Faculdade de Engenharia Industrial

VISÃO COMPUTACIONAL APLICADA AO CONTROLE DE MICRO ROBÔS

Relatório do projeto de trabalho OS N. 5886 do Departamento de Engenharia Elétrica da Faculdade de Engenharia Industrial.

Coordenador do projeto: Prof. Reinaldo Augusto da Costa Bianchi.

São Bernardo do Campo
2001

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este relatório apresenta os resultados do projeto Visão Computacional aplicada ao controle de micro-robôs, financiado pela Fundação de Ciências Aplicadas – FCA/FEI sob o Número OS 5886.

O objetivo deste projeto é o estudo de sistemas de visão computacional para o controle de robôs autônomos que apresentam comportamento inteligente, cooperando entre si, em um domínio de jogo de futebol de micro robôs. O principal problema abordado é o da integração de diversos módulos com tarefas diferentes em um sistema robótico e que utiliza Visão Computacional Propositada para perceber o mundo.

Os sistemas baseados em visão computacional têm conquistado espaço tanto nas universidades quanto na indústria, devido à intensa modernização que os sistemas de automação industriais vem sofrendo nos últimos anos. Entre os fatores que impulsionam esta moderização pode-se incluir a competitividade crescente, a rápida alteração dos produtos oferecidos ao mercado e o avanço tecnológico, entre outros, que visam aumentar a produtividade, a qualidade e a confiabilidade dos produtos. Além da área de manufatura e manipulação de matériais, outras aplicações destes sistemas incluem o trabalho em ambientes perigosos ou insalubres, o processamento de imagens de satélites e a exploração espacial.

Diversos foram os resultados alcançados por este projeto. Foram especificadas duas plataformas de desenvolvimento de um sistema de visão: a primeira, baseada em um microcomputador Pentium III com placa de aquisição de imagens e uma câmera, executando o sistema operacional Linux, e a segunda permitindo o desenvolvimento de hardware dedicado, baseado em FPGAs e VHDL. Foram realizados estudos e comparação sobre a utilização de dois tipos de algoritmos de segmentação de imagens para o desenvolvimento de um sistema de percepção visual de robôs. Finalmente, construiu-se um protótipo de micro robô, com módulos de aquisição de imagens, de controle dos motores e de controle de estratégias baseado em programação genética.

O sistema foi testado exaustivamente e os resultados experimentais indicaram algumas vantagens e desvantagens da metodologia empregada, possibilitando a definição de algumas diretrizes importantes para o projeto e implementação de sistemas robóticos inteligentes atuando no mundo real.

1. INTRODUÇÃO

Neste capítulo é apresentada uma introdução ao trabalho proposto, descrevendo os seus objetivos, justificativas, resultados e contribuições. Inicia-se o capítulo com a discussão da relação entre Visão Computacional e Inteligência Artificial, que é fundamental para a inserção do trabalho na área de Inteligência Artificial.

1.1 Visão Computacional e Inteligência Artificial

Nos últimos anos, vários pesquisadores da área de Visão Computacional (VC) têm dado muita ênfase às técnicas puramente matemáticas, como geometria e análise funcional, afastando-se das investigações básicas em Inteligência Artificial (IA), que geralmente buscam métodos mais gerais para a solução de problemas. Por um lado isto aconteceu porque as metodologias que estavam sendo transferidas de IA para VC mostravam-se grosseiramente inadequadas e, por outro, por existir uma certa confusão de conceitos entre ambas as áreas, incluindo ainda Robótica.

Esse enfoque puramente matemático para VC pode ser problemático, uma vez que tende a compartimentar o estudo, esquecendo outros aspectos que compõem um sistema inteligente, e a ignorar o que muitos autores - como por exemplo, ALOIMONOS (1994), BROOKS (1991b), ELFES (1986), FISHER (1994) e FIRBY (1997) - concordam ser um problema fundamental em IA: a integração dos diferentes aspectos da inteligência.

Muitos pesquisadores da área de VC estão hoje direcionando seus estudos de modo a reaproximá-la de IA, em um contexto bastante diferente do existente há 10 anos atrás, porque existe hoje um embasamento matemático e de engenharia muito mais sólido para a solução dos problemas de VC. Reafirmar que VC e IA possuem muitos aspectos em comum é uma conseqüência dessa linha de pesquisa.

Assim, FISHER (1994) discute a relação entre VC e IA, mostrando que ambas partilham objetivos, metodologias, ferramentas, suposições de domínios (*domain assumptions*) e embasamento biológico, psicológico e filosófico. Afirma que “as diferenças são em grande parte ilusórias, e que VC ainda possui e vai continuar possuindo uma forte conexão com IA” [FISHER, 1994, p.21]. Observa que os objetivos de VC são especializações dos objetivos de IA e que podem ser correlacionados. Entre as comparações apresentadas por FISHER (1994), tem-se:

- IA procura compreender, num dado aspecto, os processos computacionais humanos e VC procura compreender os sistemas visuais biológicos - incluindo o humano, ambos desenvolvendo metodologias, teorias e módulos que possam ser testados;
- IA também procura desenvolver ferramentas que são inteligentes, no sentido de precisar de pouca atenção de um ser humano para atuar, sendo flexíveis e possuindo conhecimento compilado. VC procura construir ferramentas que tenham habilidades de extração de informação e que sejam autônomas;
- IA procura aprimorar as habilidades dos seres humanos, possibilitando o controle do seu ambiente, enquanto VC procura desenvolver ferramentas que ampliem as habilidades perceptivas humanas;
- IA pesquisa o desenvolvimento de representações do conhecimento para

domínios diferentes e VC procura determinar qual a melhor representação para um domínio específico.

Desse modo, fica evidente que VC e IA estão profundamente relacionadas, tornando o estudo dos outros aspectos de cognição de fundamental importância para os pesquisadores de VC.

A solução para o problema de como fazer com que um robô se comporte de maneira inteligente é, há muito tempo, o sonho dos cientistas do campo de Inteligência Artificial e engenheiros da área de Robótica.

Sistemas que solucionam esse tipo de problema encontram aplicações na indústria - por exemplo, em células flexíveis de montagem ou manufatura - e em robôs móveis autônomos, que podem se locomover sem serem controlados por uma pessoa e que encontram sozinhos seus caminhos, podendo assim atuar em ambientes não-estruturados. Por exemplo, no caso de uma linha de montagem, espera-se que um robô móvel reconheça o ambiente que o cerca; localize uma peça sobre uma mesa de trabalho; identifique sua forma e orientação para poder manipulá-la; locomova-se ao longo da linha de montagem; posicione-se adequadamente para a tarefa de montagem; etc.

Esse comportamento exige que o robô interaja com o mundo à sua volta, processando uma grande quantidade de informações sensoriais (visão, tato, distância, força, etc.), e as incorpore ao seu processo de atuação em tempo real. Por isso, a utilização de robôs autônomos requer o desenvolvimento de sistemas avançados, de alto desempenho, baseados em flexibilidade e confiabilidade.

1.2 Objetivos do Projeto

O objetivo deste projeto é o estudo de sistemas de visão computacional para o controle de robôs autônomos que apresentam comportamento inteligente, cooperando entre si, em um domínio de jogo de futebol de micro robôs. O principal problema abordado é o da integração de diversos módulos com tarefas diferentes em um sistema robótico e que utiliza Visão Computacional Propositada para perceber o mundo.

Os sistemas baseados em visão computacional têm conquistado espaço tanto nas universidades quanto na indústria, devido à intensa modernização que os sistemas de automação industriais vem sofrendo nos últimos anos. Entre os fatores que impulsionam esta moderização pode-se incluir a competitividade crescente, a rápida alteração dos produtos oferecidos ao mercado e o avanço tecnológico, entre outros, que visam aumentar a produtividade, a qualidade e a confiabilidade dos produtos. Além da área de manufatura e manipulação de matériais, outras aplicações destes sistemas incluem o trabalho em ambientes perigosos ou insalubres, o processamento de imagens de satélites e a exploração espacial.

Como indicadores de resultados do projeto, três metas foram estabelecidas:

- Especificação de uma plataforma de Visão Computacional;
- Estudo de métodos e algoritmos de Visão Computacional aplicáveis ao rastreamento de objetos;
- Contrução de um sistema para o rastreamento de robôs em movimento.

Como metodologia de projeto, foram seguidos modelos propostos pela Engenharia de Software para o desenvolvimendo das aplicações. O modelo de desenvolvimento

básico utilizado é a Análise e Projeto Orientada a Objetos, com a utilização da "Unified Modelling Language" (UML) para a definição dos módulos do projeto. Outras técnicas utilizadas Constructive Cost model e os modelos de estimativa de Putnam (PRESSMAN, 1995).

As atividades realizadas incluíram o estudo de textos fundamentais e avançados da área de Inteligência Artificial, Visão Computacional e Robótica, com a apresentação (por parte dos alunos e em sistema de rodízio) e a discussão de cada artigo em reuniões semanais. Ainda nas reuniões, ocorreram a definição de tarefas de cada aluno, a discussão sobre as dúvidas e apresentação e análise dos resultados obtidos.

1.3 Domínio de aplicação: Futebol de Robôs

Os apreciadores de futebol defendem essa modalidade desportiva como apaixonante, estratégica e até artística, onde a habilidade técnica e física dos jogadores somada à estratégia tática, normalmente determinam o resultado de uma partida. Guardada as proporções, o futebol de robôs, também apaixonante e estratégico, é uma possibilidade de aplicação, depuração e desenvolvimento de técnicas de visão computacional, inteligência artificial e de outras áreas, onde a habilidade de acertar e conduzir a bola, a velocidade de deslocamento, a tomada de decisão, a capacidade de cooperação e a estratégia tática do time determinam o resultado.

Um projeto de um time de futebol robótico representa uma aplicação prática do uso de agentes autônomos com comportamento inteligente, cooperando entre si, visando a execução de uma tarefa, em geral, computacionalmente complexa. O grau de interação entre esses agentes, no meio onde se desenvolve a tarefa, depende consideravelmente das informações obtidas durante sua execução. Para tanto, são empregados sistemas de aquisição de informações que sejam adequados à situação considerada. Além disso, este domínio possui características bastante complexas, envolvendo necessidade de atuação em tempo real, tratamento de incertezas e ruídos das informações extraídas de ambientes dinâmicos, em geral imprecisas e incompletas.

Atualmente, em muitas instituições nacionais e internacionais, vários trabalhos estão em desenvolvimento nesse campo, apresentando resultados significativos tanto para a compreensão do funcionamento dos sistemas robóticos quanto para a implementação de sistemas reais nas áreas de automação industrial, entre outras.

1.4 Resultados propostos x Resultados obtidos

Analisando os resultados propostos em relação aos resultados obtidos no desenvolvimento deste projeto, a seguinte comparação pode ser estabelecida (Tabela 1.1):

Proposta	**Resultado**
Especificação de uma plataforma de Visão Computacional para o estudo.	Foram especificadas duas plataformas de desenvolvimento: um microcomputador Pentium II com placa de aquisição de imagens e uma câmera, executando o sistema operacional Linux, e o desenvolvimento de hardware dedicado, baseado em FPGAs e VHDL.

Estudo de métodos e algoritmos de Visão Computacional aplicáveis ao rastreamento de objetos.	Foi realizado o estudo e a comparação da utilização de dois tipos de algoritmos de segmentação de imagens para o desenvolvimento de um sistema de percepção visual de robôs.
Contrução de um sistema para o rastreamento de robôs em movimento.	Este item foi realizado de duas maneiras diferentes: aprimeira, com uma implementação em VHDL de algoritmos de visão Computacional tradicionais e a segunda, com a implementação de um sistema em Linguagem C.

Tabela 1.1 – Comparação das metas propostas e os resultados atingidos.

Para além dos resultados propostos, outros foram alcaçados, como é possível observar por meio da descrição abaixo (Tabela 1.2):

Resultados adicionais obtidos
Construção de um protótipo de micro robô, com módulos de aquisição de imagens, de controle dos motores e de controle de estratégias. A Figura 2.1 apresenta o protótipo construído na FEI.
Contrução de um sistema de controle baseado em algoritmos genéticos para o robô.

Tabela 1.2 – Resultado adicionais obtidos.

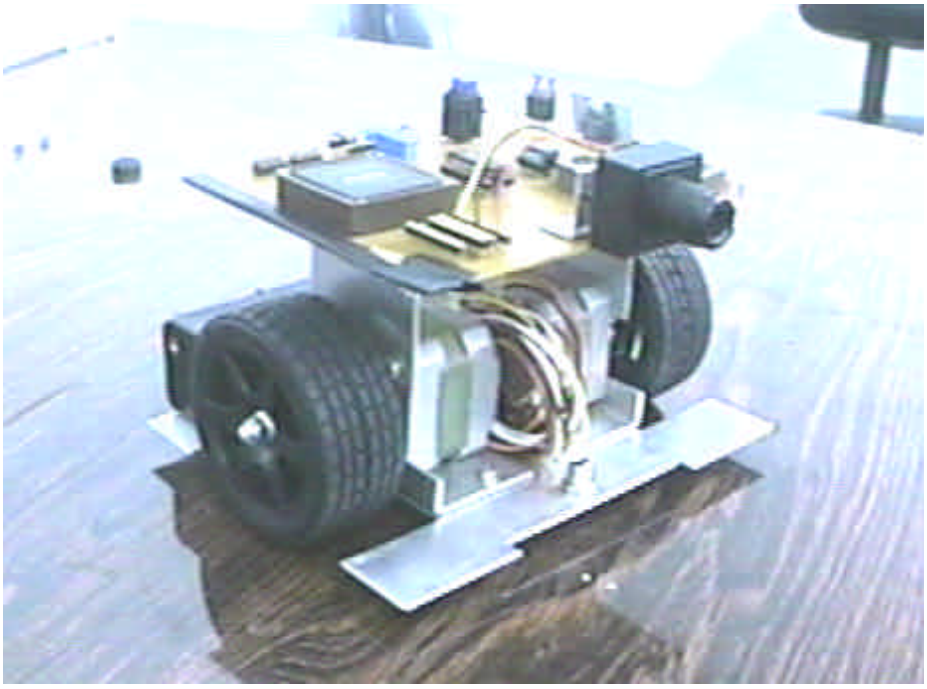


Figura 1.1 - Protótipo do Robô Autônomo para a RoboCup F180.

No que diz respeito à sistematização e divulgação dos resultados, os artigos abaixo listados foram publicados (Tabela 1.3) ou estão sendo submetidos à apreciação, ou seja, estão em fase de avaliação (Tabela 1.4).

Conferência, local e data	Título
International Conference On Engineering and Computer Education. São Paulo, Julho/2000.	The RoboCup Domain as Foundation for Practice Based, Research Integrated Studies in Electrical and Computer Engineering
Workshop de Computação – Workcomp'2000, São José dos Campos, agosto/2000	Evoluindo Agentes Jogadores de Futebol.
Congresso Brasileiro de Automática. Florianópolis, julho/2000	O Sistema de Visão Computacional de um time de Futebol de Robôs.
Simpósio de Iniciação Científica da Universidade de São Paulo, São Carlos, novembro/2000.	Um Sistema de Aquisição de Imagens baseado em FPGAs
Simpósio de Iniciação Científica da Universidade de São Paulo, São Carlos, novembro/2000.	Visão Computacional utilizando hardware e VHDL
Simpósio de Iniciação Científica da Universidade de São Paulo, São Carlos, novembro/2000.	Detecção de Velocidade baseada em Visão Computacional
Simpósio de Iniciação Científica da Universidade de São Paulo, São Carlos, novembro/2000.	Desenvolvendo Jogadores de Futebol de Robos: uma abordagem de Programação Genética
Simpósio de Iniciação Científica da Universidade de São Paulo, São Carlos, novembro/2000.	Um sistema de controle por hardware FPGA para um micro-robô.

Tabela 1.3 – Artigos publicados.

Conferência, local e data	Título
THE ROBOCUP 2001 INTERNATIONAL SYMPOSIUM. SEATTLE – USA, AGOSTO/2001.	An FPGA/VHDL-based Vision System for a Mobile Robot.
Revista	
REVISTA DA SOCIEDADE BRASILEIRA DE AUTOMÁTICA	Um sistema de controle baseado em hardware FPGA para um micro-robô.
JOURNAL OF THE BRAZILIAN COMPUTER SOCIETY	Implementing Computer Vision Algorithms in Hardware.

Tabela 1.4 – Artigos em fase de avaliação.

Finalmente, quanto à formação de especialistas na área, os alunos que participaram

do projeto como bolsistas do projeto de Iniciação Científica da FEI puderam se aperfeiçoar em temas como programação para ambiente Unix, programação em Linguagem C e JAVA, sistemas digitais, FPGAS e VHDL, sistemas de controle, etc. Além disso, foi oferecido aos alunos do programa de Iniciação Científica e a Professores desta Instituição um curso de Visão Computacional, com carga horária de 16 horas, realizado no mês de maio de 2000.

1.5 Organização do trabalho

Visando a apresentação do contexto no qual está inserido este projeto, o segundo capítulo descreve o domínio de aplicação do Futebol de Robôs e seus desafios.

Como o sistema inteligente proposto neste trabalho concentra na visão seu meio de interação perceptual com o ambiente, o terceiro capítulo deste trabalho fornece um breve histórico da área de Visão Computacional, apresentando e discutindo os paradigmas existentes. O quarto capítulo apresenta as técnicas de segmentação de imagens em visão computacional estudadas e que foram utilizadas pelas implementações de software e hardware.

A organização dos capítulos entre o quinto e o oitavo segue a organização da construção de um robô, descrevendo os módulos principais do robô construído. O quinto capítulo apresenta a implementação de algoritmos de da segmentação de imagens em software e seus resultados enquanto o sexto descreve a implementação de algoritmos de visão computacional em hardware e seus resultados. O sistema de controle baseado em programação genética é apresentado no capítulo sete. O oitavo capítulo descreve o sistema para o controle de motores de passo implementado e o robô construído.

O nono capítulo apresenta trabalhos que fundamentaram e inspiraram este projeto. Finalmente, o décimo e último capítulo apresenta as conclusões deste trabalho e discute as propostas de extensões que podem ser realizadas em trabalhos futuros.

2. O DOMÍNO DO FUTEBOL DE ROBÔS

2.1 Introdução

Os livros textos modernos de Inteligência Artificial, como RUSSELL e NORVIG (1995), começam a apresentar esta disciplina a partir de uma visão integrada. Isto não só permite que se trate os problemas desta área de pesquisa a partir de diversas abordagens, como é o resultado da compreensão, por parte dos pesquisadores, que IA não deve ser vista como segmentada.

Seguindo esta tendência, os domínios de aplicação pesquisados também começaram a mudar. Na área dos jogos, clássica em IA, criar programas eficientes para jogos de tabuleiro deixou de ser um objetivo distante: no xadrez os computadores já conseguem vencer os campeões humanos. Novos domínios fizeram-se necessários.

O futebol de robôs foi proposto por diversos pesquisadores ([KITANO, 1997], [KIM, 1998], [SANDERSON, 1997]) para criar para IA um novo desafio a longo prazo. O desenvolvimento de times de robôs envolve muito mais que integração de técnicas de IA. Segundo KRAETZCHMAR et al. (1998), "Dispositivos mecatrônicos, hardware especializado para o controle de sensores e atuadores, teoria de controle, interpretação e fusão sensorial, redes neuronais, computação evolutiva, visão, e Sistemas Multi-Agentes são exemplos de campos envolvidos nesse desafio". A figura abaixo apresenta o time de futebol de robôs da *Carnegie Mellon University*que vençeu a copa mundial de 1997.

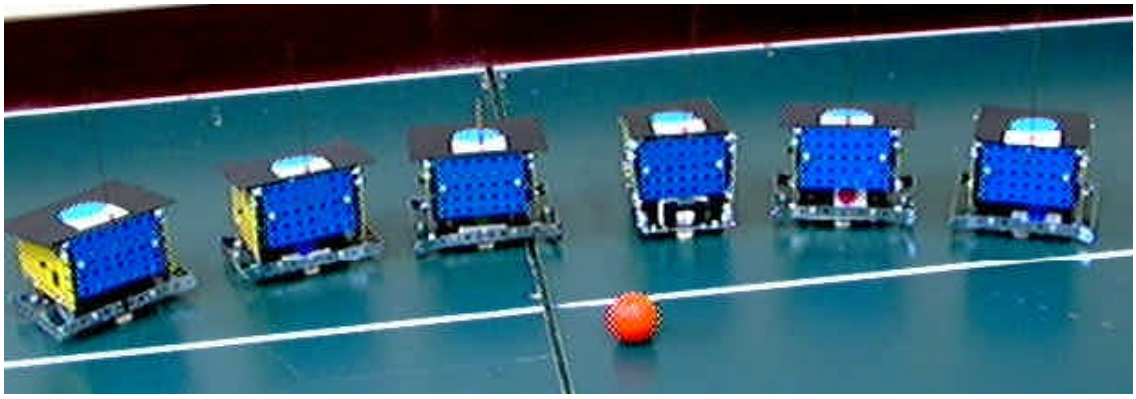


Figura 2.1 - O time CMUnited-97 vencedor da copa de 1997. [VELOSO, M.; STONE, P; HAN, K., 1998]

2.2 Descrição do Domínio do Futebol de Robôs

Partidas de futebol entre robôs constituem uma tarefa que possibilita a realização de experimentos reais para o desenvolvimento e testes de robôs que apresentem comportamento inteligente, cooperando entre si para a execução de uma tarefa, além de serem extremamente motivantes para possibilitar o surgimento de um espírito de ciência e tecnologia nas jovens gerações.

Já existem algumas competições anuais, aliadas a importantes eventos das áreas de Inteligência Artificial e Robótica, onde são observados os desenvolvimentos neste tipo de aplicação. A cada 4 anos, as copas de robôs são realizadas juntamente com as

copas mundiais de futebol, no país sede. Assim, em 1998, juntamente com a Copa Mundial de Futebol, da FIFA, foi realizada a Copa Mundial de Robôs, da FIRA, na França, patrocinada, entre outros, pela própria FIFA.

A organização chamada *Federation of International Robot-soccer Association* (FIRA) é responsável pelo estabelecimento e controle das diversas copas emergentes de futebol de robôs. Existem diversas modalidades de jogos entre robôs, com variações desde o número e o tamanho dos robôs até a capacidade computacional de cada robô. Entre estas modalidades, a MIROSOT - *Micro Robot World Cup Soccer Tournament* é a considerada a de mais fácil implementação (tem os robôs menores e de mais baixo custo), sendo a que este trabalho se destina.

Fisicamente a plataforma deste projeto é constituída por um campo para o jogo, plano, com as dimensões de 150 x 90 cm (similar a de futebol de botão) e, para cada time, uma câmara de vídeo CCD e respectivo sistema de aquisição de imagens, um computador, sistema de transmissão de dados e os 3 robôs. Estes têm as dimensões máximas de 7.5x7.5x7.5 centímetros e são compostos por 2 motores controlados por um microprocessador, bateria própria e sistema de comunicação sem fio. A Figura 2.1 apresenta o esquema básico do sistema. A bola utilizada é padronizada: uma bola de golfe laranja.

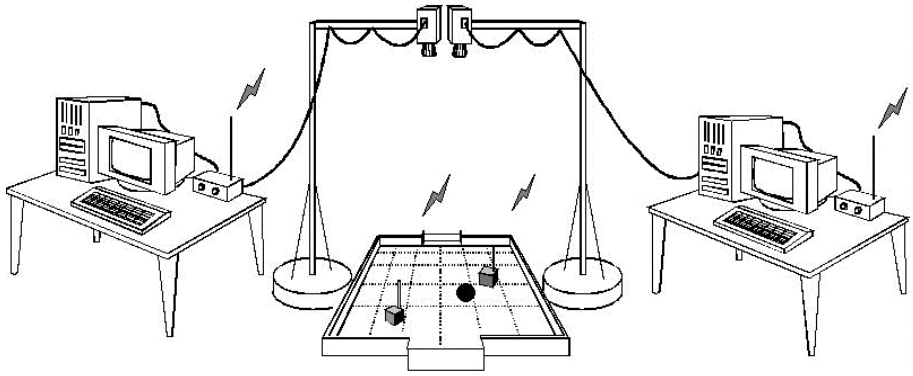


Figura 2.1 - O esquema básico de um jogo apresentado em [FIRA, 1998].

O funcionamento de cada time segue uma mesma fórmula básica: cada time realiza a aquisição da imagem através da sua câmera, processa a imagem usando técnicas de Visão Computacional para descobrir a posição de todos os robôs e da bola. Com esta imagem, um sistema de decisão define a melhor tática a aplicar e os movimentos instantâneos de cada robô. Todo este processamento é realizado em um único computador. Com a decisão de movimentação tomada, um sistema comunicação por rádio envia para os robôs uma mensagem como movimento a ser realizado.

Este processo é repetido com a decisão tomada mudando conforme a configuração encontrada na imagem, em ciclos de 60 vezes por segundo.

As outras modalidades de campeonatos, que o grupo já tem estudado incluem: *RoboCup Small League*: muito similar à Mirosot, tem times de 5 robôs de até 180 cm^2 e uma mesa similar a de tênis de mesa; *RoboCup Medium* e *Full Size*: nestas modalidades não é permitido o uso de um sistema de visão único nem de um computador central. Cada robô deve ter sua própria câmera e computador e trocam as mensagens via rádio. Nestas duas últimas modalidades é que se encontram as maiores contribuições até o momento.

2.3 Desafios e problemas a serem estudados

Segundo SHEN (1998) “Jogadores Robóticos precisam realizar processos de reconhecimento visuais em tempo real, navegar em um espaço dinâmico, rastrear objetos em movimento, colaborar com outros robôs e ter controle para acertar a bola na direção correta.” [SHEN, 1998, p.251].

Esta citação mostra a complexidade do desafio e dos problemas a resolver. Os robôs devem ser autônomos, eficientes, cooperativos, com capacidades de planejamento, raciocínio e aprendizado, além de existirem restrições de tempo real.

A construção física dos robôs envolvem diversos aspectos importantes, como o estudo de dispositivos mecatrônicos, de hardware especializado para o controle de sensores e atuadores e teoria de controle.

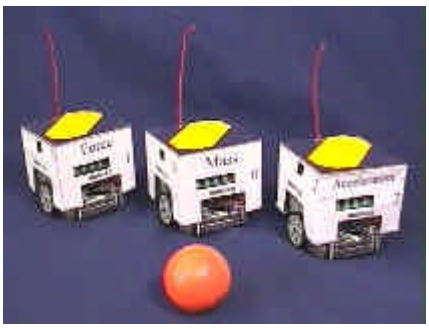


Figura 2.1 - Robôs da categoria Mirosot de fabricxação coreana.

O sistema robótico deve possuir capacidades visuais para perceber o mundo: reconhecer e localizar os robôs no jogo e a bola, descobrindo os movimentos na cena. A construção do sistema sensorial envolve: visão computacional, interpretação e fusão sensorial, redes neuronais e computação evolutiva.

Uma das principais características de um sistema para o futebol de robôs não é sua habilidade para raciocinar, mas sua habilidade para escolher ações de maneira rápida e efetivas, sendo capaz de atuar em um ambiente incerto e variável. Esse domínio pode ser caracterizado como o de uma tarefa de planejamento reativo complexa, que envolve diversos aspectos a serem estudados: a geração e execução de planos complexos para resolver objetivos específicos, a alocação de recursos *on-line*, a capacidade de lidar com problemas que surgem em tempo real e a capacidade de raciocinar com informações incompletas (nem sempre o sistema sensorial consegue detectar todos os movimentos) e eventos imprevisíveis (um toque na bola pode sair como previsto ou não: envolve aspectos físicos como atrito e aderência).

Por ser um sistema com múltiplos robôs executando uma tarefa em cooperação, este domínio é ideal para o estudo de aspectos fundamentais dos Sistemas Multi-Agentes: a modelagem distribuída do ambiente (ou como dada agente representa o conhecimento que possui do problema), a organização das sociedades e a comunicação entre os agentes, além de arquiteturas de controle.

Finalmente VELOSO (1998) afirma que “o Futebol de Robôs é excitante devido a sua natureza de sensores/atuadores em tempo real. E isso pode ser atingido usando a mais simples das arquiteturas até as mais complexas.” Isto permite que equipes com as mais diversas abordagens possam competir e medir os resultados de suas abordagens de uma maneira objetiva: o placar e a classificação no campeonato. Neste domínio a

natureza das restrições de tempo real ficam óbvias: um sistema deve ser rápido o suficiente pois se ele levar alguns segundos entre a percepção e a ação ele perde para o adversário.

O domínio do Futebol de Robôs tem adquirido importância cada vez maior na área de Inteligência Artificial pois possui todas as características encontradas em outros problemas reais complexos, desde sistemas de automação robóticos, que podem ser vistos como um grupo de robôs realizando uma tarefa de montagem, até missões espaciais com múltiplos robôs [TAMBE, 1997].

3. HISTÓRICO DA ÁREA DE VISÃO COMPUTACIONAL

Neste capítulo é apresentado um panorama da área de Visão Computacional, descrevendo os seus paradigmas, discutindo suas definições de visão e a relação entre eles.

3.1 Introdução

As primeiras tentativas de construir um sistema de visão computacional são atribuídas por JOLION (1994) à Levialdi nos anos 50. Foram trabalhos na área de física de altas energias, onde se tentava analisar imagens provenientes de experimentos em câmaras de bolhas, realizando um processamento digital de imagens.

Uma definição tradicional da área de visão computacional como disciplina foi proposta por TRIVEDI e ROSENFELD (1989): "visão computacional é a disciplina que investiga as questões computacionais e algoritmicas associadas à aquisição, ao processamento e à compreensão de imagens". Somada à Neurofisiologia e à Psicologia Perceptual, formam o conjunto das disciplinas que estudam a visão (Figura 3.1).

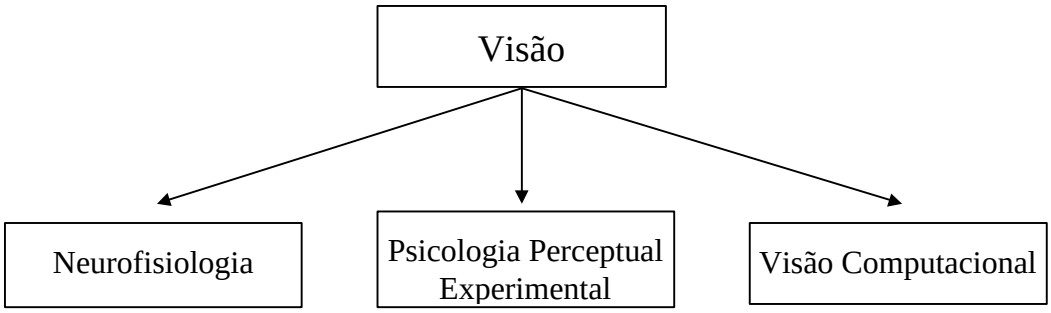


Figura 3.1 - Disciplinas que investigam a visão [TRIVEDI; ROSENFELD, 1989].

Outra definição bem aceita pelos pesquisadores da área é a de MARR (1982), que afirma: "visão é o processo que produz, a partir de imagens do mundo externo, uma descrição que é útil ao usuário e que não é repleta de informações irrelevantes" [MARR, 1982, p.31].

Uma definição mais informal e também mais polêmica da área de visão computacional é dada por TARR e BLACK (1994, p.65), que afirmam que o objetivo da área é a compreensão e modelagem de um sistema de visão de propósito geral em seres humanos e em máquinas. A crítica mais comum a essa afirmação é que o chamado "Sistema de Visão de Propósito Geral", assim como os termos "Problema Geral da Visão", "Problemas Mal Postos" e "Problema Genericamente Solucionável", não são bem definidos nem tampouco compreendidos [TSOTSOS, 1994] e que a maioria dos problemas em visão são NP-completo ou NP-hard, como provado por TSOTSOS (1989), KIROUSIS e PAPADIMITRIOUS (1985), COOPER (1992), entre outros. Além disso, TSOTSOS (1990) afirma que a visão humana não soluciona esses "problemas genéricos da visão".

Na década de 50 os primeiros pesquisadores da área acreditavam que o problema da visão seria resolvido rapidamente. A área porém se apresentou mais complexa que o imaginado inicialmente. Um dos problemas fundamentais é que visão é um problema mal posto, isto é, a imagem bidimensional de uma cena não possibilita que se construa

uma única descrição tridimensional da cena em questão, pois não existem equações geométricas suficientes para encontrar todas as incógnitas necessárias à reconstrução.

Assim, para resolver o problema visual, deve-se encontrar restrições que possam ser usadas para contorná-lo, tornando-o bem posto e possibilitando sua solução. Uma das características fundamentais que diferenciam os paradigmas existentes é a origem e grau das restrições usadas para resolver o problema visual.

A primeira teoria que tentou metodologicamente resolver o problema visual, restringindo-o, foi proposta por Marr em 1982.

3.2 A teoria de Marr

Segundo JOLION (1994), "David Marr foi o primeiro a propor (em seu livro chamado Vision [MARR, 1982]) uma metodologia completa para a visão computacional, que ficou conhecida como o paradigma de Marr".

Antes dele, Gibson foi o que mais se aproximou da proposição de uma teoria computacional para a visão, em 1966. "Sua maior contribuição foi abandonar o debate filosófico e apontar que o fato mais importante sobre os sentidos é que eles são canais para a percepção do mundo real - no caso da visão, as superfícies visíveis" [MARR, 1982, p.29].

A metodologia proposta por Marr consiste na divisão em "três níveis diferentes nos quais um dispositivo de processamento de informações deve ser compreendido" [MARR, 1982, p.24]. Neles estão contidos a teoria computacional do dispositivo (no nível superior), o algoritmo e tipo de representação usados (no nível intermediário) e os detalhes da implementação física (no nível inferior). JOLION (1994) acredita que esta metodologia de trabalho é uma das principais contribuições de Marr.

Teoria Computacional (nível superior)	**Representação e algoritmo (nível intermediário)**	**Implementação física (nível inferior)**
Qual é o objetivo da computação, por que ela é conveniente e qual é a lógica da estratégia que pode realizá-la?	Como esta teoria computacional pode ser implementada? Em particular, qual é a representação para as entradas e saídas e qual é o algoritmo para a transformação?	Como a representação e o algoritmo podem ser implementados fisicamente?

Tabela 3.1 - Os três níveis nos quais qualquer máquina realizando um processamento de informação deve ser compreendida. [MARR, 1982, p.25]

Marr ainda propôs a divisão da derivação das formas de um objeto a partir de uma imagem em três estágios de representação (de complexidade crescente) [MARR, 1982, p.37]:

1. O primeiro estágio, chamado esboço primário, representa as propriedades importantes da imagem bidimensional, como mudanças de intensidade e distribuição e organização geométrica. Algumas primitivas usadas para a

construção desta representação são: os cruzamentos de zero, terminações e descontinuidades, segmentos de bordas, linhas virtuais, grupos, organizações, curvas e fronteiras.

2. O segundo estágio, chamado esboço 2 ½D, representa algumas propriedades das superfícies visíveis em um sistema de coordenadas centrado no observador. Entre essas propriedades estão a orientação e distância do observador às superfícies visíveis (estimativas), contornos de descontinuidades destas grandezas, reflectância das superfícies e uma descrição aproximada da iluminação. Algumas primitivas usadas nesta representação são orientação local de superfícies, distância do observador, descontinuidades na profundidade e na orientação das superfícies.
3. O terceiro estágio, chamado modelo 3-D, representa a estrutura tridimensional e a organização espacial das formas observadas, usando uma representação hierárquica modular que inclui primitivas volumétricas e superficiais. As primitivas são modelos tridimensionais das formas, em um sistema de coordenadas centrado nos objetos, organizados hierarquicamente, cada um baseado em um conjunto de eixos no espaço, aos quais formas volumétricas ou de superfície são fixadas, junto com alguma informação sobre as propriedades das superfícies.

A terceira proposição de Marr é a da divisão do problema da visão em subproblemas independentes, tratados como módulos independentes, que é uma aplicação da metodologia tradicional de IA de “dividir para conquistar”. Com isso, diferentes módulos foram definidos, como estereoscopia, *shape from X*, análise de movimento, etc.

Finalmente, Marr realiza várias suposições sobre as características do mundo físico para poder restringir o problema de visão, como a existência de superfícies suaves, a existência de uma organização hierárquica da estrutura espacial, similaridade, continuidade espacial, continuidade das descontinuidades e continuidade do fluxo ótico. Esta assunção de restrições a partir do mundo físico é uma das características da teoria de Marr.

A importância da teoria de Marr para a disciplina de visão computacional é indiscutível. Para ALOIMONOS (1994), estas contribuições estabeleceram as bases que tornaram a área uma disciplina científica. Assim, a partir da teoria proposta por Marr, os pesquisadores da década de 80 começaram a trabalhar no que hoje chamamos de paradigma reconstrutivo [BLACK et al., 1993, JOLION, 1994].

3.3 O paradigma reconstrutivo

O paradigma reconstrutivo pode ser considerado uma interpretação da teoria de Marr, sobre a qual pesquisadores tiveram que fazer adições e outras suposições para poderem construir aplicações reais. Deve-se notar que, apesar do reconstrutivismo ter sua base na teoria de Marr, as duas teorias possuem diferenças.

De acordo com o paradigma reconstrutivo, o objetivo da pesquisa em visão é “a reconstrução em uma representação precisa de uma cena tridimensional observada e suas propriedades, a partir de informações contidas nas imagens, tais como: sombreamento, contornos, estereoscopia, cores, etc” [BLACK et al., 1993; JOLION, 1994]. Assim, sua finalidade principal está voltada para a reconstrução do mundo em

uma representação completa. Esta abordagem levou a muitas contribuições (principalmente teóricas), resultando em algumas técnicas matemáticas, por exemplo, relacionadas à descontinuidades e regularização.

Outra definição dos objetivos do paradigma reconstrutivo, formulada por TARR e BLACK (1994), consiste em "construir uma descrição simbólica de uma cena do mundo real em um modelo, que possa ser usado como interface para outros processos cognitivos - raciocínio, planejamento inteligente de atividades -, fazendo isto através de descrições funcionais do mundo visível" [TARR; BLACK, 1994].

O paradigma reconstrutivo justifica a necessidade de representações intermediárias como modo de reduzir o custo computacional da visão, afirmando que comportamentos complexos em ambientes desconhecidos só se tornam possíveis através do uso de representações poderosas. O reconstrutivismo normalmente objetiva a visão genérica (isto é, possibilidade de realizar comportamentos complexos) por uma abordagem de baixo para cima (*bottom-up*), baseada em dados (*data-driven*).

Por ter bases na teoria de Marr, este paradigma recebe todas as críticas feitas àquela [JOLION, 1994], além de outras específicas ao paradigma reconstrutivista [TARR e BLACK, 1994; EDELMAN, 1994]:

- não é possível sempre construir uma representação objetiva de uma cena;
- reconstruir o mundo em um modelo tão complexo quanto ele mesmo é inútil;
- não é sempre necessário construir um modelo tridimensional completo da cena para atingir um objetivo particular;
- a geometria não é o único modelo existente para representar o mundo e para sua compreensão;
- o paradigma de Marr é baseado fortemente na teoria de computação clássica de Von Neumann, com abordagem seqüencial;
- cenas não são tão simples como superfícies regulares;
- restrições podem surgir de outras fontes além do mundo externo - físico;
- devemos pensar também no que podemos obter da cena, não só o que deseja-se.
- os algoritmos de reconstrução não são robustos em presença de ruídos;
- os algoritmos são ineficientes;
- o paradigma reconstrutivista falhou em levar em conta a demanda computacional de um perceptor do mundo real;
- a reconstrução por si só não contribui em nada para a interpretação da cena;
- existem evidências psicofísicas que a hipótese reconstrutivista é inconsistente com a performance humana no reconhecimento;

Algumas destas críticas - as "dependentes de hardware" - estão sendo rebatidas por autores como TARR e BLACK (1994), que acreditam que elas são verdadeiras hoje devido somente ao estado da arte dos computadores, mas que melhoras já estão acabando com esses problemas e que "estamos entrando em um período estimulante no qual todas as críticas serão resolvidas" [TARR & BLACK, 1994]. Apesar disso, eles afirmam que essas críticas foram boas para repensar e redirecionar a pesquisa na área.

Estas críticas ainda merecem comentários:

- quanto à falta de robustez, argumentam que o uso de modelos probabilísticos propicia uma solução melhor para a reconstrução do modelo do mundo;
- para alguns problemas a referida ineficiência dos algorítmos é circunstancial e freqüentemente surgem algoritmos mais eficientes;
- quanto à subestimativa da demanda computacional, pode-se dizer que todos os pesquisadores sistematicamente subestimam o esforço necessário para se extrair alguma informação útil dos sensores;
- a inapropriação para interpretação de cenas só é verdade se é desejada uma interpretação semântica da cena, o que é difícil sob qualquer enfoque teórico;
- apesar do estudo dos modelos naturais ser interessante, pois proporcionam fonte de informações e questionamento, evidências psicofísicas contrárias por si só não invalidam a teoria: existem muitos exemplos onde a solução tecnológica encontrada não é igual à solução biológica e evolutiva, como os carros, que não possuem patas, e os aviões, que não batem asas.

Muitos pesquisadores acreditam que o paradigma reconstrutivo ainda possui fôlego, e "não só é promissor para atingir os objetivos da visão computacional, mas também possui boa fundamentação computacional e evolutiva" [TARR; BLACK, 1994]. Porém, outros pesquisadores acreditam que algo mais é necessário para resolver o problema de visão. Procurando este algo mais, novos paradigmas foram propostos. O primeiro a surgir foi a chamado visão ativa.

3.4 Visão Ativa, Animada e Qualitativa

Em 1988 surge uma nova proposta para se encontrar restrições a fim de tornar tratável o problema de visão. Nela, a visão controla o processo de aquisição de imagem "introduzindo restrições físicas que facilitam a reconstrução da informação da cena tridimensional" [ALOIMONOS, 1994]. O processo de visão passa a "procurar informação ativamente e pára de considerar que a informação seja captada acidentalmente (passivamente) pelo sensor" [BAJCSY; CAMPOS, 1992, p.31]. A este novo paradigma deu-se o nome de ativo ou visão ativa.

Este paradigma é baseado no trabalho de BAJCSY (1988), que definiu a percepção ativa como o estudo da modelagem e das estratégias de controle para percepção. Ela propôs uma nova metodologia para modelar um sistema perceptivo, fortemente influenciada pela teoria de controle. Nesta metodologia, modelam-se em três níveis os sensores, os módulos de processamento e a interação entre estes módulos. Esta modelagem pode ainda ser global, se ela é aplicada a um sistema como um todo, ou local, se ela modela algo específico, como distorções de uma lente, resolução espacial, filtros e etc.

BAJCSY ainda sugeriu uma nova proposta para a divisão da derivação das formas de um objeto, baseada no controle ativo dos sensores e na proposta original de Marr. Esta possui 5 etapas [BAJCSY, 1988, p.998]:

1. Controle dos dispositivos físicos. Os objetivos nesta etapa são foco grosseiro da cena, o ajuste da abertura, o foco no sujeito e a tentativa de encontrar distâncias a partir do foco.
2. Controle dos módulos visuais de baixo nível e o controle de um aparato binocular. Os objetivos são a segmentação da imagem 2-D, cálculo do número

de bordas e regiões da imagem.

3. Controle de um módulo visual geométrico. O objetivo é a construção do esquema 2 ½D.
4. Controle de processo de integração de várias vistas da cena e correspondência entre dados e modelos. O objetivo é a descrição de objetos tridimensionais na cena através de categorias volumétricas.
5. Controle da interpretação semântica. O objetivo é a descrição tridimensional da cena em um modelo que pode ter várias complexidades.

Muitos autores, como TARR e BLACK (1994), acreditam que a visão ativa não é um paradigma novo, totalmente diferente do reconstrutivo, mas “uma reformulação do paradigma reconstrutivo tradicional, onde somente se adiciona a exploração dinâmica do ambiente” [TARR; BLACK,1994, p.67], não abandonando todos os outros preceitos da teoria de Marr. Eles acreditam que o paradigma é “uma técnica promissora para desenvolver algoritmos robustos de visão e compreensão da visão biológica, pois aumenta a entrada de dados visuais e pode resolver problemas que são intratáveis no modo tradicional” [TARR; BLACK,1994, p.67].

Os defensores de visão ativa indicam que estas afirmações seriam verdadeiras somente se os sensores se movimentassem aleatoriamente. Mas como o paradigma incorpora na modelagem a intenção explícita do controle dos sensores, ele é fundamentalmente diferente do reconstrutivo. O controle da aquisição da informação para criar mais restrições, adquirindo mais imagens de posições diferentes, escolhidas pelo sistema, é a característica mais importante deste paradigma.

Estas novas idéias foram bem aceitas pela comunidade, que logo em seguida viu que as novas idéias ainda não tinham acabado. Assim, após o surgimento da Visão Ativa, o paradigma Animado foi proposto por Dana Ballard (1991). Neste paradigma, "o objetivo não é construir uma representação interna explícita e completa do mundo, mas exibir comportamentos que resolvam problemas” [BALLARD; BROWN, 1992, p.17].

Segundo Ballard “o comportamento visual mais importante é o controle de fixação (*gaze control*) ” [BALLARD, 1991]. O uso deste comportamento gera restrições adicionais que simplificam enormemente a solução do problema de visão.

Outra característica fundamental da visão animada é o uso exaustivo de aprendizado de comportamentos, pois ele reduz o custo computacional a longo prazo e raramente produz erros irreversíveis.

Algumas vantagens citadas por BALLARD (1991) em favor do paradigma são:

1. “sistemas de visão animada podem usar a procura física: o sistema pode mover as câmeras para chegar mais perto de objetos, mudar o foco ou o ponto de observação;
2. os sistemas podem realizar movimentos na câmera usando aproximações;
3. possibilita o uso de sistemas de coordenadas excêntricos;
4. possibilita o uso de algoritmos qualitativos;
5. o controle de fixação permite a segmentação da imagem em áreas de interesse;
6. possibilita a exploração do contexto ambiental;
7. a visão animada se adapta muito bem a algoritmos de aprendizado que usam referências indexadas” [BALLARD, 1991, p.62].

Neste item que trata das idéias que surgiram para tentar eliminar os problemas do reconstrutivismo, falta citar ainda um paradigma, o qualitativo. Nele, a idéia central é que “os comportamentos podem não necessitar de representações elaboradas do mundo tridimensional” [ALOIMONOS, 1992], aumentando assim o desempenho dos sistemas de visão que não precisam reconstruir todo o mundo físico.

Pode-se perceber que novas abordagem para solucionar o problema da visão foram introduzidas com os paradigmas apresentados acima, e foram se desenvolvendo quase paralelamente no tempo. Uma última abordagem é apresentada a seguir.

3.5 O paradigma propositado

O paradigma propositado (*purposive*) [ALOIMONOS, 1994; BLACK et al., 1993; JOLION, 1994; RIVLIN et al., 1991; SWAIN; STRICKER, 1991] surgiu como uma evolução e integração dos paradigmas ativo, animado e qualitativo, e às vezes, é confundido com algum desses. Além de reunir as idéias dos três paradigmas citados, ele propõe que “a visão deve ser considerada dentro de um contexto de tarefas que um agente deve realizar, e procura retirar dos propósitos do agente as restrições para solucionar o problema mal posto de visão” [ALOIMONOS, 1994]. Assim, a única justificativa para se selecionar o que inserir em um modelo do mundo real é a finalidade deste modelo.

Outra característica fundamental do paradigma é a grande integração da visão com outras áreas de inteligência artificial, como planejamento inteligente de atividades, raciocínio e aprendizado - resultado da crença que visão não é um problema contido em si, e que pode ser mais facilmente resolvido se for integrado com estes sistemas.

Deste modo, ao tentar solucionar o problema da visão, deve-se questionar o que se deseja reconhecer, levando a uma questão diretamente ligada às tarefas visuais, isto é, ao propósito. Assim, o pensamento propositado coloca questões cujas respostas levam a soluções de tarefas específicas e não de uso geral, tornando-se um paralelo à teoria computacional proposta pelos reconstrutivistas.

Autores como TARR e BLACK (1994), definem que o objetivo do paradigma “é construir sistemas para resolver tarefas específicas” e “o estudo das tarefas que organismos com visão podem fazer”.

Segundo ALOIMONOS (1994), “os trabalhos de Brooks e Marr estão nos finais opostos de um espectro onde a visão propositada se encontra no meio” [ALOIMONOS, 1994, p.74]. É evidente a influência dos trabalhos de BROOKS (1986 e 1991) no surgimento da visão ativa, qualitativa, animada e propositada, nas quais a decomposição em tarefas, comportamentos propositados e muitas outras características daquele trabalho são usadas.

A popularidade da abordagem propositada tem crescido bastante, uma vez que ela alcança melhores resultados práticos na interação entre robôs inteligentes e ambientes complexos.

A principal crítica da comunidade reconstrutivista em relação à propositada tem sido a de que soluções propositadas não possibilitam fácil expansão para tratamento de problemas mais sofisticados, questionando a possibilidade de adquirir comportamentos mais complexos com a simples montagem de comportamentos mais simples.

TARR e BLACK (1994) duvidam que qualquer sistema possa ser compreendido

em um contexto tão estreito como este, pois concordam com MARR (1982): “Em um sistema geral, restrições que são particulares a uma cena fazem com que o sistema aumente de complexidade para gerenciar o grande número destas restrições, ou seja, o aumento de módulos independentes - que surgem de restrições diferentes - aumenta a complexidade e que no ser humano esse número é muito grande” [MARR, 1982].

Outra questão que surge é quanto à provável duplicação de esforços ao se tentar construir módulos distintos para comportamentos diversos. Algumas outras críticas ao paradigma propositado, compiladas por TARR e BLACK (1994), são:

- “a visão propositada é enganadora, porque não possuímos ainda um conhecimento adequado de quando processamentos específicos para certas tarefas e críticos em relação ao tempo devem ser implementados e porque não compreendemos as computações que podem surgir da combinação de muitos processos;
- algumas suposições do paradigma propositado são apenas uma reformulação daquelas feitas pelo paradigma reconstrutivo;
- o propositivismo só serve para resolver problemas simples e casos particulares”

TARR e BLACK (1994) acreditam que o paradigma propositado defende uma mudança nos objetivos da área de visão computacional que é simplesmente um desvio para evitar o estudo dos problemas reais da área de visão computacional, que é “explicar e implementar comportamentos visuais complexos”. Acreditam que o paradigma defende o abandono da busca por sistemas de visão inteligentes, por não querer tratar problemas como sistema de visão de propósito geral e a visão humana, “desistindo dos objetivos de inteligência artificial, psicologia cognitiva e neurociências”. Por isso, eles afirmam que esta não é uma alternativa ao paradigma reconstrutivo.

Porém, já foram apresentadas muitas críticas contra reconstrutivistas e propositivistas. Assim, estabeleceu-se uma discussão entre reconstrutivistas e propositivistas, que tem resultado em artigos onde ânimos exaltados defendem suas abordagens como a mais apropriada para o estudo em Visão Computacional [TARR e BLACK, 1994; TSOTSOS, 1994; ALOIMONOS, 1994].

O propositivismo procura atingir a visão genérica através de uma abordagem de cima para baixo (*top-down*), orientada aos objetivos (*goal-oriented*), dirigida pelas tarefas (*task-driven*). Desta maneira, a visão de propósito geral emergirá de uma organização de diversas soluções dedicadas a diferentes tarefas visuais. Assim, o problema principal torna-se: como organizá-las e quais são as tarefas primitivas, voltando o enfoque para arquiteturas de integração dos sistemas visuais.

É na busca de soluções para este importante problema que se encontra o maior enfoque deste projeto. Para tentar resolvê-lo, foram estudados os algoritmos clássicos da área de visão computacional, apresentado no próximo capítulo.

4. TECNICAS DE SEGMENTAÇÃO DE IMAGENS EM VISÃO COMPUTACIONAL

Segundo BALLARD e BROWN (1982) “a idéia de segmentação tem suas origens nos psicólogos do grupo Gestalt, que estudavam as preferências exibidas por seres humanos no agrupamento ou organização de um conjunto de formas dispostas no campo visual”. Ainda segundo estes autores, segmentação de imagens é o termo usado em visão computacional para “o agrupamento de partes de uma imagem genérica em unidades que são homogêneas com respeito a uma ou várias características (ou atributos), que resulta em uma imagem segmentada”. [BALLARD, D.; BROWN, C., 1982, p. 116].

Existem duas maneiras básicas de realizar a segmentação de uma imagem:

- A análise baseada em similaridades das regiões da imagem. Regiões são definidas normalmente como áreas 2D conectadas, sem superposição (um pixel só pode fazer parte de uma única região).
- A análise baseada em descontinuidades na imagem, que utiliza as variações bruscas nos valores de intensidade dos pixels para particionar uma imagem.

A seguir são apresentados dois algoritmos que implementam estas abordagens diferentes para segmentação, apresentando antes alguns conceitos sobre captura e padrões de imagens.

4.1 Captura de Imagens

Um programa que capture uma imagem colorida (24 bits), no modelo de cor YUV, de tamanho variável, gera as seguintes imagens.

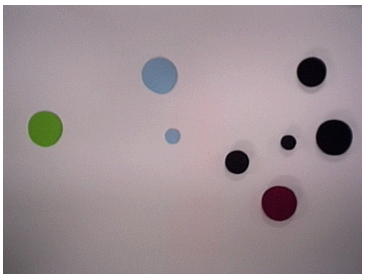
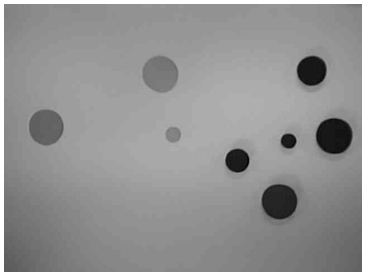
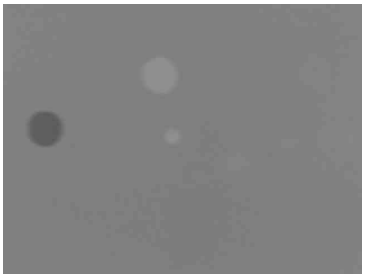


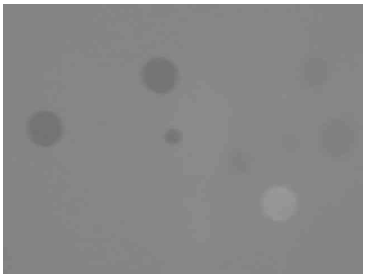
Imagem Composta



Y - Luminância



U - Crominância Azul



V - Crominância Vermelho

Figura 4.1 - Imagem composta capturada.

O modelo de cores YUV [LI, 1998] foi inicialmente usado no padrão PAL para

vídeo analógico e atualmente é usado como padrão para video digital, sendo que o JPEG e o MPEG são baseados em um modelo YUV modificado.

Este modelo define a imagem como sendo uma matriz de pontos, onde a cor de cada ponto é definido por 3 bytes: o primeiro define a luminância da imagem (Y), e os segundo (U) e terceiro (V) definem a crominância.

A luminância Y é definida a partir das cores vermelha, verde e azul, onde:

$$Y = 0.299 * \text{Vermelho} + 0.587 * \text{Verde} + 0.114 * \text{Azul}$$

A crominância é definida como a diferença entre uma cor de referência e um branco de referência para uma mesma luminância. Assim os valores de crominância são definidos para o azul e o vermelho.

$$U = \text{Azul} - Y$$

$$V = \text{Vermelho} - Y$$

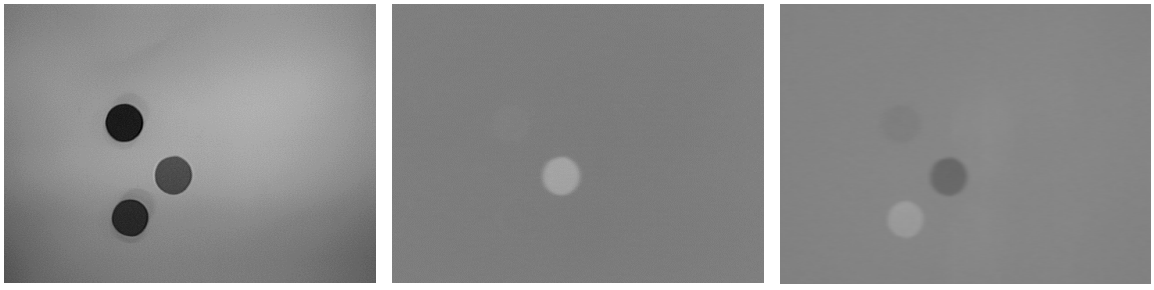
Neste modelo de cor uma imagem em níveis de cinza não possui crominância e U e V tornam-se igual a zero. Ainda, V varia do vermelho ao ciano e U do azul ao amarelo.

4.2 Segmentação baseada em similaridades: Blob Coloring

Uma maneira de realizar a segmentação é a fusão de regiões (*merging* ou *growing*). O algoritmo básico pode ser descrito como: inicie com regiões pequenas e uniformes (pixel) e siga fundindo regiões similares.

Uma implementação deste algoritmo pode ser feita com base em um algoritmo de rotulação de regiões em uma imagem do tipo *Blob Coloring,* descrito em (BALLARD, D.; BROWN, C., 1982), com adaptação para imagens coloridas.

O algoritmo implementado recebe uma imagem e cria uma lista contendo uma descrição da cena, com um “id” para cada objeto, seu tipo (dentre alguns tipos conhecidos), seu tamanho em pixels e a sua posição na imagem.



Y U V

Figura 4.1 – 3 componentes de cor de uma imagem capturada.

Ele trabalha em 3 etapas: primeiro rotula uma região; depois agrupa regiões segundo cor e proximidade em objetos; calcula a área e o centro de massa dos objetos; finalmente objetos desconhecidos (como sombras nos cantos da imagem) são retirados. O resultado da execução deste algoritmo sobre uma imagem é mostrado na Figura 4.2.

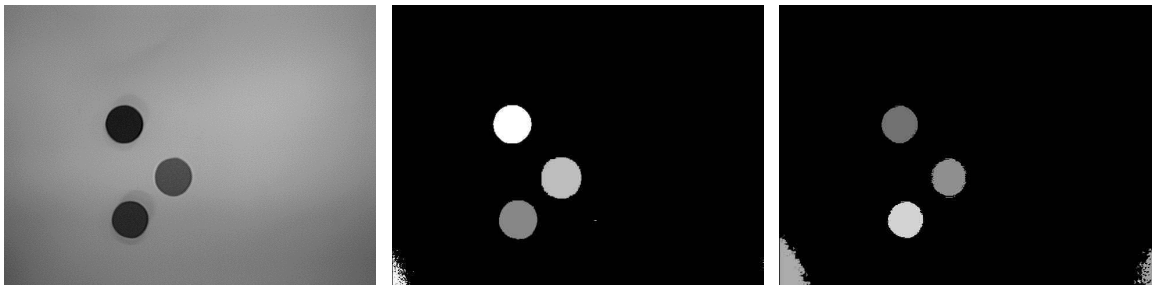


Imagem original Regiões Agrupadas Objetos na Cena

Figura 4.2 - Segmentação de uma imagem com *Blob Coloring*.

4.3 Segmentação baseada em diferenças: Vetorizador

O agente usa um algoritmo do tipo *chain code* (BALLARD, D.; BROWN, C., 1982), baseado no descrito em (RILLO, A., 1989).

Dada uma imagem capturada, de tamanho determinado, este algoritmo inicialmente filtra a imagem utilizando um limiar de cor, gerando uma imagem binária. Depois, encontra os contornos da imagem filtrada, utilizando um operador de Robert para encontrar as descontinuidades e finalmente, partir da imagem de descontinuidades, este algoritmo encontra todas as regiões da imagem que possuem um contorno fechado.

Descreve-se a seguir, os algoritmos que realizam cada parte deste processo.

4.3.1 Filtragem de Imagens por Cor (colorFilter)

A tarefa primitiva deste algoritmo é filtrar uma imagem colorida, de tamanho variável, construindo uma imagem binária. Ele recebe a imagem colorida e o valor de limiar de uma cor que se deseja filtrar no modelo YUV e atribui 1 às regiões da imagem onde a cor supere o limiar desejado e zero ao resto da imagem.

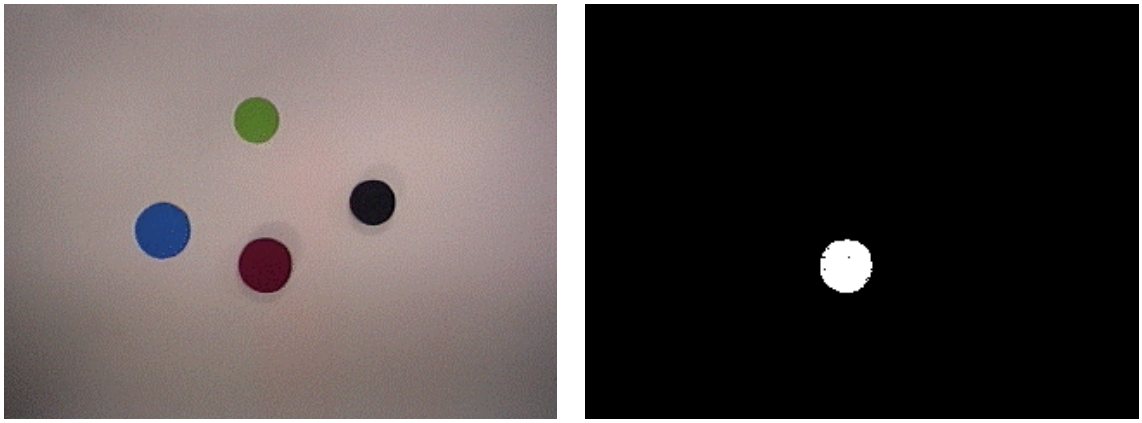


Imagem original Imagem binarizada

Figura 4.1 - Imagem binarizada para a cor vermelha.

4.3.2 Determinação de Contornos (edgeDetector)

Dada uma imagem binária, este agente determina os contornos existentes nela. É baseado no algoritmo descrito em (RILLO, A., 1989), que utiliza a definição de contornos proposta por KITCHIN e PUGH (1983). Ele verifica quatro elementos da imagem (máscara 2x2) para determinar se um ponto faz parte de um contorno ou não, criando uma nova imagem onde as bordas encontradas estão entre os elementos da imagem original.

A Figura 4.1 apresenta seis configurações básicas das dezesseis possíveis combinações de quatro elementos de uma imagem binária. As configurações na coluna esquerda não pertencem a regiões de contorno e por isso se atribui zero ao ponto correspondente da imagem de resultante. Os da coluna direita são pontos que pertencem a um contorno e resultam em um. As outras configurações possíveis podem ser obtidas pela rotação destas apresentadas.

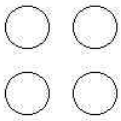
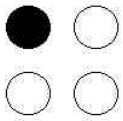
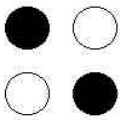
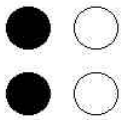
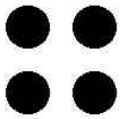
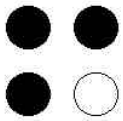
Elementos não pertencentes ao contorno (Resulta 0)	Elementos pertencentes ao contorno (Resulta 1)
	
	
	

Figura 4.1 – Definição dos elementos básicos de borda

Como este processo encontra o contorno entre os elementos da imagem, em vez de dobrar a resolução da imagem resultante - o que seria necessário para desenhar as bordas entre os pixels originais- esta é deslocada de meio pixel para a esquerda e para cima da imagem original, visando manter a resolução original.

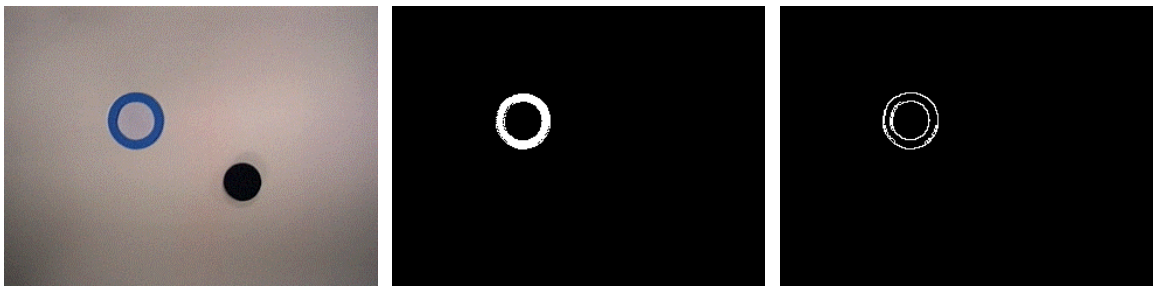


Imagem Capturada | Imagem Filtrada | Imagem dos Contornos

Figura 4.2 - Determinação de Contornos

A Figura 4.2 mostra uma imagem capturada onde existe um anel azul e uma peça preta, a imagem filtrada para a cor azul e o contorno encontrado.

4.3.3 Vetorizador (vetorizer)

Dada a imagem binária, de tamanho determinado contendo os contornos de uma imagem, este agente encontra todas as regiões da imagem que possuem um contorno fechado. Ele gera uma lista com a área, o tamanho do contorno mínimo (*chain size*), a posição do centro de área para todas as regiões fechadas da imagem. Ainda, este agente verifica se cada região possui uma forma circular ou não, utilizando a razão entre a área e o contorno calculado.

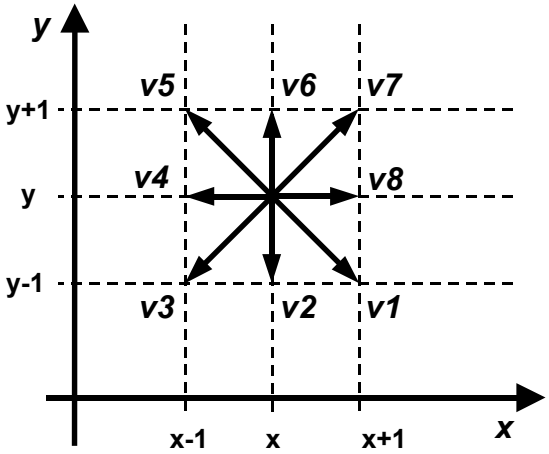


Figura 4.1 – Definição dos vetores elementares

O agente usa um algoritmo do tipo *chain code* (BALLARD, D.; BROWN, C., 1982), baseado no descrito em (RILLO, A., 1989) e em (KITCHIN, P. W.; PUGH, A., 1983). Este algoritmo considera que qualquer ponto em matriz de imagem está conectado aos oito pontos imediatamente a sua volta através de um vetor elementar, cuja direção foi rotulada (de v1 até v8) e que são apresentados na Figura 4.1.

O contorno de uma região pode ser determinado, assim como sua área calculada, através de um procedimento que segue os contornos de uma imagem, criando uma corrente de vetores ligados. Basta o ponto inicial da corrente e a seqüência dos vetores para se determinar precisamente uma região. Este procedimento segue o contorno mais externo, caso a borda possua mais de um pixel de espessura.

Para calcular a área de uma região é realizado um procedimento semelhante à integração, onde a área total é dada pela somatória da contribuição de cada vetor elementar. A Figura 4.2 mostra as contribuições dos vetores e a Tabela 4.1 apresenta os valores desta contribuição.

O perímetro de uma região pode ser encontrado facilmente através dos algoritmos baseados em *chain-codes*. O perímetro total é:

$$Perímetro = PerimetroPar + PerimetroÍmpar \cdot \sqrt{2}$$

isto é, a soma do número de vetores com rótulo par (que estão na horizontal ou vertical e por isso tem tamanho unitário) mais a soma do vetores com perímetro ímpar (que estão nas diagonais) vezes o tamanho destes, que é $\sqrt{2}$. O centro da região também

pode ser calculado, utilizando fórmulas para o cálculo de centro de área.

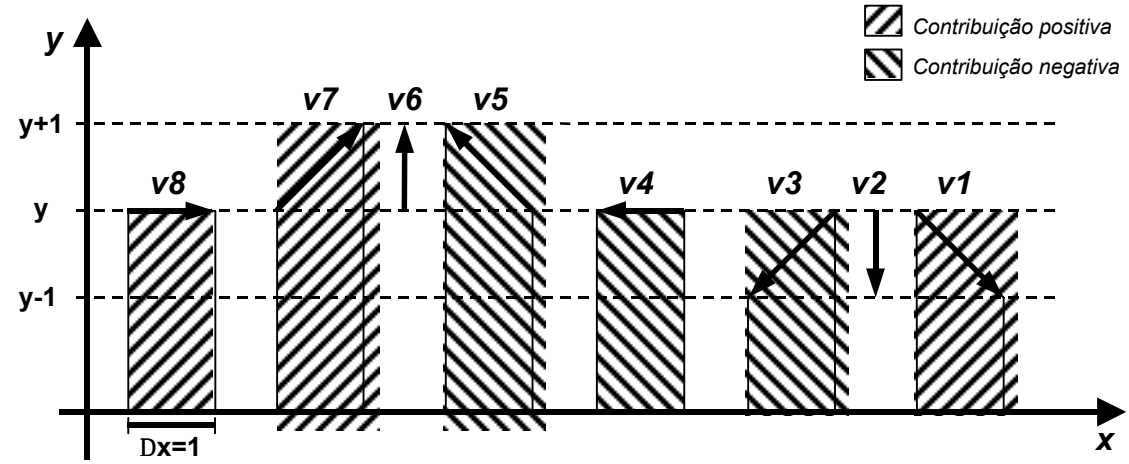


Figura 4.2 – Contribuição de cada elemento para o cálculo da área

Vetor de Direção	D **Área**
v8	y · 1
v7	(y + ½) · 1
v6	0
v5	(y + ½) · (-1)
v4	y · (-1)
v3	(y - ½) · (-1)
v2	0
v1	(y - ½) · 1

Tabela 4.1 – Valores da contribuição de área de cada elemento

A figura abaixo apresenta um exemplo de imagem vetorizada.

Imagem Capturada | Imagem Filtrada | Contornos | Imagem Vetorizada

Figura 4.3 - Vetorizador

5. SISTEMA VISUAL BASEADO EM SOFTWARE

5.1 Introdução

Esta seção descreve uma para a contrução de um sistema visual para robôs móveis baseada em software.

Para a realização deste trabalho foram implementados dois programas que, dada uma imagem previamente capturada e armazenada no disco rígido do computador, a segmentam baseados nos algoritmos *Blob Coloring* e Vetorizador.

O objetivo destas implementações é o de encontrar a bola - objeto esférico de cor laranja em imagens reais do domínio do futebol de robôs. Um exemplo das imagens que o sistema deve trabalhar estão na figura abaixo.

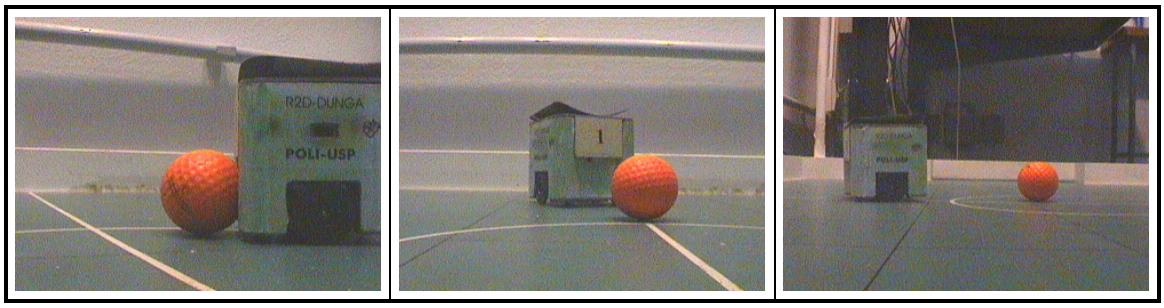


Figura 5.1 – Imagens típicas do domínio, capturadas *on-board*.

Entre os problemas que este domínio apresenta para a percepção o maior é que a câmera se encontra a bordo dos robôs e por este motivo, o sistema possui apenas uma vista parcial da área do jogo. A figura abaixo esquematiza a área visual de cada robô vista de cima e a Figura 5.3 apresenta a vista lateral.

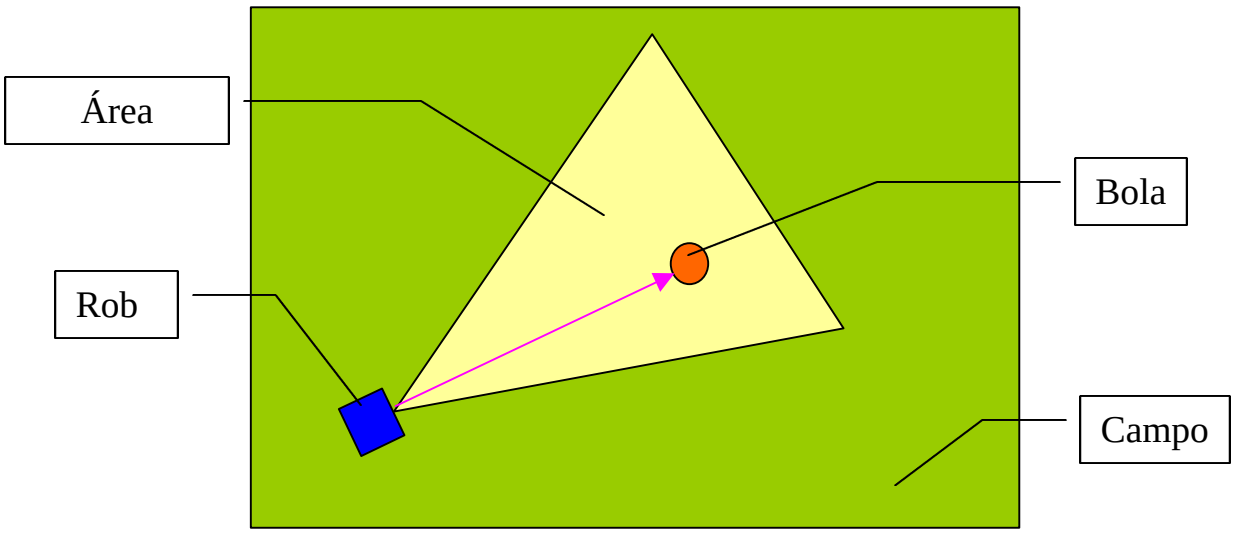


Figura 5.2 – esquema da área de visão de um robô.

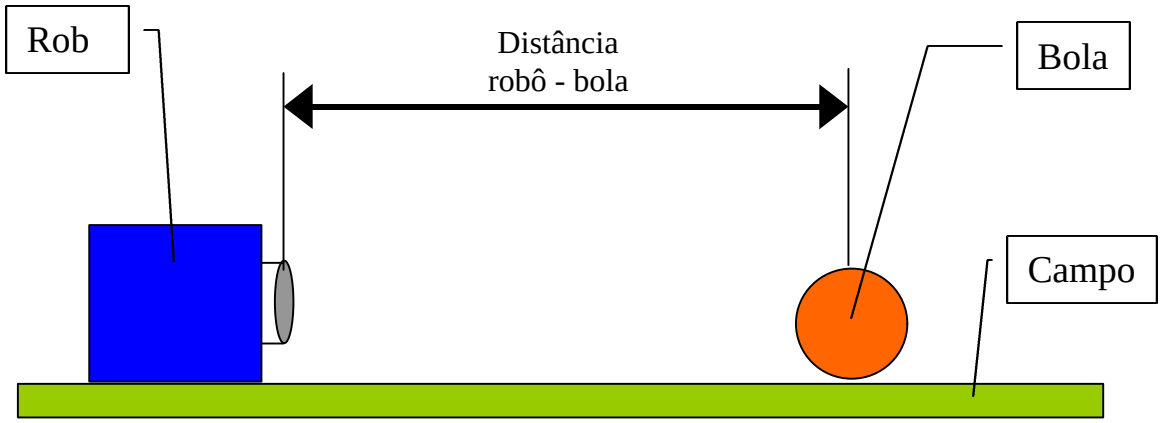


Figura 5.3 - esquema lateral da visão de um robô.

Os algoritmos foram implementados em ambiente Intel Pentium utilizando o sistema operacional Linux Conectiva 4.0 e linguagem C++ com compilador gcc. Além dos programas que implementam a segmentação foram criados diversos programas de suporte, como bibliotecas para abertura de imagens padrão PPM e programas de captura de imagem (este em ambiente Windows).

A próxima seção apresenta os resultados obtidos e a comparação entre os resultados dos dois algoritmos de segmentação implementados.

5.2 Análise do algoritmo Blob Colouring

Um dos problemas encontrados com o algoritmo de segmentação de regiões semelhantes baseado no *Blob Colouring* é que o resultado da segmentação depende muito do limite permitido para que dois pixeis de valores de cor diferente sejam considerados da mesma cor. Este limite para a semelhança é necessário pois em uma imagem real, mesmo as regiões semelhantes possuem pequenas variações na intensidade da cor de seu elementos.

Nas imagens contidas nas duas páginas seguintes é mostrado o resultado da segmentação da imagem abaixo (Figura 5.1) quando o limite em que se considera um pixel da mesma cor de outro varia de 5 níveis de qualquer um dos componentes até 50. A coluna da esquerda mostra as regiões encontradas e a coluna da direita mostra os objetos (regiões conectadas de área maior que 50 pixeis) encontrados. Além disso, o tempo de processamento e o número de regiões também é apresentado.

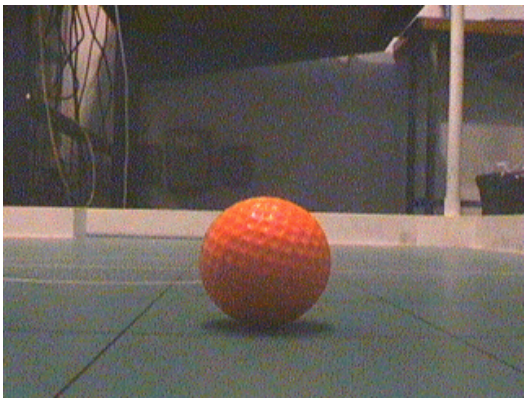
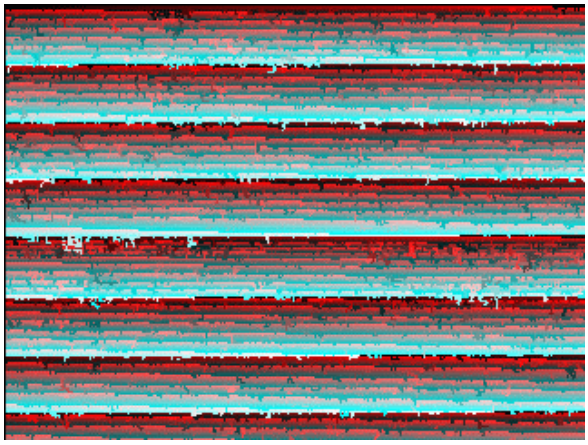
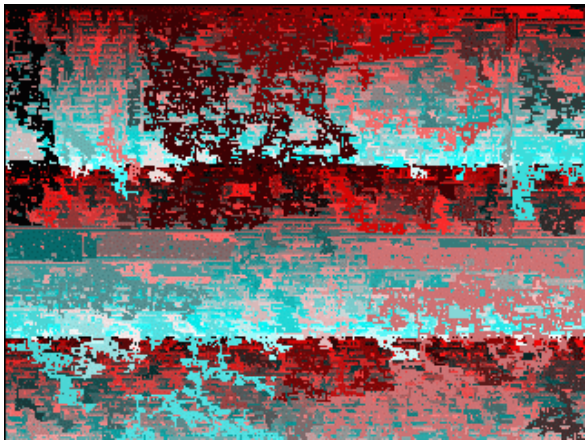
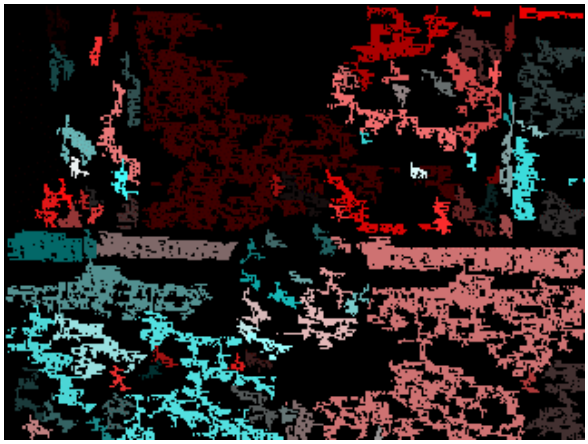
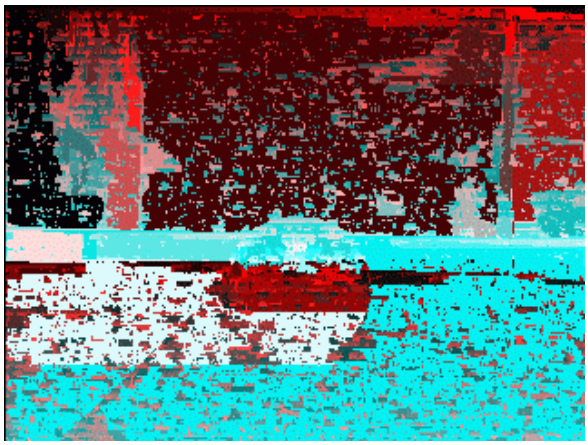
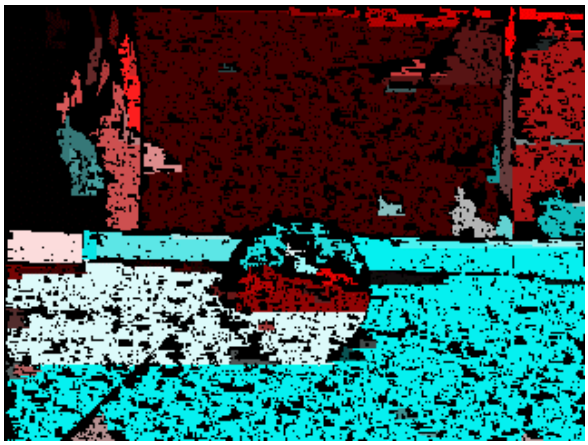


Figura 5.1 - Imagem da bola a 20 cm de distância

Regiões	***Objetos***
Diferença permitida = 5 Tempo de processamento: 1,32 segundos - Número de regiões encontradas: 49076	
	*Não existem objetos nesta imagem.*
Diferença permitida = 10 Tempo de processamento: 3,02 segundos - Número de regiões encontradas: 17557	
	
Diferença permitida = 12 Tempo de processamento: 3, 36 segundos - Número de regiões encontradas: 11047	
	

Segue →

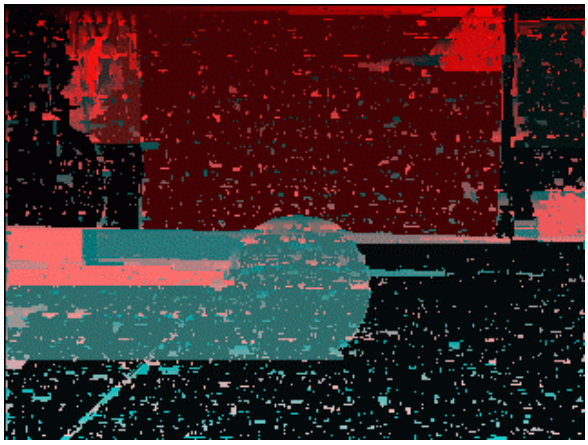
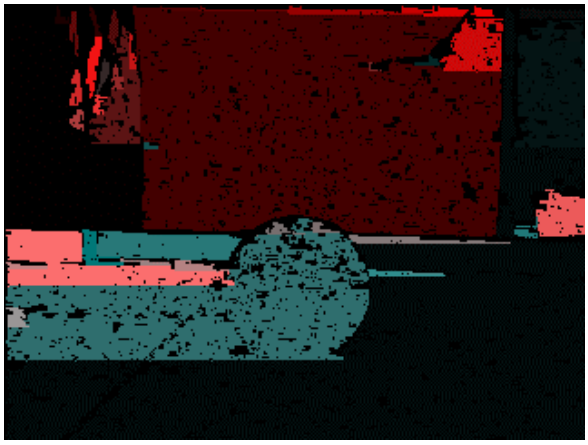
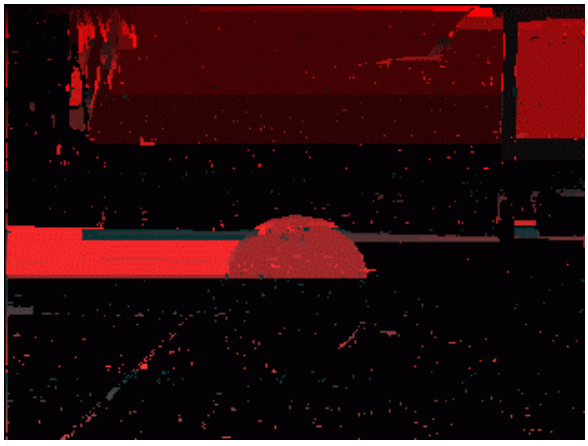
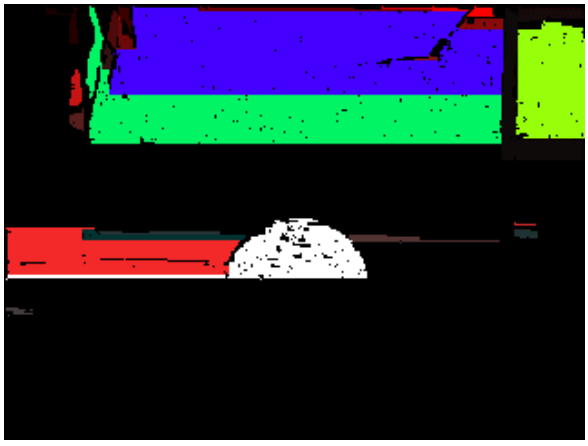
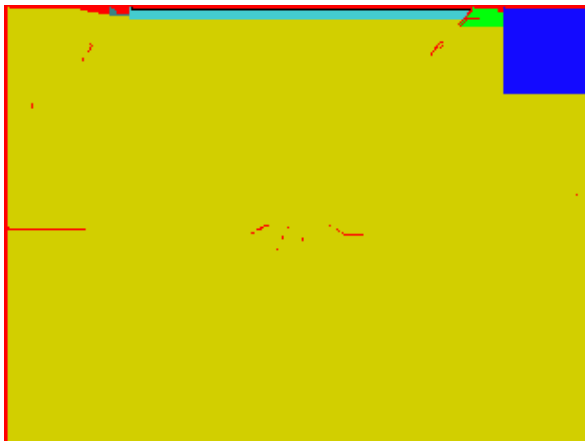
Regiões	***Objetos***
Diferença permitida = 15 Tempo de processamento: 3,9 segundos - Número de regiões encontradas: 5411	
	
Diferença permitida = 20 Tempo de processamento: 4,91 segundos - Número de regiões encontradas: 605	
	
Diferença permitida = 50 Tempo de processamento: 4,9 segundos - Número de regiões encontradas: 34	
	*Não existem objetos nesta imagem.*

Figura 5.2 – Resultados de segmentação x diferença de cor permitida.

A partir desta figura podemos concluir que:

- Quando o limite permitido para que dois pixeis de valores de cor diferente sejam considerados da mesma cor é pequeno (por exemplo, 5), a quantidade de regiões na imagem se torna enorme (~10^4) o que torna impossível encontrar objetos.
- Quando o limite permitido para que dois pixeis de valores de cor diferente sejam considerados da mesma cor é grande (por exemplo, 50), a quantidade de regiões na imagem diminui, tendendo até se tornar uma só, o que também torna impossível encontrar objetos na imagem.
- A grande variação das cores da bola, proveniente da iluminação, não permite que esta seja encontrada: ou ela é segmentada em diversas regiões diferentes (caso da diferença permitida ser igual a 12) ou ela é agrupada ao fundo (caso a diferença seja igual ou maior que 15).

Com diversos testes deste algoritmo em várias seqüências de imagens foi possível concluir que, devido a grande variação das cores dos objetos, esta abordagem não permite encontrar a bola (objeto principal do futebol de robôs) em uma imagem. A imagem abaixo mostra o problema para imagens com a bola e o robô.

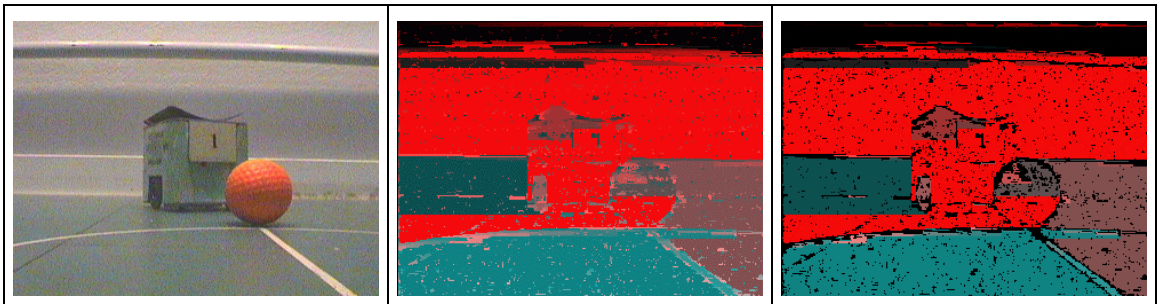


Imagem original	*Imagem com regiões*	*Imagem dos objetos*

Figura 5.3 – Resultado de imagem com robô.

5.3 Análise do algoritmo Vetorizador

O algoritmo de segmentação de imagens baseado em *Chain-Code* apresentou melhores resultados que o *Blob Coloring*. Apesar disso, este algoritmo não conseguiu encontrar a bola inteira, apenas a região central desta. Isso se deve novamente à grande variação de intensidade luminosa na imagem: a parte superior está muito clara e a inferior muito escura para serem incluídas na segmentação da bola. A Figura 5.1 mostra o resultado de uma segmentação.

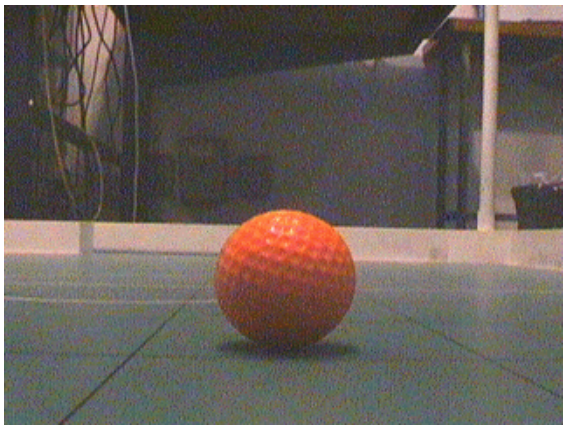


Imagem original

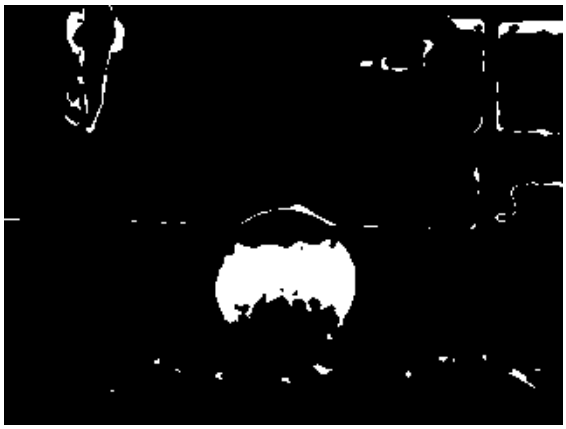


Imagem filtrada (suavização e limiarização).

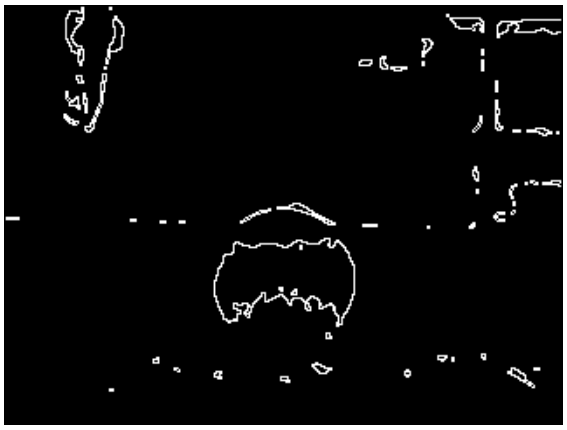


Imagem das bordas encontradas

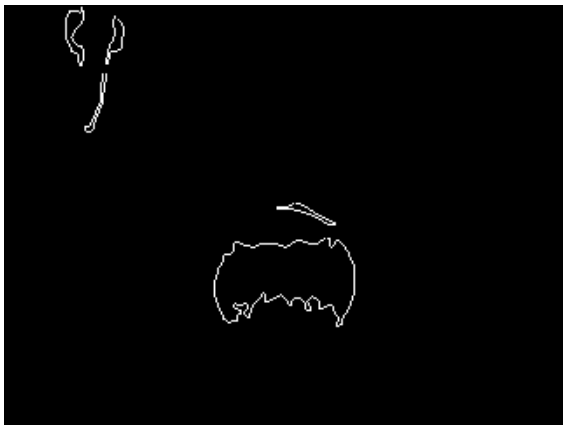


Imagem dos chain-codes encontrados.

Figura 5.1 – Resultado da segmentação por vetorização

Apesar de não serem os resultados esperados, a aplicação deste algoritmo em imagens da bola em distâncias diferentes da câmera permitiu se encontrar as laterais da bola, o que possibilitou localizar a bola na imagem ($C_x = X_{direita} - X_{esquerda}$).

Além disso, o diâmetro permite a criação de uma tabela com a variação do diâmetro da bola pela distância (Figura 5.2). Esta tabela permite que se calcule a função que relaciona a largura da bola encontrada na imagem com a distância desta da câmera.

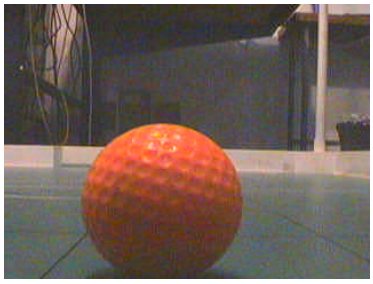
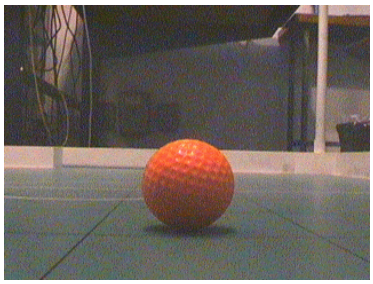
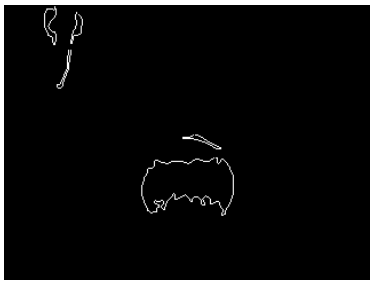
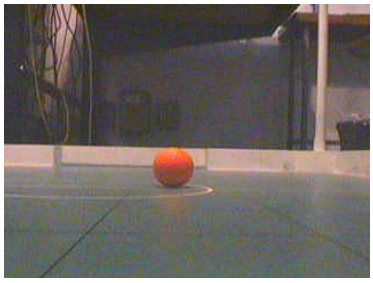
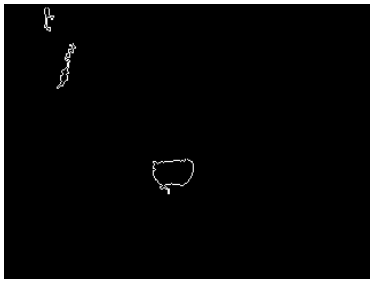
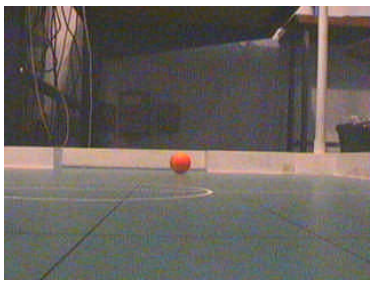
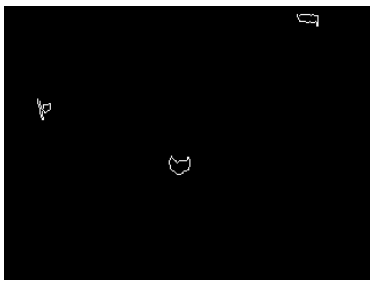
Imagem Original	*Imagem Chaincodes*	*Distância / Diâmetro*
		Distância :10 ± 0,5 cm Diâmetro: 143 ± 2 pixeis
		Distância :20 ± 0,5 cm Diâmetro: 86 ± 2 pixeis
		Distância :50 ± 0,5 cm Diâmetro: 36 ± 2 pixeis
		Distância :100 ± 0,5 cm Diâmetro: 18 ± 2 pixeis

Figura 5.2 – Distância x Diâmetro de diversas imagens.

5.4 Comparação entre os algoritmos Blob Coloring e Vetorizador.

Para a comparação de eficiência entre as implementações dos algoritmos *Blob Coloring* e vetorizador foi utilizado um computador Pentium 200 com 48 Mbytes de memória RAM. A Tabela 5.1 apresenta os resultados dos tempos de execução dos dois algoritmos, levantados para diversas imagens em 2 tamanhos.

Tamanho da imagem	***Blob Colouring* (segundos)**	**Vetorizador (segundos)**
160 x 120	1,22	0,04
320 x 240	de 1,32 até 4,91	de 0,14 até 0,16

Tabela 5.1 – Comparação de performance entre os algoritmos.

A partir desta tabela podemos notar que o algoritmo vetorizador foi mais eficiente que o *Blob Coloring*. Apesar disto, nenhum dos dois foi eficiente o suficiente para a aplicação no domínio do futebol de robôs, onde o tempo máximo de processamento é de 0,0167 segundos.

A seguir são apresentadas as conclusões deste capítulo e os possíveis caminhos para sua extensão.

5.5 Conclusões e trabalhos futuros

Os diversos testes realizados com os algoritmo de segmentação de imagens por agrupamento de regiões *Blob Coloring* e por diferenças de regiões Vetorizador com *Chain-Code* em várias seqüências de imagens foi possível concluir que:

- Devido à grande variação da intensidade das cores dos objetos, o algoritmo de *Blob Coloring* não encontra a bola (objeto principal do futebol de robôs) em uma imagem.
- Novamente devido à variação da intensidade das cores, o algoritmo vetorizador não consegue segmentar a bola perfeitamente, mas consegue a localizar na imagem e descobrir a distância desta até o robô.
- A partir dos resultados de eficiência medidos é possível concluir que o algoritmo vetorizador é mais eficiente que o *Blob Coloring*. Apesar disto, nenhum dos dois foi eficiente o suficiente para a aplicação no domínio do futebol de robôs, onde o tempo máximo de processamento é de 0,0167 segundos.

Como possíveis trabalhos futuros deve-se estudar algoritmos mais robustos e eficientes para localizar a bola na imagem, como os dedicados ao rastreamento de objetos, os que envolvem Redes Neurais Artificiais (para diminuir a influência da variação de cores) e algoritmos baseados em Lógica Nebulosa.

6. SISTEMA VISUAL BASEADO EM HARDWARE

Esta seção descreve uma outra solução a contrução de um sistema visual para robôs móveis. A solução discutida é a implementação de algoritmos de visão computacional em hardware programável . Ela está dividida em quatro partes principais. A primeira descreve a linguagem VHDL. A segunda, descreve a implementaçãso e os algoritmos e a terceira apresenta um exemplo de implementação em VHDL. Finalmente, apresentamos resultados e a conclusão deste capítulo.

6.1 VHDL

VHDL (Bhasker, 1995) é a abreviação de VHSIC Hardware Description Language (VHSIC é a abreviação de Very High Speed Integrated Circuits). VHDL é uma linguagem de descrição de hardware que pode ser usada para modelar um sistema digital, desde uma simples porta lógica até um sistema digital completo.

A necessidade pela linguagem aconteceu em 1981 no programa VHSIC. Neste programa um grande número de empresas dos Estados Unidos desenvolvia CIs para o departamento de defesa (DoD), cada uma usando uma linguagem de descrição diferente das outras. Então surgiu a necessidade de uma linguagem padronizada que todos pudessem usar.

Em 1983 um grupo de três companhias, IBM, Texas Instruments e Intermetrics, assinou um contrato com o DoD para desenvolver uma versão da linguagem.

A versão 7.2 do VHDL foi desenvolvida e apresentada ao público em 1985. Após isso houve uma crescente necessidade de fazer com que a linguagem se tornasse um padrão da indústria. Consequentemente, em 1986, a linguagem foi transferida para o IEEE para padronização.

Depois de avanços substanciais na linguagem, feitos por um grupo de indústrias, universidades e representantes do DoD, a linguagem tornou-se padrão pelo IEEE em dezembro de 1987 (IEEE Std 1076-1987). A linguagem passou a ser reconhecida como um padrão do American National Standards Institute (ANSI).

De acordo com as regras do IEEE, um padrão do IEEE precisa ser reformulado a cada 5 anos para que possa continuar a ser um padrão. Consequentemente a linguagem foi atualizada com novos recursos, a sintaxe da maioria das construções foi uniformizada, e muitas ambigüidades presentes na versão de 1987 da linguagem foram resolvidas. Esta nova versão da linguagem é conhecida como IEEE Std 1076-1993.

O DoD, desde setembro de 1988, exige que todos os seus fornecedores de Circuitos Integrados de Aplicação Específica (ASIC) apresentem descrições VHDL de seus circuitos de dos subcomponentes.

VHDL permite para o projetista implementar a arquitetura através de três métodos diferentes: o método de Fluxo de Dados, o método Comportamental e o método Estrutural.

No método de fluxo de dados o projetista define declarações Booleanas que são atribuídas a sinais. Estas declarações são executadas concorrentemente e são usadas para implementar funções Booleanas simples que dependem de um ou vários sinais binários.

O método de behavioral define processos que contêm declarações entre um par de Etiquetas “Process” e “End process”. As declarações dentro de um processo são

executadas consecutivamente e trabalham da mesma maneira que qualquer programa escrito em uma linguagem procedimental como C ou Pascal.

Finalmente, o método estrutural permite ao projetista descrever as ligações elétricas do circuito e conecta vários "componentes virtuais" dentro do FPGA. É usado para definir a interconexão lógica dos componentes de circuito.

Estes três modos podem ser combinados no chamado "modo misturado", onde o projetista pode implementar parte do sistema como um circuito digital e outra parte como um algoritmo, como foi feito neste projeto e que é descrito na próxima seção.

6.2 O sistema implementado

O sistema implementado é composto pelas seguintes partes (Figura 6.1):

- Aquisição da Imagem
- Binarização
- Determinação de contornos, com extração das bordas.
- Vetorização, com cálculo de informações sobre a imagem.
- Envio dos dados

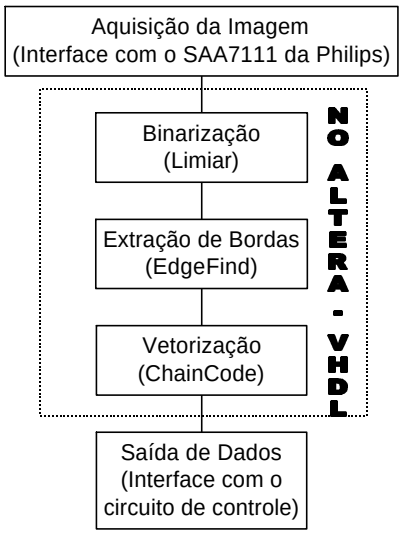


Figura 6.1 – Diagrama de Blocos do sistema implementado em Hardware.

6.2.1 Aquisição da imagem

Para a construção do sistema em hardware é necessário encontrar uma forma de extrair imagens diretamente de uma câmera de vídeo. Para isso foi utilizado um decodificador de vídeo da Philips que faz a conversão do sinal da câmera para o padrão RGB, realizando a aquisição da imagem.

O SAA7111 é um CI produzido pela Philips que tem como função receber uma entrada de vídeo analógica e transformá-la em uma saída digital com vários padrões diferentes, à escolha do usuário.

A entrada do processador de vídeo é feita por uma combinação de dois canais analógicos de pré-processamento. Estes canais possuem uma fonte de seleção, um filtro

“anti-aliasing”, um controle de ganho automático, um circuito gerador de clock (CGC), um decodificador digital multi– padrões (PAL BGHI, PAL M, PAL N, NTSC M e NTSC N), possui um circuito que controla o brilho, contraste, saturação e uma matriz espacial de cores. O SAA7111 aceita como entrada analógica os sinais CVBS ou S-vídeo (Y/C) retirados da TV ou VTR.

A saída do decodificador de vídeo pode ter vários padrões, como por exemplo, YUV (12-bit), YUV (16-bit), YUV (8-bit), RGB (16-bit), RGB (24-bit). Todos estes sinais são digitais.

A imagem utilizada pelo sistema tem resolução de 320 linhas, 240 colunas e 16 bits de cores. Ela é gerada por uma câmera de vídeo comum, capturada e decodificada pelo SAA7111. Neste trabalho foi utilizado como entrada o padrão NTSC M e como saída o padrão RGB de 16 bits.

A interface do decodificador com o FPGA que implementa a visão é feita através de 16 pinos de entrada (RGB16) da seguinte forma: 10 pinos são utilizados para as cores vermelho e azul (5 para cada) e 6 para o verde.

Este circuito é controlado pelo protocolo de vídeo I2c-bus, descrito a seguir.

6.2.2 O protocolo I²C-bus

O I^2C-bus (Inter Integrated Circuit-bus) é um protocolo de comunicação que foi desenvolvido pela divisão de semicondutores da Philips no início da década de 80. Este protocolo tem por objetivo criar um caminho mais simples para se conectar a CPU de um micro computador a um chip através da comunicação serial bidirecional, composta por dois cabos: a linha serial de dados (SDA) e a linha serial de clock (SDL).

Cada componente conectado é reconhecido pelo software com um único endereço e uma simples relação mestre/escravo é quem faz a comunicação. O mestre pode operar como transmissor ou receptor. Durante a transmissão podemos enviar mensagens sem restrição do número de bytes a serem enviados.

Para podermos configurar o CI utilizando a comunicação serial cridas três sub-rotinas que manipula os sinais SDA e SDL. Essas rotinas são:

- I2C_START = inicia transmissão
- I2C_SEND_BYTE = manda 1 byte
- I2C_STOP = termina transmissão

Para se configurar todos os parâmetros do CI são utilizadas diversas sub-rotinas em seqüência, repetindo o I2C _SEND_BYTE tantas vezes quanto necessário para enviar um bloco completo consistido por:

- Identificador (Device ID) do SAA7111;
- Endereço do primeiro parâmetro a ser programado
- Valor do primeiro parâmetro.
- Valor do parâmetro seguinte.
- Configurações restantes.

6.2.3 Binarização da Imagem

Após ser capturada, a imagem é convertida para bitmap preto e branco através de

um sistema de limiar de cores (como descrito na seção 4.3.1), que filtra e transfere para a imagem monocromática somente uma cor predefinida. Esta imagem (monocromática) é armazenada em memória para ser processada.

6.2.4 Determinação dos contornos

A extração de bordas descrita na seção 4.3.2 é implementada aqui. A imagem é analisada em partes constituídas de quatro pontos cada uma, assim sendo analisa-se primeiro os dois primeiros elementos da primeira linha, juntamente com os dois primeiros da segunda, depois os segundo e terceiro elementos da primeira e da segunda linha, e assim por diante até o final da linha. Quando termina a linha, repete-se o mesmo processo para a segunda e a terceira linha e assim por diante até que tenha encerrado as linhas.

Durante esta fase, a imagem binaria armazenada anteriormente é seqüencialmente substituída pela imagem contendo as bordas, para diminuir a necessidade do uso de memória.

6.2.5 Identificação de Objetos na Imagem

A vetorização da imagem (como descrita na seção 0) é realizada neste módulo. O resultado do módulo anterior, armazenado na área de memória reservada para a imagem, é passado para a função chaincode que transforma cada borda da imagem em uma seqüência de vetores, a partir dos quais elabora uma matriz com as informações de posição e tamanho de cada objeto.Ele faz uma varredura na imagem gerada pelo primeiro algoritmo, que contém apenas as bordas dos objetos, e cria uma matriz de vetores (chain code) para cada objeto que possua um contorno fechado. O resultado deste algoritmo aplicado à área onde se encontra a bola é mostrado a seguir:

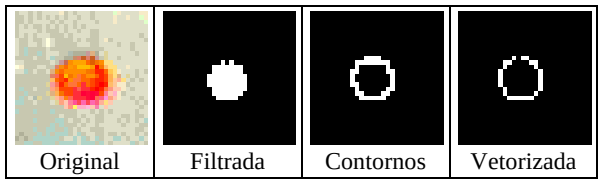


Figura 6.1 – Resultado de uma imagem Analizada pelo sistema, em suas diversas fases.

Ao final desta análise encontramos as informações sobre cada objeto, que neste caso é toda seqüência de vetores que forma um contorno fechado. Estes dados são enviados para a porta de saída do circuito.

6.2.6 Saida dos dados

Para realizar a saída dos dados foi elaborado um pacote de dados. Os dados são colocados na saída em pacotes de 32 bits, sendo 9 para a posição do objeto em relação ao eixo X, 9 para o eixo Y e 14 para o tamanho do contorno mínimo (*chain size*). Esta informação pode ser enviada a outro módulo de FPGA, como o módulo de estratégia ou um módulo que transmita a para um computador através de uma porta de comunicação serial RS-232.

6.3 Transposição dos algoritmos para VHDL

Ao implementar o projeto, os três modos para escrever uma especificação de VHDL, descritos na seção 2, foram combinados no chamado “modo-misturado”, onde o projetista pode implementar parte do sistema como um circuito digital e outra parte como um algoritmo. Como os algoritmos implementados no sistema têm um tamanho considerável, nós apresentamos aqui a simplificação do dispositivo de limiarização.

A seguir se encontra um exemplo de uma descrição em VHDL que implementa um dispositivo de limiarização de uma imagem, no caso um filtro “passa-baixa”. Foi usado o método de fluxo de dados.

```
-- Esta parte descreve os sinais de entrada e saída
Entity Threshold is
   PORT (  R, B:          IN BIT_VECTOR(0 TO 4);
           G:             IN BIT_VECTOR(0 TO 5);
           I:             OUT BIT
 );
End Threshold;

-- Esta parte defina a funcionalidade do dispositivo
Architecture Threshold of threshold Is
BEGIN
   I <= not (R(4) or B(4) or G(5));
End Threshold;
```

A primeira parte do código define os sinais de entrada e saída, onde há 5 bits para os sinais vermelhos e azuis e 6 bits para o sinal verde e apenas um para o sinal de saída. A segunda parte definea função do sistema: um dispositivo que aceitará só cores que possua os componentes vermelho, verde ou azul menos da metade da intensidade máxima. Se algum dos bits mais significativos estiver com valor verdsadeiro, o sinal de saída estará desligado.

6.4 Resultados

Estes algoritmos foram testados em um Pacote de Laboratório do Programa Universitário ALTERA, que inclui Software de desenvolvimento MAX+PLUS II (Edição de Estudante), uma placa de desenvolvimento educacional e um cabo para carga ByteBlaster.

O software MAX+PLUS II é um sistema de desenvolvimento que permite o desenvolvimento através de sistemas gráficos ou de uma linguagem de descrição de hardware, compilação e verificação do projeto, e programação de dispositivo. A placa de desenvolvimento educacional (Figura 6.1) possui dois dispositivos lógicos programáveis e reconfiguráveis. Finalmente, o ByteBlaster permite a carga de programas do MAX+PLUS na placa.

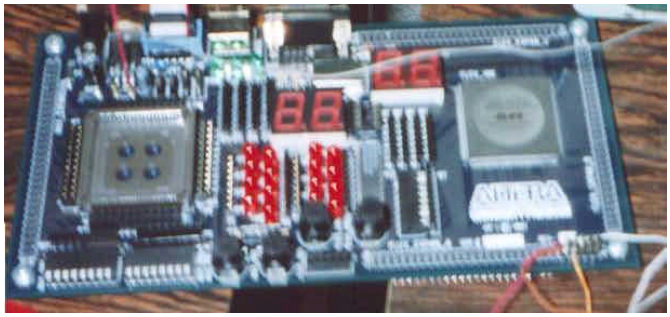


Figura 6.1 - A placa de desenvolvimento ALTERA onde o sistema foi testado.

Abaixo podemos ver um trecho da carta de tempo da fase de extração de bordas. O número de clocks gastos para a execução desta fase é o número de linhas da imagem acrescido de um, pois o método de implementação utilizado consegue processar uma linha inteira a cada ciclo. Como o clock utilizado para o processamento da imagem é de 100 MHz e nossa imagem tem 240 linhas, temos um total de 2,41µs para cada imagem.

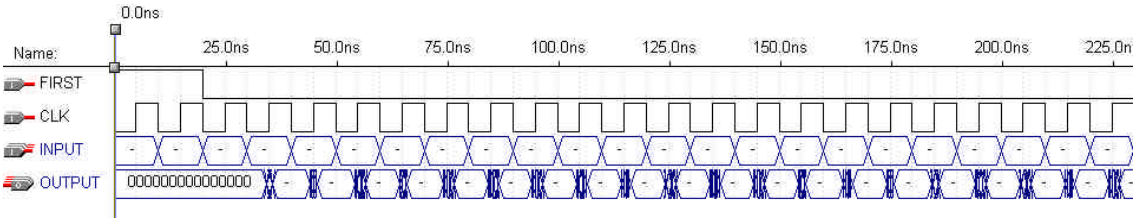
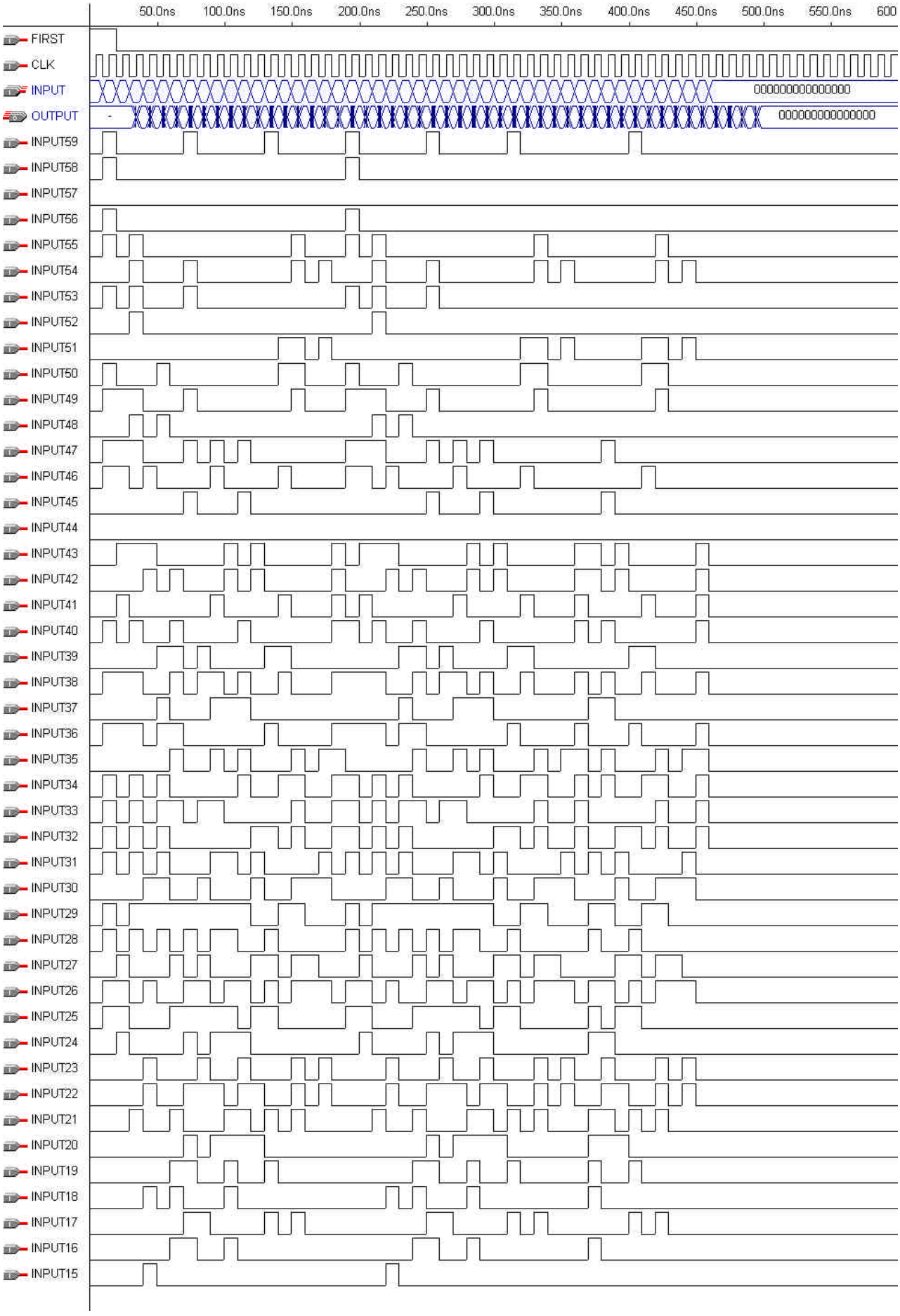


Figura 6.2 – Carta de tempo.

O sinais na carta de tempo são:

- CLK – clock do sistema;
- INPUT – vetor de entrada que é a representação de uma linha inteira;
- OUTPUT – vetor de saída no mesmo formato da entrada;
- FIRST – sinal de controle de sincronismo;

Um exemplo de carta de tempo com todas as entradas para uma imagem de 60x45 encontra-se a seguir.


50.0ns
100.0ns
150.0ns
200.0ns
250.0ns
300.0ns
350.0ns
400.0ns
450.0ns
500.0ns
550.0ns
600
FIRST
CLK
INPUT
OUTPUT
INPUT59
INPUT58
INPUT57
INPUT56
INPUT55
INPUT54
INPUT53
INPUT52
INPUT51
INPUT50
INPUT49
INPUT48
INPUT47
INPUT46
INPUT45
INPUT44
INPUT43
INPUT42
INPUT41
INPUT40
INPUT39
INPUT38
INPUT37
INPUT36
INPUT35
INPUT34
INPUT33
INPUT32
INPUT31
INPUT30
INPUT29
INPUT28
INPUT27
INPUT26
INPUT25
INPUT24
INPUT23
INPUT22
INPUT21
INPUT20
INPUT19
INPUT18
INPUT17
INPUT16
INPUT15

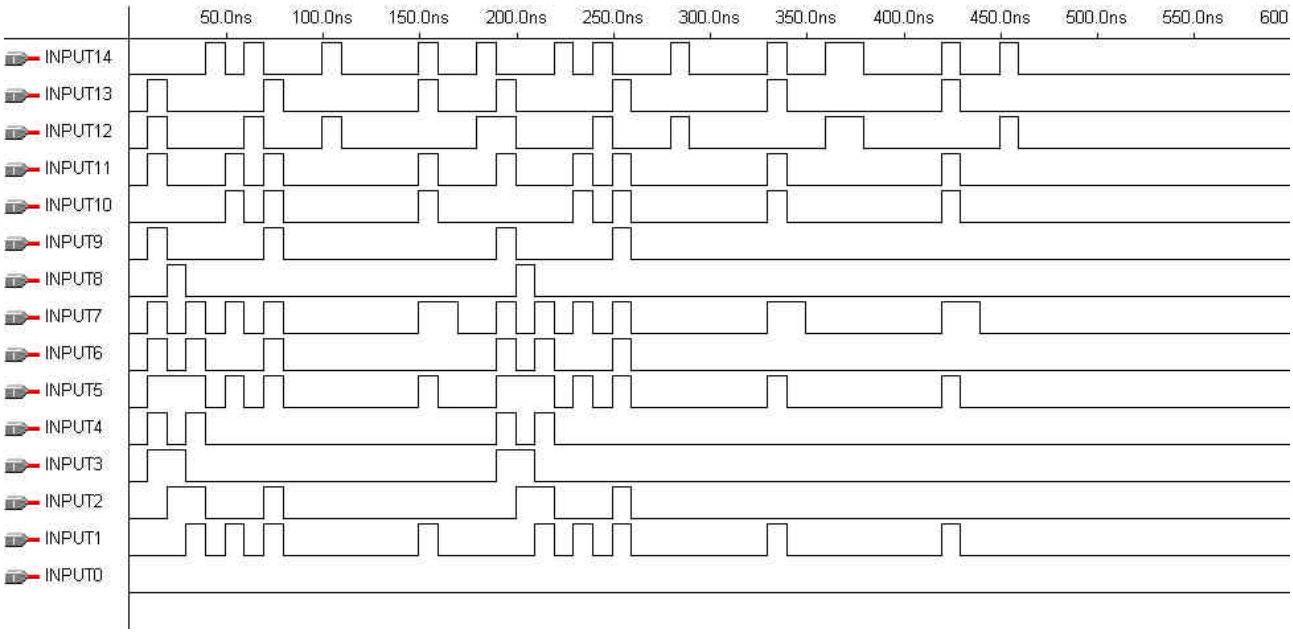


Figura 6.3 - Carta de Tempo mostrando a entrada de dados no CI Altera Flex.

A tabela abaixo mostra o tempo necessário para o processamento de cada imagem, o que nos mostra que, teoricamente, o sistema poderia processar aproximadamente 649 imagens no formato utilizado por segundo. Os mesmos algoritmos, quando implementados em linguagem C e ambiente Linux, leva 33ms para processar cada imagem, ou seja, 21,5 vezes mais tempo.

Fase	***Tempo***
Captura e Binarização	770μs
Extração de Bordas	2,41μs
Vetorização	770μs
Tempo total até os dados estarem disponíveis na saída	**1,54ms**

Tabela 6.1 – Tempo necessário para cada módulo.

Assim sendo, quem determinará o número de quadros processados por segundo poderá ser a câmera, o circuito de captura ou o circuito que processará os dados, já que é certo que todos eles terão capacidade inferior à apresentada pela fase de tratamento e identificação de objetos na imagem.

6.5 Conclusão

Uma comparação da eficiência do sistema implementado em hardware com os mesmos algoritmos implementados em linguagem C, os quais já forneciam um desempenho muito bom (da ordem de 30 quadros por segundo), mostrou que o primeiro sistema alcançou um desempenho superior com a mesma qualidade. Associado com o baixo custo de componentes de FPGA (menor que uma placa Vídeo for Windows), o

sistema resultante é um dispositivo de pequeno tamanho que serve bem para robôs autônomos.

Através dos resultados obtidos pode-se concluir que a implementação em Hardware Programável torna o sistema extremamente eficiente e que essa eficiência não pode ser aproveitada plenamente pelo conjunto do robô, pois os componentes restantes não conseguem alcançar o mesmo nível de desempenho.

A avaliação das imagens resultante permitiu concluir que a qualidade dos resultados obtidos também foi satisfatória.

Como trabalho futuro em VHDL deseja-se implementar outros algoritmos de Visão Computacional em VHDL, como por exemplo, o *Blob Colouring* [BALLARD E BROWN, 1982].

7. SISTEMA DE CONTROLE

O sistema de controle de um robô decide, a partir da sua percepção, quais ações devem ser realizadas. Para estudar estes sistemas geralmente são implementados simuladores do ambiente que se deseja estudar.

Atualmente na Robocup existe a liga de simulação, onde é utilizado um sistema cliente-servidor que gera um campo virtual e controla as ações dos jogadores. Um cliente comanda um jogador no campo e controla suas ações. As comunicações entre o cliente e o servidor são feitas por UDP/IP. Os clientes podem ser escritos em qualquer sistema de programação que tenha interface UDP/IP. O Servidor de Futebol Multiplataforma pode ser usado em sistemas como SunOS 4.1.x, Solaris 2, DEC OSF/1, NEWS-OS, Linux and Irix 5. Com base nesta simulação foi implementado o sistema de controle descrito neste capítulo, baseado em programação Genética.

7.1 Introdução aos Algoritmos Genétimos (AG) e à Programação Genética (PG)

7.1.1 Evolução e Seleção

De acordo com a Teoria da Evolução de Darwin, os organismos mais adaptados ao mundo em que vivem são os que terão maiores chances de sobrevivência.

Os organismos que se tem hoje são conseqüência da evolução de outros organismos inferiores que se extinguiram e que sem eles, provavelmente, não existiriam. Cada criatura nessa cadeia é o produto de uma série de “acidentes” que têm acontecido continuamente sobre a pressão seletiva do ambiente, ou seja, é fruto do ambiente em que vive e não o contrário. Por muitas gerações, a variação aleatória de características e a seleção natural formaram o comportamento de indivíduos e espécies para que estes suprissem as demandas do seu ambiente.

Segundo Koza (1992), o processo evolutivo ocorre na natureza quando quatro condições básicas são satisfeitas:

- “Um indivíduo tem a habilidade de se reproduzir;
- Existe uma população desses indivíduos;
- Existe alguma variedade entre esses indivíduos;
- Alguma diferença em habilidade de sobrevivência está associada com a variedade.”

Em PG será aproveitado o fato de que a evolução na natureza é criativa, pois produz muitas vezes resultados inesperados, impensáveis e não-lineares, diferente do modo de programação usual. Este fato refere-se às formas de evolução utilizadas em PG como Reprodução, Cruzamento e Mutação.

7.1.2 Fitness

Fitness é um parâmetro que mede o “encaixe” da espécie em seu ambiente. Existem várias formas de *fitness*, a usada é: “Quanto maior o *fitness* do indivíduo maior sua probabilidade de sobrevivência e adaptação ao ambiente”. Com este dado obtido, os indivíduos podem ser organizados de forma descendente, então são eliminados os piores

e os melhores continuam para serem reproduzidos ou cruzados.

O *fitness* na Programação Genética é usado para que os piores indivíduos sejam descartados.

7.2 Programação Genética

7.2.1 Indivíduo

Em PG o indivíduo é definido como uma árvore (representações de árvores serão mostradas a seguir) de funções e terminais, que constituem suas características e comportamento no ambiente para o qual foi desenvolvido, cada função é um ramo e cada terminal uma folha do indivíduo.

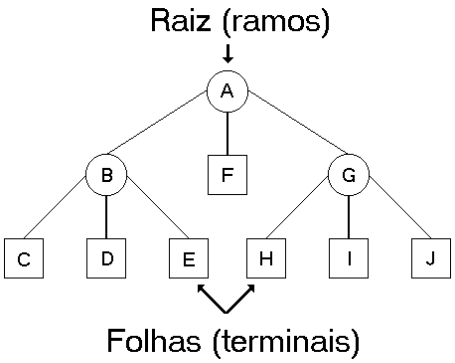


Figura 7.1 – Exemplo de Indivíduo

Funções: podem ser condições, sensores, operações aritméticas, booleanas, etc. São usadas funções para captar informações sobre o ambiente (mundo) e, convenientemente, para melhorar o desempenho de cada indivíduo.

Terminais: geralmente são variáveis ou constantes. Os terminais serão usados para fazer o indivíduo agir no ambiente.

7.2.2 Reprodução

Os melhores indivíduos da geração atual são simplesmente copiados para a nova geração. O objetivo é que os melhores indivíduos não sejam perdidos.

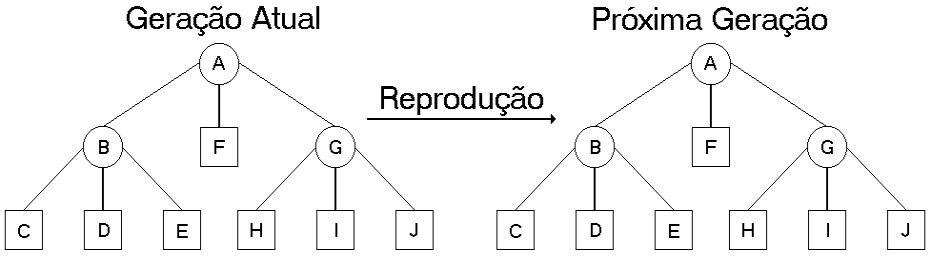


Figura 7.1 – Exemplo de Reprodução

7.2.3 Cruzamento

São escolhidos dois indivíduos (pais) a partir do *fitness*, selecionando pontos de cruzamento, e deles criados dois descendentes que vão estar na próxima geração. O objetivo é que algo de novo seja criado usando o que há de melhor, os pais também são preservados para a próxima geração.

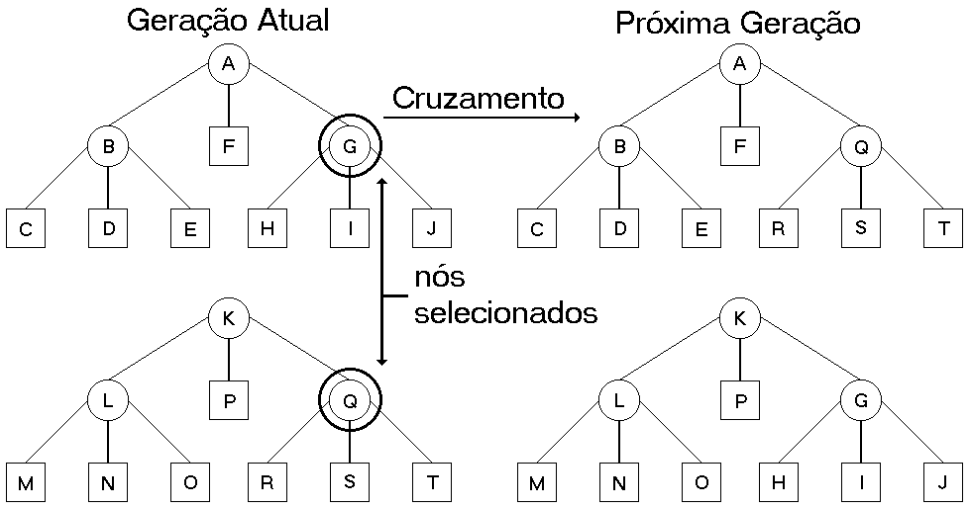


Figura 7.1 – Exemplo de Cruzamento

7.2.4 Mutação

Um indivíduo é selecionado, depois um ponto de mutação, e a partir dele é criado um novo ramo. O objetivo é óbvio: melhorar um indivíduo já bom, mas com isso ele pode perder desempenho e ser eliminado. Observação: isso não impede que o indivíduo seja salvo antes de ser aplicado à mutação, o que normalmente acontece.

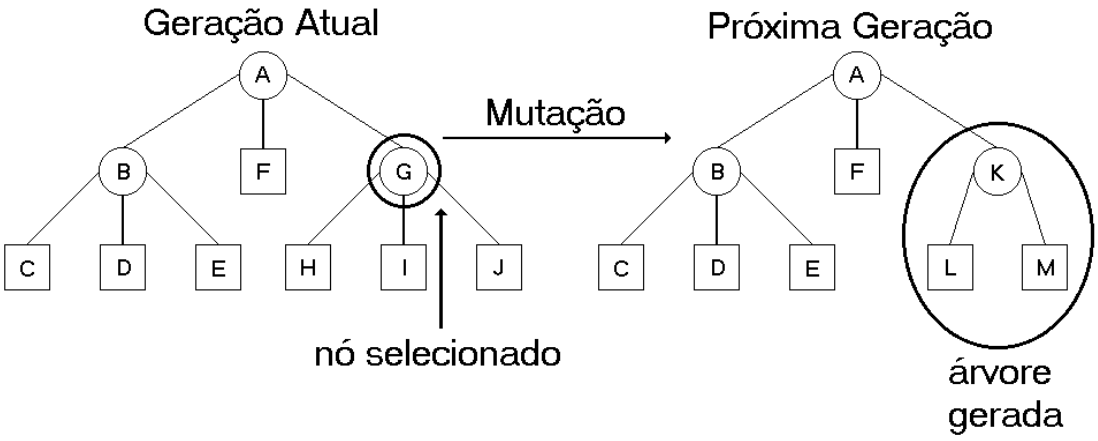


Figura 7.1 – Exemplo de Mutação

A mutação é pouco utilizada porque não melhora, relevantemente, o desempenho normal de uma execução.

7.3 Algoritmo Genético

7.3.1 Indivíduo

Em AG os indivíduos (exemplos serão mostrados a seguir) são representados por códigos de comprimento fixo chamados de cromossomos, geralmente o cromossomo é dividido em partes que identificam características importantes do indivíduo, neste exemplo ele é representado em código binário, mas também podem ser usados números reais ou até mesmo o alfabeto.

0010	1110	0001	0011	1011

Figura 7.1 - Exemplo de Cromossomo

A forma e o comprimento do cromossomo são a forma de representação do indivíduo. Características que devemos escolher para o obtermos o melhor resultado final. O cromossomo acima tem comprimento igual à ‘20’ e é dividido em cinco características.

7.3.2 Reprodução

Funciona da mesma forma que a reprodução em PG.

0010	1110	0001	0011	1011

Geração Atual

0010	1110	0001	0011	1011

Próxima Geração

Figura 7.1 – Exemplo de Reprodução

7.3.3 Cruzamento

São escolhidos dois indivíduos que serão os pais. Então é escolhido apenas um ponto de cruzamento, pois o comprimento dos indivíduos é fixo e depois disso é efetuada a troca dos códigos.

		Ponto de Cruzamento		
0010	1110	0001	0011	1011

0011	1111	0011	0011	1010

Geração Atual

0010	1110	0011	0011	1010

0011	1111	0001	0011	1011

Próxima Geração

Figura 7.1 – Exemplo de Cruzamento

Observação: O ponto de cruzamento pode ser qualquer um entre ‘1’ e o comprimento do cromossomo menos ‘1’ (L – 1).

7.3.4 Mutação

É necessário apenas um indivíduo que, depois de escolhido o ponto de mutação, é aplicada uma das seguintes mudanças:

Complemento: o valor da característica selecionada aleatoriamente é complementado.

0010	1110	0001	0011	1011

Geração Atual

0010	1110	1110	0011	1011

Próxima Geração

Figura 7.1 – Exemplo de Complemento

Incremento: O valor da característica selecionada aleatoriamente é incrementado.

0010	1110	0001	0011	1011

Geração Atual

0010	1110	0010	0011	1011

Próxima Geração

Figura 7.2 – Exemplo de Incremento

Decremento: O valor da característica selecionada aleatoriamente é decrementado.

0010	1110	0001	0011	1011

Geração Atual

0010	1110	0000	0011	1011

Próxima Geração

Figura 7.3 – Exemplo de Decremento

Finalmente, apresentamos na figura a seguir um Fluxograma de Funcionamento da Programação genética e dos Algoritmos Genéticos.

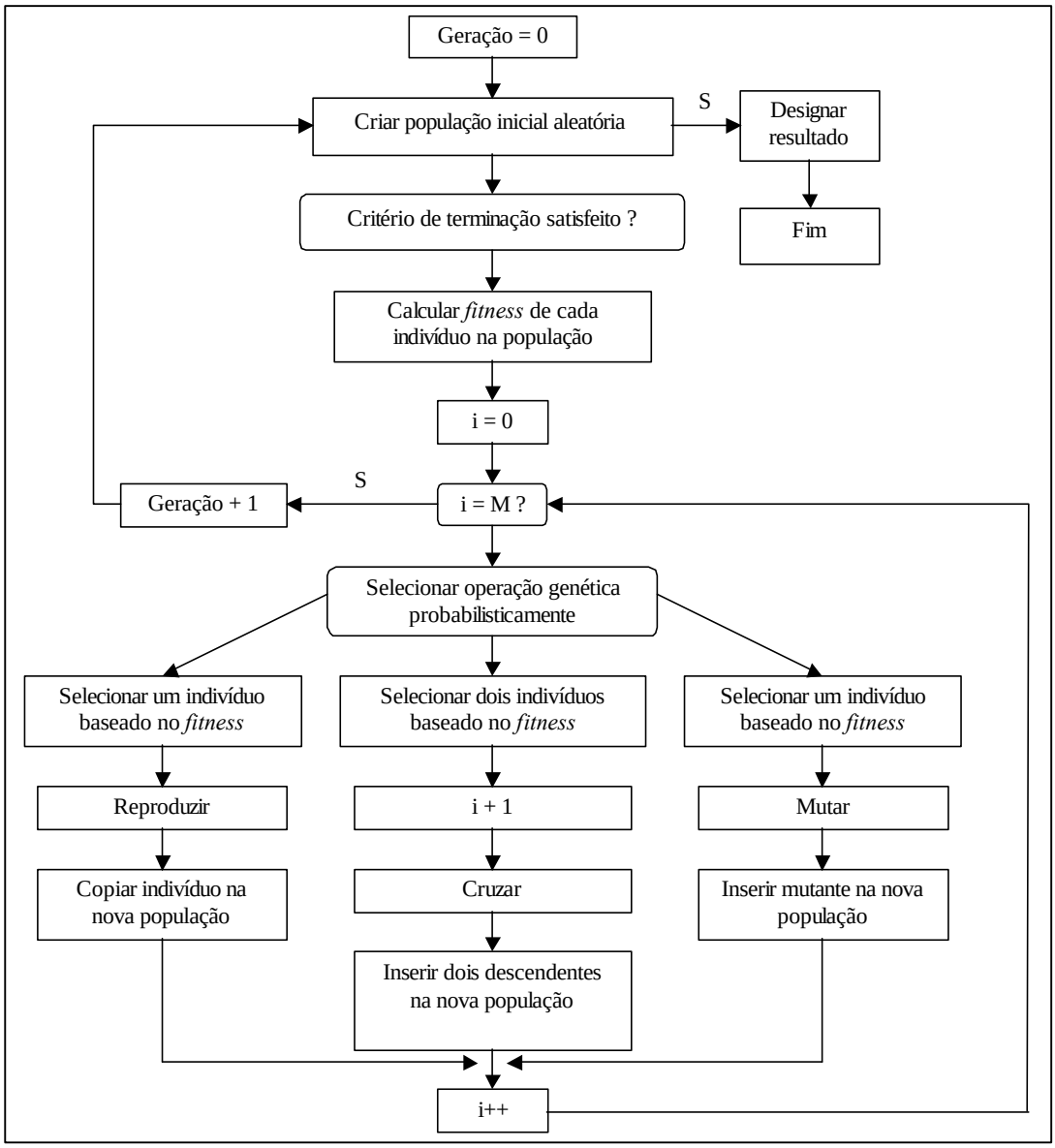


Figura 7.4 – Funcionamento da AG e da PG

7.4 Programas Implementados

7.4.1 Agente seguidor de paredes com visão global e ambiente restrito

A implementação de um agente Seguidor de Paredes é necessária por se tratar de um comportamento básico de movimentação de agentes inteligentes no domínio do Futebol de Robôs e por ser possível o aproveitamento das características desenvolvidas nesse agente para criar um time de futebol.

O objetivo principal desse comportamento é desenvolver um indivíduo que seja capaz de percorrer toda a lateral do ambiente de simulação no tempo estipulado.

Com as características da PG foi criado o programa do robô seguidor de paredes no qual o robô deve visitar o maior número de quadrados adjacentes à parede dentro de um ambiente de simulação em um número máximo de passos. É considerado um passo cada vez que o robô executar um terminal (agir), e é considerada uma execução cada vez que a árvore completa do robô for percorrida uma vez, ou seja, existem vários passos dentro de uma execução e opta-se por não pará-la antes que termine, por isso têm-se robôs com mais passos, mas não com menos.

O indivíduo deve percorrer a sala sempre próximo à parede e completar sua volta em no máximo ‘200 passos’.

A cada simulação o indivíduo começa em posições e direções diferentes para evitar que ele percorra o caminho automaticamente sem verificar o ambiente em que está.

Fitness

O *fitness* é calculado da seguinte forma:

```
fitness = fit;
```

Na qual ‘fit’ é o número de posições do caminho ideal que o robô visitou nas cinco execuções.

O *fitness* máximo é ‘300’ e representa o número de células do caminho ideal, ou seja, se um robô conseguir percorrer o caminho ideal sem se desviar muito da sua trajetória, certamente terá um *fitness* igual ou próximo ao máximo.

Funções

- PROGN3 (3): executa três ramos em seqüência;
- PROGN2 (2): executa dois ramos em seqüência;
- IFWALL (I): executa seu ramo esquerdo se não for detectada parede pelo robô (a no máximo 1 passo de distância) e caso contrário executa seu ramo direito.

Terminais

- WALKFRONT (F): faz robô dar um passo à frente;
- WALKBACK (B): faz robô dar um passo para trás;
- RIGHT (R): faz robô virar à direita;
- LEFT (L): faz robô virar à esquerda.

Outros Parâmetros

- População (M) = 500;
- Número de Gerações = 71;
- Probabilidade de Cruzamento = 70%;
- Probabilidade de Reprodução = 30%;
- Matriz de simulação = 20 x 20;
- Posicionamento do robô em números inteiros;

- Limite de complexidade (no sorteio) = 120;
- Ângulo de virada = 90º;
- Número de execuções por indivíduo = 5.

Resultados

Gráficos Evolutivos

O gráfico abaixo ilustra a evolução dos indivíduos ao se passarem ‘71’ gerações de reprodução e cruzamento. Pode-se observar que na geração ‘69’ obteve-se o primeiro indivíduo com *fitness* máximo designado igual à 300. As oscilações devidas a aleatoriedade utilizada na medição do *fitness*. Por fim, a média da geração que também melhora, mas se mantém mais constante e estabiliza por volta da geração ‘62’.

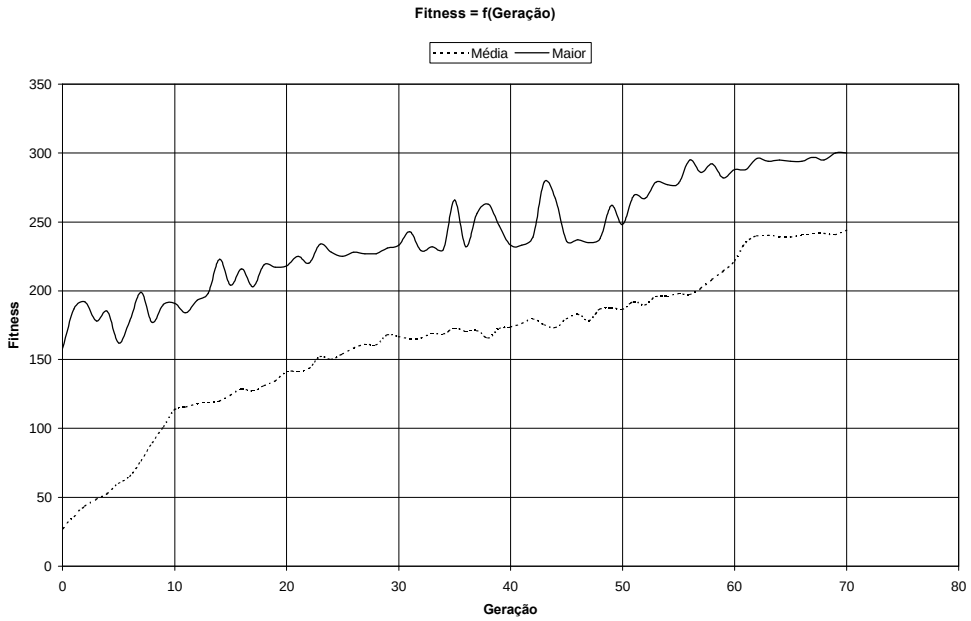


Figura 7.1 – Gráfico Evolutivo de ‘71 gerações’

No próximo gráfico temos um exemplo de uma execução aonde o primeiro indivíduo com *fitness* ‘300’ aparece na ‘geração 6’, muito prematuro, por isso, e também devido à aleatoriedade esse indivíduo não mantém seu desempenho. Ainda vemos oscilações até a geração ‘20’ aonde o melhor se estabiliza. Nesse gráfico também podemos perceber que devido à solução ter sido encontrada muito cedo obtemos uma média final dos indivíduos muito melhor que a do caso anterior.

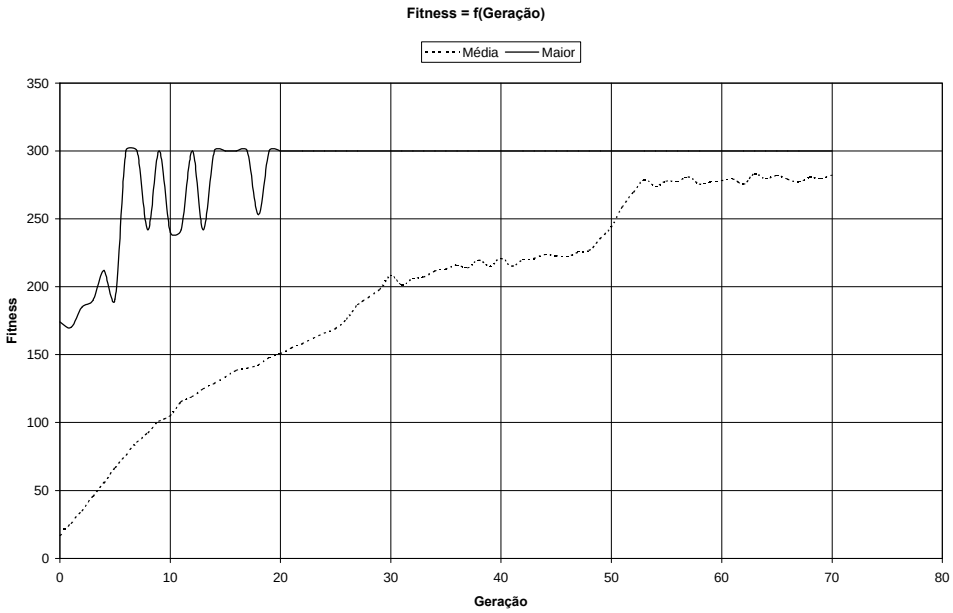


Figura 7.2 – Gráfico Evolutivo de ‘71 gerações’ com evolução antecipada

E no último gráfico que foi simulado com apenas ‘51’ gerações, o exemplo de uma evolução no qual o melhor indivíduo não é nem prematuro nem muito atrasado, ele ocorre na 22ª geração e se estabiliza.

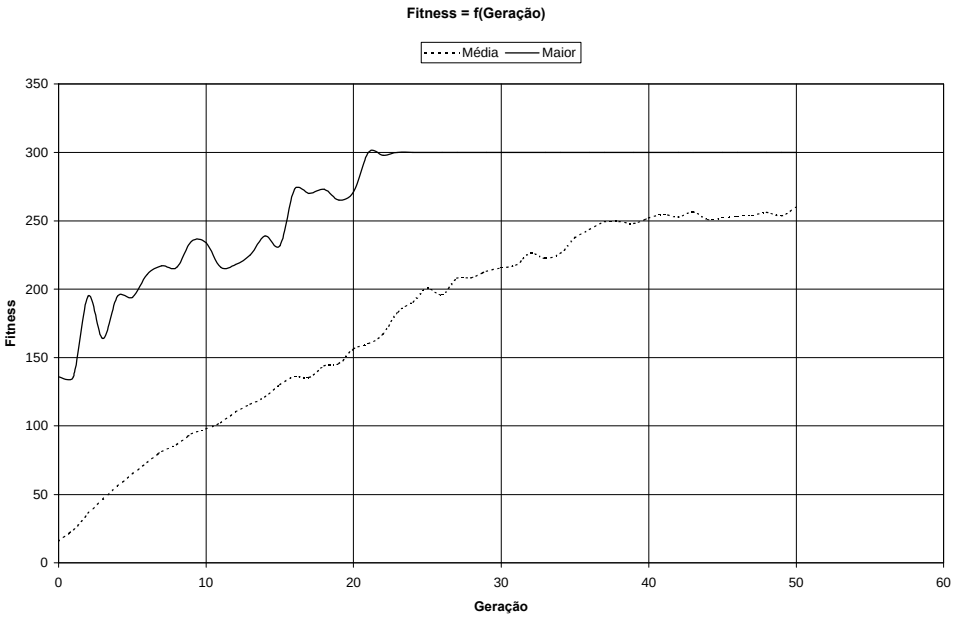


Figura 7.3 – Gráfico Evolutivo de ‘51 gerações’

Exemplos de Caminhos

Para entendermos as figuras é necessário que se saiba a maneira como foram representadas:

- As paredes são a parte representada pela cor preta;
- A área livre para circulação do robô é representada pela cor branca;
- As outras cores representam a direção do robô durante o percurso:
    - Vermelho: para cima;
    - Amarelo: esquerda;
    - Azul: para baixo;
    - Verde: direita.

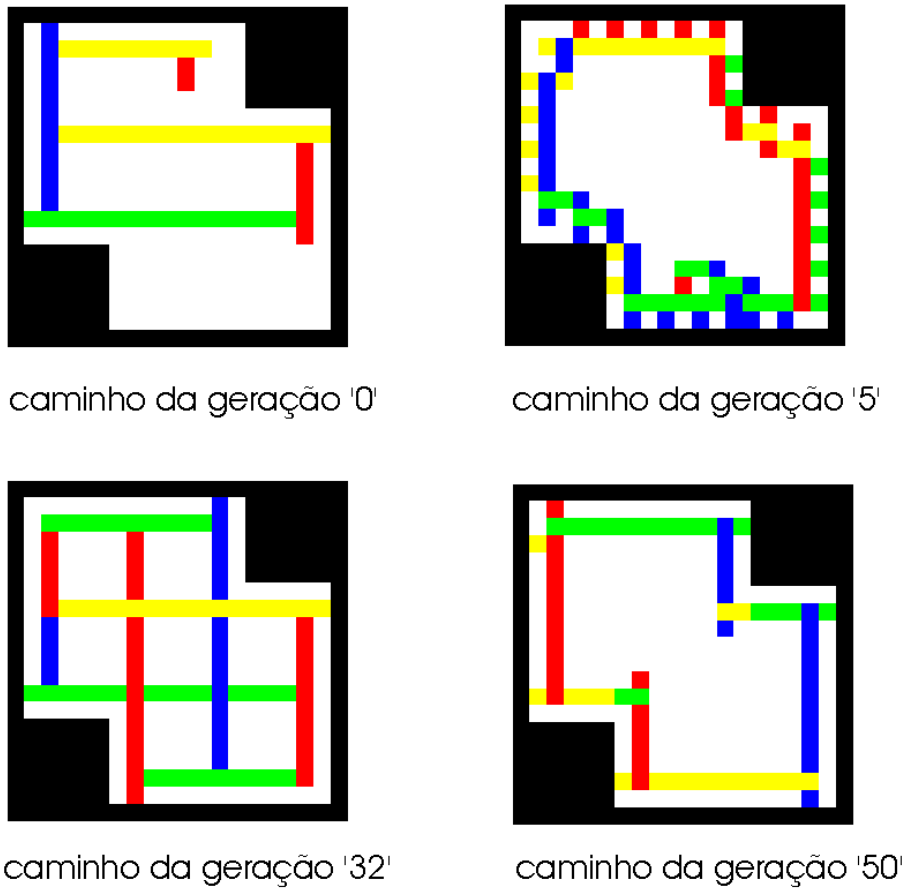


Figura 7.4 – Exemplos de caminhos percorridos pelos agentes

‘Geração 0’: vemos que o indivíduo não consegue detectar quinas, por isso percorre um caminho que possui mérito dentro da sua geração mas não é o suficiente.

‘Geração 5’: ao contrário do indivíduo da ‘Geração 0’ esse indivíduo procura pela quina diversas vezes o que torna seu caminho confuso e de execução lenta.

‘Geração 32’: nesse indivíduo vemos um caminho “limpo”, que não executa muitos testes antes de se movimentar, mas ainda não é um indivíduo excelente exatamente por poupar movimentos na detecção das quinas e esbanjar movimentos andando em linha reta.

‘Geração 50’: esse é o indivíduo excelente por que não se movimenta além do necessário para a detecção das quinas e cantos, podemos ver que sempre próximo a eles

o indivíduo se desvia um pouco de seu caminho retornando logo em seguida, sem desperdício de movimentos.

Exemplos de Indivíduos

Abaixo é mostrado o melhor indivíduo dessa simulação na forma em que ele é escrito em arquivo depois de ter sido evoluído, e os dados do segundo e terceiro melhores. Escrito dessa forma ele pode ser reutilizado em novos testes:

Indivíduo 1

33FFILBI3RI22BR223BBRR2FF2LFFRIIF32I3FF3BRL32RLBII33RIIF3ILLFRFF3LB3FB2BFRLBLRBB

PONTOS DE COMPLEXIDADE = 159
FITNESS = 300

Indivíduo 2

PONTOS DE COMPLEXIDADE = 79
FITNESS = 300

Indivíduo 3

PONTOS DE COMPLEXIDADE = 425
FITNESS = 300

Podem ser notadas as diferenças na complexidade dos três indivíduos, o primeiro possui um tamanho que é próximo à média, o terceiro é muito grande (ou muito complexo) e o segundo tem aproximadamente a metade do tamanho do primeiro.

Deve ser lembrado que todos possuem *fitness* máximo portanto deve ser escolhida uma das soluções apresentadas, se houver necessidade de um indivíduo que é apenas suficiente para suprir as necessidades mínimas, a escolha mais óbvia seria o segundo que apresentará um tempo de execução menor, mas se o caso for um indivíduo que venha a ter bom comportamento em outros ambientes primeiramente deveremos testá-los nesse novo ambiente e então o melhor deve ser escolhido.

7.4.2 Agente seguidor de paredes com visão global

Esta simulação possui as mesmas características da simulação anterior exceto pelo fato de apresentar maior complexidade de resolução devido à grande área a ser percorrida.

O indivíduo deve percorrer a sala, sempre próximo à parede e completar sua volta em no máximo ‘5000 passos’.

Fitness

O *fitness* é calculado da seguinte forma:

```
fitness = fit – ( unfit / 500 )
```

Na qual ‘fit’ é o número de posições do caminho ideal que o robô visitou e ‘unfit’ é o número de vezes em que ele se desviou do caminho (penalização) ideal ou visitou uma célula mais de uma vez, por esse número ser “relativamente” grande ele é dividido por ‘500’ para diminuir sua influência no *fitness* final.

O *fitness* máximo é ‘780’ e representa o número de células do caminho ideal, ou seja, se um robô conseguir percorrer o caminho ideal sem se desviar muito da sua trajetória, certamente terá um *fitness* próximo ao máximo.

Funções

- PROGN3 (3): executa três ramos em seqüência;
- PROGN2 (2): executa dois ramos em seqüência;
- IFWALL (I): executa seu ramo esquerdo se não for detectada parede pelo robô (a no máximo 1 passo de distância) e caso contrário executa seu ramo direito.

Terminais

- WALKFRONT (F): faz robô dar um passo à frente;
- WALKBACK (B): faz robô dar um passo para trás;
- RIGHT (R): faz robô virar à direita;
- LEFT (L): faz robô virar à esquerda.

Outros Parâmetros

- População (M) = 500;
- Probabilidade de Cruzamento = 70%;
- Probabilidade de Reprodução = 30%;
- Matriz de simulação = 200 x 200;
- Posicionamento do robô em números reais;
- Limite de complexidade (no sorteio) = 1000;
- Ângulo de virada = 30º;
- Número de execuções por indivíduo = 2.

É importante que seja dito que o programa permite que o robô se vire em qualquer ângulo que seja definido, o ângulo de 30º foi adotado depois de executadas várias simulações.

Resultados

Gráficos Evolutivos

O primeiro gráfico, do *fitness* em função da geração, mostra a evolução dos indivíduos após ‘51’ gerações (a geração ‘0’ também é contada). Pode se observar como a média acompanha a reta dos maiores após as gerações e como é mais estável não apresentando variações bruscas. Por exemplo: na ‘geração 9’ o maior indivíduo possui *fitness* igual à ‘400’ e na ‘geração 11’ o *fitness* é igual à ‘530’ o que representa um

grande salto, isso devido ao cruzamento.

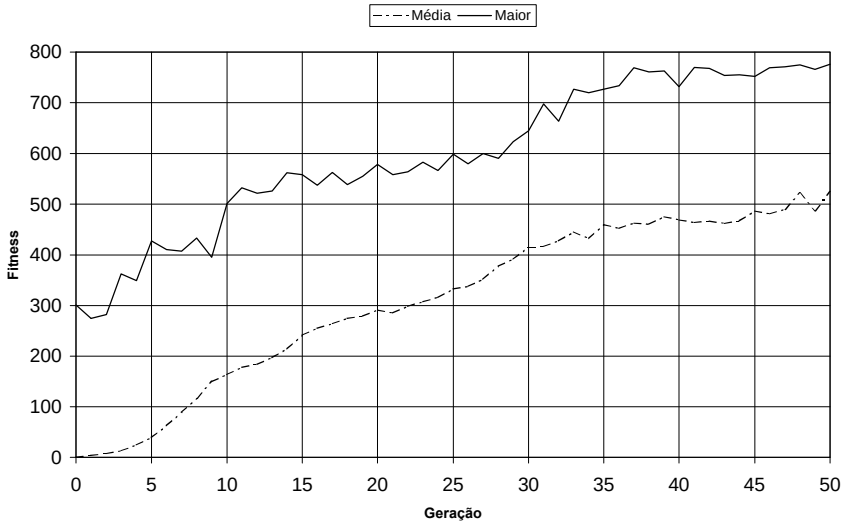


Figura 7.1 – Gráfico Evolutivo ’51 gerações’

No próximo gráfico, em uma simulação de ‘42’ gerações, nota-se claramente uma queda na média do *fitness*. Isso ocorre devido a uma tentativa de renovação dos indivíduos executada no meio do processo de evolução, essa renovação é chamada de “decimação”. Na decimação apenas ‘10%’ deles são aproveitados e os outros ‘90%’ são criados novamente. Apesar dessa tentativa e do *fitness* inicial ser próximo à ‘500’ no final da simulação não se conseguiu indivíduos tão bons como os da execução anterior. Isso demonstra como a PG é muito dependente dos seus primeiros indivíduos, mas também da variedade existente.

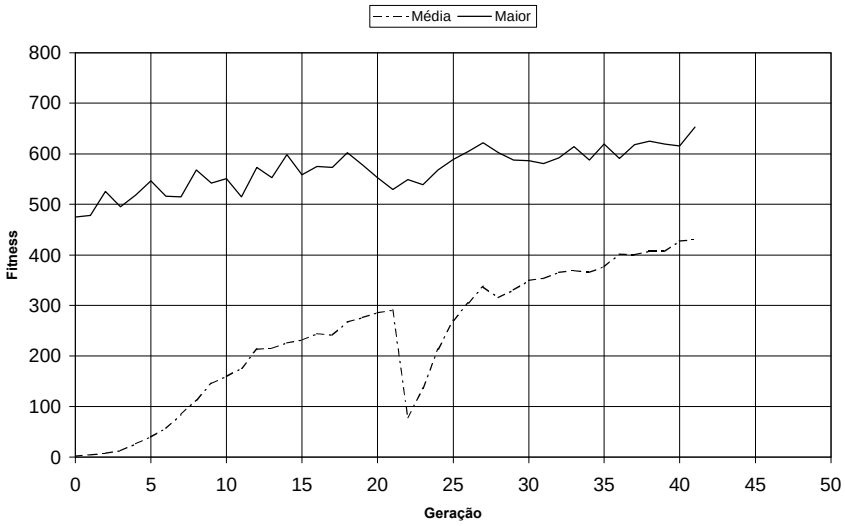


Figura 7.2 – Gráfico Evolutivo ’42 gerações’com decimação na 21ª geração

Exemplos de Caminhos

‘Geração 10’: o indivíduo dessa geração apesar de poder acompanhar a parede e

detectar o obstáculo à sua frente, não percebe quinas, apenas caminha até encontrar algo que impeça seu movimento, isso faz com que ele tenha algumas características que podem ser aproveitadas na próxima geração.

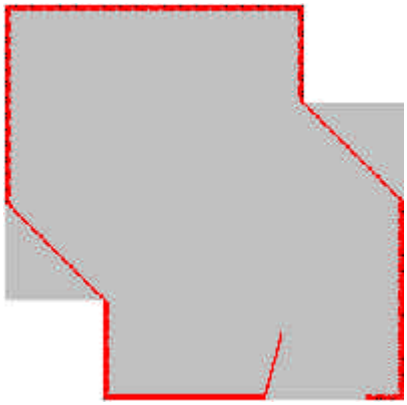


Figura 7.3 – Caminho Geração 10.

‘Geração 26’: esse indivíduo percorre muito bem o caminho, mas por virar-se muitas vezes para encontrar quinas, acaba não percorrendo o caminho todo. Ele demonstra o aperfeiçoamento que ocorreu desde a décima geração na detecção de quinas.

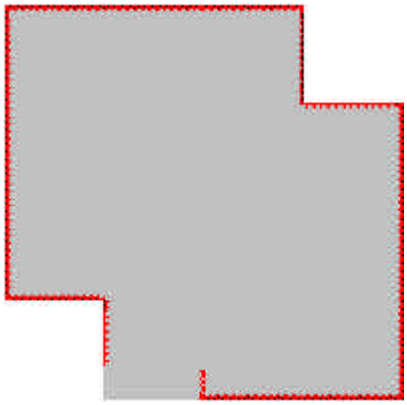


Figura 7.4 – Caminho Geração 26

‘Geração 48’: esse é o indivíduo excelente dessa execução, pois percorre todo o caminho dentro do número de movimentações estipulado. Consegue detectar as quinas que foram problema na 10ª geração e não perde tempo com isso como na 26ª geração.

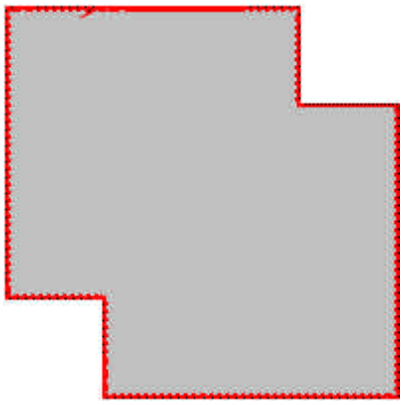


Figura 7.5 – Caminho Geração 48

Exemplos de Indivíduos

Abaixo é mostrado o melhor indivíduo dessa simulação na forma em que ele é escrito em arquivo depois de ter sido evoluído, e os dados do segundo e terceiro melhores. Escrito dessa forma ele pode ser reutilizado em novos testes:

Indivíduo 1

3IR22222FIF3B2II2FII2B2LI2B2B32F2LRRI22BL3IBIFIFBFB323L3B3B
FB2F32RFF23IIB3FLIFRLR2L3IILLILFILLRIF23B2II32RLIFRBB2RIIBB33IR
223I3IILL2I2I33BBRBRBIRRL3IB32LRBRRBI2BRLRFR2L3222FRIILLIFFRIF
LRIF3BIBRLRFFR2FRRL3IILBILFIFLRR3IB2I32LIRR22ILRBRILBBBBLFLF
BFBLR2IBRRRRIILLIFFRFI33FLIFR2LIRBB3ILLFIR3BBR

PONTOS DE COMPLEXIDADE = 290

FITNESS = 776

Indivíduo 2

PONTOS DE COMPLEXIDADE = 110

FITNESS = 766

Indivíduo 3

PONTOS DE COMPLEXIDADE = 44

FITNESS = 761

Podem ser notadas as diferenças na complexidade dos três melhores indivíduos. O primeiro é o maior de todos, mas que também é o que possui maior *fitness.* O segundo mesmo tendo *fitness* menor não apresenta tanta inferioridade e possui menos da metade de pontos de complexidade do primeiro, o terceiro por sua vez também não perde muito em desempenho e possui complexidade inferior à metade do segundo. Deve ser lembrado também que todos possuem *fitness* muito próximos ao máximo, mesmo tendo sido descontados desvios, portanto cabe a nós escolher uma das soluções que nos foram apresentadas, se quiser um indivíduo que é apenas suficiente para suprir as nossas

necessidades atuais a escolha mais óbvia seria o terceiro que apresentará um tempo de execução menor, mas se o caso for um indivíduo que venha a ter bom comportamento em outros ambientes primeiramente ele deverá ser testado nesse novo ambiente.

7.4.3 Agente seguidor de bola com obstáculos e com visão global

O agente Seguidor de Bola foi escolhido para ser o segundo passo da evolução dos Agentes Jogadores de Futebol, pois apresenta maior dificuldade que o Seguidor de Paredes e adiciona elementos como a bola e obstáculos.

Além das funções de desvio de paredes e de movimentação, que foram reaproveitadas, adiciou-se apenas uma função que relaciona a movimentação do robô com a movimentação da bola, alterando o ambiente e a maneira como se mede o *fitness*.

Com as características da PG foi criado o programa do robô seguidor de bolas aonde o robô deve alcançar a bola e tocá-la o maior número de vezes possíveis dentro da sua execução. O ambiente de simulação continua do mesmo tamanho, apesar de ter sido adicionados obstáculos, os passos são contados da mesma forma. A execução da árvore também não é parada enquanto não terminar.

O indivíduo deve percorrer a sala buscando pela bola em no máximo ‘2000 passos’. Os passos são reduzidos em relação ao caso anterior especificamente porque o robô não precisa se movimentar tanto para alcançar a bola, e para tornarem a representação final mais clara.

A cada simulação o robô e a bola começam em posições aleatórias.

Fitness

O *fitness* foi calculado de diversas maneiras. Abaixo, algumas das melhores maneiras e os motivos pelos quais são considerados melhores são apresentadas.

A fórmula que serviu de base para as outras é:

```
fitness = hits * k – Σ(n / Dini);
```

Aonde ‘hits’ é o número de vezes que o robô atingiu a bola; n é o número de passos do robô e Dini é a distância inicial medida antes do robô se movimentar. O valor ‘n / Dini’ (chamado de penalização) é recalculado a cada vez que o robô atinge a bola, por isso é usada a somatória desses valores no *fitness*. ‘k’ é uma constante a ser definida em cada caso, e será mostrado como ela influi no resultado.

O número de hits máximo é estimado em ‘15’, com base na área de movimentação, no número de execuções e na distância máxima que a bola pode ter do robô.

Funções

- PROGN3 (3): executa três ramos em seqüência;
- PROGN2 (2): executa dois ramos em seqüência;
- IFWALL (I): executa seu ramo esquerdo se não for detectada parede pelo robô (a no máximo 1 passo de distância) e caso contrário executa seu ramo direito.

Observação: são as mesmas do caso anterior.

Terminais

- WALKFRONT (F): faz robô dar um passo à frente;
- WALKBACK (B): faz robô dar um passo para trás;
- RIGHT (R): faz robô virar à direita;
- LEFT (L): faz robô virar à esquerda;
- ALIGN (A): direciona o robô para a bola.

Outros Parâmetros

- População (M) = 500;
- Probabilidade de Cruzamento = 70%;
- Probabilidade de Reprodução = 30%;
- Matriz de simulação = 200 x 200;
- Posicionamento do robô em números reais;
- Limite de complexidade (no sorteio) = 1000;
- Ângulo de virada = 5º;
- Número de execuções por indivíduo = 1.

O ângulo de 5º foi adotado por se tratar um valor suficiente e que permite bastante precisão na movimentação buscando a bola.

Resultados

Gráficos Evolutivos

No primeiro gráfico (50 gerações) nota-se uma grande diferença em relação ao Seguidor de Paredes: a variação exagerada do *fitness* do melhor indivíduo, isso se deve ao fato do robô interagir-se com outro objeto que também se movimenta, que é mais rápido que ele e que reflete em outros objetos, ou seja, executa-se apenas uma simulação de cada robô e a posição inicial aleatória influi no resultado. De qualquer forma o *fitness* nesse caso é mais útil para informar quantas vezes o robô consegue tocar a bola dentro da execução estipulada.

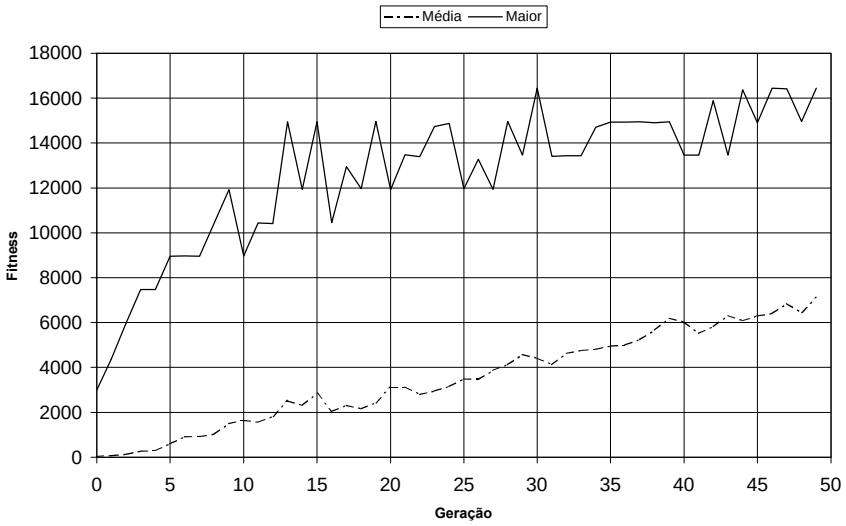


Figura 7.1 – Gráfico Evolutivo ’50 gerações’ k = 1500

No segundo gráfico (100 gerações com decimação) foi executada a decimação já citada, na ‘50ª geração’. Como no caso anterior tem-se a variação do *fitness* do melhor indivíduo e também é percebido que essa variação após algum tempo permanece dentro da mesma faixa, com poucas exceções. Essa simulação foi feita para observar a convergência final dos dados e a eventualidade de ter indivíduos por vezes tão bons, e como estes não poderiam ter perdido suas características de uma geração para outra. Até a centésima geração a média de *fitness* continua aumentando e a faixa em torno de onde varia o indivíduo com *fitness* máximo, também. Algumas vezes a posição inicial, sorteada de certa forma, pode favorecer o desempenho final e isso não foi retirado, pois a sorte também influi na evolução.

O último gráfico (50 gerações) mostra uma simulação onde ‘k = 1000’. Isso fez com que a somatória das penalizações dos indivíduos tivesse maior significância no *fitness* e com isso selecionou melhor aqueles que deveriam continuar, ocasionando uma variação um pouco menor nos melhores. Pode-se notar indivíduos com *fitness* muito próximo à ‘14000’ o que indica que atingiram a bola ‘14’ vezes. No caso anterior com ‘k = 1500’ teve-se indivíduos com *fitness* próximo à ‘23000’ o que indica ‘15’ hits. Ou seja, obteve-se uma melhora em um aspecto (maior estabilidade dos dados finais) e em termos de *fitness* não se perdeu, pois a execução com ‘k = 1500’ foi feita duas vezes e somente na segunda execução (100 gerações) que se obteve indivíduos tão bons.

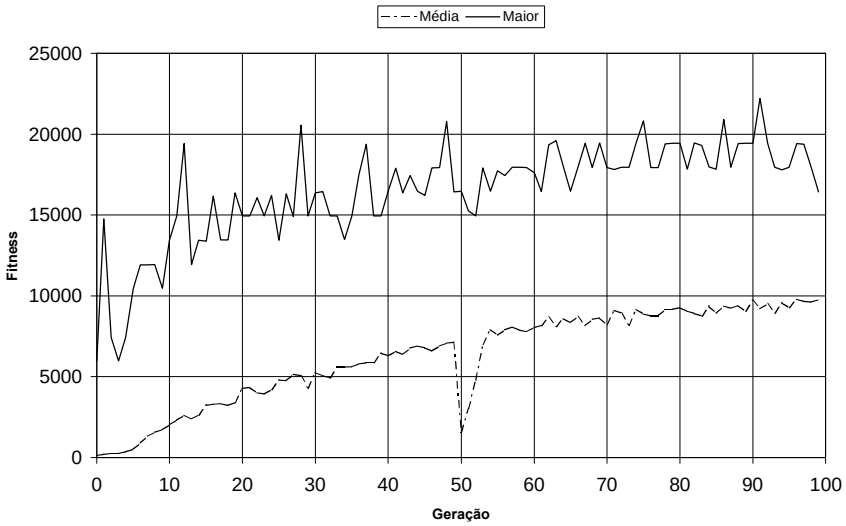


Figura 7.2 – Gráfico Evolutivo ’100 gerações’ com decimação - k = 1500

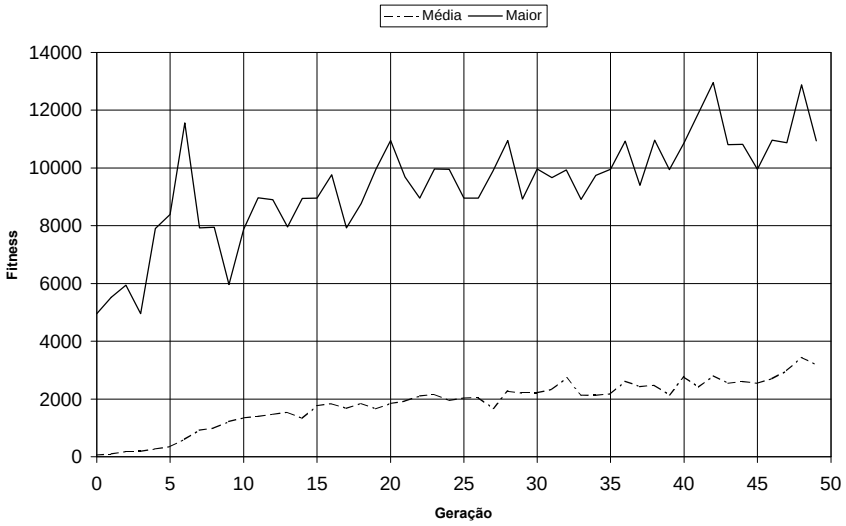


Figura 7.3 – Gráfico Evolutivo ’50 gerações’ k = 1000

Exemplos de Caminhos

As figuras a seguir foram extraídas da simulação com ‘k = 1500’. Os quadrados brancos são obstáculos para o indivíduo e para a bola.

Nessas figuras o caminho feito pelo robô é representado pela cor vermelha. O amarelo representa o princípio do caminho, e o verde a posições da bola.

‘Geração 1’: Esse indivíduo toca poucas vezes na bola, mas pode perceber como ele já a persegue, apesar de não ter tido problemas com os obstáculos.

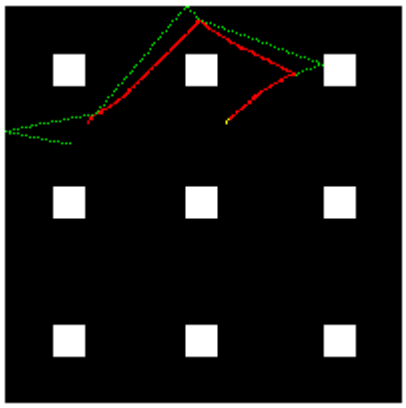


Figura 7.4 – Caminho Geração 1

‘Geração 8’: Exemplo de um caminho onde o indivíduo persegue muito bem a bola, por partir de uma distância maior que o exemplo anterior e mesmo assim possuir *fitness* maior, formando uma parábola com o seu caminho.

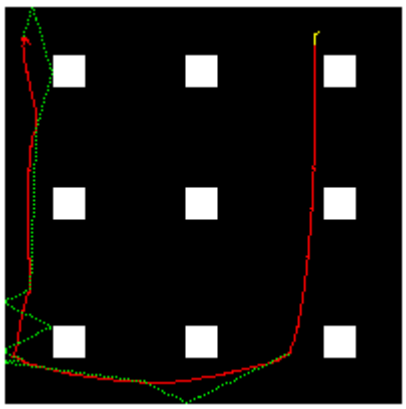


Figura 7.5 – Caminho Geração 8

‘Geração 13’: Indivíduo que inicia a busca pela bola já com um obstáculo entre ele e seu destino e utiliza suas funções e treinamento para desviar-se.

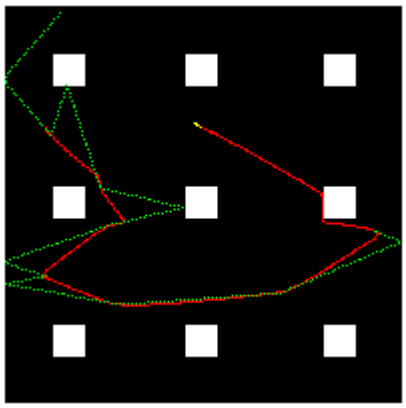


Figura 7.6 – Caminho Geração 13

Geração 41’: Indivíduo bastante evoluído que consegue por várias vezes atingir a bola já que o *fitness* o estimula a fazer isso. Nota-se também que este indivíduo não

perde tempo ao seguir a bola e sempre tenta a melhor trajetória. Pouco antes de a execução terminar ele ainda buscava atingir a bola mais uma vez e fez uma pequena curva dentro da trajetória da bola.

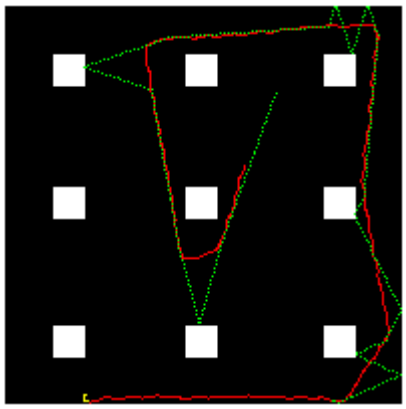


Figura 7.7 – Caminho Geração 41

Exemplos de Indivíduos

Abaixo é mostrado o melhor indivíduo da simulação com ‘k = 1500’ na forma em que ele é escrito em arquivo depois de ter sido evoluído, e os dados do segundo e terceiro melhores. Escrito dessa forma ele pode ser reutilizado em novos testes:

Indivíduo 1

3IIR2FF323I2ILIB32R3FRRR22L3RAF2BRFR323I2ILIBILLIAAIR32F2IF
3FFRFFF33RB2RRF33FFRILI23I2IRIB32R3FR32AR32F2IIAL2FRFAF22IALF
2BRR22L3RF2IBF323IFIRR33RBII3FBBIRAFF33FF22F2IIAL3F3RRFRFIRAI
232FRR2B2BR33FBIRF3L2RF3FFR2RBA2BIR32FRFBAIRFII333F2LFFFRBA
I2IBIIAL3FF3FFR3FFRAFL2BRFR323I2ILIB33BIAR22FF3A2IIAL3FFRLFF2F
RIAAIR32F2IF3FFRRFF33RF2RRF33FFRILR2BIR32FR3FFRBFLII3FLBIRAF
33FFRL32FRR2FIBFAIRF32RIIFRIFF33AIL3A2FFFFLFR2BIR32FR2FRBAIR
FII3FBBIRFF33FFRL3AR2FIBFAIRFII3L33FA3AILIFRFFFRIRAFLRF

PONTOS DE COMPLEXIDADE = 477

FITNESS = 16436

Indivíduo 2

PONTOS DE COMPLEXIDADE = 696

FITNESS = 16433

Indivíduo 3

PONTOS DE COMPLEXIDADE = 432

FITNESS = 16431

É muito clara a diferença na complexidade dos indivíduos do Seguidor de Bola em relação aos indivíduos do Seguidor de Paredes principalmente por se tratar de uma tarefa mais difícil de ser executada. Os indivíduos do Seguidor de Bola possuem uma complexidade muito maior.

A complexidade do indivíduo não pode ser diretamente relacionada com o seu desempenho no ambiente. Os três indivíduos acima têm *fitness* muito próximos e o segundo se destaca por ter uma complexidade consideravelmente maior.

7.4.4 Agente seguidor de bola com obstáculos e com visão local

Esta simulação possui as mesmas características da simulação anterior exceto pelo fato exigir uma maior habilidade do agente a ser desenvolvido.

Como a visão é limitada, o agente não tem mais a visão do ambiente inteiro e precisa buscar a bola da maneira mais rápida e desviando dos obstáculos, pois estes também atrapalham sua visão.

Fitness

O *fitness* nessa simulação foi calculado da mesma forma que na simulação anterior. Isto porque o objetivo da evolução ainda é o mesmo, só as dificuldades é que aumentaram.

A fórmula do cálculo do *fitness* é:

```
fitness = hits * k – Σ(n / Dini);
```

Aonde ‘hits’ é o número de vezes que o robô atingiu a bola; n é o número de passos do robô e Dini é a distância inicial medida antes do robô se movimentar. O valor ‘n / Dini’ (chamado de penalização) é recalculado a cada vez que o robô atinge a bola, por isso é usada a somatória desses valores no *fitness*. ‘k’ é uma constante que foi mantida com o valor de ‘1500’ de acordo com o que concluímos na simulação anterior.

Funções

- PROGN3 (3): executa três ramos em seqüência;
- PROGN2 (2): executa dois ramos em seqüência;
- IFWALL (I): executa seu ramo esquerdo se não for detectada parede pelo robô (a no máximo 1 passo de distância) e caso contrário executa seu ramo direito;
- IFBALL (C): executa o ramo esquerdo se enxergar bola e caso contrário executa o ramo direito.

Observação: foi acrescentada apenas uma função (ifball).

Terminais

- WALKFRONT (F): faz robô dar um passo à frente;
- WALKBACK (B): faz robô dar um passo para trás;
- RIGHT (R): faz robô virar à direita;
- LEFT (L): faz robô virar à esquerda;

- ALIGN (A): direciona o robô para a bola.

Observação: iguais às do caso anterior.

Outros parâmetros

- População (M) = 500;
- Probabilidade de Cruzamento = 70%;
- Probabilidade de Reprodução = 30%;
- Matriz de simulação = 200 x 200;
- Posicionamento do robô em números reais;
- Limite de complexidade (no sorteio) = 1000;
- Ângulo de virada = 5º;
- Número de execuções por indivíduo = 1;
- Ângulo de visão = 30° para cada lado.

Resultados

Gráficos Evolutivos

Nesse primeiro gráfico, como já havia sido notado na simulação anterior, existe uma grande variação no desempenho do melhor agente, parecendo até que este "desaprendeu" de repente o que havia aprendido. Isso não é verdade, mais uma vez isso se deve à grande gama de possibilidades em que o ambiente pode se apresentar para o agente, antes que ele comece a buscar pela bola.

O principal é que pode ser percebida a evolução, em outras simulações adiante serão mostrados os efeitos de um desenvolvimento mais longo (com mais gerações).

No segundo gráfico, é notada uma maior estabilidade na evolução e um melhor resultado final, e que também seja notado que se trata apenas de uma outra simulação com o mesmo programa. Esse gráfico serve para ser mostrado que não se tem um padrão muito correto de evolução, exceto o que já foi citado.

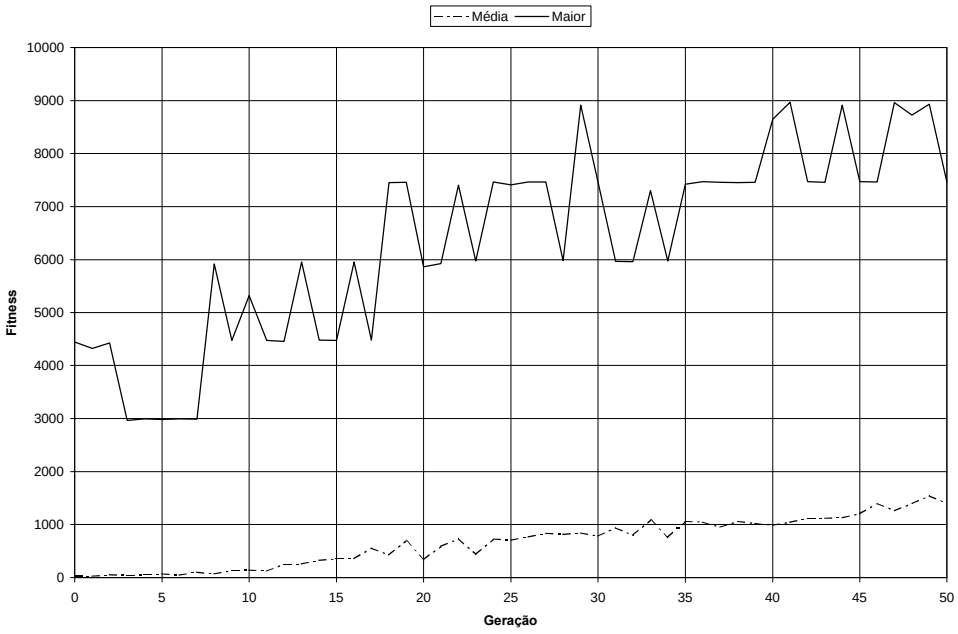


Figura 7.1 – Gráfico Evolutivo ’51 gerações’ (Tempo de execução = 1:42:35)

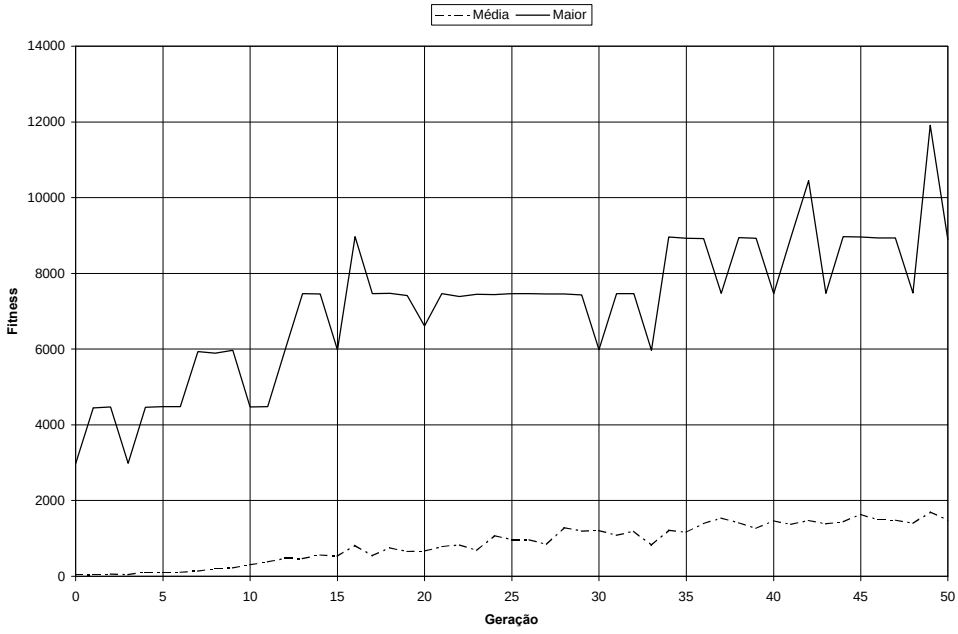


Figura 7.2 – Gráfico Evolutivo ’51 gerações’ (Tempo de execução = 1:37:18)

Outro ponto interessante desse gráfico é um salto de desempenho na ‘49ª geração’ o que pode indicar que se a simulação proseguisse, provavelmente seriam obtidos, daí em diante, agentes com *fitness* que variassem entre ‘9000’ e ‘12000’.

Já no terceiro gráfico, que se trata de uma simulação de ‘101 gerações’, é perceptível sua semelhança com o segundo gráfico até a ‘50ª geração’, após isso o detalhe importante é que a evolução tende a se estabilizar entre uma faixa de valores,

mas até a ‘100ª geração’ ela ainda existe. Outro fato importante mas que também não pode ser tomado como regra é que a faixa de variação dos melhores agentes diminui.

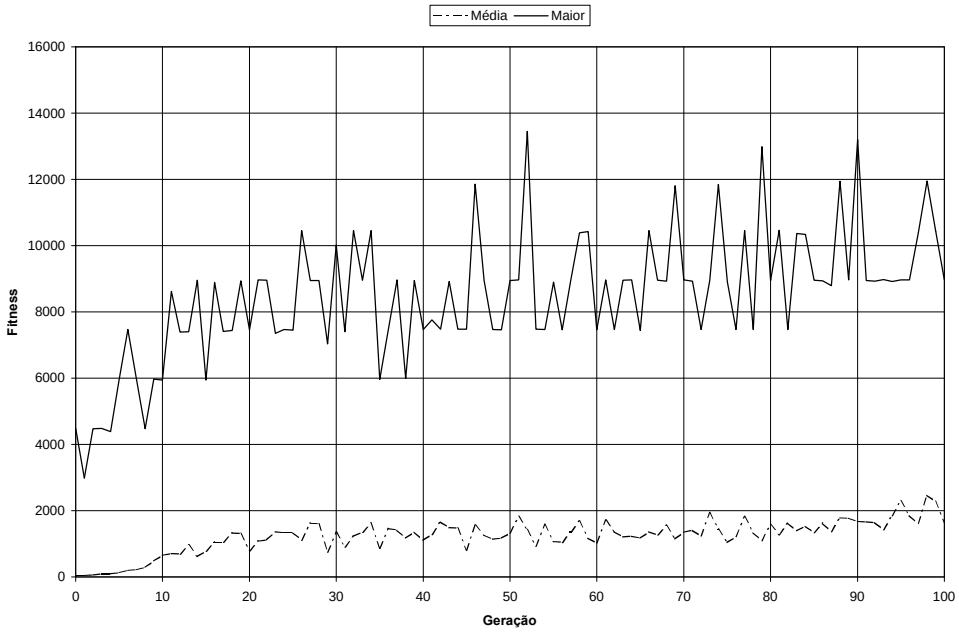


Figura 7.3 – Gráfico Evolutivo ‘101 gerações’ (Tempo de execução = 2:57:45)

Um ponto marcante que podemos perceber nessas simulações e que as diferenciam das outras é o baixo valor da média dos indivíduos, que nem chega a alcançar o desempenho do melhor indivíduo da primeira geração. Isso se deve principalmente ao alto grau de dificuldade dessa simulação. Mas como se trata de uma evolução espelhada na natureza, o que se espera é que se esses mesmos agentes forem deixados evoluindo durante mais tempo, os melhores devem estabilizar em um valor próximo ao ótimo e a média não deve ficar muito longe disso.

Exemplos de Caminhos

Nos exemplos de caminhos que serão apresentados a prioridade é que sejam mostrados os que exemplifiquem o desenvolvimento e sejam de fácil entendimento com breves descrições.

Geração 1: por se tratar de um agente com pouca evolução, nessa figura o agente após atingir a bola duas vezes se perde atrás de um obstáculo.

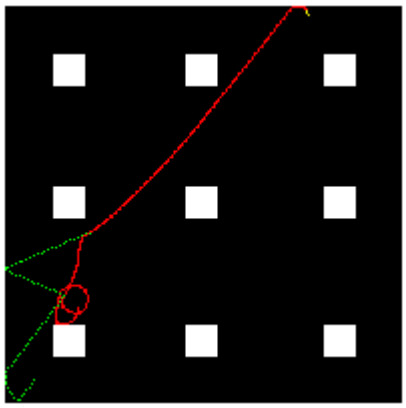


Figura 7.4 – Caminho da Geração 1

Geração 7: esse agente se vira uma vez procurando a bola e não a encontra, então segue em uma direção, começa a se virar novamente dessa vez encontrando a bola, daí segue para atingi-la.

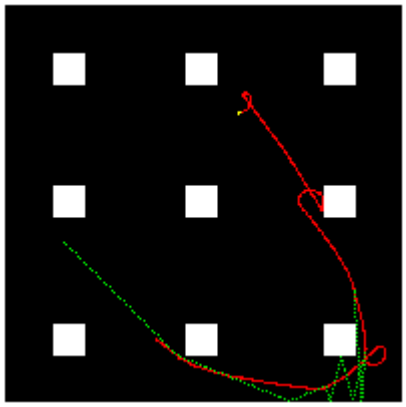


Figura 7.5 – Caminho da Geração 7

Geração 12: nesse exemplo o agente se vira buscando a bola, como todos fazem desde as primeiras gerações, e depois que a encontra não mais a perde seguindo-a até o fim da sua execução.

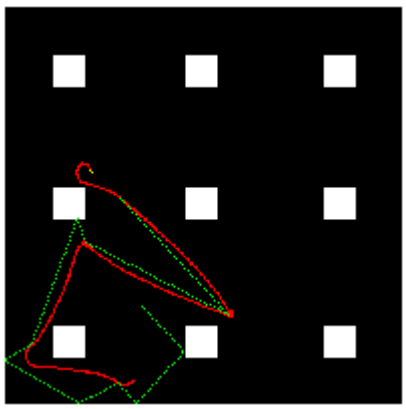


Figura 7.6 – Caminho da Geração 12

Geração16: esse agente atinge a bola duas vezes e por algum motivo perde sua localização, o que acontece então é que ele começa a se virar procurando a bola, demonstrando uma boa evolução no que diz respeito a localizar, ou no caso, achar novamente a bola.

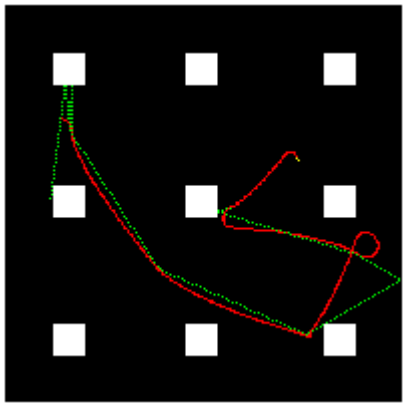


Figura 7.7 – Caminho da Geração 16

Geração 41: esse é apenas um exemplo de um agente que já consegue quase que perfeitamente seguir a bola, daqui em diante o principal ponto de evolução serão: o aumento do número de vezes em que o agente atinge a bola e a economia na movimentação até que se atinja a bola.

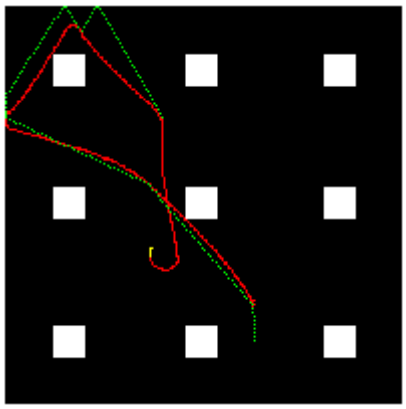


Figura 7.8 – Caminho da Geração 41

Geração 50: o que deve ser observado nessa figura é que o agente perde a visão da bola porque se depara com um obstáculo, mas logo depois faz o conhecido movimento de busca da bola e a reencontra.

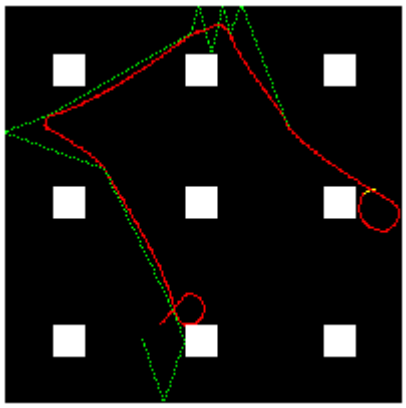


Figura 7.9 – Caminho da Geração 50

Geração 70: nesse exemplo que se quer ressaltar é o tipo de movimentação do agente, que como no agente implementado anteriormente, esse agente desenvolve uma certa capacidade em economizar movimentos para atingir a bola. O que se observa na figura abaixo é uma movimentação que quase “prevê” onde a bola será atingida novamente.

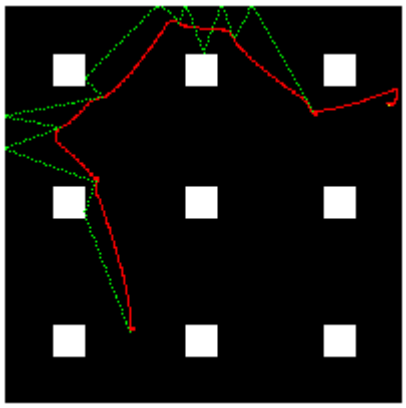


Figura 7.10 – Caminho da Geração 70

Geração 81: esse último exemplo mostra uma boa perseguição da bola e também as características boas que se mantiveram dos agentes anteriores, como a economia de movimentos e um bom número de vezes atingindo a bola.

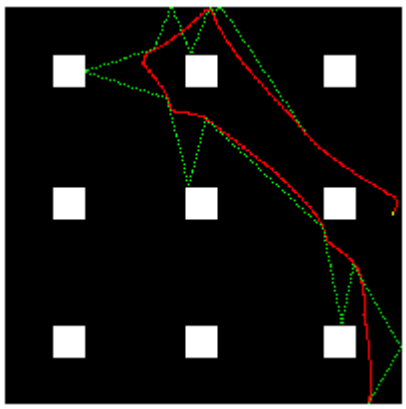


Figura 7.11 – Caminho da Geração 81

Exemplos de Indivíduos

Abaixo é mostrado o melhor indivíduo da simulação de ‘101 gerações’ na forma em que ele é escrito em arquivo depois de ter sido evoluído, e os dados do segundo e terceiro melhores. Escrito dessa forma ele pode ser reutilizado em novos testes:

Indivíduo 1

3F33FCA32LLACI23FRBFBIIFL3I232FIICAF22BIIRRLBR3CA23FLII2F
2BLIBIL2IILR2BAF3I232FIBR3CA23FLIICA32LL2I3B2IBCARR2LFRBCAIIF
L3I232FIICAF22BIIR3RCA3R3ICARIR2BIBCAIRFI23FCACA3FBBRRFALRL
BR3CA23FLII2F2BLIBIL2IILR2BCARFIL22LBIL223CAIFB3IRBFBLL3FB2I3
3LILRBFL2IIR2CA3R2FIICBB3LL2LBCAR32B2LLLFCAR3FRLLRF223L2FR
RCABLCAICAFFFBILCA2I3L3FFR3B33FLFBLBLBRLIRIL2IILR2BCARFIL2
2L222IFB23FCALB2LRR2F3RIF2RLI323BRRFL23CACACAF2BI3F2RBB3B
RFLLI3LCA3FIR3233FFF3B2ILRFBCA2B322LRFLLRLACAR3BBCA3R333I
LRLFBBIRR3IILB23BFFBBLLRIR23CAFBB3FB2ICA2FFCALRF223L2FRRB
LCAICAFFFBILCA2I3L3FFR3BRBLLRLF22L3IB3CAFLLBFLLFBILCA2I3L3
FF2I32B3RCF3FB232I33CF2BR3B2RIBRRFIFFBLI3IB2LFILBFLB3RIRR33RI
BCA3CA32FBR2LCALBR2I2BLILBCA3RBB3CA2I2LR3CARRLLRLRI23IR2
3LBCAI2L3LL2I22CAR3FI3ICALII32B3R3CAF2BI2LCA3FRII3I2F3RLRICA2
BCAIICA2IRBCAF3B2CABBCA3RABLLFRLRLBRRBFLBRFRFFBLBRBFB
RRBFCALBLLRLFLBCAB

PONTOS DE COMPLEXIDADE = 832

FITNESS = 8966

Indivíduo 2

PONTOS DE COMPLEXIDADE = 1023
FITNESS = 8902

Indivíduo 3

PONTOS DE COMPLEXIDADE = 2583
FITNESS = 7476

Podemos perceber que a complexidade dos agentes sofreu um considerável aumento, não tanto no melhor agente da simulação, mas no segundo e terceiro agentes. A tarefa mais complexa exigiu uma maior elaboração dos indivíduos, principalmente se compararmos com o melhor indivíduo da primeira simulação, que era o seguidor de paredes em um ambiente simples. Para a resolução daquele problema, o mínimo necessário foi um agente com 79 pontos de complexidade.

7.5 Conclusão

A partir dos resultados obtidos conclui-se que, usando Programação Genética,

pode-se criar indivíduos que apresentam os comportamentos básicos para atuar no domínio do Futebol de Robôs, como seguir paredes e bolas.

A Programação genética permite a criação de indivíduos que produzam os resultados desejados desde que as habilidades necessárias para realização das tarefas e a relação dos indivíduos com o ambiente sejam conhecidas. Assim, um aspecto muito importante na Programação Genética é a simulação do ambiente, que deve incluir as dificuldades e dados relevantes do ambiente, pois os indivíduos criados só estarão bem preparados para reagir a situações conhecidas.

Para a obtenção de indivíduos capazes de realizarem outras tarefas deve-se realizar novos treinamentos com base nos mesmos indivíduos (se puderem ser aproveitadas características adquiridas com o treinamento anterior) ou em novos (como foi feito para o seguidor da bola). Ainda, para que um mesmo indivíduo seja capaz de executar duas tarefas, é necessário treiná-lo nos dois ambientes onde atuará, pois suas características somente permanecem ou se perpetuam quando as mesmas são necessárias para o seu “encaixe” no ambiente. Mudando o ambiente de um indivíduo pode tornar muita das características, antes essenciais para a sua sobrevivência, desnecessárias.

É importante destacar que nenhum tipo de reedição foi utilizado para eliminar os ramos desnecessários ou reduzir a complexidade dos indivíduos, sendo os indivíduos apresentados e seus comportamentos resultados dos conceitos de Programação Genética implementados.

A função de *fitness* é essencial para o controle dos indivíduos que permanecerão nas próximas gerações. Ela controla a velocidade da “evolução”, o resultado final e complexidade dos indivíduos A escolha de uma função de *fitness* que não selecione corretamente os indivíduos que permanecerão acarreta no maior tempo necessário para evoluir uma população e até na criação de populações que não atuam como desejado.

Entre os possíveis trabalhos futuros em estudo encontram-se:

- A evolução de comportamentos mais complexos como: goleiro, defesa, ataque, etc.
- A inserção de comunicação entre os agentes para a formação de times.
- E finalmente, a implementação destes agentes no Simulador Oficial da RoboCup.

8. PROTÓTIPO DE ROBÔ AUTÔNOMO

Este capítulo apresenta a construção de um protótipo de micro-robo autônomo para a categoria F-180 da RoboCup. Para isso, inicialmente se estudou o controle de um motor de passo. Depois se apresenta a implementação deste controle em VHDL.

8.1 Motores de passo

Motor de passo é um atuador eletromecânico incremental. A energização seqüencial de cada enrolamento individual, a partir de uma fonte CC, realizadas por chaves semicondutoras (transistores bipolares ou Mosfets) comandadas por pulsos digitais, provoca um deslocamento discreto e síncrono com os mesmos, denominados "passos", como conseqüência do alinhamento magnético dos pólos nas estruturas do rotor e estator do motor. A cada pulso de alimentação aplicado em uma bobina do estator, o rotor estaciona em uma posição estável e única, guardando entre um passo e outro a mesma precisão, sem erros acumulativos.

Na Figura 8.1 pode-se observar que o estator possui seis pólos onde cada par de pólos opostos possui um enrolamento comum e o rotor apenas quatro pólos, este motor é conhecido como motor de relutância variável. Cada par de pólos com enrolamento comum é chamado de fase, conseqüentemente esta figura1 trata de um motor de três fases. Cada fase é energizada por uma fonte de tensão continua através das chaves I, II e III.

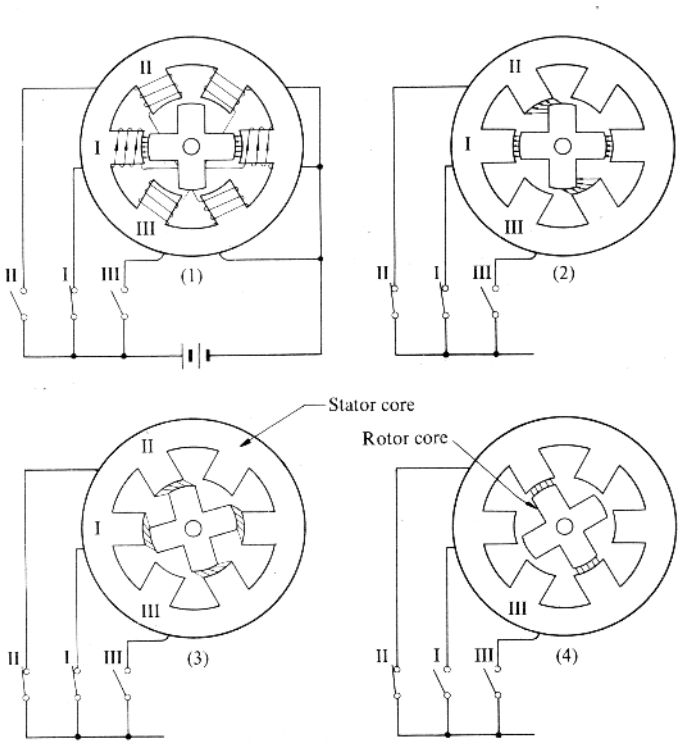


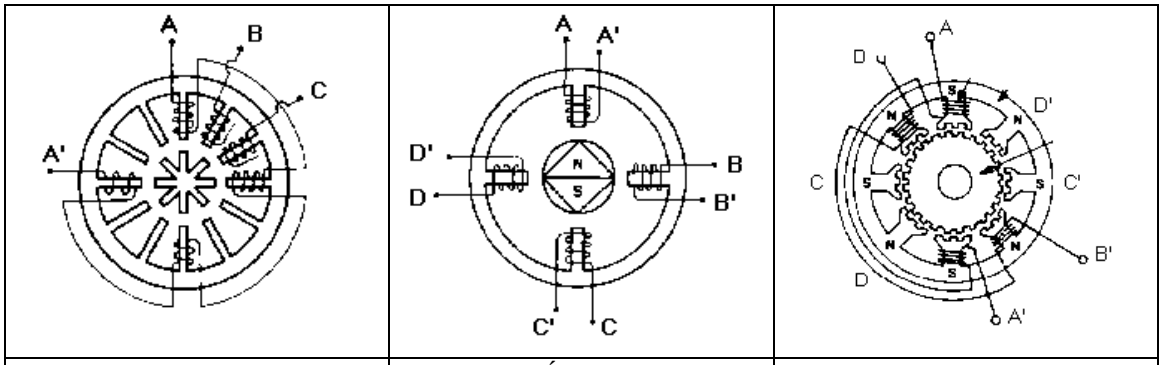
Figura 8.1 – Motor de passo de relutância variável.

No primeiro estado o par de pólos chaveado por I está excitado, onde o fluxo magnético está representado por linhas entre os pólos do estator e os pólos do rotor. Neste estado os pólos do estator e do rotor estão alinhados e em equilíbrio, onde há a

menor relutância para o fluxo magnético. Quando a chave II é acionada, dois novos pólos serão excitados, estado (2), e o rotor irá se movimentar a fim de entrar em equilíbrio, determinando novamente a menor relutância para o fluxo magnético nesta nova situação, estado (3). Por fim a chave I é desacionada, estado (4), completando um passo (aproximadamente 15^o).

O torque do motor de passo depende da freqüência de energização seqüencial de cada enrolamento individual, onde quanto maior a freqüência menor o tempo que o motor tem para mudar o ângulo acarretando a perda de torque.

Existem três tipos básicos de motores de passo, são eles: Motor de Passo de Relutância Variável, Motor de Passo de Imã Permanente e Motor de Passo Híbrido. A figura abaixo mostra estes três tipos de motores.



Motor de Relutância Variável	Motor de Ímã Permanente	Motor Híbrido

Figura 8.2 – Três tipos de motores de passo.

Motor de Passo de Relutância Variável

Este tipo de motor apresenta no rotor e estator uma estrutura multipolar em aço-doçe com conjuntos de bobinas convenientemente interligadas (fases) e enroladas nas peças polares do estator.

Os núcleos do estator e do rotor possuem números diferentes de pólos (dentes), de forma que a relutância do circuito é função da posição do rotor em relação ao estator. À medida que uma das fases é excitada, o núcleo do rotor gira em relação ao estator até atingir a posição de mínima relutância (como no exemplo anterior).

Motor de Passo de Ímã Permanente:

Este tipo de motor apresenta um rotor permanentemente magnetizado com estrutura cilíndrica e um estator com pólos (eventualmente com dentes) sobre as quais se encontram enroladas as bobinas.

Uma característica importante deste motor é torque de retenção, ou seja, torque sem excitação das bobinas.

Motor de Passo Híbrido

Este tipo de motor é uma combinação dos motores de relutância variável e do motor de imã permanente.

No estator deste motor existem pólos, onde sobre cada pólo existem duas ou mais bobinas pertencentes a fases distintas, utilizadas para reforçar ou enfraquecer o fluxo

produzido pelo ímã rotórico.

A estrutura rotórica é cilíndrica com ímãs permanentemente magnetizados longitudinalmente. Cada pólo do ímã é envolto por um anel de aço coberto de dentes. Tais anéis encontram-se desalinhado entre si de um passo de meio dente.

8.2 Técnica de Acionamento

A alimentação de um motor de passo pode ser unipolar ou bipolar. A alimentação unipolar é geralmente utilizada em motores pequenos e permite passagem de corrente unidirecional pelos enrolamentos. A alimentação bipolar, ou em ponte H, permite que a corrente circule em ambos os sentidos, possibilitando um conjugado maior e operação regenerativa.

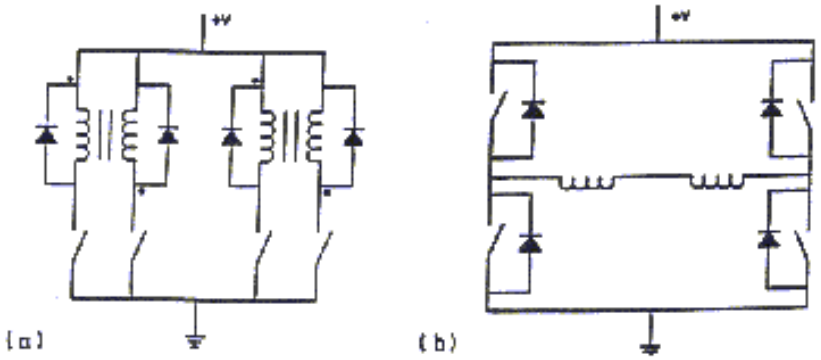


Figura 8.1 - Acionamento unipolar e bipolar

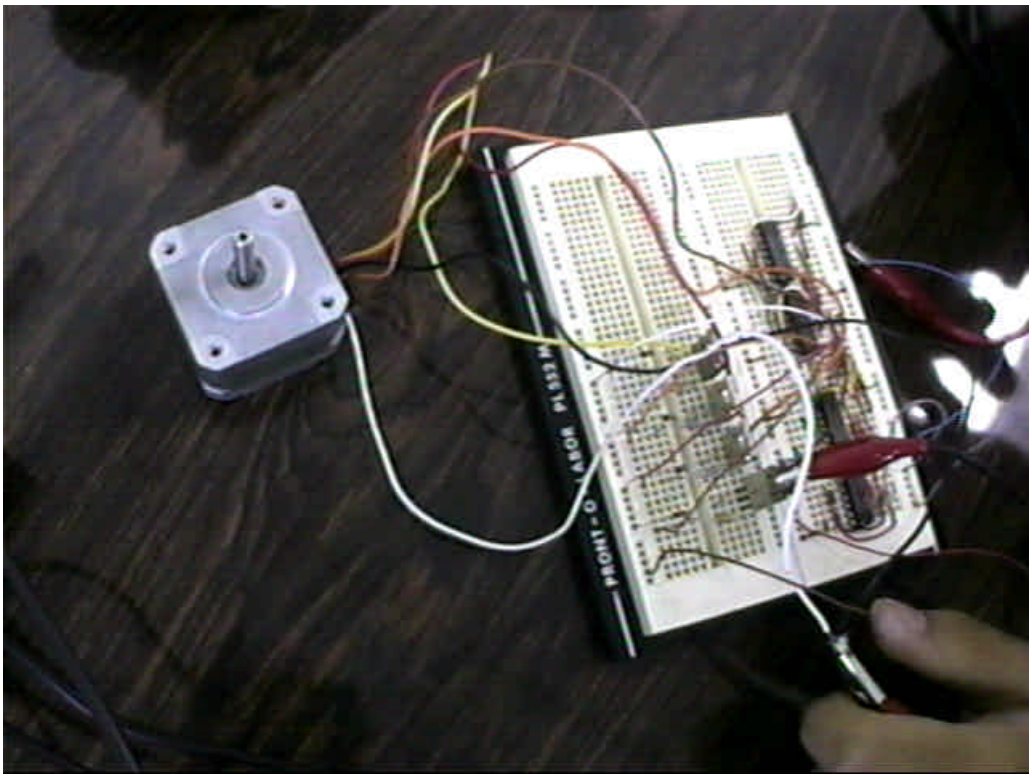


Figura 8.2 – Motor de passo e circuito montado em proto-board.

Existem três modos básicos para geração de pulsos (Figura 8.3), que foram testados em uma montagem em proto-board (Figura 8.2):

- Modo 1: Uma fase é alimentada por vez, de modo que a posição de equilíbrio das fases coincida com a posição de equilíbrio dos ímãs, fornecendo passo pleno.
- Modo 2: Duas fases são alimentadas simultaneamente e a posição de equilíbrio dos ímãs é diferente do Modo 1, fornecendo passo pleno, mas com conjugado $\sqrt{2}$ vezes maior.
- Modo 3: É obtido combinando-se os Modos 1 e 2, ou seja, alternando-se a alimentação de uma única fase com a alimentação simultânea de duas fases. Neste caso o avanço de passo corresponde à metade do passo produzida pelos Modos 1 e 2

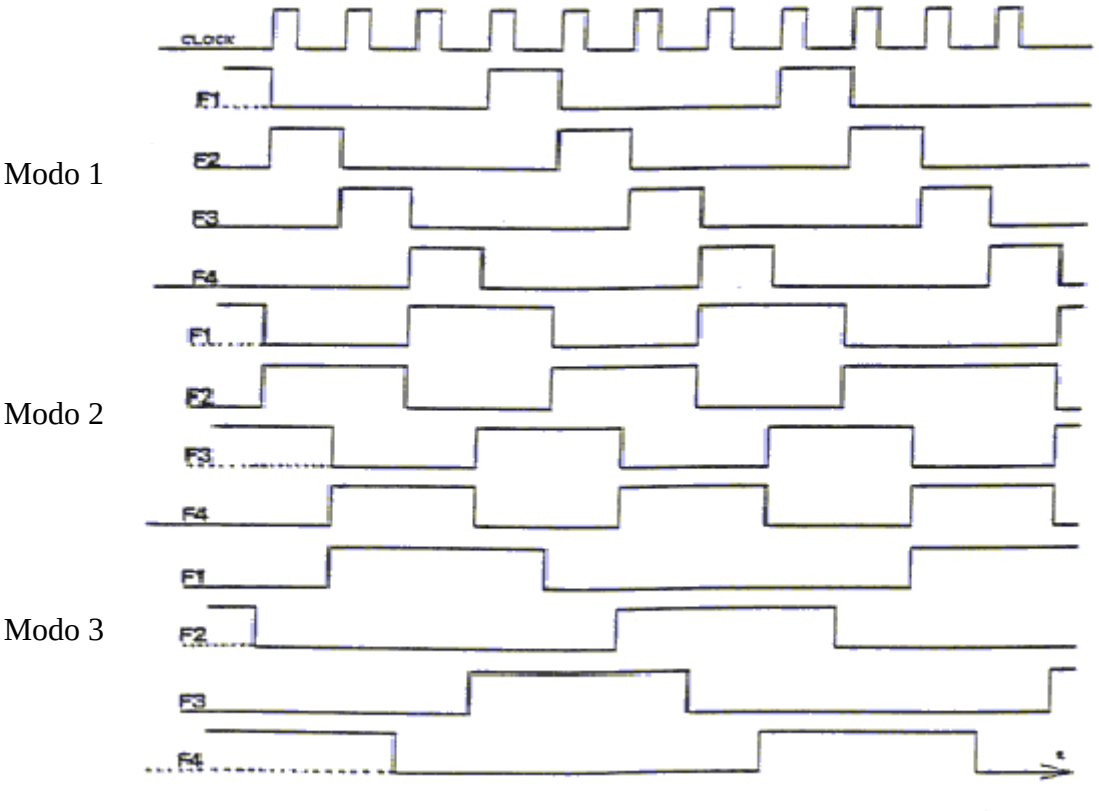


Figura 8.3 – Modos de acionamento de um motor de passo.

8.3 Implementação em VHDL

A implementação do circuito consiste em um sistema lógico que controle os dois motores de passo e um drive de potência que forneça corrente e tensão necessária para os motores, ambos pertencentes ao robô e fixos em uma base de metal (Figura 8.1), observando os seguintes pré-requisitos:

- Os motores não devem começar trabalhar em uma freqüência muito alta (alta velocidade), pois se não houver esse controle pode haver um escorregamento no motor (não dar um "passo", por exemplo);

- Os motores devem girar tanto em sentido horário e em sentido anti-horário, pois assim o robô poderá fazer curvas em qualquer direção e girar em torno do próprio eixo;
- Os motores não devem parar de uma só vez, para também não haver escorregamento.

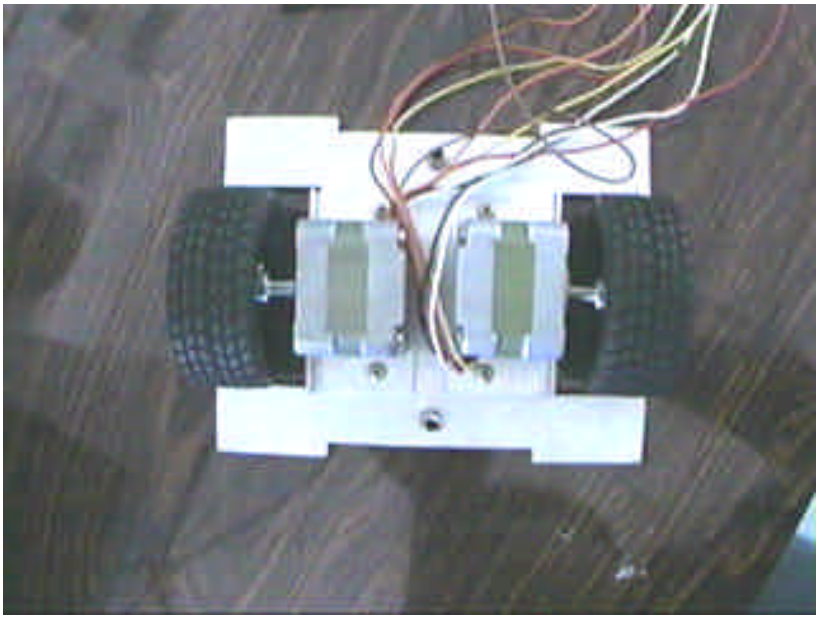


Figura 8.1 - Os motores do robô fixos em uma base de metal.

O sistema lógico que controla os motores pode ser observado na Figura 8.2.

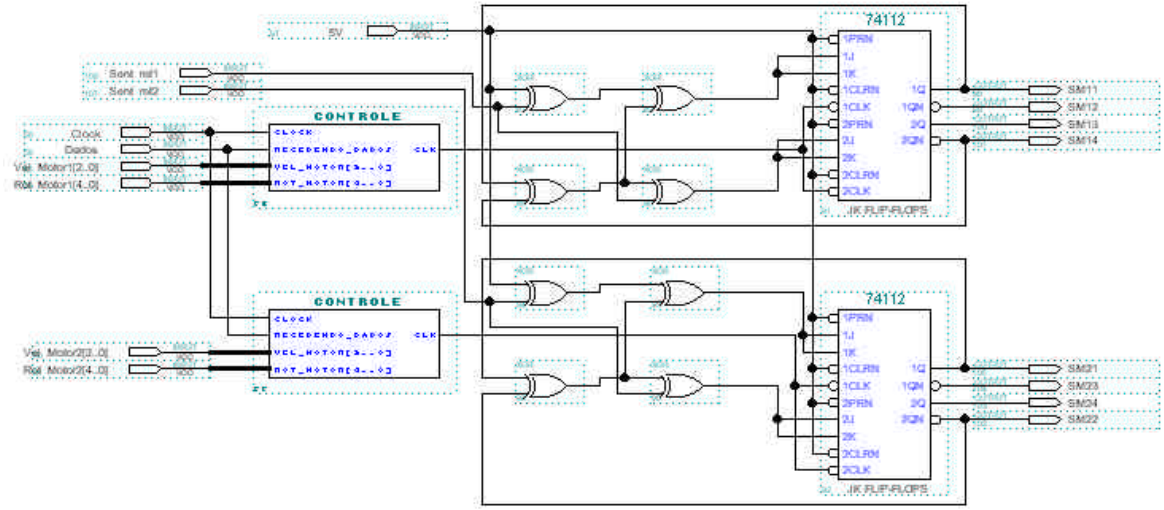


Figura 8.2 – Circuito de controle implementado.

O bloco de controle define e a velocidade e a rotação dos motores e no bloco seguinte ocorrem as divisões de pulsos e do sentido de rotação para cada enrolamento do motor (pólos). Os seguintes dados de entrada, que permitem ao robô realizar manobras no plano, são utilizados pelo circuito:

- Clock de 300 Hz, necessário para temporizar a seqüência de pulsos gerados pelo circuito para os motores.
- Sentido dos motores, para determinar se o motor rotacionará em sentido horário ou sentido anti-horário.
- Velocidade dos motores, o robô e capaz de desenvolver 5 velocidades diferentes em ambos os sentido e esse dado é necessário para controlar essa velocidade
- Rotação do robô, com essa informação o robô e capaz de rotacionar em torno de seu eixo um determinado ângulo.

Um drive de potência é necessário para fornecer a corrente e tensão necessária para os enrolamentos dos motores. Este foi construído com transistores Mosfets IRF530 e resistências. A Figura 8.3 apresenta como o circuito foi montado:

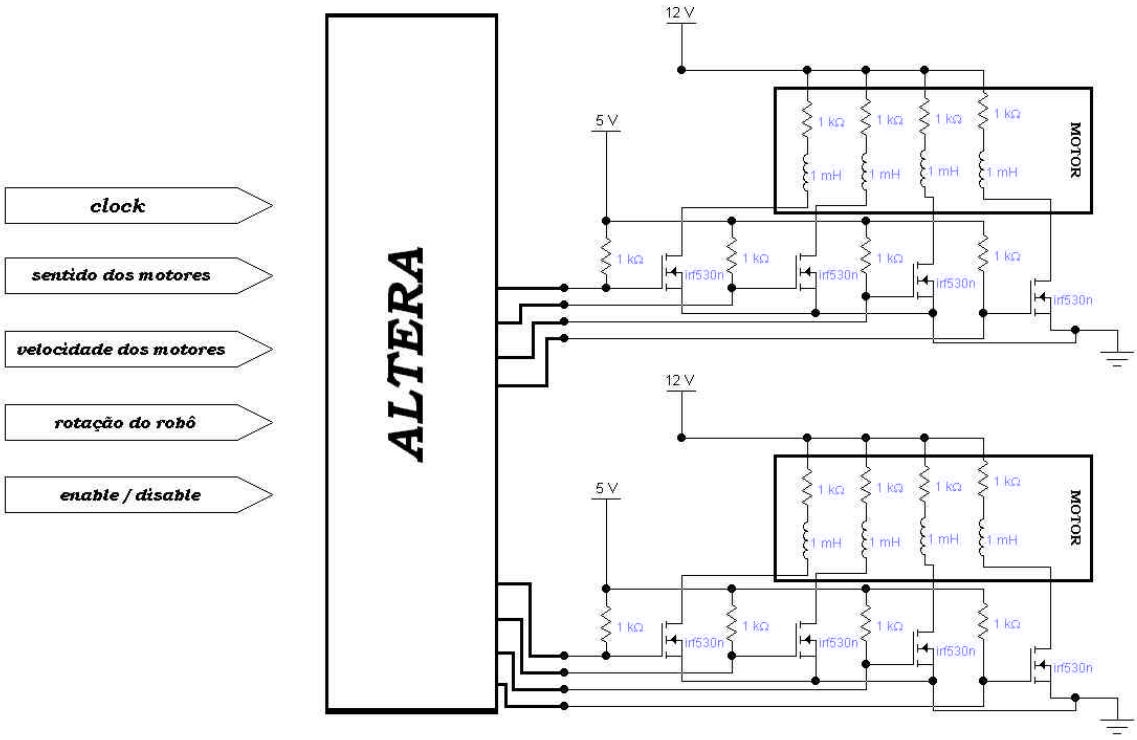


Figura 8.3 – Circuito de potência dos motores.

8.4 Resultados da simulação

Através de simulação utilizando o Software ALTERA Max+Plus 2 foi possível observar o funcionamento do circuito projetado. Nela pode-se observar que o circuito depende da entrada "Dados" aceitar os dados enviados, necessário para casos onde o robô receba duas vezes a mesma informação, por exemplo, virar duas vezes para o mesmo lado com o mesmo número de passos. Observa-se também que há aceleração e desaceleração dependendo dos dados enviados ao controle, pois como já visto anteriormente os motores não podem começar a trabalhar em freqüências altas.

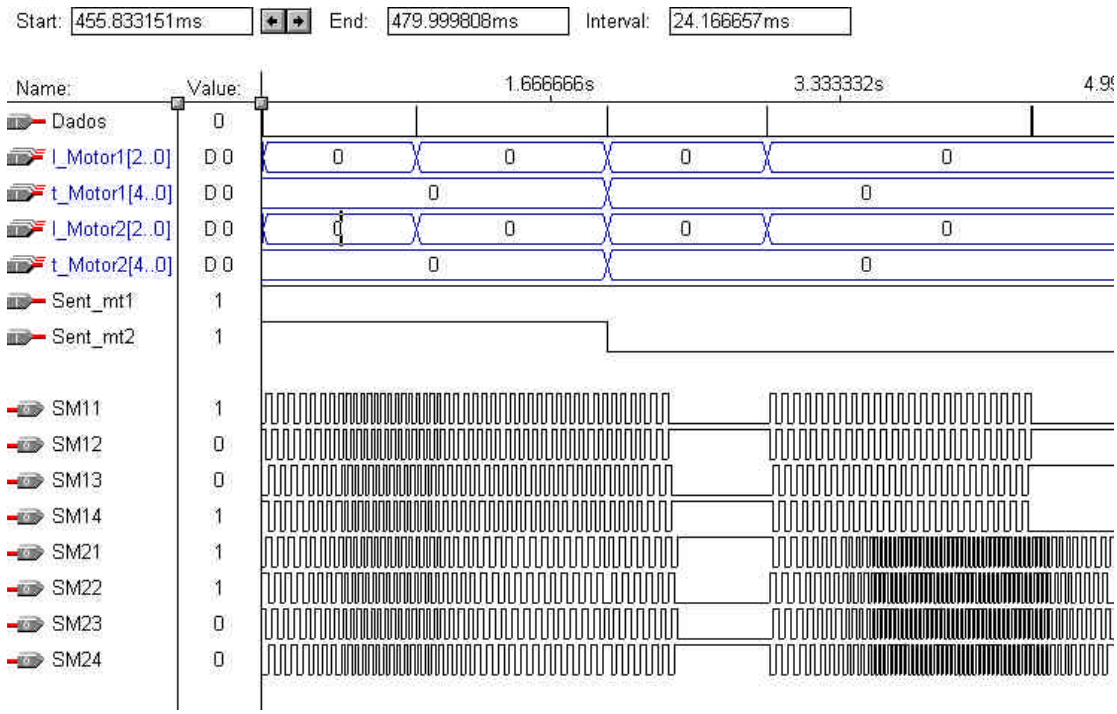


Figura 8.1 – Simulação do controle dos motores.

A figura abaixo apresenta o protótipo resultante deste estudo, com o sistema de controle dos motores e a câmera para aquisição de imagens.

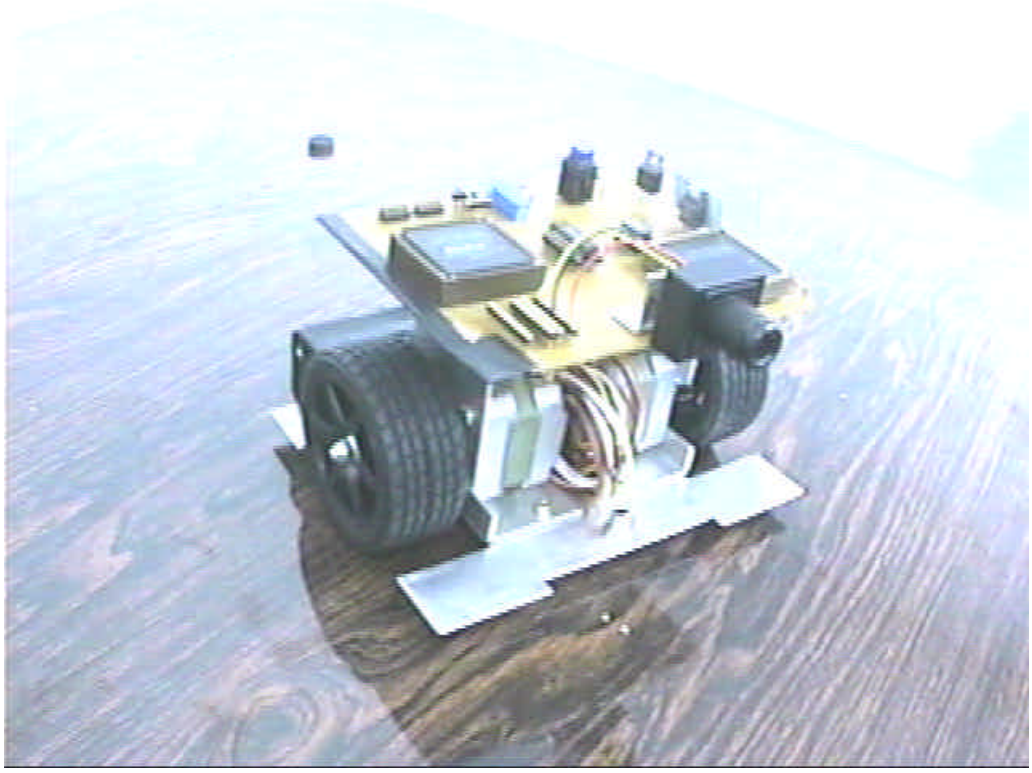


Figura 8.2 - Protótipo do Robô Autônomo para a RoboCup F180.

8.5 Conclusão

A partir dos resultados da simulação e de experiências realizadas com a base móvel pode-se concluir que:

- O circuito descrito é capaz de controlar um motor de passo observando acelerações e desacelerações para que não ocorra perda de passos no motor.
- A implementação em VHDL permite a fácil modificação do sistema e fácil integração com os outros componentes do sistema robótico (visão e estratégia de controle).

Entre as possibilidades de trabalho futuro deve-se estudar a adoção de motores mais econômicos ou motores DC visto que o motor de passo tem um consumo elevado de energia.

9. TRABALHOS CORRELATOS

São discutidos neste capítulo os trabalhos que serviram de base e inspiração para o aqui proposto, dos quais muitas idéias e críticas tiveram origem. Entre os trabalhos que influenciaram esta proposta, dois tiveram maior peso: o de BROOKS (1986, 1991) e o de BOISSIER e DEMAZEAU (1992, 1994).

Este capítulo é iniciado com o trabalho mais controverso e conhecido, o de Rodney Brooks, que tem influenciado muitas áreas de IA, como a de Planejamento Inteligente, onde o paradigma reativo surgiu decorrente destas idéias. Ainda, como já afirmado ao ser apresentada a área de Visão Computacional, o paradigma propositado para VC surge como uma conseqüência do seu trabalho [ALOIMONOS, 1994]. Assim, são apresentadas resumidamente as idéias mais importantes do trabalho de Brooks.

BROOKS (1986, 1991) tem sido um entusiasta de uma abordagem baseada em comportamentos em AI. Ele propôs a chamada *Subsumption Architecture* [BROOKS, 1986], onde um sistema é construído pela composição de camadas dedicadas a tarefas específicas, com um comportamento específico, e onde cada camada interage diretamente com o mundo através de percepção e ação. Nesta arquitetura, "camadas de um nível podem assumir o papel de uma camada em um nível mais baixo suprimindo as saídas desta", e ainda, "camadas em níveis inferiores continuam a funcionar enquanto adiciona-se camadas de níveis mais alto" [BROOKS, 1986, p.152]. Ainda, cada camada é implementada em uma Máquina de Estado Finita (FSM), sendo decomposta em unidades de processamento de informação paralelas e independentes, interligadas por hardware, e que trocam mensagens entre si.

Deve-se enfatizar que a principal característica da *Subsumption Architecture* é permitir que todas as camadas percebam e atuem diretamente no mundo, um método diferente do que se fazia até então, onde só uma camada na base do sistema percebia e atuava, e a decomposição era feita em níveis de processamento, como percepção, navegação e planejamento.

O trabalho de BROOKS (1991) possui muitos outros aspectos importantes, desejados para os sistemas criados com a arquitetura aqui proposta. Entre elas, BROOKS acredita que é necessário que os sistemas:

1. "Dominem apropriadamente e de maneira rápida as mudanças em seu ambiente dinâmico, sejam robustos em relação ao seu ambiente e tenham um propósito em sua existência.
2. Sejam construídos para o mundo real e nele testados.
3. Sejam construídos incrementalmente: os módulos devem ser adicionados ao sistema um de cada vez, e o sistema deve ser testado exaustivamente antes da adição de um novo módulo". [BROOKS, 1991, p.140 e p.145]

A proposta de Brooks foi importante como uma reação às arquiteturas robóticas que estavam sendo desenvolvidas até então, que tinham a tendência de se tornar cada vez mais complexas. ELFES (1986) é um exemplo dos sistemas que estavam sendo construídos, onde muito tempo era gasto com o seu próprio gerenciamento. Como reação a esta tendência, um sistema simples e funcional como o de Brooks foi bem recebido.

O segundo conjunto de trabalhos que influencia esta proposta é o apresentado por BOISSIER e DEMAZEAU (1992) e (1994).

Inicialmente, BOISSIER e DEMAZEAU (1992) propõem uma análise de alguns sistemas de visão computacional baseada em DAI-MAS. Nesta, cada sistema é dividido em agentes focais (focus-agents), agentes de nível (level-agents) e agentes básicos (basic-agents). Os agentes focais surgem com uma divisão vertical do sistema em suas tarefas, e os agentes de nível surgem com a divisão horizontal do sistema, em seus níveis de representação. Os agentes básicos são o resultado da interseção dos agentes de foco e de nível, sendo os blocos básicos do sistema. Pode-se relacionar estes agentes aos paradigmas de Visão Computacional notando que os agentes de nível seguem uma abordagem reconstrutiva e os agentes focais são propositados.

Depois deste trabalho de análise, eles apresentaram em 1994 a ASIC (*Architecture for Social and Individual Control*): "uma arquitetura de controle baseada na abordagem Multi-Agentes, que foi usada para construir o sistema MAVI, que integra diferentes módulos visuais em cooperação" [BOISSIER; DEMAZEAU, 1994, p.107].

A descrição do funcionamento do sistema MAVI mostra como ele tem uma abordagem reconstrutiva: "o primeiro agente alimenta o sistema com uma seqüência de imagens a serem interpretadas e controla a câmera; o segundo reconstrói as características bidimensionais (bordas de constrastes e agrupamento perceptual) a partir das imagens e um terceiro agente constrói uma descrição simbólica da cena a partir das características bidimensionais" [BOISSIER; DEMAZEAU, 1994, p.113]. Como exemplo das primeiras aplicações do sistema MAVI em BOISSIER e DEMAZEAU (1994b) é mostrado como o sistema pode ser utilizado para a realização de detecção de contornos em imagens.

Um outro trabalho importante é o de ELFES (1986), que concorda com BROOKS no sentido em que acredita ser "essencial investigar como os subsistemas são integrados em uma arquitetura global" [ELFES, 1986, p. 135].

ELFES (1986) propõe uma arquitetura de controle onde "processos independentes comunicam através de mensagens em um *blackboard* e guardam informações relevantes no mesmo" [ELFES, 1986, p.144]. O sistema é dividido em níveis de processamento (controle robótico, interpretação sensorial, integração sensorial, ...) e utiliza representações para a modelagem do mundo real. Pode-se notar ainda uma característica da abordagem tradicional em seu sistema de teste, chamado Dolphin, que tem como propósito estudar a extração de características do mundo com sonares e não apresenta um comportamento ou realiza uma tarefa específica.

Nos últimos dois anos surgiram vários trabalhos que utilizam a abordagem baseada em agentes tanto para controle de sistemas robóticos como para sistemas de visão computacional. Entre os trabalhos que utilizam agentes para o controle de sistemas robóticos que influenciaram este projeto estão os abaixo.

NEVES e OLIVEIRA (1997) propõem uma arquitetura multi-agentes para o controle de um robô móvel autônomo. Neste trabalho, o controle de um agente possui três níveis distintos, o cognitivo, o reativo e o reflexivo, onde em cada nível existe uma comunidade de agentes. Assim, os agentes do nível cognitivo orientam os agentes do nível reativo, para que o sistema apresente uma "reação orientada", enquanto os agentes no nível reflexivo são responsáveis pelos comportamentos mais elementares. Uma característica muito forte neste trabalho é o estudo da utilização de aprendizado para a inserção de novos comportamentos no sistema.

GARCIA-ALEGRE e RECIO (1997) apresentam as diretrizes utilizadas na

implementação de Agentes de Comportamento Básicos para o domínio de navegação visualmente guiada. Neste sistema existe uma separação entre o sistema de visão e o sistema de locomoção, e os vários agentes necessários para cada sistema são implementados separadamente. Por exemplo, no sistema de locomoção existem agentes básicos como “Pare”, “Ande” e “Volte”, enquanto no sistema de visão existem agentes básicos chamados de “Saccadic”, “Contorno” e “Centralize”. Estes dois sistemas interagem através de um agente que coordena a utilização dos agentes básicos para que o sistema apresente um comportamento.

Tanto MATRIC (1998) quanto VELOSO et al. (1998) utilizam DAI-MAS para modelar a coordenação em sistemas compostos por múltiplos robôs. O primeiro trabalho se preocupa com o aprendizado de comportamentos nesta classe de sistemas, enquanto o segundo estuda a percepção distribuída e a colaboração de agentes autônomos múltiplos em um ambiente incerto e dinâmico.

Finalmente, FIRBY et al (1995) testaram a abordagem de visão propositada em um robô móvel que realiza tarefas no ambiente de um escritório real, e em particular a tarefa de coleta de lixo, e analisam a adaptação deste tipo de sistema a novas situações. A arquitetura proposta integra um planejador reativo e um sistema de controle baseado em capacidades inteligentes que utiliza rotinas visuais para perceber o ambiente. Este trabalho trata um robô como um agente, e conclui que a abordagem é viável para tal tarefa e que este tipo de sistema leva vantagem ao integrar ação e percepção.

Outros trabalhos que foram importantes para a realização do trabalho proposto e que devem ser citados são :

Os trabalho de SHNEEBELI (1992) e sua continuação em XAVIER (1996) apresentam uma abstração para controladores baseados em agentes onde estes são distribuídos em 3 categorias: Agentes Sensores, Agentes de Comportamento e Agentes Atuadores. Assim, um sistema pode ser construído utilizando classes de agentes sensores e atuadores. Na seqüência deste trabalho, FREIRE et al (1997) utiliza esta arquitetura para construir um sistema sensorial ultra-sônico para um robô móvel.

Como exemplo de utilização da *Subsumption Architecture* objetivando o controle de um sistema de Visão Computacional, PINHANEZ (1994) propõe o seu uso em um sistema com um foco de atenção móvel objetivando a detecção de áreas de alta concentração de bordas em uma cena. O artigo descreve atentamente o uso de métodos baseados em comportamentos para o projeto e a implementação da estrutura de controle de um sensor parecido com a fóvea, que apresenta resultados interessantes.

RIVLIN et al. (1991) estudam o problema do reconhecimento de objetos quando este é considerado no contexto de um agente operando em um ambiente. Eles tratam principalmente da tradução das intenções do sistema em um conjunto de comportamentos, propondo uma metodologia para a construção de sistemas com intenções, comportamentos e tarefas em um ambiente determinado.

FERGUSON (1992) propõe uma arquitetura para o controle de agentes móveis autônomos, na qual um sistema é dividido com base nos seus níveis de representação em três camadas, uma reativa, uma de planejamento e uma de modelagem.

E finalmente, NOREILS e CHATILA (1995) também introduzem uma arquitetura de controle para robôs móveis, e ainda fazem um bom levantamento dos trabalhos relacionados a arquiteturas de controle de robôs.

10. CONCLUSÃO E TRABALHOS FUTUROS

Devido à dificuldade inerente à construção de sistemas de IA e aos resultados limitados obtidos em domínios artificiais bem-comportados, realizar experiências em domínios reais é uma boa maneira de buscar compreensão sobre os sistemas construídos. Experiências podem prover: (i) confirmação preliminar de partes de uma teoria de raciocínio; (ii) sugestões de possíveis modificações para a teoria, para o ambiente onde ela está sendo testada e para o sistema robótico inserido no ambiente. Além disso, podem sugerir um grande número de experiências adicionais, possibilitando a expansão e o fortalecimento da teoria original (HANKS et al, 1993).

Os resultados das experiências realizadas, que foram apresentados em cada capítulo, podem ser resumidos aqui.

Quanto aos algoritmos de Visão Computacional estudados, diversos testes realizados em várias seqüências de imagens permitiram concluir que o algoritmo de *Blob Coloring* não encontra a bola (objeto principal do futebol de robôs) em uma imagem. Já o algoritmo vetorizador não consegue segmentar a bola perfeitamente, mas consegue a localizar na imagem e descobrir a distância desta até o robô e é mais eficiente que o *Blob Coloring*.

Quanto à implementação do sistema em hardware com os mesmos algoritmos implementados em linguagem C, uma comparação da eficiência mostrou que o primeiro sistema alcançou um desempenho superior com a mesma qualidade. Associado com o baixo custo de componentes de FPGA (menor que uma placa Vídeo for Windows), o sistema resultante é um dispositivo de pequeno tamanho que serve bem para robôs autônomos. A avaliação das imagens resultante permitiu concluir que a qualidade dos resultados obtidos também foi satisfatória.

Quanto ao controle de um robô utilizando a Programação Genética, pode-se criar indivíduos que apresentam os comportamentos básicos para atuar no domínio do Futebol de Robôs, como seguir paredes e bolas. A Programação genética permite a criação de indivíduos que produzam os resultados desejados desde que as habilidades necessárias para realização das tarefas e a relação dos indivíduos com o ambiente sejam conhecidas.

Durante o projeto e desenvolvimento do sistema implementado foram aprendidas algumas lições importantes sobre o desenvolvimento de um sistema robótico. Esses conhecimentos, alguns dos quais são assunto de discussão por outros autores [BOND; GASSER, 1988; GARCIA-ALEGRE, 1997], podem ser usados no suporte ao desenvolvimento de novos sistemas e consistem em:

- As dependências entre subproblemas afetam o projeto dos agentes, nos fluxos de dados, nos processos decisórios e nas ações.
- Os conflitos que surgem a partir de ações incompatíveis e da divisão dos recursos do sistema podem colocar restrições nas maneiras de se decompor um sistema e forçando o projetista a considerar a decomposição em diferentes dimensões (temporal, espacial ou níveis de abstração).

Durante o desenvolvimento deste projeto muitas sugestões de aperfeiçoamentos foram discutidas. As sugestões de trabalhos futuros mais importantes são:

- Os algoritmos de Visão Computacional implementados são simples. Como possíveis trabalhos futuros deve-se estudar algoritmos mais robustos e

eficientes para localizar a bola na imagem, como os dedicados ao rastreamento de objetos, os que envolvem Redes Neurais Artificiais (para diminuir a influência da variação de cores) e algoritmos baseados em Lógica Nebulosa.

- Implementar esses mesmos algoritmos citados acima em Linguagem C e VHDL.
- Quanto à Programação Genética, pode-se evoluir comportamentos mais complexos como: goleiro, defesa, ataque, etc. e se inserir a comunicação entre os agentes para a formação de times.
- A implementação de agentes para o Simulador Oficial da RoboCup.
- Aumentar o nível de inteligência e raciocínio de forma que os agentes possam cumprir tarefas mais deliberativas e menos reativas.
- A construção de novos protótipos, com outros tipos de motores, como motores de corrente contínua.
- Deve ser implementada uma maneira de realizar a confirmação sensorial que verifica se uma ação foi realizada. Isso fecharia o laço de controle, configurando o sistema como um servo mecanismo baseado em visão.

Finalmente, o objetivo principal deste projeto — o estudo de sistemas de visão computacional para o controle de robôs autônomos que apresentam comportamento inteligente, cooperando entre si, em um domínio de jogo de futebol de micro robôs — foi completamente atingido, fato que pode ser constatado pelo número e pela qualidade das publicações realizadas.

11. REFERÊNCIAS BIBLIOGRÁFICAS

AGGARWAL, J. K.; MARTIN, W. N. The role of representation and reconstruction in vision: is it a matter of definition? **CVGIP: Image Understanding**, v.60, n.1, p.100-2, July 1994.

ALOIMONOS, Y., ed. Special issue on purposive, qualitative and active vision. **CVGIP: Image Understanding**, v.56, n.1, 1992.

ALOIMONOS, Y. What I have learned. **CVGIP: Image Understanding**, v.60, n.1, p.74-85, July 1994.

ALOIMONOS, Y.; ROSENFELD, A. A response to "Ignorance, myopia and naiveté in computer vision systems" by R. C. Jain and T. O. Binford. **CVGIP: Image Understanding**, v.53, n.1, p.120-24, 1991.

ANDRAKA CONSULTING GROUP. "What is an FPGA?" http://users.ids.net/~randraka/whatisan.htm

ARCELLI, F.; DE SANTO, M.; DI SALVO, S. Software Agents for Computer Vision: a Preliminary Discussion. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 31, Hawaii, 1998. **Proceedings**. Los Alamitos, IEEE, 1998. v. 5, p. 9-17.

BAJCSY, R. Active perception. **Proceedings of the IEEE,** v.78, n.8, p.996-1005, 1988.

BAJCSY, R.; CAMPOS, M. Active and exploratory perception. **CVGIP: Image Understanding**, v.56, n.1, p.31-40, July 1992.

BALLARD, D. H. Hierarchic Recognition of Tumor in Chest Radiographs. Basel, Birkhauser-Verlag, 1976.

BALLARD, D. Animate vision. **Artificial Intelligence,** v.48, p.57-86, 1991.

BALLARD, D. H.; BROWN, C. M. **Computer Vision.** Englewood Cliffs, Prentice-Hall, 1982.

BALLARD, D.; BROWN, C. M. Principles of animate vision. **CVGIP: Image Understanding**, v.56, n.1, p.3-21, July 1992.

BIANCHI, R. A. C. **Uma Arquitetura de Controle Distribuída para um Sistema de Visão Computacional Propositada.** São Paulo, 1998. Dissertação (Mestrado) -

Escola Politécnica, Universidade de São Paulo.

BLACK, M. J.; ALOIMONOS, J.; HORSWILL, I.; SANDINI, G.; BROWN, C.M.; MALIK, J.; TARR, M.J. Action, representation, and purpose: re-evaluating the foundations of computational vision. In: INTERNATIONAL JOINT CONFERENCE ON AI, 1993. **Proceedings.** 1993. p.1661-6.

BOISSIER, O.; DEMAZEAU, Y. A distributed artificial intelligence view on general purpose vision systems. In: DEMAZEAU, Y; WERNER, E. (eds.) **Decentralized AI-3.** Amsterdam, Elsevier, 1992. p.311-330.

BOISSIER, O; DEMAZEAU, Y. ASIC: An architecture for social and individual control and its application to Computer Vision. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, 1994. **Proceedings.** 1994. p.107-18.

BOISSIER, O; DEMAZEAU, Y. Finding Contours Using a Multi-Agent System. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1994. **Proceedings.** 1994.

BOND, A. H; GASSER, L. **Readings in Distributed Artificial Intelligence**. Morgan Kaufmann, San Mateo, CA, 1988.

BOUCHER, A.; GARBAY, C. A multi-agent system to segment living cells. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 13th., 1996. **Proceedings**. Los Alamitos, IEEE, 1996. v.3, p.558-62.

BROOKS, R. A. A robust layered control system for a mobile robot. **IEEE Journal of Robotics and Automation**, VRA-2, n.1, p.14-23, March 1986.

BROOKS, R. A. Intelligence without representation. **Artificial Intelligence**, v.47, p.139-59, 1991.

BROOKS, R. A. **Intelligence without Reason**. MIT AI MEMO No.1293, 1991b.

BROOKS, R. A.; STEIN, L. A. **Building Brains for Bodies**. MIT AI MEMO No. 1439, 1993.

BROWN, C. M. Toward General Vision. **CVGIP: Image Understanding**, v.60, n.1, p.89-91, July 1994.

BROWN, S.; ROSE, J. Architecture of FPGAs and CPLDs: A Tutorial. **IEEE Design**

and Test of Computers, v. 13, no. 2, pp. 42-57, 1996.

CHAIB-DRAA, B. Industrial Applications of Distributed AI. In: HUHNS, M. N.; SINGH, M. P. (eds.) **Readings In Agents**. São Francisco, Morgan Kaufmann, 1997.

CHENG, T. K.; KITCHEN, L.; LIU, Z.; COOPER, J. An Agent-Based Approach for Robot Vison System. Technical Report no. 95/341995, Computer Science Department, University of Melbourne, Austrália, 1995.

CHRISTENSEN, H. I.; MADSEN, C. B. Purposive Reconstruction: A Reply to "A Computational and Evolutionary Perspective on the Role of Representation in Vision" by M. J. Tarr and M. J. Black. **CVGIP: Image Understanding**, v.60, n.1, p.103-8, July 1994.

COOPER, M. **Visual Occlusion and the Interpretation of Ambiguous Pictures**. Chicester, Ellis Horwood, 1992.

DEMAZEAU, Y; MÜLLER, J. Decentralized Artificial Intelligence. In: DEMAZEAU, Y; MÜLLER, J. (eds.) **Decentralized AI 1.** Amsterdam, Elsevier, 1990. p.3-13.

DROGOUL, A. When Ants Play Chess (or Can Strategies Emerge from Tactical Behaviors?). In:CALSTELFRANCHI, C; MÜLLER, J. P. (eds.) **From reaction to cognition - Lecture Notes in AI 957**. Berlin-Heidelberg, Springer-Verlag, 1995. p. 13-27.

EDELMAN, S. Representation without reconstruction. **CVGIP: Image Understanding**, v.60, n.1, p.92-4, July 1994.

ELFES, A. A distributed control architecture for an autonomous mobile robot. **Artificial Intelligence**, Computational Mechanics Publications, v.1, n.2, 1986, p.135-44.

FERGUSON, I. A. Toward an architecture for adaptative, rational, mobile agents. **Decentralized AI 3**. Amsterdam, Elsevier, p.249-61, 1992.

FIRA - Federation of International Robot-Soccer Association. **The rules of Mirosot.** FIRA, 1998. (http://www.fira.net/fira/98fira/rules.html)

FIRBY, R. J. Opening the door to robotic agents. **IEEE Expert,** v.12, n.2, p.7-9, Mar. 1997.

FIRBY, R. J.; KAHN, R. E.; PROKOPOWICZ, P. N.; SWAIN, M. J. Collecting trash:

a test of Purposive Vision. Submitted to the Workshop on Vision for Robots, Pittsburg, PA, 1995.

FISHER, R. B. Is Computer Vision still AI? **AI Magazine**, p.21-7, Summer, 1994.

FISCHLER, M. A. The modeling and representation of visual information. **CVGIP: Image Understanding**, v.60, n.1, p.98-9, July 1994.

FRANKLIN, S. **Artificial Minds.** Cambridge, MA, MIT Press, 1995.

FRANKLIN, S.; GRAESSER, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 3., Heidelberg, Alemanha, 1996. **Proceedings.** Berlin, Springer Verlag, 1996.

FREIRE, E. O.; BASTOS, T. F.; DINNIKIV, V.; SHNEEBELI, H. J. Sistema Sensorial Ultra-Sônico para Robô Móvel com Controle Baseado em Agentes. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 3º. Vitória, 1997. **Anais**. Vitória, UFES-SBA, 1997. p.471-6.

GARCIA-ALEGRE, M. C.; ROCIO, F. Basic Agents for Visual/Motor Coordination of a Mobile Robot. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 1, Marina Del Rey, 1997. **Proceedings**. Marina Del Rey**,** ACM, 1997.

GASSER, L. Foreword. In: HUHNS, M. N.; SINGH, M. P. (eds.) **Readings In Agents**. São Francisco, Morgan Kaufmann, 1997a. p.v-vi.

GEIST, G. A.; KOHL, J. A.; PAPADOPOULOS, P. M. PVM and MPI: a Comparison of Features. **Calculeteurs Paralleles,** v. 8, n. 2, 1996.

HANKS, S.; POLLACK, M. E. ; COHEN, P. R. Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures. **AI Magazine**, v. 17, n. 42, 1993.

HAYES-ROTH, B. An Architecture for Adaptive Intelligent Systems. **Artificial Intelligence (Special Issue on Agents and Interactivity)**, 72, p.329-65, 1995.

HENDLER, J.; MUSLINER, D. **Research in the Autonomous Mobile Robotics Laboratory**, University of Maryland, College Park, Maryland, 1994.

HEWITT, C. Viewing Control Structures as Patterns of Passing Messages. **J. Artificial Intelligence**, v.8, n.3, p. 323-64, June 1977.

HUHNS, M. N.; SINGH, M. P. Agents and Multiagent Systems: Themes, Approaches, and Challenges. In: HUHNS, M. N.; SINGH, M. P. (eds.) **Readings In Agents**. São Francisco, Morgan Kaufmann, 1997.

JAIN, R. C. Expensive Vision. **CVGIP: Image Understanding**, v.60, n.1 p.86-8, July 1994.

JAIN, R. C.; BINFORD, T. O. Ignorance, myopia and naiveté in computer vision systems. **CVGIP: Image Understanding**, v.53, n.1, p.112-17, 1991.

JENNINGS, N. R et al. Using ARCHON to develop real-world DAI applications. **IEEE Expert**, v. 11, n, 6, p. 64-70, Dec 1996.

JOLION, J.M. Computer vision methodologies. **CVGIP: Image Understanding**, v.59, n.1, p.53-71, Jan. 1994.

KENJO, T. **Stepping motors and their microprocessor control Monographs in Electrical and Electronic Engineering**. OXFORD, 1992.

KIM, J.H.; VADAKKEPAT, P.; VERNER, I.M. "FIRA Robot World Cup Initiative and Research Directions". **Int. J. of Robotics and Automation Systems**, 1998.

KIROUSIS, L.; PAPADIMITRIOUS, C. The complexity of recognizing polyhedral scenes. In: ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 1985. **Proceedings**. Los Alamitos, IEEE, 1985, p. 175-85.

KITANO, H. et al. "RoboCup: A challenge Problem for AI". **AI Magazine**, v. 18, n. 1, p. 73-85, Spring 1997.

KITCHIN, P. W.; PUGH, A. Processing of Binary Images. In: PUGH, A. (editor) **Robot Vision.** Berlin, Springer, 1983. P. 21–42.

KOZA, J. **Genetic programming: on the programming of computers by means of natural selection.** Cambridge, MA, MIT Press, 1992.

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

LI, Z. N. **Multimedia Systems Class Notes**. - School of Computing Science - Simon Fraser University, 1998. (http://www.cs.sfu.ca/fas-info/cs/CC/365/li/material/notes/Chap3/Chap3.3/Big.html)

MARR, D. **Vision**. New York, Freeman, 1982.

MATRIC, M. J. Coordination and learning in multirobot systems. **IEEE Intelligent Systems**, v.13, n.2, p.6-8, April 1998.

MEDEIROS, A. A. D.; CHATILA, R. Priorities and data abstraction in hierarchical control architectures for autonomous robots. In: WORKSHOP ON INTELLIGENT ROBOTICS, 1º, Brasília, 1997. **Proceedings.** Brasília, 1997. p. 207-220.

NEVES, M. C.; OLIVEIRA, E. A Control Architecture for an Autonomous Mobile Robot. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 1, Marina Del Rey, 1997. **Proceedings**. Marina Del Rey, ACM, 1997.

NOREILS, F. R.; CHATILA, R. G. Plan execution monitoring and control architecture for mobile robots. **IEEE Transactions on Robotics and Automation**, v.11, n.2, April 1995.

PINHANEZ, C. S. Behavior-based Active Vision. **International Journal of Pattern Recognition and Artificial Intelligence**, v.8, n.6, p.1493-1526, 1994.

PRESSMAN, R. **Engenharia de Software.** São Paulo, Makron Books, 1995.

RILLO, A.H.R.C. **Sistema de Visão Binária com Reconhecimento de Peças Parcialmente Oclusas.** São Bernardo do Campo, 1989. Dissertação (Mestrado) - Faculdade de Engenharia Industrial.

RILLO, A.H.R.C., **RECTRI: um sistema de reconhecimento tridimensional a partir de uma única imagem de intensidade luminosa.** São Paulo, 1994. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo.

RILLO, A. H. R. C., BIANCHI, R. A. C., MOREIRA Jr, B., FERRAZ, F. "Integrando Visão e Comportamento: Uma aplicação de reconstrução propositiva". In:. CONGRESSO BRASILEIRO DE AUTOMÁTICA, 11., São Paulo, 1996. **Anais.** São Paulo, SBA, 1996. pp. 573-8.

RILLO, M.; RILLO, A.H.R.C.; COSTA, L.A.R. The LSI assembly cell. In: IFAC/IFIP/IFORS/IMACS/ISPE Symposium on information control problems in manufacturing technology, 7º, Toronto, 1992. **Proceedings.** IFAC, 1992. p. 361-5.

RIVLIN, E.; ALOIMONOS, Y.; ROSENFELD, A. **Purposive Recognition: a Framework.** Technical Report CAR-TR-597, CS-TR-2811, DACA76-89-C-0019,

IRI-9057934, University of Maryland, Dec. 1991.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach.** Englewood Cliffs, Prentice Hall, 1995.

SANDERSON, A. "Micro-Robot World Cup Soccer Tournament (MiroSot)". **IEEE Robotics and Automation Magazine**, pg.15, December 1997.

SANDHOLM, T.; LESSER, V. Issues in automated negotiation and eletronic commerce: extending the contract net framework. In: INTERNATIONAL CONFERENCE IN MULTIAGENTS SYSTEMS, 1995, **Proceedings,** 1995. p. 328-335.

SANDINI, G.; GROSSO, E. Why purposive vision? **CVGIP: Image Understanding**, v.60, n.1 p.108-12, July 1994.

SHEN, W. et al. Integrated Reacive Soccer Agents. In: ROBOCUP WORKSHOP, 2, Paris, 1998. **Proceedings.** FIRA, 1998.p. 251-264.

SHNEEBELI, H. A. **Die Steuerung von Mehrfinger-Greifersystemen**. Karlsruhe, Alemanha, 1992. Tese (Doutorado) - Universidade de Karlsruhe.

SICHMAN, J. S.; DEMAZEAU, Y; BOISSIER, O. When can knowledge-based systems be called agents? In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., Rio de Janeiro, 1992. **Anais.** Rio de Janeiro, SBC, 1992. p.172-85.

SMITH, R. G. The Contract Net Protocol: high-level communication and control in a distributed problem solver. **IEEE Transactions on Computers**, v.29, n.12, p.1104-13, Dec 1980.

SMITH, D. C.; CYPHER, A.; SPOHRER, J. KidSim: Programming Agents Without a Programming Language. **Communications of the ACM**, v.37, n.7, p.55-67, 1994.

SPINU, C.; GARBAY, C.; CHASSERY, J. M. A multi-agent approach to edge detection as a distributed optimization problem. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 13th.,1996. **Proceedings**. Los Alamitos, IEEE, 1996. v.2, p. 81-5.

STONE, P.; VELOSO, M. Multiagent Systems: A Survey from a Machine Learning Perspective. CMU Internal Report, February, 1997.

Sun Microsystems Computer Corporation **Sun Video User's Guide**. Revision A, August, 1994.

SWAIN, M.J.; STRICKER, M. **Promising Directions in Active Vision.** Technical Report CS 91-27, University of Chicago, Nov. 1991.

TALUKDAR, S.; SOUSA, P.; MURTHY, S. Organization for computer-based agents. **Eng. Int. Syst.**, v.2, p. 75-87, 1993.

TAMBE, M. Implementing Agent Teams in Dynamic Multi-Agent Environments. **Applied Artificial Intelligence**, v12, March 1998.

TARR, M. J.; BLACK, M. J. A computational and evolutionary perspective on the role of representation in vision. **CVGIP: Image Understanding**, v.60, n.1, p.65-73, July 1994.

TARR, M. J.; BLACK, M. J. Reconstruction and purpose**.** **CVGIP: Image Understanding**, v.60, n.1, p.113-8, July 1994b.

TRIVEDI, M.M.; ROSENFELD, A. On making computers "See". **IEEE transactions on systems, man and cybernetics,** v.19, n.6, p.1333-6, Nov. 1989.

TSOTSOS, J. K. Complexity of Perceptual search tasks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 11, Detroit, 1989. **Proceedings**. AAAI, 1989. 1571-7.

TSOTSOS, J. K. A Complexity Level Analysis of Vision. **Behaviour and Brain Sci**, v.13, p.423-55, 1990.

TSOTSOS, J. K. There is no one way to look at vision. **CVGIP: Image Understanding**, v.60, n.1, p.95-7, July 1994.

VAN DE VELDE, W. Toward learning robots. **Robotics and Autonomous Systems**, v.8, n.1, pp 1-6, Nov. 1991.

VELOSO, M.; STONE, P; HAN, K. The CMUnited-97 Robotic Soccer Team: Perception and Multiagent Control. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2, Minneapolis, 1998. **Proceedings.** AAAI, 1998.

WALTZ, D. I. "*Generating semantic descriptions from drawings of scenes with shadows*". In **The Psicology of Computer Vision**. New York, McGraw-Hill, 1975.

WERNER, E. The design of Multi-Agent Systems. In: DEMAZEAU, Y; WERNER, E.

(eds.) **Decentralized AI-3.** Amsterdam, Elsevier, 1992. p.3-28.

WOOLDRIDGE, M.; JENNINGS, N. R. Agent Theories, Architectures, and Languages: a Survey. In: WOOLDRIDGE, M.; JENNINGS, N. R. (eds.) **Intelligent Agents.** Berlin, Springer-Verlag, 1995. p.1-22.

XAVIER, J. E. M. **Uma Estrutura para a Construção de Sistemas de Controle Baseados em Agentes para Robôs Móveis**. Vitória, 1996. Dissertação (Mestrado) - Universidade Federal do Espírito Santo.