicar qual é a posição exata de cada objeto no ambiente e também reconhecer todos os objetos que estão neste ambiente.

A posição em que se encontra um objeto é informada através da coordenada (x,y). É a partir da coordenada (x,y), que corresponde à posição da bola, que o sistema de exploração do ambiente define qual deve ser a trajetória que o robô deve executar. O sistema de exploração do ambiente, além de informar a posição dos objetos, deve também informar qual é o melhor caminho que o robô deve seguir.

É o sistema de exploração do ambiente que define qual será o robô que receberá a mensagem de comando de que deve direcionar-se em direção à bola, ou de que deve ocupar uma posição estratégica. Cabe ao sistema de exploração decidir qual deve ser o robô que receberá este comando. É também de responsabilidade do sistema de exploração perceber qual é o robô que está mais bem posicionado em campo, ou até mesmo direcionar um robô para que ele possa, em condições futuras, tornar-se um agente útil.

Desta forma, o sistema consiste em explorar todo o ambiente, informando a posição exata de cada objeto, bem como quais são os espaços que se encontram livres, para permitir um melhor desempenho por parte dos robôs e da dinâmica do jogo.

3.4.3 Sistema de definição de estratégia

O sistema de definição de estratégia para os robôs é considerado como sendo uma das etapas de grande importância dentro de uma aplicação voltada para o futebol de robôs. A definição da estratégia no controle dos robôs tanto pode ser empregada em robôs reativos quanto em robôs cooperativos. Nesta etapa do desenvolvimento, é necessário utilizar conhecimentos multidisciplinares, envolvendo as áreas de Inteligência Artificial e cooperação multiagentes.

Para modelar a estratégia a ser adotada neste trabalho, é utilizada uma máquina de estados finitos, que se refere aos agentes: robôs da própria equipe e robôs da equipe adversária.

Em uma partida de futebol de robôs, a bola pode estar em diversos estados, sendo que em um único instante, pode estar em apenas um estado. Pode-se citar como estados possíveis para a bola: livre, disputa, de posse de um robô da equipe, de posse de um robô da equipe adversária, gol. Estes estados citados dependem da posição dos robôs e também da posição em que a bola se encontra no ambiente.

A bola se encontra no estado livre quando em um raio pré-definido não se encontra nenhum robô, ou seja, nenhum robô está próximo o suficiente para que se possa definir que uma determinada equipe esteja de posse da bola.

A bola se encontra no estado de disputa quando se tem em um raio pré-definido ao menos um robô de cada equipe. A bola se encontra de posse de um robô da equipe quando em um raio pré-definido se tem apenas um único robô da equipe próximo da bola, ou robôs pertencentes à mesma equipe. A bola se encontra de posse de um robô da equipe adversária quando em um raio pré-definido se tem apenas um único robô da equipe adversária próximo da bola, ou robôs pertencentes à mesma equipe adversária. A bola se encontra no estado gol quando ela ultrapassar a linha da meta que corresponde à marcação do campo independente da equipe.

Tanto a bola como o(s) robô(s), independente da equipe a que pertençam, devem apresentar dois estados que possam identificar sua localização dentro do campo durante uma partida de futebol. Um dos estados é o eixo x, tendo como base as metas.

Pode-se definir três áreas: área de defesa, meio e ataque. O outro estado é o eixo y, tendo como base as laterais do campo. Pode-se também definir três áreas: lateral esquerda, meio e lateral direita. É necessário conhecer estes estados para cada objeto, pois é a partir destes estados que é definida a estratégia que cada robô deve realizar.

A partir do momento em que se conhece o estado de cada agente, é possível definir estratégias que identificam em que nível ou estado uma partida de futebol de robôs se encontra, com o auxílio de máquinas de estados finitos [57]. A transição de um estado em um instante t, para um estado em um instante t + 1, é realizada tendo como base o estado em que se encontra a bola e os robôs. Dentre as ações de transição de um estado para outro, podem-se citar: captura da bola, quando esta se encontra de posse da equipe adversária, ou mesmo em uma área considerada livre e é capturada pelo robô que passa a ter o controle ou a posse absoluta da bola; domínio de bola, que é o ato de deslocar-se de um lado a outro do campo guiando a bola; deslocamento, que consiste no movimento que o robô realiza durante uma partida de futebol para ocupar uma determinada posição estratégica no campo sem o domínio da bola; chute, movimento realizado pelo robô com o intuito de lançar a bola em direção ao gol da equipe adversária; seguir adversário, em que o deslocamento de um robô tem como princípio acompanhar os movimentos realizados por um outro robô da equipe adversária. Estas mudanças de estratégias de ação utilizam como base o goleiro, a defesa e o ataque.

A estratégia do goleiro consiste em intervir no movimento e deslocamento da bola quando ela se aproxima de sua meta. O movimento que impede que a bola ultrapasse a meta (a linha que separa o campo) é feito de forma coordenada, utilizando a velocidade com que a bola está se deslocando em sua direção e o ângulo de direção de projeção formado pela bola em um instante t e um instante t + 1. A estratégia do goleiro deve ser precisa o suficiente para saber se o ângulo de projeção que a bola está tomando oferece um perigo de gol, ou não. O goleiro deve levar em consideração o instante t, o instante t + 1, e a meta, traçar uma reta e ver se estes três pontos formam um ângulo correspondente a 180^0. Caso este ângulo obtido seja igual a 180^0 ou próximo deste valor, este oferece perigo de gol. Sendo assim, o goleiro deve se deslocar em direção a este ângulo, procurando estar na mesma linha que divide os instantes de deslocamento de projeção da

bola e a meta, e impedir que a bola continue a deslocar-se nesta direção. A Figura 3.2 apresenta a estratégia do goleiro.

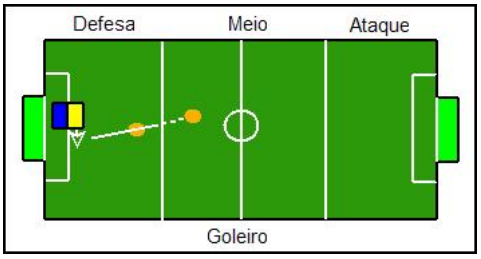

Figura 3.2: Estratégia do goleiro.

A estratégia da defesa utiliza como informação o estado em que se encontra a bola. Ela procura identificar em que posição ou região do campo a bola está posicionada. A área do campo é dividida em três segmentos conforme se pode observar na Figura 3.3.

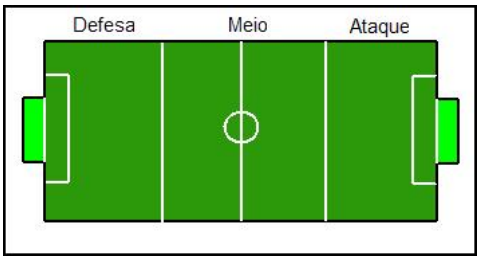

Figura 3.3: Áreas que dividem o campo.

A área de defesa que correspondente ao espaço do campo vai da área da meta da equipe até a linha que divide o campo em três partes; a área meio é representada pela região central do campo, e a área de ataque corresponde à parte do campo que está mais próxima da meta da equipe adversária. A estratégia da defesa consiste em saber de qual área para qual área a bola está se deslocando, ou seja, partindo do princípio de que a bola se encontra na área de ataque e de posse do time adversário em um instante t e em um instante t + 1, ela se encontra na área meio. Sendo assim, é traçada uma reta

que intercepta os dois pontos e estima um terceiro ponto na mesma seqüência em que a bola está se deslocando, passando pela mesma reta. A partir da informação deste terceiro ponto, é possível prever qual será a trajetória da bola. De posse desta informação, o robô que atua na defesa, deve se deslocar na direção deste ponto com o objetivo de recuperar a bola.

A estratégia do ataque pode se encontrar em dois estados distintos: o primeiro estado da estratégia do ataque está relacionado com a posição em que se encontra o alvo. Neste caso, a meta da equipe adversária, considerando que o robô se encontra atrás da bola, e estando de posse da mesma, isto significa dizer que, o robô está em uma posição em que ele pode conduzir a bola em direção à meta da equipe adversária. Sua função consiste em procurar atingir seu objetivo, que é fazer o gol. No entanto, se a bola está de posse da equipe adversária, o robô que tem como função atacar, acaba fazendo o papel de defensor, bloqueando ou impedindo a passagem do robô da equipe adversária. Desta forma, este posicionamento serve para fortalecer a defesa. O segundo estado da estratégia do ataque consiste no modo de condução: envolve o processo de rotação e deslocamento do robô em relação à posição em que ele ocupa em determinado instante no ambiente, isto é, dado que o robô se encontra na posição intermediária, bola, robô, meta da equipe adversária, ou qualquer posição que não forme um ângulo de 180^0, utilizando como base as posições: do robô, da bola e da meta da equipe adversária, é preciso corrigir sua posição para que ele possa conduzir a bola na direção da meta da equipe adversária. A Figura 3.4 apresenta os estados da estratégia de ataque.

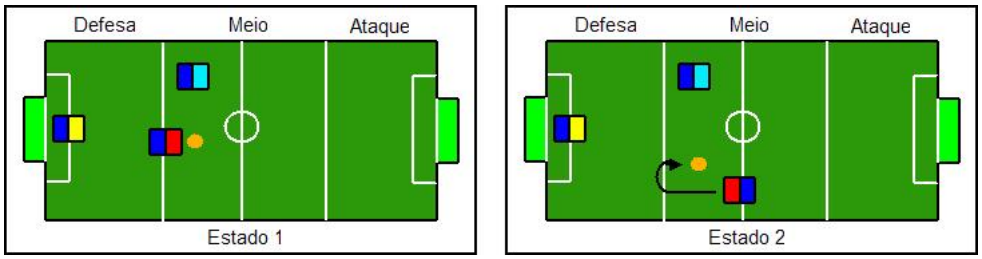

Figura 3.4: Estados da estratégia de ataque.

No entanto, cada estratégia (goleiro, defesa e ataque) deve levar em

consideração a posição em que o robô se encontra e qual deve ser a nova posição ou a posição para a qual ele deve se mover. Considerando o estado atual da bola, um vetor correspondente à velocidade da bola é obtido especificamente para esta posição ou para uma posição em que foi fornecida pelo método de decomposição de células aproximadas [57]. Partindo do princípio de que a bola não está de posse de um jogador da equipe, estando a bola em uma área livre ou de posse de um jogador da equipe adversária, então, pode-se aplicar este método. Este método tem como principal função construir uma trajetória para o robô, levando em consideração a possibilidade de encontrar outros objetos inseridos neste campo e tendo como princípio procurar não colidir com estes objetos, no caso, os robôs. A colisão de um robô da equipe com outro robô da equipe adversária pelas regras oficiais da Robocup é considerada como sendo falta [64].

3.4.4 Sistema de definição de trajetória

A definição de qual deve ser a trajetória que o robô deve executar é dada a partir do momento em que o sistema de exploração do ambiente, juntamente com o sistema de visão, indicar a posição exata em que a bola e os robôs se encontram. É a partir da informação que corresponde à coordenada da bola, que o sistema de definição de estratégia escolhe qual é o robô que receberá a mensagem de comando, indicando o caminho a ser seguido em direção à bola, ou até mesmo, qual deve ser a posição que o robô deve ocupar em situações onde o robô faz parte da defesa.

A partir da coordenada da bola, o sistema de definição de trajetória define a velocidade que o robô receberá para deslocar-se até a posição desejada. A velocidade definida pertence a um intervalo [0.0 a 0.7] em que o valor 0.0 corresponde à velocidade nula, e o valor 0.7 corresponde à maior velocidade que o robô receberá. Juntamente com a definição de velocidade, é necessário definir o deslocamento angular que o robô receberá. O deslocamento angular pertence a um intervalo de [-0.7 a 0.7]. O valor pertencente ao intervalo [-0.7 a -0.1] corresponde ao movimento girar à direita, e o valor pertencente ao intervalo [0.1 a 0.7], ao movimento girar à esquerda. Dessa forma, toda mensagem de comando enviada, é composta de dois valores (Velocidade linear, Veloci-

dade angular).

Após a definição do vetor de valores que corresponde à velocidade linear e angular, o sistema envia para o robô em forma de comando esta mensagem. Esta informação de comando é executada pelos atuadores do robô. Após o envio da primeira mensagem de comando e a execução deste comando pelos atuadores, é necessário que o sistema refaça novamente seu ciclo para uma possível correção de trajetória se necessário.

3.4.5 Sistema de navegação

O sistema de navegação é responsável pelo controle das mensagens de comando que correspondem ao vetor de velocidade, que é enviado para os atuadores do robô. Este sistema é composto pelo processo de inicialização e finalização do comando de execução dos atuadores. O sistema de navegação recebe o vetor de velocidade que representa a velocidade linear e angular do sistema de definição de trajetória, processa estas informações e envia as mensagens de comando, utilizando o sistema de comunicação para que sejam executadas pelos atuadores do robô. A estratégia do sistema de navegação é responsável pelos passos de atuação e controle de todas as mensagens de comando que são enviadas para serem executadas no robô. Os passos que correspondem à atuação do robô no ambiente convertem o valor do vetor de velocidade que corresponde à velocidade linear e angular que foi enviado pelo sistema de comunicação em uma força, que é atribuída a cada roda do robô. Este processo de conversão do valor que corresponde à velocidade linear e angular é implementado no sistema de controle do próprio robô. Dessa forma, cada mensagem de comando que representa a velocidade de rotação deve ser convertida pelo módulo *PWM (Pulse Width Modulation)* do próprio microcontrolador de cada motor.

De maneira semelhante ao sistema de navegação, o servo motor utilizado para controlar o mecanismo de chute, deve ser acionado. Para este motor, é necessário apenas enviar as mensagens no instante em que o mecanismo de chute deve ser acionado. O motor que controla o mecanismo de chute recebe apenas duas mensagens de comando. Uma corresponde ao momento de chute e a outra, de passe.

3.5 Módulo de comunicação

Para que haja troca de informações entre a entidade central e os robôs, faz-se necessário um módulo de comunicação. O conjunto de robôs possui uma unidade de comunicação sem fio para realizar a troca de mensagens. Desta forma, permite estabelecer a comunicação em uma determinada freqüência estabelecida, tanto para a recepção das mensagens de comandos por parte dos robôs quanto para transmissão de informações entre entidade central e robôs.

O próprio sistema operacional do robô é quem gerencia este meio de comunicação através de uma rede virtual do tipo *token ring*. Este sistema de comunicação transmite a uma taxa de 9.600 *bps*; também apresenta a característica de tolerância a falhas. Esta rede virtual, denominada de *EyeNet*, permite conectar diversos robôs com a mesma estação de comunicação. A comunicação é realizada através do envio de pacotes em forma de comandos da entidade central para os robôs. Estes pacotes de 8 *bits* possuem um *check-sum* para verificação de possíveis erros de transmissão. No entanto, este meio de comunicação não pode ser considerado como sendo um meio de comunicação confiável, pois ele não possui um protocolo de comunicação que permita a retransmissão de pacotes perdidos. Desta forma, a certeza de que um pacote foi enviado corretamente deve ser gerenciada através de uma camada que se encontra um nível acima [26].

As mensagens que correspondem aos comandos do vetor de velocidade (linear, angular), que são enviados da entidade central para os robôs por pacotes de 8 *bits*, são divididas em dois sub-pacotes de 4 *bits*. O primeiro sub-pacote de 4 *bits* é responsável pelo controle do motor esquerdo, que corresponde à velocidade linear, e o outro sub-pacote de 4 *bits* é responsável pelo controle do motor direito, que corresponde à velocidade angular.

O sistema operacional do robô disponibiliza as funções *RadioInit()* e *RadioTerm()* que correspondem aos processos de iniciar e finalizar a comunicação via rádio. Ao mesmo tempo também oferece as funções de *RadioSend(), RadioCheck()* e *RadioRecv()*, responsáveis pelo envio e recebimento dos pacotes.

3.6 Módulo do agente

Ao definir a arquitetura para os agentes locais, optou-se pela implementação de uma arquitetura puramente reativa, baseada na Arquitetura de *Subsumption*, proposta por Brooks na década de 80 [18]. Este modelo reativo apresenta muitas propriedades que são consideradas adequadas. Dentre as que se pode citar está a capacidade de execução de forma rápida e eliminar a necessidade de planejamento.

Os comportamentos foram montados conforme à Arquitetura de *Subsumption*, sendo que sua posição está de acordo com a importância na arquitetura e com a necessidade de fornecer informações para as camadas superiores. A arquitetura proposta contém cinco camadas que são: vagar sem informação, avançar com informação, aproximar-se da bola, conduzir bola e chutar, conforme se pode observar na Figura 3.1. As camadas que compõem a arquitetura do robô representam uma estrutura hierárquica, sendo que a camada que se encontra em mais alto nível assume a atividade que foi designada à camada inferior, quando uma determinada configuração dos sensores indicar uma situação que seja considerada propícia para sua atuação. Desse modo, as camadas de mais alto nível suprimem as saídas das camadas dos níveis inferiores.

Para definir os autômatos das camadas adotou-se uma máquina de estados finitos, a máquina de Moore [47], pois este modelo gera uma palavra que corresponde à saída, para cada estado do autômato definido. A máquina de Moore é um autômato finito determinístico, onde as saídas estão associadas aos estados.

3.6.1 Sistema de visão

O sistema de visão do robô é composto por uma câmera embarcada, cuja função principal é detectar a bola e as metas. O algoritmo a ser desenvolvido deve apresentar a característica de diferenciação das cores. Portando, é preciso identificar a bola, que possui a cor laranja, a meta da equipe adversária, que possui a cor amarela e a meta da própria equipe, que possui a cor azul. É necessário também que o algoritmo apresente a característica de alternância de cores no processo de identificação das metas, para uma possível troca de campo.

As imagens capturadas pela câmera encontram-se no formato RGB. Portanto, o algoritmo a ser desenvolvido para o sistema de visão deve apresentar a característica de diferenciação de cores para que se possa realmente capturar somente as cores que interessam. O *display* de LCD do robô apresenta as imagens capturadas pela câmera em tons de cinza. Sendo assim, é possível desenvolver um algoritmo que mostre em forma de histograma as imagens capturadas no *display* do robô. Outra característica que o sistema de visão deve apresentar é informar qual é a posição em que a bola se encontra no ambiente. Esta posição é calculada em função das coordenadas (x,y), que correspondem à distância que o robô se encontra da bola.

3.6.2 Sistema de sensores

O sistema de sensores utilizado pelo robô é composto por três sensores de distância, ficando um sensor na parte da frente, um sensor à direita e o outro sensor à esquerda do robô. A utilização deste conjunto de sensores se faz necessária para um maior poder de controle reativo do agente na navegação. Estes sensores são responsáveis por detectar obstáculos que possam surgir na dinâmica do jogo.

A informação que permite a identificação dos obstáculos são disponibilizadas pelos sensores que indicam onde um obstáculo pode estar a certa distância à frente, à esquerda ou à direita do robô. Assim, faz-se necessário classificar a informação da existência de um obstáculo em faixas úteis, sendo estas pertencentes a três valores aceitáveis, que podem ser definidos através da indicação de que um obstáculo pode estar: perto, médio, longe.

3.6.3 Sistema dos atuadores

O sistema de atuadores dos robôs é composto por dois motores de passo, que são controlados pelo módulo de *PWM* do próprio microcontrolador do robô. Cada motor está conectado a um *encoder*, que é responsável pela realimentação de velocidade, juntamente com um controlador PI do sistema operacional. Sendo assim, é possível realizar o controle de velocidade, que é exercido sobre cada roda. No entanto, os robôs

possuem duas rodas com características diferentes, sendo que, uma é responsável pelo controle angular e a outra pelo controle linear. Desta forma, não é possível ter um grau de certeza com relação ao controle de posição dos robôs, confiando unicamente nos dados coletados pelos sensores.

O sistema composto pelos atuadores é responsável pelo controle dos motores de passo para navegação no ambiente. O robô também possui dois servos motores, um para controle da câmera e outro para controlar o mecanismo de chute.

Para que ocorra o deslocamento dos robôs no ambiente, é necessário o envio de mensagens em forma de comando de maneira independente para cada motor de passo. Estes comandos servem para indicar ao robô qual deve ser sua trajetória, isto é, se ele deve seguir em frente, retornar, girar à direita, ou girar à esquerda. Desta forma, todo o controle do robô está concentrado nestes comandos. Ao mesmo tempo, é possível definir estes comandos através de um controle de velocidade, quer dizer, este controle pode ser definido como sendo uma velocidade lenta, normal ou rápida. Simultaneamente, pode-se definir um comando de parada. Sendo assim, fica definido o seguinte conjunto de comandos válidos para os motores de passo: frente, retornar, girar à direita, girar à esquerda, parar e mover-se com velocidade lenta, normal e rápida.

O servo motor utilizado para controlar a câmera apresenta duas posições: normal e baixo. Sendo assim, tem-se somente dois valores que são responsáveis por este controle: o primeiro leva em consideração se a câmera está na posição horizontal, considerado como sendo um posicionamento normal, permitindo que o robô possa visualizar a bola e o gol de forma ampla. Por outro lado, o posicionamento baixo é utilizado para que o robô possa visualizar a bola quando ela está próxima ou de posse do robô. Portanto, o conjunto de comandos que é utilizado para controlar o servo motor da câmera pode ser definido como sendo alto, baixo.

O servo motor utilizado para controlar o mecanismo de chute apresenta três posições: chutar, abaixar e passar. Dessa forma, este servo motor possui três valores que são responsáveis por este controle. O primeiro comportamento é utilizado para levantar o mecanismo de chute e serve para chutar a bola. O segundo comportamento é utilizado para abaixar o mecanismo de chute. Este comportamento se faz necessário

em função da possibilidade do mecanismo de chute, por algum motivo, por exemplo, um choque entre robôs, permanecer fora de sua posição que é estar abaixado. E o terceiro e último comportamento é utilizado em momentos onde o robô pretende passar a bola para um robô da mesma equipe. Neste terceiro comportamento, o servo motor atua com uma força intermediária. Sendo assim, o conjunto de comandos que é utilizado para controlar o servo motor de chute fica definido da seguinte forma: chutar, abaixar, passar.

3.6.4 Camada vagar sem informação

A camada vagar sem informação representa o comportamento de mais baixo nível do robô na arquitetura proposta. Este comportamento corresponde a vagar pelo ambiente desviando dos obstáculos que possam surgir, sendo obstáculos as laterais do campo e os robôs. A Tabela 3.1 apresenta os estados, condições e ações para a camada vagar sem informação.

O comportamento vagar sem informação emerge quando o robô não está de posse da informação que corresponde à posição em que a bola se encontra. Este comportamento é iniciado no "Estado Inicial", que corresponde ao E1. A troca de estados ocorre através das ações executadas pelo robô. A Figura 3.5 apresenta o autômato da camada vagar sem informação.

A partir do E1, o robô tem duas ações válidas, "Vagar" ou "Desviar". A ação "Vagar" é executada pelo robô quando ele não tem obstáculos na sua frente e, deste modo, ele passa para um "Estado Vagando", que corresponde ao E2. Já a ação "Desviar" é executada quando o robô tem obstáculo a sua frente e, deste modo, ele passa para um "Estado Desviar", que corresponde ao estado E3. O robô permanece no E2, através da ação "Vagar", até que ele encontra um obstáculo. A partir do instante em que o robô detecta um obstáculo, executa a ação "Desviar" e passa para o estado E3. A partir do E3, o robô pode retornar para o E2 através da ação "Vagar", ou ir para o "Estado à Esquerda", que corresponde ao E4, por meio da ação "Girar à esquerda", ou ir para o "Estado Ré", que corresponde ao E5, por intermédio da ação "Ré", ou ir para o "Estado à Direita", que corresponde ao E6, por intermédio da ação "Girar à direita". A ação "Ré" é executada

Tabela 3.1: Estados, condições e ações da camada vagar sem informação

Estado	**Condição**	**Ação**
E1 - Estado inicial	Não tem obstáculo	Vagar
E1 - Estado inicial	Tem obstáculo	Desviar
E2 - Estado vagando	Não tem obstáculo	Vagar
E2 - Estado vagando	Tem obstáculo	Desviar
E3 - Estado desviar	Não tem obstáculo	Vagar
E3 - Estado desviar	Tem obstáculo à esquerda	Girar à direita
E3 - Estado desviar	Tem obstáculo à direita	Girar à esquerda
E3 - Estado desviar	Tem obstáculo à dir./esq./frente	Ré
E4 - Estado à esquerda	Não tem obstáculo	Vagar
E4 - Estado à esquerda	Tem obstáculo à direita	Cont. girando à esquerda
E4 - Estado à esquerda	Tem obstáculo à esquerda	Girar à direita
E4 - Estado ré	Tem obstáculo à esquerda	Girar à direita
E5 - Estado ré	Tem obstáculo à dir./esq./frente	Continuar ré
E5 - Estado ré	Não tem obstáculo à direita	Girar à direita
E5 - Estado ré	Não tem obstáculo à à esquerda	Girar à esquerda
E6 - Estado à direita	Não tem obstáculo	Vagar
E6 - Estado à direita	Tem obstáculo à direita	Cont. girando à esquerda
E6 - Estado à direita	Tem obstáculo à esquerda	Girar à direita
E6 - Estado ré	Tem obstáculo à direita	Girar à esquerda

no instante em que os três sensores indicarem a presença de obstáculos. Desta forma, é necessário que o robô retorne. A partir do estado E4, o robô pode ir para o E2 pela ação “Vagar”, ou ir para o “Estado Ré”, que corresponde ao E5, através da ação “Girar à direita”, ou ir para o “Estado à Direita”, que corresponde ao E6, através da ação “Girar à direita”, ou permanece no E4 através da ação “Continuar girando à esquerda”. A partir do estado E5, o robô pode permanecer no mesmo estado com a execução da ação “Ré”,

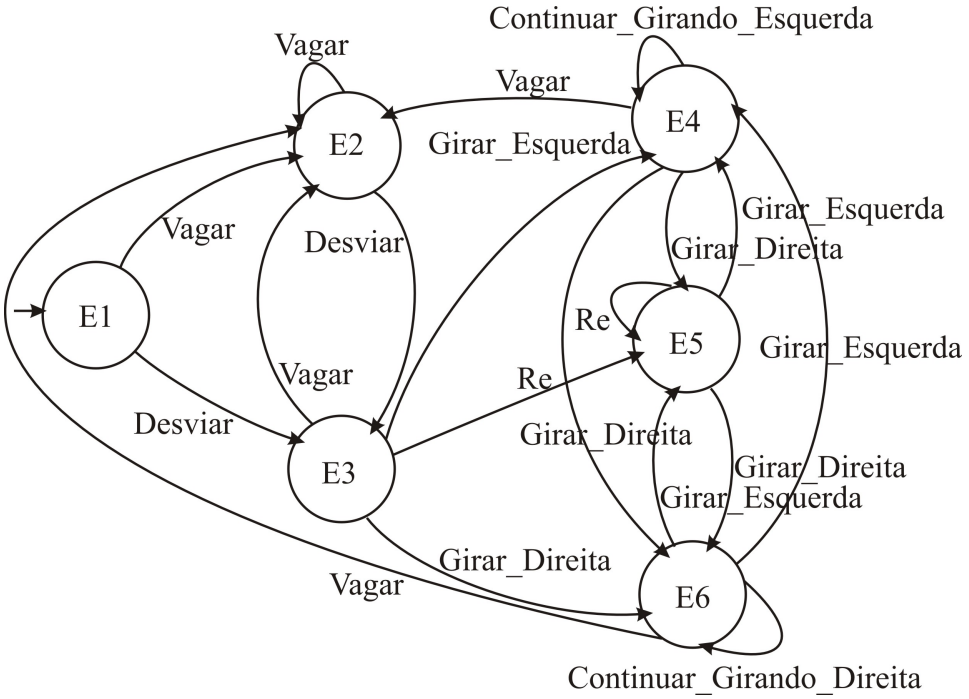

Figura 3.5: Autômato da camada vagar sem informação.

ou ir para o “Estado à Esquerda”, por intermédio da ação “Girar à esquerda”, ou ir para o “Estado à direita”, pela ação “Girar à direita. A partir do estado E6, o robô pode ir para o E2 através da ação “Vagar”, ou ir para o “Estado Ré”, por intermédio da ação “Girar à esquerda”, ou ir para o “Estado à Esquerda”, por intermédio da ação “Girar à esquerda”, ou permanece no E6 através da ação “Continuar girando à direita”. No instante em que o sistema de visão indicar a posição em que bola se encontra, a camada avançar com informação é ativada e sobrescreve a camada vagar sem informação.

3.6.5 Camada avançar com informação

A camada avançar com informação inicia sua atuação a partir do momento em que o sistema do robô está de posse da informação da posição em que a bola se encontra, ou seja, esta camada parte do princípio de que o robô está vendo a bola ou está sendo informado pelo sistema que está alocado na entidade central da posição em que a bola se encontra. A Tabela 3.2 apresenta os estados, condições e ações para a camada

avançar com informação e, a Figura 3.6 apresenta o autômato da camada avançar com informação.

Tabela 3.2: Estados, condições e ações da camada avançar com informação

Estado	**Condição**	**Ação**
E7 - Com objetivo	Ter objetivo	Avançar
E7 - Com objetivo	Ter objetivo	Calcular_Trajetória
E8 - Estado em Direção à bola	Ter objetivo	Ajustar_Trajetória
E8 - Estado em Direção à bola	Bola na esquerda	Girar à esquerda
E8 - Estado em Direção à bola	Bola na direita	Girar à direita
E9 - Estado à esquerda	Bola na esquerda	Permanece_Girando_Esquerda
E9 - Estado à esquerda	Bola na direita	Girar à direita
E9 - Estado à esquerda	Bola no centro	Avançar
E10 - Estado à direita	Bola na direita	Permanece_Girando_Direita
E10 - Estado à direita	Bola na esquerda	Girar à esquerda
E10 - Estado à direita	Bola no centro	Avançar

A partir da informação da posição da bola, o comportamento é iniciado no “Estado com Objetivo”, que corresponde ao E7. O estado E7 tem duas ações válidas, “Avançar” e “Calcular Trajetória”. A ação “Avançar” é executada no momento em que o robô não está de posse da informação da posição da bola, ou no momento em que ele perde a informação que corresponde à posição em que a bola se encontra. A perda da informação que corresponde à posição da bola pode ser causada por diversos motivos. Dentre estes motivos, pode-se destacar: a bola se movimenta saindo da frente do sistema de visão local e o sistema de visão global não está informando a posição em que a bola se encontra, ou um robô da equipe adversária passou em frente ao sistema de visão local. Já a ação “Calcular Trajetória”, é executada quando o robô vai ao encontro da bola. Com a aplicação desta ação, é possível mudar para o “Estado em Direção à bola”, que corresponde ao E8.

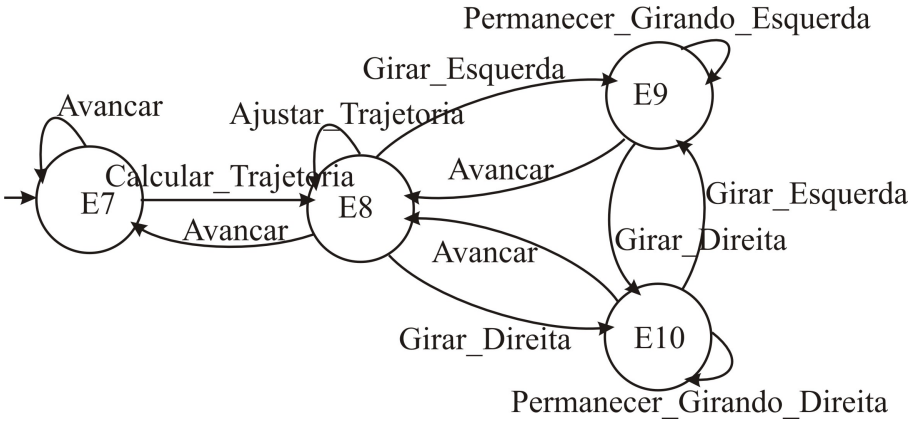

Figura 3.6: Autômato da camada avançar com informação.

A partir do estado E8, é possível se deslocar para três novos estados, o "Estado à Direita", que corresponde ao E10, com a execução da ação "Girar à direita", o "Estado à Esquerda", que corresponde ao E9, executando a ação "Girar à esquerda", ou retornar para o E7 através da ação "Avançar", ou permanecer no E8, pela ação "Ajustar trajetória". A partir do estado E9, o robô pode permanecer no mesmo estado por meio da ação "Permanecer girando à esquerda", que corresponde a continuar girando à esquerda, ou ir para o estado E10, através da ação "Girar à direita", ou retornar para o estado E8, por intermédio da ação "Avançar". A partir do estado E10, o robô pode permanecer no mesmo estado através da ação "Permanecer girando à direita", que corresponde a continuar girando à direita, ou ir para o estado E9, pela ação "Girar à esquerda", ou retornar para o estado E8, através da ação "Avançar". A camada avançar com informação é executada até o momento em que o robô possui uma trajetória linear com velocidade angular zero, que o conduz em direção a bola e, desta forma, é ativada a camada aproximar-se da bola.

3.6.6 Camada aproximar-se da bola

A camada aproximar-se da bola é executada a partir do momento em que o robô está se aproximando da bola. É esta camada a responsável pelo momento

de aproximação do robô com relação à posição em que a bola se encontra. A camada aproximar-se da bola se faz necessária em função da importância que envolve o instante de transição do estado em que o robô não está de posse da bola e o instante em que o robô está de posse da bola. Esta transição de estado deve ser o mais suave possível, visto que, a chegada do robô na bola em uma velocidade constante pode resultar em um toque na bola de forma que a mesma seja conduzida para uma posição distante do robô.

O processo de redução de velocidade somente é aplicado para o robô que apresenta o comportamento de atacante. Sendo assim, este comportamento é acionado somente a partir do instante em que a entidade central informa ao robô, que ele está ocupando uma posição no ambiente a partir da linha que divide o campo. Por isso, o robô passa a exercer o comportamento de atacante e se faz necessário aplicar o comportamento de redução de velocidade para que o robô possa permanecer com a posse da bola. Por outro lado, o comportamento de redução de velocidade não é aplicado em caso do robô encontrar-se em uma posição no ambiente antes da linha que divide o campo, pois esta posição no ambiente é considerada como sendo uma área de defesa e, desta forma, é necessário que o robô apenas retire a bola desta área. A Tabela 3.3 apresenta os estados, condições e ações para a camada aproximar-se da bola.

Tabela 3.3: Estados, condições e ações da camada aproximar-se da bola

Estado	**Condição**	**Ação**
E11 - Estado Longe da Bola	Robô recebe a posição da bola	Avançar
E11 - Estado Longe da Bola	Robô não está de posse da bola	Avançar_Lento
E11 - Estado Longe da Bola	Robô está de posse da bola	Avançar_Rápido
E12 - Estado Intermediário à Bola	Robô recebe a posição da bola	Avançar
E12 - Estado Intermediário à Bola	Robô não está de posse da bola	Avançar_Lento
E12 - Estado Intermediário à Bola	Robô está de posse da bola	Avançar_Rápido
E13 - Estado Próximo da Bola	Robô recebe a posição da bola	Avançar
E13 - Estado Próximo da Bola	Robô não está de posse da bola	Avançar_Lento

O autômato da camada aproximar-se da bola possui três estados, que são: “Estado Longe da Bola”, “Estado Intermediário à Bola” e “Estado Próximo da Bola”, conforme pode ser observado na Figura 3.7, que apresenta o autômato da camada aproximar-se da bola.

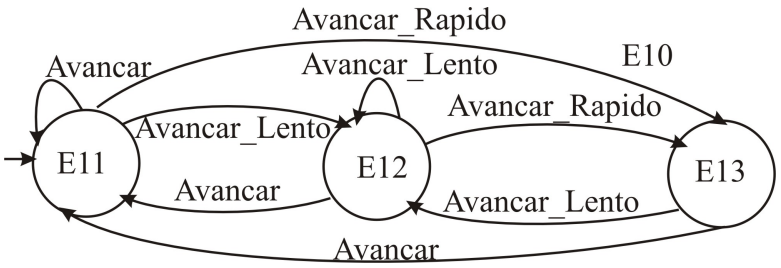

Figura 3.7: Autômato da camada aproximar-se da bola.

O “Estado Longe da Bola”, que corresponde ao E11, consiste na execução da ação “Avançar” até o momento em que o robô é informado pelo sistema da entidade central de sua posição no ambiente. A partir da informação que corresponde ao comportamento (ataque, defesa) a ser executado pelo robô, o estado E11 apresenta duas ações válidas: “Avançar com velocidade lenta” ou “Avançar com velocidade rápida”. A ação “Avançar com velocidade lenta” é executada a partir do instante em que o robô assume o comportamento de atacante e o sistema de visão local indica uma posição considerada como sendo próxima da bola. Desta forma, é preciso aplicar o comportamento de redução de velocidade linear para que o robô possa se aproximar da bola de forma suave. Esta ação é executada até o momento em que o sistema de visão local indicar novamente uma nova posição para o robô como sendo bem próximo da bola. Por outro lado, a ação “Avançar com velocidade rápida” é executada no instante em que o robô assume o comportamento de defensor. Já o “Estado Intermediário à Bola”, que corresponde ao E12, consiste em permanecer vagando com velocidade linear lenta, ou retornar à velocidade linear constante, através da ação “Avançar”. O “Estado Próximo da Bola”, que corresponde ao E13, consiste em permanecer, avançando com velocidade linear constante, ou retornar à velocidade linear lenta, através da ação “Avançar com velocidade lenta”.

3.6.7 Camada conduzir bola

A camada conduzir bola somente é executada a partir do momento em que o robô está de posse da bola. No entanto, esta camada é responsável pelo controle da bola, ou seja, estando o robô de posse da bola e necessitando conduzi-la, esta camada é acionada. É esta camada a responsável pelo controle mais fino do robô. Isto significa dizer que é a camada conduzir bola que informa quando a bola está em uma posição mais à direita ou mais à esquerda do mecanismo de chute do robô. A Tabela 3.4 apresenta os estados, condições e ações para a camada conduzir bola.

Tabela 3.4: Estados, condições e ações da camada conduzir bola

Estado	**Condição**	**Ação**
E14 - Estado seguir em frente	bola no centro	avançar
E14 - Estado seguir em frente	bola à direita	girar à direita
E14 - Estado seguir em frente	bola à esquerda	girar à esquerda
E15 - Estado seguir à esquerda	bola no centro	avançar
E15 - Estado seguir à esquerda	bola à direita	girar à direita
E15 - Estado seguir à esquerda	bola à esquerda	girar à esquerda
E16 - Estado seguir à direita	bola no centro	avançar
E16 - Estado seguir à direita	bola à direita	girar à direita
E16 - Estado seguir à direita	bola à esquerda	girar à esquerda

O autômato da camada conduzir bola possui três estados que são: “Estado Seguir em Frente”, “Estado Seguir à Esquerda” e “Estado Seguir à Direita”, conforme pode ser observado na Figura 3.8, que apresenta o autômato da camada conduzir bola.

O “Estado Seguir em Frente”, que corresponde ao E14, consiste de três ações válidas: “Avançar”, “Girar à direita” e “Girar à esquerda”. A ação “Avançar” é executada toda vez que a bola se encontra em uma posição considerada central ao mecanismo de chute do robô. Por outro lado, a ação “Girar à direita” é executada toda vez que a bola

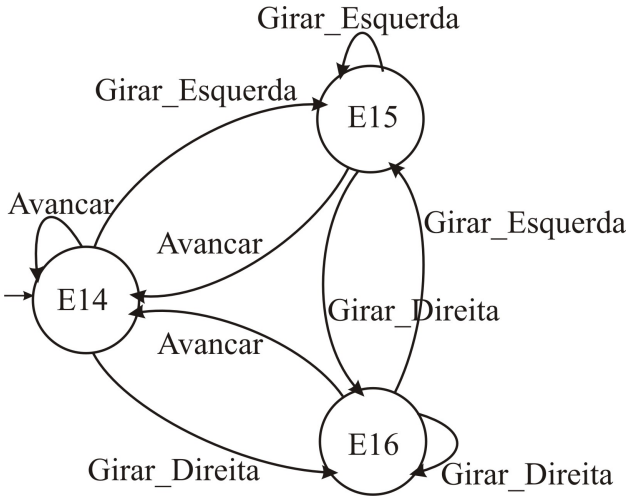

Figura 3.8: Autômato da camada conduzir bola.

se encontra em uma posição considerada à direita do mecanismo de chute. Através desta ação, o robô é atualizado para o "Estado Seguir à Direita". Da mesma forma, a ação "Girar à esquerda" é executada toda vez que a bola se encontra em uma posição considerada à esquerda do mecanismo de chute. Através desta ação, o robô é atualizado para o "Estado Seguir à Esquerda".

A partir do "Estado Seguir à Esquerda", que corresponde ao E15, também têm-se três ações válidas: "Avançar", "Girar à direita" e "Girar à esquerda". A ação "Avançar" é executada toda vez que a bola retorna a uma posição considerada central ao mecanismo de chute do robô e, desta forma, o robô é atualizado para o "Estado Seguir em Frente". Por outro lado, a ação "Girar à direita" é executada toda vez que a bola se encontra em uma posição considerada à direita do mecanismo de chute. Através desta ação, o robô é atualizado para o "Estado Seguir à Direita". Já a ação "Girar à esquerda", é executada toda vez que é necessário permanecer girando à esquerda, sendo que o robô fica no mesmo estado.

A partir do "Estado Seguir à Direita", que corresponde ao E16, também têm-se três ações válidas: "Avançar", "Girar à direita" e "Girar à esquerda". A ação "Avançar" é executada toda vez que a bola retorna a uma posição considerada central ao

mecanismo de chute do robô e, desta forma, o robô é atualizado para o “Estado Seguir em Frente”. Por outro lado, a ação “Girar à esquerda” é executada toda vez que a bola se encontra em uma posição considerada à esquerda do mecanismo de chute. Através desta ação, o robô é atualizado para o “Estado Seguir à Esquerda”. Já a ação “Girar à direita”, é executada toda vez que é necessário permanecer girando à direita, sendo que o robô permanece no mesmo estado.

3.6.8 Camada chutar

A camada chutar somente é executada a partir do momento em que o robô está de posse da bola. No entanto, esta camada é responsável pelo controle do mecanismo de chute, quer dizer, que estando o robô de posse da bola e em condições de chutar, esta camada é acionada. A Tabela 3.5 apresenta os estados, condições e ações para a camada chutar bola.

Tabela 3.5: Estados, condições e ações da camada chutar

Estado	**Condição**	**Ação**
E17 - Estado abaixado	Permanecer abaixado	Abaixar
E17 - Estado abaixado	Chute	Chutar bola
E17 - Estado abaixado	Passe	Passar bola
E18 - Estado passe	Aguarda um instante	Voltar
E19 - Estado levantado	Aguarda um instante	Voltar

O autômato da camada chutar possui três estados: “Estado Abaixado”, “Estado Passe” e “Estado Levantado”, conforme pode ser observado na Figura 3.9, que apresenta o autômato da camada chutar.

O primeiro estado “Abaixado”, que corresponde ao E17, apresenta três ações válidas: “Abaixar”, “Passar bola” e “Chutar bola”. A ação “Abaixar” consiste em permanecer com o mecanismo de chute em uma posição abaixada. Esta ação é executada para verificar se o mecanismo de chute encontra-se nesta posição. Caso o mecanismo de

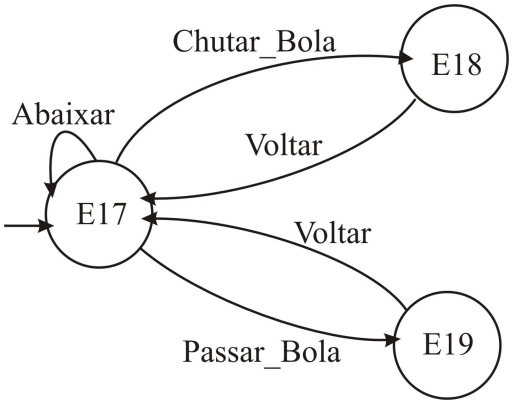

Figura 3.9: Autômato da camada chutar.

chute não esteja na posição estabelecida, esta ação é executada para que o mecanismo fique posicionado. Este estado do mecanismo de chute permanece nesta posição quando: *a)* o robô não está de posse da bola; *b)* quando não se encontra em condições de passar a bola para o seu companheiro de equipe; *c)* quando não se encontra em condições de chutar a bola na meta da equipe adversária. Por outro lado, a ação "Chutar bola" é executada somente quando o robô se encontra em condições de chutar a bola na meta da equipe adversária. Este comportamento somente é acionado quando: o sistema de visão local indicar uma posição próxima e alinhada à meta da equipe adversária e, assim, desta forma o comportamento chutar é executado, ou quando o sistema de visão global, indicar que este comportamento deve ser acionado. A ação "Chutar bola" leva o mecanismo de chute para o "Estado Levantado", que corresponde ao E18. Este estado aguarda um instante e executa a ação "voltar", que retorna o controle do mecanismo de chute para o "Estado Abaixado". Da mesma forma, a ação "Passar bola" leva o mecanismo de chute para o "Estado Passe", que corresponde ao E19. Este estado aguarda um instante e executa a ação "Voltar", que retorna o controle do mecanismo de chute para o "Estado Abaixado".

Capítulo 4

Implementação da Arquitetura

4.1 Introdução

A estratégia utilizada para a realização deste trabalho teve como uma primeira etapa a divisão dos recursos que compõe este sistema. Dentre os recursos utilizados para a implementação desta arquitetura, tem-se: um microcomputador com sistema operacional Windows XP, onde foi implantado o módulo da entidade central que é responsável pelo controle do sistema de visão, pelo nível estratégico, nível de ação e pelo sistema de comunicação; o sistema de visão composto por uma câmera de vídeo colorida JVC com saída S-VHS. As imagens capturadas são digitalizadas através de uma placa de captura PCI da *PixelView PV-TV304P*. A imagem digitalizada apresenta uma resolução de 640 x 480 *pixels* em formato RGB (*Red, Green, Blue*), com taxa de captura de 29.97*fps*. Três robôs *SoccerBot (EyeBot)* contendo sistema de visão local, sensores e atuadores. A seguir, são apresentadas as estratégias utilizadas bem como os resultados que foram obtidos no desenvolvimento desta aplicação, utilizando as ferramentas de desenvolvimento Dev-C++ e a biblioteca de visualização *OpenCV*.

4.2 Implementação da entidade central

O sistema que corresponde ao módulo da entidade central é responsável pelo controle do sistema de visão e dos níveis estratégico e de ação. Cabe ao sistema alocado na entidade central explorar, planejar, definir e deliberar a estratégia que será executada pelos robôs. A definição de qual deve ser a estratégia que o robô deve executar é dada a partir do momento em que o sistema de exploração do ambiente, juntamente com o sistema de visão global, indicar a posição em que a bola e os robôs se encontram no ambiente. É a partir da informação que corresponde à coordenada da posição da bola, que o sistema de definição de estratégia, define uma trajetória que será executada pelo robô. Após a definição da estratégia, o robô receberá a mensagem de comando que indica a trajetória a ser seguida em direção à bola, ou até mesmo, qual deve ser a posição em que o agente deve ocupar em situações onde ele faz parte da equipe de defesa. No módulo da entidade central também estão comandos que correspondem ao início (iniciar e reiniciar a partida), comandos de parada (gol, faltas, cobrança de faltas, pênaltis e finalizar a partida).

4.2.1 Sistema de visão

O sistema de visão utilizado neste módulo é um sistema de captura global, que recebe um conjunto de *pixels* que representam uma imagem capturada do campo, com os robôs e a bola, que estão em movimento em um dado instante. Este processo de captura é realizado por uma câmera que está posicionada a uma distância de 4m acima do campo, tendo este sistema de visão a função de capturar a posição dos robôs e da bola [64]. Os robôs e a bola estão sendo representados na Figura 4.1 que mostra o quadro do ambiente capturado pelo sistema de visão global para o processamento e tratamento das imagens.

Neste quadro, pode-se observar o campo, a bola e quatro objetos que representam os robôs. Cada objeto que representa um robô é identificado através de duas tarjas coloridas, uma para representar a cor da equipe e outra para representar a cor que identifica cada jogador individualmente. Pode-se também observar que o nível de luminosidade que incide sobre o campo é diferenciado, isto é, o centro do campo recebe maior

Figura 4.1: Representação do ambiente.

índice de luminosidade. No entanto, próximo às margens do campo, este nível de luminosidade vai diminuindo. Isto ocorre em função da fonte luminosa estar localizada em apenas um ponto. Dessa forma, é preciso utilizar um sistema de iluminação de forma uniformemente distribuída para que seja possível minimizar este tipo de problema.

4.2.2 Implementação do sistema de exploração do ambiente

O processo de exploração do ambiente consiste em localizar cada objeto que está inserido no ambiente. Para localizar cada objeto o sistema desenvolvido recebe um quadro em formato RGB, converte esta imagem para o formato HLS, separa esta imagem em três canais, CanalH, CanalL, CanalS, onde o CanalH representa o tom da imagem. O CanalL representa o nível de luminosidade e o CanalS representa o nível de saturação. Após a divisão da imagem em três canais, é possível aplicar um limiar de corte, que é responsável pelo intervalo aceito no processo de classificação de um determinado objeto. Dependendo da posição em que o objeto se encontra, o nível de luminosidade que incide sobre este objeto é variável. Sendo assim, o sistema aceita como limiar de corte uma faixa de valores que corresponde ao intervalo aceito no processo de classificação do objeto. Este processo de escolha do limiar de corte é realizado no processo de configuração do sistema através de um clique de mouse sobre a cor do objeto

que se pretende classificar e sendo feito para todos os objetos. Após ter sido selecionada, a cor que corresponde a um objeto determinado, o algoritmo realiza o processo de erosão e dilatação dos objetos. O processo de erosão ao redor dos objetos é necessário para que seja possível remover pontos isolados que representam falsos objetos, já a dilatação serve para uma melhor visualização do objeto na imagem.

Depois do processo de calibração da câmera, responsável pela definição dos limiares, o algoritmo inicia o processo de reconhecimento dos objetos. Este reconhecimento é realizado em tempo real, utilizando os limiares definidos no processo de calibração. O processo que envolve o reconhecimento e a posição do objeto bola é apresentado na Figura 4.2, que mostra o quadro original capturado à esquerda e o objeto que corresponde à bola encontrada no quadro do lado direito.

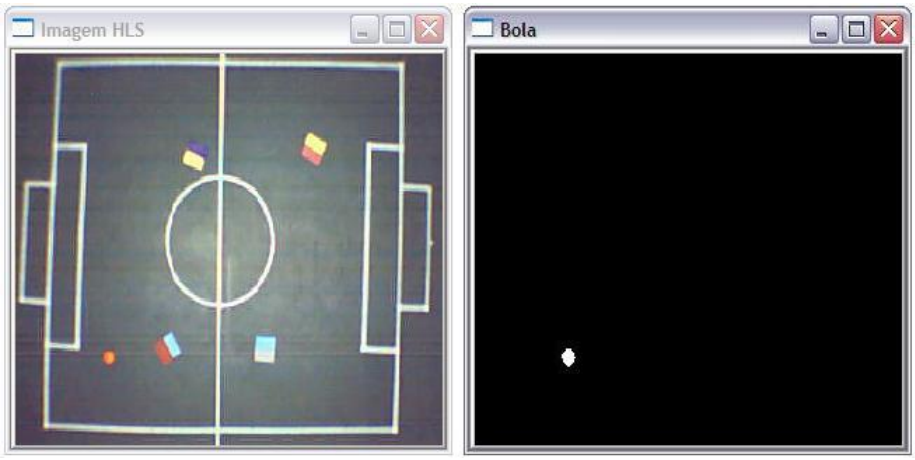

Figura 4.2: Captura da bola.

Pode-se observar que o sistema desenvolvido detectou o objeto desejado, bem como sua posição. Para este caso, em específico a bola, percebe-se também que o formato do objeto encontrado é maior do que o objeto original. Isto não é problema, visto que o sistema só precisa saber a posição em que o objeto se encontra no ambiente. É a posição do objeto que corresponde à coordenada (x,y), que será repassada através de uma mensagem de comando para os robôs e não o tamanho do objeto.

O processo de localização dos robôs que correspondem aos jogadores

da equipe de cor azul é semelhante ao processo de reconhecimento da bola. Este processo envolve o reconhecimento dos jogadores que fazem parte desta equipe, sendo realizado através da aplicação de dois sub-processos de calibração da câmera, que correspondem à definição do limiar de corte, aceito para o reconhecimento dos robôs na imagem. A definição desses limiares de corte é semelhante ao processo descrito para localização da bola. No entanto, o reconhecimento de cada jogador da equipe azul é realizado em duas etapas.

A primeira etapa consiste na verificação de toda a imagem para saber quantos objetos fazem parte desta equipe, ou seja, todos os objetos que possuem a cor azul. O sistema foi desenvolvido assumindo que uma partida de futebol de robôs pode ser jogada independente do número de jogadores, ou seja, pode-se jogar uma partida com apenas um robô de cada equipe, ou mais jogadores. Desta forma, não se tem a necessidade de ajustar o sistema desenvolvido de acordo com a quantidade de jogadores. Neste caso, são encontrados dois objetos. A Figura 4.3 mostra o resultado do primeiro processo na identificação dos objetos que correspondem à equipe de cor azul.

Figura 4.3: Jogadores da equipe de cor azul.

A partir do momento em que foram localizados, na imagem original, todos os possíveis valores que correspondem a um objeto de cor azul, o algoritmo desenvolvido seleciona uma mini-região ao redor de cada objeto de cor azul e passa a realizar o

processamento apenas nestas mini-regiões. O processamento acontece apenas nas mini-regiões em função da necessidade de melhorar o desempenho do sistema.

A segunda etapa consiste em verificar a cor de cada jogador da equipe. Este mecanismo é realizado apenas nas mini-regiões. Com isso, ganha-se tempo de processamento, pois a cor que identifica cada jogador, que faz parte de uma determinada equipe, está próxima da cor que define a equipe. Por isso, processar apenas uma pequena parte da imagem é o suficiente. O tamanho da mini-região é definido com base no tamanho dos objetos. Neste caso, tem-se um jogador que possui a identificação azul-vermelho, e outro robô com a identificação azul-cinza. A Figura 4.4 mostra o resultado da aplicação da segunda etapa de processamento na identificação individual de cada jogador, que compõe uma determinada equipe.

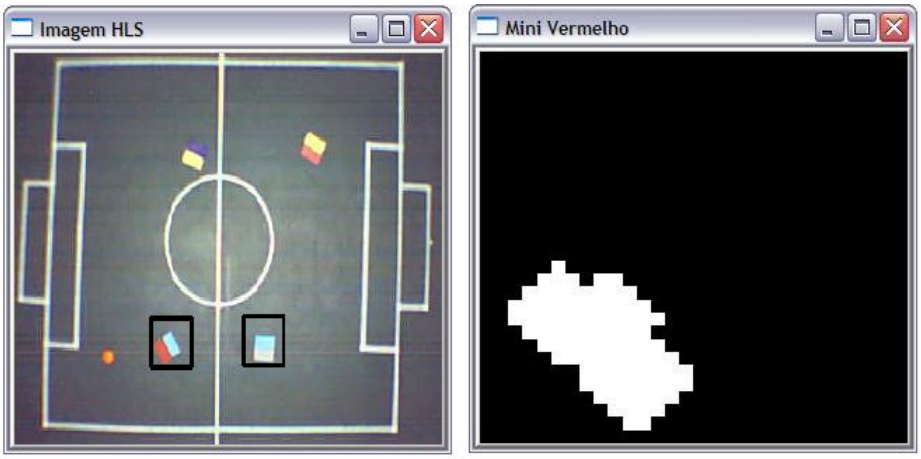

Figura 4.4: Identificação do jogador.

Observa-se que a região marcada ao redor de cada jogador da equipe azul é o suficiente para identificar o jogador. Assim o resultado do segundo processo, que é apresentado na figura à direita, é o processamento da mini-região da imagem marcada à esquerda. Verifica-se também que a imagem à direita representa apenas a cor vermelha do jogador azul-vermelho. Este mesmo processo é aplicado para identificar o jogador azul-cinza.

Após terem sido identificados, todos os jogadores de cada equipe com

a aplicação da primeira regra de processamento, aplica-se a segunda regra de processamento, que identifica individualmente cada jogador que compõe a equipe. Na seqüência, é necessário unir estas informações para que se possa identificar o lado que corresponde à parte da frente do robô. Neste caso, é preciso identificar a direção apenas dos robôs da própria equipe, já que o objeto bola é identificado através de seu deslocamento no ambiente. Quanto aos objetos que correspondem aos robôs da equipe adversária, identifica-se apenas sua posição para evitar possíveis colisões.

O processo que reconhece a parte da frente do robô é composto pela união das duas partes que identificam cada agente, ou seja, a cor da equipe e a cor individual. De posse da informação da cor da equipe e da cor individual, o sistema desenvolvido traça uma linha que indica a direção do robô, conforme apresentado pela Figura 4.5, que mostra a linha que identifica a parte da frente do jogador.

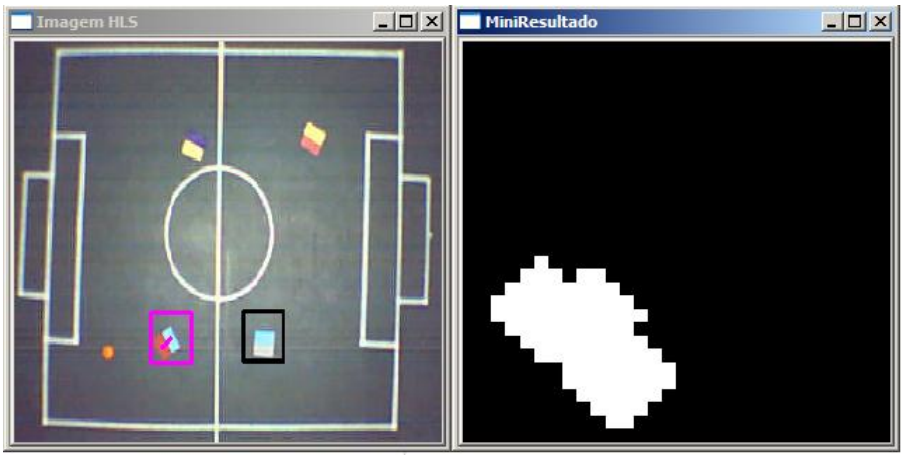

Figura 4.5: Linha que identifica a parte da frente do jogador.

O sistema desenvolvido utiliza o centro da imagem que representa a cor da equipe como sendo o ponto base. A partir deste ponto, traça uma linha em direção a um segundo ponto que identifica a cor individual.

De posse da informação de direção, o sistema transmite ao robô informações necessárias para o processo de correção de sua posição, isto é, o robô é informado da posição em que se encontra a bola no campo bem como qual deve ser sua

postura (defensor ou atacante) em relação ao decorrer da partida. Neste caso, como comportamento têm-se duas situações: defender ou atacar, sendo que cada situação é parte da estratégia da equipe.

4.2.3 Implementação do sistema de definição de estratégia

A implementação do módulo de estratégia consiste na definição de qual deve ser o comportamento que é executado pelo robô. Para a aplicação em questão, e considerando que a equipe possui três robôs *EyeBot*, adotaram-se dois comportamentos: um para o goleiro e outro para os jogadores de linha. Para os jogadores de linha, tanto eles podem assumir o comportamento de defesa quanto de ataque.

A definição da estratégia a ser adotada utiliza como base as informações fornecidas pelo sistema de visão global, que apresenta o ambiente (campo), e pelo sistema de exploração do ambiente, que apresenta o estado de cada agente (robôs e bola) no ambiente. Nessa implementação, foi utilizada uma máquina de estados, sendo que os nós representam os estados em que cada agente pode se encontrar no ambiente, e a transição entre os estados ocorre através das ações executadas na dinâmica do jogo.

A identificação dos objetos: para definir uma estratégia é necessário identificar todos os objetos inseridos no ambiente, e transformar a informação que corresponde à posição do objeto em uma nova informação que corresponde ao vetor de distância que está associado ao estado de cada agente. Dessa forma, um conjunto de estados e transições (ações) é definido. A Tabela 4.1 apresenta os estados em que a bola pode se encontrar no ambiente, bem como, a descrição e as ações que são executadas pelos robôs com relação à bola.

A atribuição do comportamento: a partir do instante em que o sistema de definição de estratégia possui os valores das coordenadas (x,y) que correspondem à posição da bola e dos robôs da equipe é definido, pelo processo de planejamento, o comportamento que corresponde a um conjunto de ações que podem ser executadas para cada agente da equipe. Dentre os comportamentos têm-se: o do goleiro, que é fixo para um robô da equipe, o de defesa e de ataque, que podem ser alternados entre os robôs que

Tabela 4.1: Estados em que a bola pode se encontrar no ambiente.

Estado da bola	**Descrição dos robôs**	**Ações dos robôs**
Livre	Nenhum robô está próximo da bola	Em direção à bola
Disputa	Um ou mais robô(s) de cada equipe	Capturar bola
De posse da equipe	Um único robô da equipe	Conduzir bola
De posse do adversário	Um único robô da equipe adversária	Bloquear bola
Meta	O robô está próximo da meta adversária	Chutar

atuam na linha. Cada robô possui sua própria máquina de estados e, ao mesmo tempo, um conjunto de ações válidas para execução.

O comportamento do goleiro consiste em ficar posicionado em um ponto estratégico. Este ponto é o centro da meta da equipe que ele está defendendo. A Figura 4.6 apresenta a estratégia do goleiro.

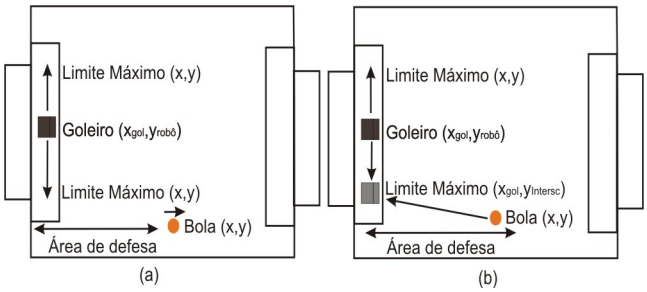

Figura 4.6: Estratégia do goleiro.

O algoritmo implementado para o comportamento do goleiro consiste em verificar a distância em que a bola se encontra da meta. Quando a bola está se distanciando da meta (Figura 4.6a), a estratégia do goleiro é retornar ou permanecer no centro da meta. Por outro lado, quando a bola está se aproximando da meta (Figura 4.6b), a estratégia do goleiro é se posicionar no ponto (x,y) que corresponde à intersecção da trajetória da bola com a linha do gol, para impedir a passagem da bola.

Os comportamentos dos robôs que atuam na linha dependem da in-

formação que corresponde à posição em que o robô se encontra no ambiente. A Figura 4.7 apresenta a estratégia da defesa e de ataque.

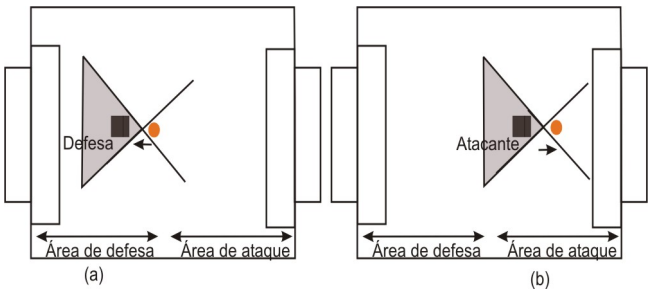

Figura 4.7: Estratégia da defesa e de ataque.

O algoritmo implementado para o comportamento de defesa (Figura 4.7a) consiste basicamente em impedir que o robô da equipe adversária possa conduzir a bola em direção à meta da própria equipe. A estratégia de defesa independe do local em que o robô se encontra, isto é, mesmo estando no meio de campo, o robô pode assumir o comportamento de defesa. Esta definição de comportamento leva em consideração o sentido de deslocamento da bola.

Já o comportamento de atacante (Figura 4.7b) é composto de duas ações: atacar ou posicionar-se. A ação atacar é aplicada quando o robô se encontra atrás da bola. Dessa forma, é necessário apenas que o robô ajuste sua trajetória para conduzir a bola, até um ponto considerado próximo à meta da equipe adversária, e acionar o mecanismo de chute (chutar). Por outro lado, a ação posicionar-se é aplicada toda vez que o robô não se encontra atrás da bola. Dessa forma, é necessário primeiramente posicionar o robô atrás da bola, e a partir dessa posição, aplicar a ação atacar. A definição de qual ação o robô deve executar depende da posição em que a bola e a meta da equipe adversária se encontram.

4.2.4 Implementação do sistema de definição de trajetória

A implementação do módulo de definição de trajetória depende do comportamento que é assumido pelo robô. Assim, definiu-se três comportamentos, do goleiro,

de defesa e de ataque. É necessário conhecer o conjunto de ações válidas para o comportamento. Dessa forma, é preciso saber a posição alvo que corresponde ao ponto para onde o robô deve se deslocar e o ponto onde o robô se encontra.

Para este algoritmo, e sendo que as informações necessárias para o deslocamento do robô são coletadas pelo conjunto de sensores e processadas pela camada vagar sem informação no próprio *EyeBot*, utilizou-se então um método que corresponde apenas à definição das ações válidas, isto é, qual deve ser a ação necessária para que seja possível o robô atingir o ponto alvo. Foi implementado, dessa forma, dada à característica da arquitetura que consiste em deixar os níveis estratégico e de atribuição de comportamento (ação), na entidade central, e comportamentos reativos no próprio robô.

Assim, para o comportamento do goleiro, o conjunto de ações válidas é: permanecer parado, deslocar-se para um ponto à direita, deslocar-se para um ponto à esquerda. Dessa forma, quando pretende-se que o robô ocupe um ponto alvo, é necessário apenas enviar uma mensagem de comando, informando a ação necessária para ocupar este ponto. Por outro lado, o comportamento dos jogadores de linha apresentam as seguintes ações: “Parar”, “Seguir em frente”, “Girar à direita” e ”Girar à esquerda”.

Este método foi escolhido por ser simples, uniforme e conveniente para o processo de construção de uma possível trajetória com precisão. A forma utilizada na definição da estratégia consiste em definir a ação, enviar para o sistema de navegação, que é responsável pelo processo de conversão dessa ação em um vetor de velocidade que é enviado para ser executado pelos atuadores do robô.

4.2.5 Implementação do sistema de navegação

A implementação do sistema de navegação é responsável pelo controle das mensagens de comando que correspondem ao vetor de velocidade que é enviado para os atuadores do *EyeBot*. Este sistema recebe o vetor de velocidade que representa à velocidade linear e angular do sistema de definição de trajetória, processa estas informações, e verifica se realmente é necessário enviar esta mensagem para o robô. O processo de verificação consiste em saber se realmente a informação coletada pelo sis-

tema visão global e processada pela entidade central é importante para o deslocamento do robô. O grau de importância é atribuído após verificar o sentido de deslocamento do robô. Pois, em situações onde o vetor de velocidade definido apresenta valores que correspondem a uma mensagem seguir em frente e sendo que o robô já está se deslocando neste mesmo sentido, esta mensagem não tem necessidade de ser enviada. Dessa forma, as mensagens que são enviadas pelo sistema de comunicação primeiramente são verificadas. Após esta análise, apenas as mensagens consideradas relevantes são enviadas pelo sistema de comunicação para que sejam executadas pelos atuadores do *EyeBot*. A estratégia do sistema de navegação conta com a informação fornecida pelo sistema de visão global, para ajustar a trajetória de navegação do robô nos instantes em que a bola, dada à dinâmica do jogo, não se encontra mais no ponto identificado pelo sistema de visão local, isto é, a trajetória fornecida pelo sistema de navegação global é responsável pelo direcionamento do robô no ambiente. Todas as mensagens transmitidas da entidade central para o robô são enviadas diretamente para a camada avançar com informação. Estas mensagens possuem ordem de prioridade maior e sendo assim sempre são executadas pela camada avançar com informação. Esta prioridade é atribuída em função de que o sistema de visão global possui a característica de poder visualizar todo o ambiente do jogo; já a camada avançar com informação é responsável pela precisão no controle da navegação do robô.

4.3 Implementação do módulo de comunicação

A implementação do módulo de comunicação é responsável pela troca de mensagens entre a entidade central e os robôs. A forma utilizada na implementação consiste em definir um tamanho fixo para as mensagens de comunicação. Após esta definição, verifica-se se a porta serial da entidade central “COM 1 ou COM 2” está funcionando. Da mesma forma, é preciso verificar se o rádio do robô *EyeBot* está recebendo e transmitindo informações. Esta checagem acontece antes de iniciar o envio de mensagens que correspondem aos comandos. A partir do instante em que tanto o sistema de comunicação da entidade central quanto dos robôs estão em pleno funcionamento, inicia-

se a troca de mensagens entre a entidade central e os robôs.

A troca de mensagens acontece da seguinte forma: o rádio do *EyeBot* verifica se existe mensagem disponível e identifica se esta mensagem pertence ao robô. Quando existem mais do que uma mensagem disponível para o mesmo robô, somente a última mensagem enviada é executada; o restante é descartado. Para o caso em que não existem mensagens disponíveis, o sistema de comunicação do robô fica aguardando uma nova mensagem.

Uma mensagem de comando qualquer é composta de quatro argumentos: *(msg, param, from e type)*. O argumento *msg* representa um comando que corresponde à informação da mensagem. Esta mensagem de comando pode ser: parar, seguir em frente, girar à direita, girar à esquerda e retornar. O argumento *param* corresponde ao comando que verifica na condição do sistema de controle do *EyeBot* se este comando é válido e qual deve ser a função a ser executada. O argumento *from* contém a informação que corresponde à identidade de origem da informação, e o argumento *type* representa o tipo desta informação, ou seja, se o comando que está sendo enviado é um comando de controle remoto ou de jogo. Para o argumento *type*, estão disponíveis dois tipos de informações. Um tipo serve para posicionar o robô em campo quando o jogo está parado, e o outro, quando o jogo está em andamento.

Após o recebimento de uma mensagem de comando, o sistema de comunicação do robô envia para a entidade central a confirmação do recebimento dessa mensagem. Este sistema de comunicação permanece em funcionamento até receber a mensagem de comando *"RADIOTerm();"* que corresponde à finalização do processo de transmissão.

4.4 Implementação do módulo do *EyeBot*

O módulo do *EyeBot* é composto por uma câmera CCD, três sensores e cinco comportamentos separados em forma de camadas, além dos atuadores. A implementação do módulo de inicialização do *EyeBot* consiste no processo de checagem dos componentes de hardware do robô e da integração de todas as camadas que fazem parte do

software embarcado. A checagem dos componentes de hardware do robô é responsável por indicar se existe algum problema com relação ao rádio, à câmera, aos motores e ao mecanismo de chute. É este módulo que informa caso um desses componentes não esteja conectado ou não funcionando. Já o processo de integração das camadas, une todas as camadas que fazem parte do sistema embarcado, bem como da liberação de memória após a execução de cada camada.

4.4.1 Implementação do sistema de visão

A implementação do sistema de visão para o robô *SoccerBot (EyeBot)* foi desenvolvida com o objetivo de que o robô reconheça apenas três cores: laranja, azul e amarelo. A cor laranja corresponde à cor da bola, e as cores azul e amarelo representam as cores das metas. A forma utilizada na implementação consiste em procurar pelo tom da cor correspondente e armazenar este valor em uma variável. Após a análise de todo o quadro da imagem, o algoritmo traça um histograma para o eixo horizontal e outro, para o eixo vertical e combina os histogramas. Dessa forma, é possível obter um valor aproximado do centro da imagem que corresponde à posição aproximada do objetivo.

Para que seja possível construir um histograma do eixo horizontal e outro histograma do eixo vertical da imagem que está sendo analisada, é necessário armazenar a informação que diz respeito à coordenada inicial em que o tom da cor correspondente foi encontrado, bem como o ponto que marca a coordenada final. A partir do instante em que se conhece este intervalo, que correlaciona à coordenada em seu ponto inicial e final, e dividindo este intervalo ao meio, é possível obter o ponto da coordenada que corresponde ao centro da imagem da bola no eixo em que está sendo analisado.

O valor obtido da variável que armazena o ponto da coordenada correspondente ao centro da imagem, adequa-se ao valor da distância em que a bola se encontra do robô, tanto para o eixo horizontal quanto para o eixo vertical.

A informação que se refere ao valor da distância em que a bola se encontra do robô é obtida e atualizada toda vez que um quadro é capturado e analisado. Ao mesmo tempo, essa informação é disponibilizada para as camadas: avançar com

informação, aproximar-se da bola, conduzir bola e chutar.

4.4.2 Implementação do sistema de sensores

O modelo de robô *SoccerBot Plus (EyeBot)* possui três sensores *PSD (Position Sensitive Detection)*, que são responsáveis por indicar a distância em que um obstáculo se encontra do robô. A forma utilizada na implementação consiste primeiramente em definir um valor de distância em que seja aceitável para detectar todos os obstáculos. Este valor de distância depende da posição em que o sensor está posicionado no *EyeBot*. Por exemplo: para o sensor que está posicionado na parte da frente, é necessário que o sensor indique a presença de um obstáculo com um valor de distância maior. Dessa forma, é possível evitar a colisão. Já para os sensores posicionados à direita e à esquerda, este valor de distância deve ser menor, visto que estes sensores auxiliam apenas em desviar de obstáculos nos movimentos rotacionais do robô.

O processo que envolve a identificação dos obstáculos consiste na pré-definição dos valores que indicam a presença ou a ausência de um obstáculo. Portanto, um obstáculo pode estar a uma distância considerada: perto, média ou longe. Assim, definem-se dois valores para cada sensor: um valor menor indicando a presença de obstáculo e outro valor maior, indicando a ausência de obstáculo.

A partir da definição dos dois valores pré-definidos para cada sensor, o algoritmo desenvolvido verifica a leitura do sensor e compara o valor obtido com os valores pré-definidos. Caso o valor lido esteja abaixo do menor valor pré-estabelecido para o sensor, o algoritmo desenvolvido indica a presença de um obstáculo. Por outro lado, se o valor lido estiver acima do maior valor pré-estabelecido para o sensor, o algoritmo desenvolvido indica a ausência de obstáculo. Já o valor que corresponde ao intervalo que vai do menor valor pré-estabelecido até o maior valor pré-estabelecido para cada sensor, faz-se necessário para que a máquina de estados possa ser atualizada.

A informação que corresponde ao valor da distância em que um obstáculo se encontra do robô é atualizada em tempo-real. E ao mesmo tempo, esta informação é disponibilizada para o sistema de atuadores.

4.4.3 Implementação do sistema dos atuadores

O sistema implementado para os atuadores é responsável pelo controle dos robôs no ambiente. Este sistema controla os dois motores DC: um é responsável pelo controle da velocidade linear e o outro, pelo controle da velocidade angular. Dessa forma, quando se deseja que o robô se desloque pelo ambiente, é preciso passar em forma de comando um vetor de velocidade que é composto pela velocidade linear e velocidade angular.

A forma utilizada na implementação consiste primeiramente em definir os valores, tanto para a velocidade linear quanto para a velocidade angular. A escolha destes valores depende do movimento desejado. Por exemplo, para o movimento seguir em frente, a velocidade linear 0.4 e a velocidade angular 0.0; para o movimento girar à direita, a velocidade linear 0.0 e a velocidade angular -0.4; para o movimento girar à esquerda, a velocidade linear 0.0 e a velocidade angular 0.4. No entanto, esses valores em situações em que o robô necessita calcular o ângulo em direção à bola, ou desviar de obstáculos são variáveis. O cálculo destes valores variáveis é realizado com base em funções de espaço geométrico (ver seção 4.4.5).

O algoritmo desenvolvido define a velocidade linear e a velocidade angular para um estado desejado. A partir dessa definição, os atuadores executam este comando até que um novo comando seja atribuído.

4.4.4 Implementação da camada vagar sem informação

O sistema implementado para a camada vagar sem informação foi desenvolvido com o objetivo de que o robô possa vagar pelo ambiente a partir das informações adquiridas do seu conjunto de sensores. Esta camada é a de mais baixo nível na arquitetura embarcada proposta. Sua aplicação se faz necessária para que o robô possa vagar pelo ambiente independente de possuir a informação da posição em que a bola se encontra e ao mesmo tempo de forma reativa.

O funcionamento da camada vagar sem informação verifica a distância em que um obstáculo se encontra do robô. Cada robô é equipado com um conjunto de três

sensores, e a distância entre o robô e os obstáculos é fornecida através da leitura destes sensores. O sistema desenvolvido verifica se os sensores: frontal, à direita e à esquerda possuem a informação de presença ou ausência de obstáculo. Ao informar a presença ou ausência de obstáculos, que o algoritmo define qual é o estado em que o robô se encontra.

Com a informação coletada pelo sensor frontal, a máquina de estados define qual ação deve ser executada. No caso da informação do sensor frontal indicar que um obstáculo encontra-se longe do robô, isto é, o valor fornecido é superior 150mm, a ação a ser executada é vagar pelo ambiente à procura da bola. Dessa forma, é possível que o robô inicie sua navegação pelo ambiente. Já para o caso onde o sensor frontal indicar a presença de um obstáculo, a ação a ser executada é desviar deste obstáculo. Para que seja possível desviar do obstáculo frontal, é necessário verificar as informações dos sensores à direita e à esquerda. Para este caso, o algoritmo decide de forma randômica se verifica primeiro o sensor à direita ou à esquerda. Em caso dos três sensores indicarem a presença de obstáculos, a ação a ser executada é retornar.

O algoritmo implementado consiste dos estados possíveis em que o robô pode se encontrar durante uma partida de futebol, e quais são as ações que devem ser executadas para cada estado possível. Estes estados e suas ações são executados até o momento em que o sistema de visão detectar a presença da bola. A partir da informação da posição da bola, o robô passa a ter um objetivo e, desta forma, é ativada a camada avançar com informação.

4.4.5 Implementação da camada avançar com informação

O sistema implementado para a camada avançar com informação foi desenvolvido com o objetivo de que esta camada somente seja acionada a partir do momento em que o robô possui a informação da posição em que a bola se encontra. A informação da posição em que a bola se encontra é fornecida através do sistema de visão embarcado. É necessário informar também que, para o caso onde o sistema de visão global indicar a presença da bola, a mensagem que corresponde ao vetor de velocidade linear e angular definida pela entidade central é enviada em forma de mensagem de comando diretamente

para esta camada. Assim, a camada avançar com informação passa a ter um objetivo.

O funcionamento da camada avançar com informação é iniciado no instante em que o sistema de visão embarcado indicar a posição da bola. Dessa forma, o algoritmo desenvolvido executa uma ação, que é responsável por calcular a trajetória de direção para o robô. O cálculo da trajetória de direção toma como referência o centro da posição da bola. A partir deste ponto de referência, o algoritmo desenvolvido verifica esta posição para saber se este ponto se encontra: no centro da imagem; ou à esquerda; ou à direita. Caso o ponto de referência pertença ao intervalo considerado como sendo o centro da imagem, a mensagem de comando enviada aos atuadores consiste simplesmente em avançar. Já para o caso onde a posição do ponto de referência não pertence ao intervalo considerado como sendo o centro da imagem, independente deste ponto estar à esquerda ou à direita, é necessário calcular a nova trajetória de direção para o robô. Se a posição em que a bola se encontra é à direita, a função que calcula esta trajetória corresponde a equação 4.1. Por outro lado, se a posição em que a bola se encontra é à esquerda, a função que calcula esta trajetória corresponde a equação 4.2. A variável x é a abscissa do ponto onde se encontra a bola e y é sua coordenada. Os ângulos são medidos em radianos.

$$\begin{aligned} angulo &= atan(sen(y)/cos(x)) \\ tanangulo &= (angulo - (\pi/2))/2 \end{aligned} \tag{4.1}$$

$$\begin{aligned} angulo &= atan(sen(y)/cos(x)) \\ tanb &= \pi - angulo \\ anguloa &= \pi - tanb \\ angulos &= (tanb - anguloa)/2 \end{aligned} \tag{4.2}$$

Observa-se que o cálculo da nova trajetória de direção, somente é realizado no caso da bola se encontrar em uma posição considerada ou à direita ou à esquerda

do robô. Por outro lado, a partir do momento em que o sistema de visão não indicar mais a presença da posição em que a bola se encontra, o sistema desenvolvido retorna sua atividade de processamento para a camada vagar sem informação, que passa a ter o controle de planejamento da trajetória para o robô.

4.4.6 Implementação da camada aproximar-se da bola

O sistema implementado para a camada aproximar-se da bola foi projetado com o objetivo de atender à deficiência que o robô tem no momento em que passa do estado em que não está de posse da bola para o estado em que está de posse da bola. A abordagem do robô em relação à bola deve ser o mais suave possível, pois uma abordagem em alta velocidade pode resultar em uma colisão entre o robô e a bola, fazendo com que a bola seja lançada para uma posição mais distante do robô. Portanto, o algoritmo desenvolvido para a camada aproximar-se da bola, possui como sua principal função prever este momento de chegada do robô na bola, procurando reduzir o impacto gerado.

O algoritmo da camada aproximar-se da bola, inicia sua execução a partir do momento em que o sistema de visão indicar a posição da bola. No entanto, seu comportamento somente é acionado a partir do momento em que o robô está se deslocando em direção à bola. Este comportamento consiste em verificar a distância em que a bola se encontra do robô, e a partir do instante em que o robô vai se aproximando da bola, e encontra-se em uma distância considerada como sendo um intervalo aceito para aplicar o comportamento de redução de velocidade. Um intervalo aceito para aplicar o processo de redução consiste em um valor entre 65mm à 50mm. Dentro deste intervalo definido, é possível aplicar o processo de redução de velocidade que consiste na mudança do valor linear de 0.4 para 0.2. Abaixo do valor de 50mm, uma posição considerada pelo robô próxima da bola ocorre à troca do valor linear de 0.2 para 0.4. Assim, sempre que o robô estiver se aproximando da bola, este comportamento é acionado. Portanto, o comportamento do algoritmo consiste em tornar o movimento do robô um pouco mais lento até o momento que a abordagem acontece. A partir do momento em que a abordagem aconteceu, a velocidade do robô retorna às condições normais de navegação.

O comportamento que envolve a redução de velocidade somente é aplicado no momento em que o robô pretende ficar de posse da bola e a redução de sua velocidade acontece uma única vez e, ao mesmo tempo, em uma situação bem específica, quer dizer, apenas no instante em que o robô vai tocar na bola. Para o caso onde o robô faz parte da defesa, e sendo que é necessário retirar a bola dessa região, o comportamento de redução de velocidade não é acionado. Quando o robô está de posse da bola, esta informação é enviada para a camada conduzir bola que passa a ter o controle do robô.

4.4.7 Implementação da camada conduzir bola

O sistema implementado para a camada conduzir bola é responsável pelo controle que o robô exerce na condução da bola. Esta camada controla o robô nos momentos em que a bola está sob o seu domínio. O controle exercido por este algoritmo é aplicado no instante em que a bola está escapando do domínio do robô. Isto significa dizer que, quando a bola está em uma das extremidades do mecanismo de chute que é externo ao robô. Por isso, é preciso controlar a bola em direção ao seu objetivo, que é a meta da equipe adversária.

O algoritmo da camada conduzir bola inicia sua execução a partir do momento em que o robô possui o domínio da bola e o sistema de visão indica que a bola está próxima do mecanismo de chute. No entanto, o comportamento implementado pelo algoritmo conduzir bola, é responsável por verificar se a bola está em uma região considerada como sendo o centro do mecanismo de chute. Caso a bola não esteja no centro do mecanismo de chute, é necessário ajustar o movimento do robô para que a bola retorne para o centro. Sendo assim, o algoritmo verifica em qual das extremidades do mecanismo de chute a bola se encontra e ajusta a nova trajetória do robô, para que a bola possa ficar no centro do mecanismo de chute. Este ajuste utiliza as mesmas fórmulas que são utilizadas na camada avançar com objetivo. A única diferença é que os valores para o ajuste da camada conduzir bola são menores em função da proximidade do robô com a bola. Assim, é possível que o robô possa conduzir a bola até o seu objetivo, que é a meta da equipe adversária.

4.4.8 Implementação da camada chutar

O sistema implementado para a camada chutar é responsável pelo movimento do mecanismo de chute. Este sistema controla o movimento que é executado pelo servo motor de chute. Dentre os movimentos executados pelo sistema que controla o movimento do mecanismo de chute está: levantar o mecanismo de chute, abaixar o mecanismo de chute e levantar de forma moderada o mecanismo de chute.

O algoritmo da camada chutar inicia sua execução com o mecanismo de chute abaixado, mesmo que o mecanismo de chute esteja fora da posição considerada abaixada. O primeiro passo do algoritmo é verificar a posição do mecanismo de chute e deixar o mecanismo de chute na posição abaixada. O estado abaixado do mecanismo de chute permanece a maior parte do tempo em que o robô está em movimento no ambiente. No entanto, o mecanismo de chute somente é acionado em duas condições, quando o robô está em condições de chutar a bola na meta da equipe adversária ou quando pretende passar a bola para um robô da mesma equipe. Para o caso onde o robô chuta a bola em direção à meta da equipe adversária, o robô deve estar em condições de chute. Estas condições são: visão da meta da equipe adversária e posição do robô considerada como sendo próxima da meta da equipe adversária. Já para o caso onde o robô passa a bola para um outro robô da mesma equipe, esta condição depende apenas da dinâmica do jogo.

A decisão que envolve o acionamento do mecanismo de chute é atribuída tanto ao robô quanto à entidade central. Para o caso onde a decisão parte da entidade central, é necessário enviar uma mensagem através do sistema de comunicação, que corresponde ao comando responsável pelo acionamento do mecanismo de chute. Este comando é interpretado pela camada chutar e executado pelo agente, que é responsável pelo controle do mecanismo de chute.

Capítulo 5

Resultados

Neste capítulo são descritos os resultados e a forma utilizada para a validação desses resultados. Todos os resultados descritos nesta seção foram obtidos através da observação feita no *display* de LCD dos robôs *EyeBots*. Para se chegar a cada resultado apresentado, foi necessário configurar, no sistema desenvolvido dos robôs, comandos de impressão das mensagens de execução. Assim, foi possível observar o desempenho dos robôs no ambiente e realizar estes ensaios.

Os resultados obtidos com o desenvolvimento dessa aplicação demonstraram um desempenho satisfatório, pois o sistema desenvolvido que está hospedado na entidade central é consistente em relação à localização da bola e dos robôs. Este sistema processa as imagens em tempo real e não apresenta um melhor desempenho em função do sistema de captura utilizado, que opera a uma taxa de 29.97 *fps*.

Em relação à definição de deslocamento envolvendo os níveis, estratégico e de comportamento, percebeu-se que em função da dinâmica de jogo, o resultado obtido para o deslocamento do robô em relação a sua direção, em geral, converge para o mesmo ponto. Dessa forma, a mensagem de comando enviada pelo nível estratégico é simplesmente descartada pelo robô. Por outro lado, quando os níveis estratégicos e de comportamento não convergem para um ponto em comum, e sendo que o robô não possui a informação que correspondem à posição da bola, ele descarta as mensagens enviadas pelo nível de comportamento do sistema embarcado e executa as mensagens enviadas

pelo nível estratégico que correspondem à informação enviada pela entidade central. Assim, este modelo de arquitetura atende aos objetivos que consistem na construção de um modelo de arquitetura híbrida e distribuída, atuando em tempo real.

Com relação aos resultados obtidos com o desenvolvimento do sistema embarcado, constatou-se que realmente os robôs atuam no ambiente conforme foi proposto. A implementação das camadas com seus estados e ações são considerados suficientes para que o robô vague pelo ambiente sem colidir com obstáculos e, simultaneamente, ao encontro da bola, tendo como objetivo capturar e conduzir a bola até à meta da equipe adversária.

Com relação aos resultados obtidos com o desenvolvimento do sistema de visão do *EyeBot*, observou-se que este sistema é consistente em relação à localização da bola e das metas. No entanto, o tempo necessário para processamento de um quadro capturado pela câmera do *EyeBot* e a identificação da cor desejada pertence ao intervalo [0.22s, 0.25s]. Este tempo é atribuído aos recursos de hardware oferecidos pela câmera do *EyeBot*. Pois, somente o processo que envolve a captura do quadro pela câmera do *EyeBot*, necessita de um tempo aproximado de 0.2s, ou seja, a câmera utilizada, dado sua característica, captura até 5 *fps* [26]. Dessa forma, torna-se difícil melhorar o sistema de visão desenvolvido, visto que a margem de melhoria possível é de até 0.03s. A Figura 5.1 apresenta no *display* de LCD do *EyeBot* a imagem da bola capturada após o processamento.

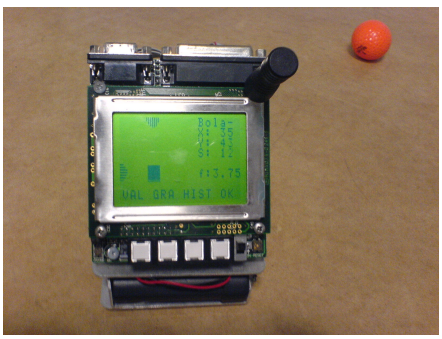

Figura 5.1: Imagem da bola no *display*.

O sistema desenvolvido para os sensores dada sua simplicidade, que consiste apenas em indicar a presença ou a ausência de um obstáculo e sua distância, comportou-se de forma eficiente. No entanto, dada à característica do conjunto dos sensores *EyeBot* que se está utilizando, percebe-se que em situações onde o robô choca-se contra a parede, e sendo este choque fruto de uma colisão imprevista causada por outro robô ou qualquer outro problema que possa surgir, o sensor do *EyeBot* não consegue indicar que o robô está em colisão, quer dizer, para uma distância menor de 5cm, o sensor do *EyeBot* indica ausência de obstáculo.

O resultado obtido com o desenvolvimento do sistema dos atuadores verificou que, para uma velocidade linear maior do que 0.7, os robôs *EyeBot* não se comportam de forma adequada. Pois, valores superiores 0.7 simplesmente são mal interpretados pelos atuadores do robô, ocasionando em diversos momentos desequilíbrio com relação a sua trajetória. Também constatou-se que uma velocidade linear que pertence ao intervalo [0.1, -0.1] para movimentos angulares, em que o robô necessita girar à direita ou girar à esquerda, nem sempre é adequada. Em situações onde o robô está parado, a velocidade linear que pertence ao intervalo [0.1, -0.1], nem sempre consegue superar o atrito exercido sobre o ponto de apoio que o robô possui. A Tabela 5.1 apresenta o melhor conjunto de valores para solucionar o problema de velocidade linear e angular para o *EyeBot*.

Tabela 5.1: Melhor conjunto de valores encontrado para a velocidade linear e angular do *EyeBot*.

Ação	**Velocidade linear**	**Velocidade angular**
Navegação para frente	0.4	0.0
Navegação para frente rápido	0.6	0.0
Navegação para frente lento	0.2	0.0
Navegação de ré	-0.4	0.0
Girar à direita	variável	-0.4
Girar à esquerda	variável	0.4

Os resultados obtidos com relação à camada vagar sem informação, mostraram que realmente o robô vaga pelo ambiente conforme foi proposto. Os testes, realizados com a execução desta camada demonstraram que, conforme os sensores indicam posições diferentes para os obstáculos, o robô reage através da execução de uma nova ação. Por exemplo, quando o sensor frontal indicar a presença de um obstáculo, e os sensores à direita e à esquerda não indicarem a presença de obstáculos, imediatamente, o sistema executa uma das ações válidas, que consiste em girar à direita ou à esquerda. Com a implementação dos estados projetados, constatou-se de que os estados e suas ações são considerados suficientes para que o robô vague pelo ambiente sem colidir com os obstáculos, referiu-se como sendo obstáculos os robôs e as paredes. Averiguou-se também que, em situações em que o valor pré-definido no algoritmo do sistema de sensores é um valor muito alto, superior a 190mm, o robô não chega a aproximar-se das laterais, ou este valor sendo um valor muito baixo, inferior a 70mm, o robô não consegue detectar o obstáculo a tempo de evitar uma possível colisão. Dessa forma, foi necessário um ajuste destes valores para se obter o melhor desempenho por parte do robô. A Tabela 5.2 apresenta os melhores valores para solucionar o problema de leitura dos sensores para se obter o melhor desempenho com a camada vagar sem informação.

Tabela 5.2: O melhor conjunto de valores encontrado para os sensores do *EyeBot*.

Sensor	**Nível baixo**	**Nível alto**
Frontal	135	150
Direito	70	100
Esquerdo	70	100

Dentre outros possíveis resultados obtidos para esta camada vagar sem informação, é necessário informar que a máquina de estados do *EyeBot* executa suas ações por um determinado tempo, dada a característica do próprio hardware do *EyeBot* [26]. Dessa forma, o número de camadas que está sendo executado em um determinado tempo, influencia no desempenho do robô.

Os resultados obtidos com relação à camada avançar com informação, apresentaram que o algoritmo proposto para calcular a nova trajetória, que é em direção à posição da bola, realmente vai ao encontro do que havia sido proposto, ou seja, ao encontro da bola. No entanto, constatou-se que nem sempre o robô atinge este alvo. Esta deficiência é atribuída à máquina de estados do *EyeBot*, pois ela executa uma ação por um tempo aproximado de 1s. Este tempo é atribuído em função de que cada servo motor é atualizado periodicamente a cada 20ms [26], e sendo que o *EyeBot* utiliza: dois motores (motor A, motor B), um servo motor para a câmera e um servo motor para o mecanismo de chute e, também, necessita atualizar a máquina de estados. Então cada ação será executada por aproximadamente 1s. Neste tempo de execução da máquina de estados, ficou comprovado que, dependendo do movimento exercido pela bola, ou seja, seu deslocamento no ambiente, o resultado obtido do deslocamento do robô em direção a uma posição em que a bola não esteja mais, pode ocorrer. Para este tipo de problema não se encontrou uma solução, visto que, para resolvê-lo, é necessário desenvolver um novo sistema operacional para o conjunto de robôs *EyeBot*, com características diferenciadas para o sistema de deslocamento do robô. Uma alternativa para tentar solucionar este tipo de problema vem do sistema de navegação da entidade central, que ao perceber que o robô está se distanciando do alvo que é o seu objetivo, a bola, envia uma nova mensagem de comando informando que o robô deve seguir uma nova trajetória. A Tabela 5.3 apresenta os resultados que foram obtidos através de um conjunto de ensaios realizados com relação à camada avançar com informação.

Esta camada apresenta um conjunto de 13 ensaios. Para cada ensaio foram coletadas as coordenadas (x,y) em que a bola se encontra e além disso o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. A partir da posição em que a bola se encontra nas coordenadas (x,y), o sistema desenvolvido para a camada avançar com informação define um vetor de velocidade que corresponde à velocidade linear e à velocidade angular, que será executada pelos atuadores. Estes resultados representam as possíveis posições em que a bola pode se encontrar no ambiente. Estes dados foram extraídos do *display* do *EyeBot* após o processamento desta camada.

Com relação aos resultados obtidos com o desenvolvimento da camada

Tabela 5.3: Resultados obtidos da camada avançar com informação.

Ensaios	**Coordenada X**	**Coordenada Y**	Tam. Bola	Vel. Linear	Vel. Angular
1	9	37	7	0.2	0.23
2	11	56	4	0.2	0.19
3	15	47	8	0.2	0.31
4	44	49	17	0.2	0.73
5	67	56	7	0.2	0.85
6	24	46	10	0.2	0.46
7	21	32	10	0.4	0
8	20	26	8	0.4	0
9	3	1	2	0.2	-0.39
10	11	3	5	0.2	-0.65
11	30	9	15	0.2	-0.68
12	55	8	12	0.2	-0.75
13	31	16	15	0.2	-0.54

aproximar-se da bola, percebeu-se que a quantidade de variáveis possíveis que podem influenciar o momento da chegada do robô na bola é muito grande. Observou-se que o robô encontra dificuldade com relação à forma como esta abordagem acontece. Em situações em que a bola se encontra parada, ou segue a mesma trajetória de direção do robô, a abordagem acontece de forma perfeita. No entanto, em situações em que a bola se encontra em movimento e na direção contrária à trajetória de deslocamento do robô e ao mesmo tempo, dependendo da velocidade da bola, o momento da abordagem do robô em relação à bola nem sempre acontece de forma suave. Esta deficiência acontece em função de que a máquina de estados está executando uma determinada ação, e o tempo envolvendo a colisão é muito pequeno. Dessa forma, a máquina de estados ainda não foi atualizada para o novo estado e, sendo assim, pode ocorrer uma colisão envolvendo o momento em que o robô chega à bola. No entanto, mesmo que ocorra uma colisão,

dependendo da forma como ela acontece, o robô permanece com a posse da bola. A Tabela 5.4 apresenta os resultados obtidos da camada aproximar-se da bola.

Tabela 5.4: Resultados obtidos da camada aproximar-se da bola.

Ensaios	**Coordenada X**	Coordenada Y	Tam. Bola	Vel. Linear	Vel. Angular
14	66	26	22	0.4	0
15	65	41	21	0.1	0
16	64	29	18	0.1	0
17	49	32	15	0.1	0
18	40	32	12	0.1	0
19	39	32	11	0.4	0

Esta camada apresenta um conjunto de 6 ensaios. Para cada ensaio, foram coletados as coordenadas (x,y) em que a bola se encontra e o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. Nesta camada, o sistema utiliza apenas a coordenada (x) para ativar o comportamento de redução de velocidade. Dessa forma, o sistema desenvolvido verifica a coordenada (x) e a partir de um valor inferior a 66, que corresponde à distância em que a bola se encontra do robô, aplica o comportamento de redução de velocidade, e a partir do instante em que o valor da distância é inferior 40, retorna à velocidade normal de navegação. Estes resultados representam as possíveis posições em que a bola pode se encontrar no ambiente. Estes dados também foram extraídos do *display* do *EyeBot* após o processamento da camada aproximar-se da bola.

Com relação aos resultados obtidos com o desenvolvimento da camada conduzir bola, constatou-se que o algoritmo proposto para ajustar a nova trajetória do robô, que é responsável pela condução da bola, realmente vai ao encontro do que havia sido proposto, ou seja, a camada conduzir bola realiza o processo de correção da trajetória do robô nos momentos em que a bola está saindo do controle do mecanismo de chute do próprio robô. Também verificou-se que o comportamento desta camada é semelhante ao

comportamento da camada avançar com informação. No entanto, a diferença que existe é que a bola está próxima do robô e vale lembrar também o motivo pelo qual esta camada foi projetada, que é ajustar a curva do robô no momento em que o robô está de posse da bola, mas, no entanto, a bola está deslizando por uma das laterais do mecanismo de chute. Dessa forma, faz-se necessário redirecionar o robô para que ele possa permanecer com a posse da bola. A Tabela 5.5 apresenta os resultados obtidos da camada conduzir bola.

Tabela 5.5: Resultados obtidos da camada conduzir bola.

Ensaios	Coordenada X	**Coordenada Y**	Tam. Bola	Vel. Linear	Vel. Angular
20	72	53	11	0.2	0.94
21	73	48	7	0.2	0.90
22	71	20	19	0.4	0.0
23	72	16	12	0.4	0.0
24	71	6	13	0.2	-0.74
25	70	3	11	0.2	-0.76

Esta camada apresenta um conjunto de 6 ensaios. Para cada ensaio, foram coletados as coordenadas (x,y) em que a bola se encontra e o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. Nesta camada, o sistema utiliza apenas a coordenada (y) para ativar o comportamento responsável por planejar a nova trajetória do robô. Dessa forma, o sistema desenvolvido verifica a coordenada (y) e ao perceber que a bola não se encontra em um intervalo aceito como sendo o centro do mecanismo de chute, calcula uma nova trajetória, que é responsável por redirecionar o robô, para que a bola possa retornar ao centro do mecanismo de chute. A partir de um valor inferior a 15, que corresponde a um ponto à direita do mecanismo de chute, o algoritmo calcula a nova trajetória para o robô. A nova trajetória deve direcionar o robô à direita, dessa forma, a velocidade angular será negativa. Por outro lado, a partir de um valor superior a 45, que corresponde a um ponto à esquerda do mecanismo de chute, o algoritmo calcula a nova trajetória para o robô. A nova trajetória deve direcionar a robô

à esquerda, desta forma, a velocidade angular será positiva. Estes resultados representam as possíveis posições em que a bola pode se encontrar no ambiente. Estes dados também foram extraídos do *display* do *EyeBot* após o processamento da camada conduzir bola.

Finalmente, a partit dos resultados obtidos com o desenvolvimento da camada chutar bola, percebeu-se que o algoritmo desenvolvido para controlar o mecanismo de chute, que é responsável pelo toque na bola em direção à meta da equipe adversária, ou mesmo para passar a bola para um robô da mesma equipe, apresenta uma característica de movimento lento. Ou seja, apesar do servo motor que controla o mecanismo de chute estar em condições de movimentar-se com total velocidade e, sendo que, em nenhum momento o algoritmo desenvolvido apresenta controle de velocidade de movimento, apenas as condições de abaixar, passar e levantar o mecanismo de chute, percebe-se que a bola não é lançada com muita força. Dessa forma, faz-se necessário aproximar o robô o máximo possível da meta da equipe adversária para que possa ser ativado o comportamento chutar em gol.

Também constatou-se que apesar do algoritmo desenvolvido apresentar a condição de passar a bola para o jogador da mesma equipe, este comportamento nem sempre acontece, ou acontece em momentos considerados inadequados. Para poder explorar melhor este comportamento, faz-se necessário implementar condições relacionadas à cooperação multiagentes. Cooperação esta, que indique quais são os momentos em que o robô da mesma equipe se encontra em uma melhor condição para receber a bola e continuar com a condução da bola em direção à meta da equipe adversária.

Capítulo 6

Conclusão

6.1 Considerações finais

Neste trabalho, descreveu-se e analisou-se o desenvolvimento de uma arquitetura híbrida para sistemas multiagentes e sua aplicação no futebol de robôs. Esta arquitetura é adaptada com um sistema de visão global e dotada de sensores e visão local para captura das informações. O processamento das imagens é realizado tanto nos robôs quanto na entidade central.

A arquitetura híbrida distribuída para processamento das informações permitiu tratar as informações em locais estratégicos e possibilitou um ganho com relação ao desempenho do sistema. Assim, tanto o sistema de visão global quanto o sistema de visão local foram beneficiados, pois esta arquitetura permitiu os níveis estratégicos e de ação serem precisos e, ao mesmo tempo, os níveis de comportamento e execução se beneficiarem com relação ao domínio do ambiente.

Esta arquitetura foi testada na V Competição Brasileira de Robótica (CBR2007). Apesar dos resultados obtidos com relação à classificação na competição não terem sido dos melhores, a participação serviu de experiência para testar a implementação da arquitetura proposta em um ambiente oficial da *RoboCup*. As falhas ocorridas durante a competição são atribuídas ao baixo desempenho oferecido pelo robô *EyeBot*, principalmente, com relação ao desgaste das baterias e à arquitetura oferecida pelo hardware, pois

a arquitetura do hardware não permite retirar o atrito produzido pelo ponto de apoio do robô. Assim, este modelo de robô apresenta um desempenho considerado fraco para uma competição oficial da *RoboCup*. Por outro lado, o software desenvolvido passou por uma série de melhorias, tendo como objetivo ajustar pontos considerados relevantes. O contato com os demais competidores do evento serviu para consolidar idéias e trouxe novas inspirações para o desenvolvimento e aplicação de novas ferramentas.

A arquitetura proposta neste trabalho é uma solução adequada para controle híbrido e distribuído de sistemas que operam em tempo real e sistemas mutiagentes capazes de jogar futebol.

6.2 Trabalhos futuros

Tanto no modelo da arquitetura híbrida e distribuída proposta quanto na implementação do sistema, poderiam ser realizadas diversas modificações, tendo como objetivo aperfeiçoar este trabalho e torná-lo mais eficiente. Dessa forma, apresentam-se algumas propostas para trabalhos futuros:

- Implementação de um novo sistema de definição de estratégia, procurando oferecer um maior poder na tomada de decisões;
- Implementação de um novo sistema de comunicação que permita operar em outras freqüências;
- Desenvolver um novo protótipo de hardware que permita oferecer maior domínio sobre a arquitetura do robô, autonomia no controle dos motores, utilizando novos tipos de sensores: infravermelho, toque, câmera digital;
- Testar esta arquitetura em um novo protótipo de hardware.

Referências Bibliográficas

[1] ABREU, A. & CORREIA, L. Behavior-based decision control in autonomous vehicles: A fuzzy approach using khepera, in: Fuzzy Systems. IEEE, vol. 1, p. 140-145, 2000.

[2] AHO, A. V. et al. Data Structures and Algorithms. Addison - Wesley, 1983.

[3] ALBUS, J., & PROCTOR, F. G. A reference model architecture for intelligent hybrid control systems. Proceedings of the International Federation of Automatic Control. San Francisco, 1996.

[4] ANTSAKLIS, P. J.; PASSINO, K. M. & WANG, S. J. Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues. Journal of Intelligent and Robotic Systems, vol. 1, p. 315-342, 1989.

[5] ARLEO, A. & GERSTNER, W. Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity. Biological Cybernetics, Special Issue on Navigation in Biological and Artificial Systems, 83:287-299, 2000.

[6] ARKIN, R. C. Behavior-Based Robotics. MIT Press, 1998.

[7] ARKIN, R. C. Integrating Behavioural, Perceptual, and World Knowledge in Reactive Navigation. p. 105-122, 1990.

[8] ASADA, M. et al. The Robocup Physical Agent Challenge: Phase I. Applied Artificial Intelligence, 1997.

[9] ASADA, M. & KINATO, H. RoboCup-98: Robot Soccer World Cup II, volume 1604 of Lecture Notes in Computer Science, Springer-Verlag. 1999.

[10] BAUDRILLARD, J. Tela Total. Porto Alegre, Editora Sulina, 1997.

[11] BARRETO, J. M. Inteligência Artificial no Limiar do Século XXI, 3 ed. Florianópolis, 2001.

[12] BLÁZQUEZ, J.; SHEN, Q. & TUSON, A. Tuning fuzzy membership functions with neighbourhood search techniques. IEEE. International Conference on Intelligent Engineering Systems, p. 337-342, 1999.

[13] BISHOP, C. M. Neural Networks for Pattern Recognition, 2 ed. Oxford University Press Inc. - Bookcraft Ltd, Walton Street, Oxford OX2 6DP, New York, 1996.

[14] BITTENCOURT, G. Inteligência Artificial: Ferramentas e Teorias. Editora da UFSC, 1998.

[15] BOTELHO, S. et al. FURGBOL Construindo Robôs Autônomos Holonômicos para Jogar Futebol. VI Simpósio Brasileiro de Automação Inteligente. Bauru, Setembro, 2003.

[16] BRADY, J. M. et al. Vision and the oxford agv, Proc. of Image Processing, London, p. 267-286, 1988.

[17] BRADY, J. M. & WANG, H. Vision for mobile robots, Proceeding of Royal Society, London, 1992.

[18] BROOKS, R. A. A Robust Layered Control System for a Mobile Robot, Massachusetts Institute of Technology, September, 1985.

[19] BROOKS, R. A. A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, vol. 2, n.1, p. 14-23, March, 1986.

[20] BROOKS, R. A. Intelligence Without Reason. In: IJCAI-91, Morgan Kaufmann, San Mateo CA USA. Proceedings, p. 569-595, 1991.

[21] CHENG, G. & ZELINSKY, A. A Reactive Vision-based Navigation System for a Mobile Robot. Workshop for Reactive Systems, 1997.

[22] CONNELL, J. H. A Hybrid Architecture Applied to Robot Architecture. Technical report, IBM, 1991.

[23] COSTA, H. & PEGORARO, R. Construindo Robôs Autônomos para Partidas de Futebol. O Time Guaraná. SBA Controle e Automação v. 11, n. 3, Dezembro, 2000.

[24] COSTA, A. L. & BITTENCOURT, G. UFSC-team: A Cognitive Multi-Agent Approach to the RoboCup'98 Simulator League. RoboCup'98 Workshop - Team Description, p. 371 - 377. Springer, Lecture Notes in Artificial Inteligence, vol. 1694.

[25] DRIANKOV, D. F. Fuzzy logic techniques for autonomous vehicle navigation. Physica-Verlag, Heidelberg, 2001.

[26] EYEBOT - Online Documentation. http://robotics.ee.uwa.edu.au/eyebot/, Último acesso em Junho 2007.

[27] FIRA. Disponível por WWW em URL: http://www.fira.net/. Último acesso em Junho de 2007.

[28] FRANKLIN, S. & GRAESSER. Is it an Agent or Just a Program?: A Taxonomy for Autonomous Agents. University of Memphis, 1996.

[29] GAT, E. Integration Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-Word Mobile Robots. In Proceedings of the 1992 American Association for Artificial Intelligence Conference, p. 809-815, 1992.

[30] GIRALT, G. et al. Remote operated autonomous robots, International Symposium in Intelligent Robotics. 1991.

[31] GOLDBERG, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Mass, 1989.

[32] GUPTA, G. S. et al. Real-Time Identification and Predictive Control of Fast Mobile Robots Using Global Vision Sensing. IEEE. Transaction on Instrumentation and Measurement, vol. 54, no 1. February, 2005.

[33] HENDERSON, T. & HANSEN, C. The specification of distributed sensing and control. Journal of Robotic Systems 2-4 p. 387-396, 1985.

[34] HOLLAND, J. Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[35] HU, H. & BRADY, M. A Parallel Processing Architecture for Sensor-Based Control of Intelligent Mobile Robots. Robotics and Autonomous Systems 17 p. 235-257, 1996.

[36] HU, H. & BRADY, J. M. A bayesian approach to real-time obstacle avoidance for an intelligent mobile robot. International Journal of Autonomous Robots 1-1 p. 67-102, 1994.

[37] HU, H. & BRADY, M. Planning with uncertainty for a mobile robot, Proc. of 2nd International Joint Conference on Automation, Robotics and Computer Vision, Hyatt Regency, Singapore, 1992.

[38] HUANG, H. W. et al. Visual navigation and obstacle avoidance using a steering potential function. Robotics and Autonomous Systems 54 p. 288-299, 2006.

[39] KARR, C. L. Fuzzy-evolutionary systems. Handbook of Evolutionary Computation, Institute of Physics Publishing and Oxford University Press, Bristol, New York, p. D2.1:1-2:9, 1997.

[40] KIM, J. H. et al. A Cooperative Micro Robot System Playing Soccer: Design and Implementation. Robotics and Automation Systems, Elsevier, South Korea, p. 177-189, 1997.

[41] KIM, J. H. et al. A Cooperative Multi-agent System and Its Real Time Application To Robot Soccer. Proceeding of the IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, p. 638-643, 1997.

[42] KIM, J. H. et al. FIRA Robot World Cup Initiative and Research Directions. Int. J. of Robotics and Automation Systems, 1998.

[43] KITANO, H. et al. RoboCup: A challenge Problem for AI. AI Magazine, vol. 18, n. 1, p. 73-85, Spring, 1997.

[44] KRAETZCHMAR, G. et al. The ULM Sparrows: Research into Sensorimotor Integration, Agency, Learning, and Multiagent Cooperation. In: ROBOCUP WORKSHOP, 2, Paris, 1998. Proccedings. FIRA, p. 459-465, 1998.

[45] LEWIS, M. A. & FAGG, A. H. & SOLIDUM A. Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot. IEEE. International Conference on Robotics and Automation. Nice, France. May, 1992.

[46] MAES, P. & BROKS, R. A. Learning to Coordinate Behaviors, AAAI Journal, p. 796-802, 1990.

[47] MENEZES, P. B. Linguagens Formais e Autômatos. 5 ed. Editora Sagra Luzzatto, Porto Alegre, 2005.

[48] MEYSTEL, A. Intelligent Control: Issues and Perspectives. Proceeding of the IEEE Workshop on Intelligent Control, p. 1-15, 1985.

[49] MURPHY, R. R. Biological and Cognitive Foundations of Intelligent Sensor Fusion. IEEE. Transactions on Systems, Man, and Cybernetics, vol. 26, No 1, January 1996.

[50] MURPHY, R. R. Use of Scripts for Coordinating Perception and Action, IROS-96, Japan, p. 156-161, Nov. 1996.

[51] MURPHY, R. R., & HAWKINS, D. K., & SCHOPPERS, M. J., Reactive Combination of Belief Over Time Using Direct Perception. IJCAI-97, Japan, p. 1353-1358, 1997

[52] MURPHY, R. R. Dempster-Shafer Theory for Sensor Fusion in Autonomous Mobile Robots. IEEE. Transactions on Robotics and Automation, vol. 14, No 2, April, 1998.

[53] MURPHY, R. R. Introduction to AI Robotics. The MIT Press, London, England, 2000.

[54] NA, Y. K. & OH S. Y. Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification. Autonomous Robots 15, p. 193-206, 2003.

[55] NAUCK, D. & KRUSE, R. A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. Conference on Neural Networks, San Francisco, p. 1022-1027, 1993.

[56] NEHMZOW, U. Mobile Robotics: A Practical Introduction. Springer Verlag, London, 2000.

[57] NEVES, S. R. et al. Visão Computacional e Estratégias de Controle Aplicadas ao Futebol de Robôs, Robocup Brasil, 2004.

[58] NORBERT, J. Decision marking and image processing robot soccer - the challenge of the FIRA MIROSOF league, 2006.

[59] NOVAK, G. & SPRINGER, R. An Introduction to a Vision System used for a MiroSOT Robot Soccer System. IEEE International Conference on Computational Cybernetics, p. 101-108, 2004.

[60] PARKER, L. E.; BEKEY, G. & BARHEN, J. Distributed Autonomous Robotics Systems 4, Springer, Berlin, 2000.

[61] PAZOS, F. Automação de Sistemas e Robótica. Editora Axcel, 2002.

[62] PIO, F. V. et al. Arquitetura FURGBOL aplicada ao Soccerserver. EnRI, III Encontro de Robótica Inteligente, 2006.

[63] REZENDE, M. F. Desenvolvimento de um Robô Móvel Autônomo Inteligente Utilizando a Arquitetura de Subsumption. Uberlândia - MG: Centro de Ciências Exatas e Tecnologia, Universidade Federal de Uberlândia, Dissertação (Mestrado), 102 páginas, 1992.

[64] ROBOCUP. Disponível por WWW em URL: http://www.robocup.org. Último acesso em Junho de 2007.

[65] RUSSEL, S & NORVIG, P. Inteligência Artificial. Editora Campus, 2004.

[66] SAHOTA, M. K. & MACKWORTH, A. K. Can Situated Robots Play Soccer? In: Proceedings of Canadian IA, 1994.

[67] SAHOTA, M. K. et al. Real-time Control of Soccer-playing Robots Using Off-board Vision: the Dynamite Testbed. IEEE, 1995.

[68] SALIM, A.; FUENTES, O. & MUÑOZ, A. Development of Local Vision-based Behaviors for a Robotic Soccer Player. IEEE. Proceedings of the Fifth Mexican International Conference in Computer Science, 2004.

[69] SANDERSON, A. Micro-Robot World Cup Soccer Tournament (MiroSot). IEEE. Robotics and Automation Magazine, p. 15, December, 1997.

[70] SELVATICI, A. H. P. & COSTA, A. H. R., A Hybrid Adaptive Architecture for Mobile Robots Based on Reactive Behaviors. Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, 2005.

[71] SCHWARTZ, W. Reconhecimento em tempo real de agentes autônomos em futebol de robôs. VI Simpósio Brasileiro de Automação Inteligente - SBAI, 2003.

[72] SHEN, W. et al. Integrated Reactive Soccer Agents. In: RoboCup Workshop, 2, Paris, 1998. Proceedings. FIRA, p. 251-264, 1998.

[73] SHIM, H. S. et al. A Hybrid Control Structure for Vision Based Soccer Robot System. International J. Intelligent Automation and Soft Computing, p. 89-101, 2000.

[74] SHIM, H. S. et al. Designed Distributed Control Architecture for Cooperative Multi-agent System and Its Real-Time Application to Soccer Robot. Journal of Robotics and Autonomous System, vol. 21, N. 2, p. 149-165, September, 1997.

[75] SILVA, F. A. et al. Estratégia para o Controle dos Robôs EyeBot do UFSC-Team: Categoria Small Size do Futebol de Robôs. Simpósio Brasileiro de Automação Inteligente, Campo Grande. Competição Brasileira de Robótica, 2006.

[76] SKARMETA, A. & JIMENEZ, F. Generating and tuning fuzzy rules using hybrid systems. Proc. of the 6th International Conference on Fuzzy Systems, vol. 1, p. 247-252, 1997.

[77] SONY. Disponível por WWW em URL: http://www.sony.com. Último acesso em Julho de 2007.

[78] TAKAHASHI, Y. & ASADA, M. Vision-Guided Behavior Acquisition of a Mobile Robot by Multi-Layered Reinforcement Learning. IEEE. International Conference on Intelligent Robots and Systems, 2000.

[79] TAMBE, M. Implementing Agent Teams in Dynamic Multi-Agent Environments. Applied Artificial Intelligence, vol. 12, March, 1998.

[80] VADAKKEPAT, P. et al. Evolution of fuzzy behaviors for multi-robotic system. Robotics and Autonomous Systems. Singapore, July, 2006.

[81] VELOSO, M. et al. The CMUnited-7 Robotic Soccer Team: Perception and Multi-agent Control. In: International Conference on Autonomous Agents, 2, Minneapolis, Proceedings. AAAI, 1998.

[82] VELOSO, M. et al. CMUnited97: RoboCup97 SmallRobot World Champion Team. February, 1998.

[83] WANG, Z. D. et al. LOGUE: an architecture for task and behavior object transmission among multiple autonomous robots. Robotics and Autonomous Systems 44 p. 261-271, 2003.

[84] WATSON, J. Psychology as a Behaviorist Views It, Psychological Review, 20, 158-77, 1913.