Sistema de Visão Baseado em Redes Neurais Artificiais para o Controle de Robôs Móveis [1]

Marcos Gonçalves Quiles

Orientadora: *Profª Drª Roseli Aparecida Francelin Romero*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC/USP, como parte dos requisitos necessários para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Novembro/2004

[1] Trabalho realizado com apoio financeiro do CNPq.

Candidato: Marcos Gonçalves Quiles

A Comissão Julgadora:

Profa. Dra. Roseli Aparecida Francelin Romero _____

Prof. Dr. Eduardo do Valle Simões _____

Prof. Dr. Marco Henrique Terra _____

Este documento foi preparado utilizando-se o formatador de textos LaTeX. Sua bibliografia é gerada automaticamente pelo BibTeX, utilizando o estilo Chicago.

Resumo

Sistemas de Visão Computacional (SVCs) são interessantes ferramentas para a navegação de robôs. Este trabalho propõe um SVC baseado em redes neurais multi-camadas para o controle de um robô móvel. Dois módulos principais compõe o SVC: módulo de visão e módulo de controle de navegação. O primeiro é dividido em três partes: pré-processamento, segmentação e reconhecimento das imagens. Este módulo processa as imagens (obtidas por uma câmera) que podem conter diversos objetos com cores diferentes, retornando a posição ou a forma e a posição de um dos objetos, o qual corresponde a cor especificada. O processamento das imagens é realizado por redes neurais multi-camadas. O *módulo de controle* é responsável por validar os resultados do módulo de visão e conduzir o robo utilizando os dados provenientes do módulo de visão. O objetivo do sistema proposto é capacitar um robô a realizar as tarefas: seguir um objeto de cor determinada, seguir um objeto de cor e forma determinada ou ainda navegar pelo ambiente seguindo um objeto de cor determinada evitando obstáculos. Vários experimentos são apresentados, em um ambiente real, utilizando o robô Pioneer I, para mostrar as vantagens e desvantagens do sistema proposto.

Abstract

Computer Vision Systems (CVSs) are important tools for robot navigation. This work proposes a CVS based on Multi-Layer Neural Networks for a mobile robot control. Two main modules compose the CVS: vision module and navigation control module. The first is divided into three parts: preprocessing, segmentation and image recognition. This module processes the images (obtained by a camera) that can contain several objects with different colors, returning the position or the shape and position of one of these objects, which corresponds to a specified color. The image processing is carried out by multi-layer neural networks. The control module validates the results from the vision module and leads the robot using the data from the vision module. The aim of the proposed system is to enable a robot to perform the tasks: to follow an object with a specified color, to follow an object with a specified color and shape or even to navigate in an environment following an object with a specified color avoiding obstacles. Some experiments are presented in a real environment with the robot Pioneer I in order to show the advantages and disavantages of proposed system.

Aos meus pais Izaias e Lucinda, por todo amor, força e incentivo que me deram em todos os momentos da minha vida.

A meu grande amor Camila, uma pessoa maravilhosa que eu tive a oportunidade de conhecer e me apaixonar, (*I can't live without anymore*).

A Nayara, por despertar em mim um lado paterno que antes não existia.

Aos meus irmãos, porque nem mesmo a distância que nos separa é capaz de quebrar o laço de sangue que existe entre nós.

Amo muito vocês,
Marcos Quiles.

Agradecimentos

Primeiramente, a minha mais que orientadora Roseli Aparecida Francelin Romero, por todo o apoio, incentivo e orientação durante esses quase dois anos de mestrado. Vou ser sempre grato a você por me conduzir nesse caminho que me transformou em um Mestre.

A dois grandes professores e amigos, Eduardo Simões e Alexandre Delbem, por todo incentivo e apoio que muito ajudaram no meu crescimento científico e pessoal, e por todas as oportunidades a mim concedidas.

Ao meu amigo Luiz Antônio de Campos, por sempre estar disposto a me ajudar com seus bons conselhos.

Ao meu amigo Alan Gavioli, por toda amizade, companheirismo e paciência nesses últimos anos.

A Reginaldo, Denise e Família por todo o carinho que me deram e pala forma que me receberam em suas casas, vou ser sempre grato a vocês.

A Eduardo pelo incentivo e ajuda concedida na revisão de alguns trechos da dissertação.

Aos companheiros Andrew, Vinícius e Ednaldo, pela amizade construída durante o mestrado e pelas conversas e discussões muito produtivas.

Ao Gedson, pela ajuda com a parte de campos potenciais, pelas dicas sobre o meu sistema.

Ao grande Mauro, pela ajuda com os Algoritmos Evolutivos e pela amizade que nós construímos.

Aos meus amigos Rodrigo e Débora, por todo o apoio durante o início do meu mestrado.

Ao Edson, pelas dicas, pela consultoria do LaTeXe por estar sempre disposto a me ajudar.

A todos os professores que tive por toda a minha vida, pois, cada um de vocês foram um pouco responsáveis por esta conquista.

A todos do Labic pelo companheirismo e por toda ajuda concedida quando necessário.

E por fim ao CNPq pelo apoio financeiro concedido.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo

1

Introdução

O desenvolvimento de robôs, que possam auxiliar ou substituir os seres humanos em tarefas de risco ou onde a precisão de determinados movimentos é essencial, tem atraído a atenção de diversos setores da sociedade. Nas últimas décadas, com a modernização e crescimento dos recursos computacionais, os pesquisadores têm buscado a elaboração de robôs inteligentes, que estejam aptos a interagir de forma amigável com os seres humanos e que sejam capazes de atuar de forma autônoma em ambientes desestruturados (Pagello et al. 2002), em contraste com os primeiros robôs estritamente mecânicos, comumente instalados em ambientes de chão de fábricas.

Para diversas aplicações reais, como a tarefa de distribuição de peças dentro de uma indústria, robôs fixos não se mostram adequados. Assim, o desenvolvimento de robôs móveis é fundamental tanto para essas pequenas tarefas como para objetivos maiores, como, por exemplo, no caso da exploração espacial, marítima, ou mesmo no controle de um incidente em uma usina nuclear. Nesses ambientes, os robôs móveis poderão substituir os seres humanos, ou pelo menos, realizar uma exploração prévia das condições do ambiente antes que pessoas possam atuar.

Para o controle de robôs móveis, diversos mecanismos isolados ou mesmo a combinação deles têm sido utilizados. É o caso dos sonares, dispositivos de visão, laser, infravermelho, entre outros. Este trabalho se concentra no estudo de modelos de redes neurais para o desenvolvimento de um sistema de visão, aplicado ao controle de um robo móvel.

Diversos trabalhos têm sido desenvolvidos utilizando redes neurais multicamadas para a construção de sistemas de visão, objetivando o controle de robôs nas mais diversas atividades. No trabalho apresentado por Pomerleau (1995), um sistema foi construído com o objetivo de guiar um veículo autonomamente sobre uma pista. Em um segundo trabalho (Blank & Ross 1997), foi desenvol-

vido um sistema para capacitar o robô a seguir uma bola sobre uma superfície. Em Jonsson et al. (1997), um robô foi treinado para caminhar em um corredor desviando dos possíveis obstáculos. Tyler & Czarnecki (1999) apresentaram um sistema de visão 1D para o futebol de robôs. Um outro trabalho utilizando visão foi proposto em Waldherr et al. (2000) no qual uma interface para o controle de um robô baseado em gestos foi desenvolvida. Medeiros & Romero (2002) utilizaram uma rede neural multi-camadas para capacitar um robô do tipo Pioneer I a percorrer sobre um caminho delimitado por duas faixas laterais brancas.

Entretanto, dos diversos trabalhos citados acima, apenas o trabalho desenvolvido por Waldherr et al. (2000) trata cores no sistema de visão, e o tratamento de cores pode ser de fundamental importância em diversos domínios, pois esta informação adicional pode representar características não expressas quando apenas a luminância é utilizada (níveis de cinza). Por exemplo, no desenvolvimento de aplicações relacionadas ao futebol de robôs, a análise de cores pode ser um fator decisivo (Simões & Costa 2000b; Simões & Costa 2000a).

Segundo Cheng et al. (2001), a segmentação é um dos primeiros passos em análise de imagens e reconhecimento de padrões, sendo considerada uma componente essencial com impacto direto no resultado final do processamento.

No desenvolvimento de um sistema de visão com tratamento de cores, a segmentação pode ser fundamental a fim de destacar os objetos de interesse presentes nas imagens analisadas, o que pode facilitar a realização de uma determinada tarefa pelo sistema, por exemplo o reconhecimento de faces. Contudo, uma das maiores dificuldades no desenvolvimento de um sistema de segmentação de imagens coloridas está nas variações de luminosidade presentes em ambientes reais, fator este que está intimamente relacionado a forma como as cores estão representadas (espaço de cores) e ao método responsável pela realização da segmentação.

Vários trabalhos relacionados ao processo de segmentação de imagens coloridas, utilizando as mais diversas técnicas e espaços de cores, têm sido desenvolvidos, entre as quais: Redes Neurais (Simões & Costa 2000a; Simões & Costa 2000b), (Li et al. 2003), (Ong et al. 2002); Sistemas *Fuzzy* (Chen & Lu 2002) e *Multithresholding* (Papamarkos et al. 2000). Entretanto, nenhuma dessas técnicas é capaz de produzir resultados satisfatórios para todos os domínios (Cheng et al. 2001). Além disso, a grande maioria dos métodos de segmentação disponíveis não são capazes de atuar em tempo real, impossibilitando assim, sua aplicação no desenvolvimento de sistemas para controle de robôs.

Este trabalho propõe um Sistema de Visão Computacional (SVC) baseado em redes neurais para o controle de robôs móveis com câmera embarcada. Como

será detalhado nos capítulos seguintes, para o desenvolvimento deste trabalho dois módulos principais são implementados: o módulo de visão e o módulo de controle.

O módulo de visão é responsável pelo pré-processamento das imagens capturadas, segmentação e pelo reconhecimento das imagens. A segmentação é implementada utilizando-se um método de classificação de cores baseado em redes neurais multi camadas. A fase de reconhecimento das imagens, também realizada por redes neurais multi-camadas, possui duas tarefas: informar a posição do objeto ou informar a posição e a forma do objeto.

Com base nas informações obtidas a partir do módulo de visão, o módulo de controle foi elaborado para agregar as seguintes capacidades ao robô:

- perseguir um objeto de cor determinada;
- perseguir um objeto de cor e forma determinada;
- navegar em um ambiente buscando um objeto de cor determinada e ao mesmo tempo desviando de obstáculos.

Após o desenvolvimento do sistema, diversos experimentos foram realizados com o robô Pioneer I em um ambiente real, com o objetivo de validar o SVC desenvolvido.

Esta dissertação está organizada da seguinte forma. No Capítulo 2, são apresentados alguns trabalhos relevantes que utilizaram técnicas visão computacional baseados em redes neurais artificiais aplicados ao controle de robôs móveis. Além disso, também é apresentada uma breve seção sobre segmentação de imagens coloridas. No Capítulo 3, estão descritas as técnicas de computação que foram utilizadas na elaboração deste trabalho. O sistema proposto é apresentado no Capítulo 4, onde os dois módulos que compõe o sistema são detalhados. Os experimentos realizados, com o objetivo de estabelecer os parâmetros do sistema proposto, bem como os resultados obtidos a partir dos experimentos realizados com o robo, são descritos no Capítulo 5. Finalmente, as conclusões e perspectivas de trabalhos futuros são apresentadas no Capítulo 6.

Capítulo

2

Sistemas de Visão Computacional

A visão é certamente um dos mais poderosos sensores que o ser humano e diversos animais utilizam em sua locomoção. Porém, o custo computacional necessário no desenvolvimento de um sistema de visão artificial inviabilizou durante muitos anos a pesquisa nesta área. Nos últimos anos, com o grande aumento do poder computacional e o barateamento dos periféricos necessários ao desenvolvimento de um sistema de controle baseado em visão, diversos pesquisadores têm se voltado à elaboração de técnicas que auxiliem na compilação de um sistema de visão artificial, tanto aplicado à robótica como em outras áreas. Por exemplo, na classificação de padrões, reconhecimento de faces e objetos, controle de pragas em lavouras, etc.

Segundo Nalwa (1993) o termo **Visão Computacional** pode ser definido como:

> "A *Visão Computacional descreve automaticamente as estruturas e as propriedades de uma cena tridimensional do mundo a partir de uma ou mais imagens bidimensionais deste*".

Apesar da visão não ser necessária no desenvolvimento de vários tipos de robôs móveis, ela pode ser de extrema utilidade em diversas tarefas, principalmente quando for necessária uma maior resolução do mundo externo. No caso de sensores como o sonar e o laser, não é possível realizar a distinção de cores e texturas presentes no ambiente. Além disso, a visão é um sensor passivo, diferenciando dos demais sensores citados, que emitem energia.

Figura 2.1: Veículo robotizado - ALVINN (Pomerleau 1995)

2.1 Sistemas de Visão Aplicados ao Controle de Robôs Móveis

Diversos trabalhos têm sido produzidos, principalmente na última década, acoplando sistemas de visão computacional ao controle de robôs.

Em Pomerleau (1991, 1992, 1993a, 1993b, 1995) e Pomerleau et al. (1991) uma rede neural foi utilizada na implementação de um sistema de visão para controlar um veículo autonomamente *ALVINN*[1] (Figura 2.1). O controle do veículo é realizado com base nas imagens adquiridas por uma câmera situada no teto deste. Cada imagem (quadro) passa por um pré-processamento e em seguida é submetida à rede neural que desempenha o papel de controlar a trajetória do robô (automóvel) pela estrada, isto é, indicar ao robô qual a direção que este deve seguir evitando obstáculos. O pré-processamento realizado nas imagens tem como função reduzir a quantidade de dados que são enviados à rede neural, como as imagens já pré-processadas apresentadas pela Figura 2.2. O modelo de rede neural utilizado neste trabalho foi uma MLP (Capítulo 3. Seção 3.1.5) totalmente conectada, treinada por algoritmo de retropropagação (Haykin 2001) com 960 elementos na camada de entrada, 4 elementos na camada oculta e 30 elementos na camada de saída (Figura 2.3). Os 960 elementos de entrada são os valores dos pontos da imagem já pré-processada (30X32) e os 30 elementos de saída representam a direção instantânea em que o veículo deve ser direcionado no intuito de manter o veículo na estrada e de evitar colisões com obstáculos próximos. O treinamento da rede neural foi realizado de forma *on-line*, em que um motorista guia o veículo pela estrada enquanto a rede neural aprende o comportamento deste. Este sistema desenvolvido possibilitou que o veículo navegasse em rodovias pavimentadas com e sem marcações e em rodovias não pavimentadas a uma velocidade de até 55 milhas/h.

[1] Autonomous Land Vehicle In a Neural Network

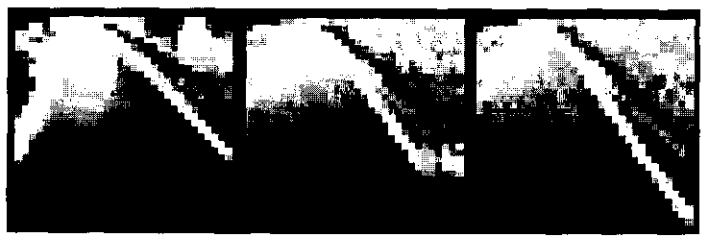


Figura 2.2: Imagem pré-processada (Pomerleau 1995)

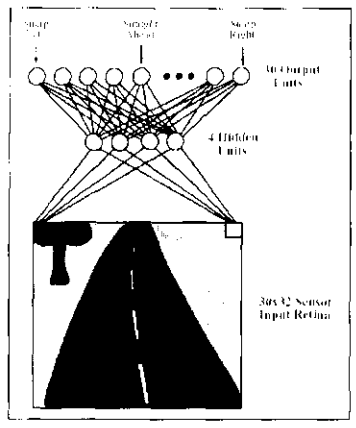


Figura 2.3: Arquitetura da rede neural - ALVINN (Pomerleau 1995)

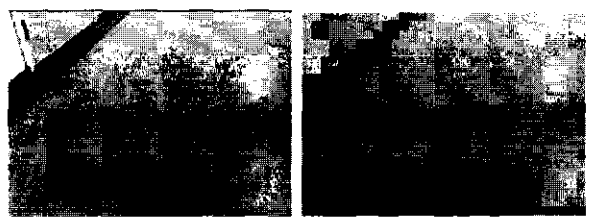


Figura 2.4: Imagem vire à direita (Jonsson et al. 1997)

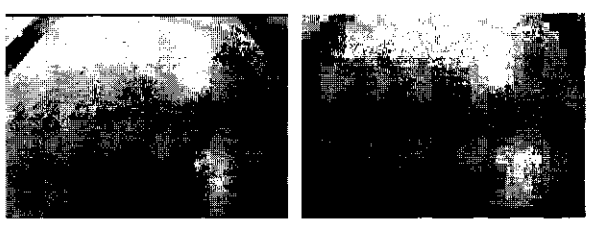


Figura 2.5: Imagem siga em frente (Jonsson et al. 1997)

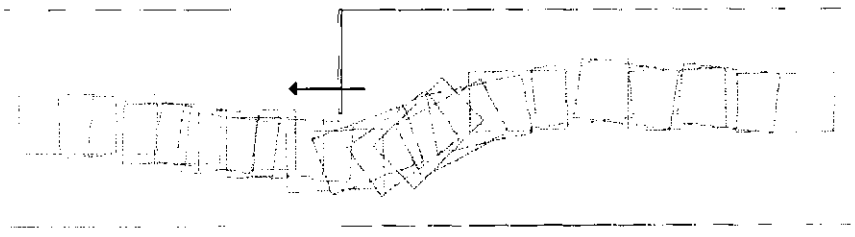


Figura 2.6: Robô desviando da porta (Jonsson et al. 1997)

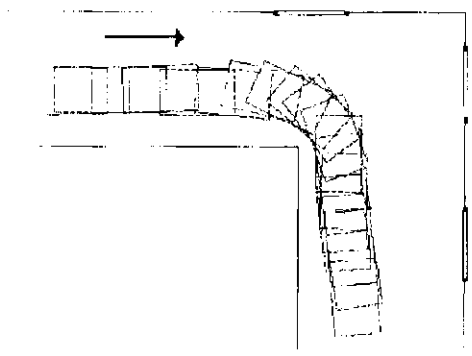


Figura 2.7: Robô realizando uma curva (Jonsson et al. 1997)

Um outro sistema de navegação de robôs baseado em visão foi proposto em Jonsson et al. (1997). O objetivo deste trabalho foi capacitar um robô a caminhar por um corredor desviando de obstáculos em ambientes internos (*indoor*). Para a realização das duas tarefas, uma rede neural do tipo MLP é treinada utilizando o algoritmo *Rprop* (Riedmiller & Braun 1992; Riedmiller 1994a; Riedmiller 1994b) (Seção 3.1.6), já que o tradicional algoritmo de retropropagação não apresentou bons resultados. As imagens são captadas à uma freqüência de 7 quadros/s por uma câmera CCD posicionada no topo do robô. Essas imagens são reduzidas à 32X23 pontos em 256 níveis de cinza, sem que nenhum processamento adicional seja realizado (Figuras 2.4 e 2.5). Diferente da forma de treinamento realizado em Pomerleau (1995), neste trabalho a rede neural foi treinada com uma base de imagens capturadas e rotuladas a partir do ambiente de trabalho do robô. O Rótulo de cada uma das imagens foi configurado como: vire a esquerda, vire a direita ou siga em frente.

Este trabalho apresentou bons resultados mesmo para ambientes desconhecidos para o robô, demonstrando a capacidade de generalização da rede MLP utilizada. As Figuras 2.6 e 2.7 mostram a capacidade de navegação do robô em um ambiente real. Ainda deve ser ressaltado que mesmo não sendo explicitamente programado para realizar curvas, o robô foi capaz de realizá-la sem problemas (Figura 2.7). É relatado no artigo que em apenas algumas ocasiões o robô não foi capaz de realizar a manobra no tempo correto, colidindo com o obstáculo.

Técnicas de redes neurais artificiais também foram utilizadas em Blank & Ross (1997). Neste trabalho, um módulo de visão foi incorporado a um sistema de controle de robos baseado em comportamento[2] *fuzzy*. O sistema de controle do robôs funciona da seguinte forma: **IF door-on-left THEN turn-left** onde **door-on-left** é representado por uma variável *fuzzy* que indica um valor entre

[2]Do inglês: *Behavior Based*

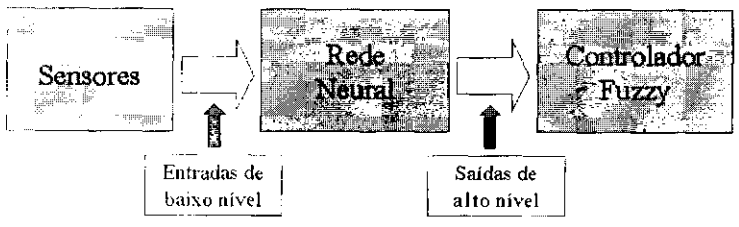


Figura 2.8: Sistema de aprendizagem baseado em redes neurais

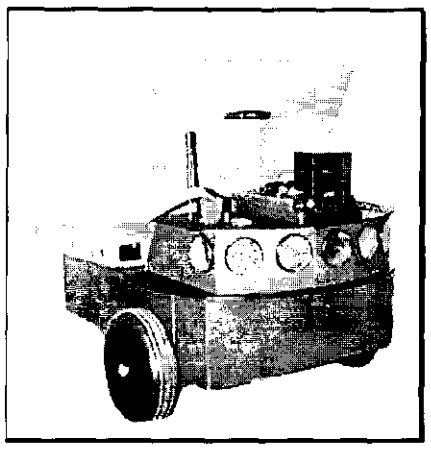


Figura 2.9: Robô Pioneer I com *laptop* e câmera *on-board* (Blank & Ross 1997)

0 e 1. O módulo de visão incorporado ao sistema de controle teve como função implementar um sistema de aprendizagem responsável pelo reconhecimento de imagens provenientes da câmera e indicar, a partir do processamento destas, qual o nível de ativação das variáveis *fuzzy* que estão incorporadas aos comportamentos. Assim, a rede neural que implementa o sistema de visão é responsável por mapear as entradas de baixo nível dos sensores (no caso a câmera CCD) em entradas de alto nível para o controlador (como no exemplo acima: *door-on-left*) como é indicado na Figura 2.8. Este sistema teve como objetivo capacitar o robô (Figura 2.9) a seguir uma bola sobre uma determinada superfície.

Para isso foi utilizada uma rede neural do tipo MLP onde as entradas da rede são as componentes RGB de cada um dos pontos da imagem pré-processada e as saídas da rede neural (4 elementos na camada de saída) indicam em qual quadrante a bola se encontra (Figura 2.10). No pré-processamento, a imagem foi reduzida ao tamanho de 14X48 pontos e as componentes RGB foram binarizadas. Após o processamento de cada quadro pelo sistema de visão, cada uma das saídas da rede neural, que representam os quadrantes da imagem, são submetidas aos comportamentos implementados no controlador. Os comportamentos foram implementados da seguinte maneira:

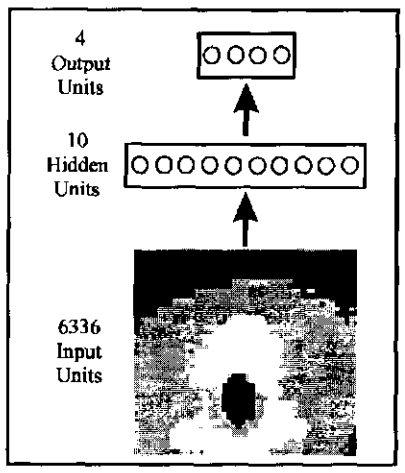


Figura 2.10: Arquitetura da rede neural do sistema de aprendizagem (Blank & Ross 1997)

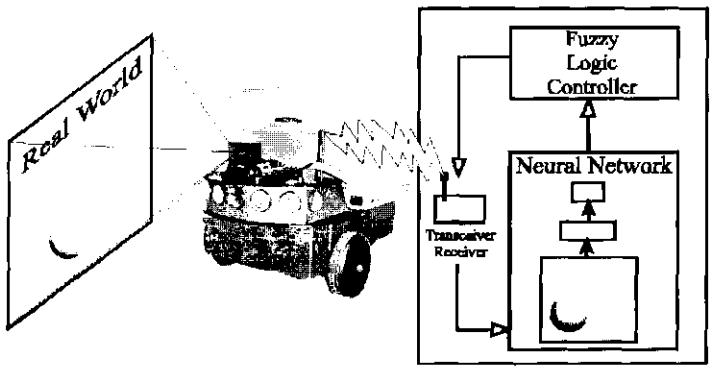


Figura 2.11: Esquema de funcionamento do controlador (Blank & Ross 1997)

IF (q1 OR q3) THEN Turn Left
IF (q2 OR q4) THEN Turn Right

onde **q1..q4** representam as saídas da rede neural e respectivamente os quadrantes da imagem. O sistema implementado (Figura 2.11) permitiu ao robô seguir a bola pelo chão, porém algumas falhas foram apresentadas. Por exemplo, o comportamento do robo quando a bola não se encontra em nenhum dos quadrantes não foi tratado pelo sistema.

Em Tyler & Czarnecki (1999), um controlador baseado em visão aplicado a um robó jogador de futebol foi desenvolvido. Um robô Khepera foi utilizado nos experimentos. A visão do robó consiste em um *array* 1D de 61 pontos em 256 níveis de cinza, em que 30 desses foram utilizados para a construção do módulo de visão. Uma rede neural artificial do tipo MLP foi utilizada em sua implementação. A rede neural, construída com uma topologia 30-12-6 (30 elementos de entrada, 12 na camada oculta e 6 na camada de saída) tem como função indicar

a parte da bola que está visível na imagem (esquerda, direita, ou nenhuma parte presente na imagem) e a parte do gol visível (esquerda, direita, ou não visível na imagem). Os elementos de entrada da rede neural foram obtidos a partir da normalização dos pontos obtidos do dispositivo de captura do robô Khepera. Para testar o módulo de visão, um controlador simples foi desenvolvido. Neste controlador foram implementados dois módulos (*seek-goal* e *seek-ball*), ambos os módulos também foram implementados via rede neural. Os resultados obtidos foram satisfatórios, demonstrando que mesmo um controlador baseado em um sistema de visão simples pode executar razoavelmente pequenas tarefas.

Em Waldherr et al. (2000) uma interface para o controle de um robô baseada em gestos foi proposta. Neste trabalho, uma câmera é responsável pela captura das imagens utilizadas pelo sistema na identificação dos gestos realizados por um humano que está posicionado em frente do robô. Com base nestes gestos, o robô obedece a comandos como: siga-me, pare, etc. Dois módulos compõe este sistema. O primeiro módulo, responsável por encontrar a pessoa no campo visual do robô. Quando nenhum humano está no campo visual do robô, este realiza uma busca em uma área central da imagem até detectar a face de uma pessoa. A identificação da face é realizada através de um modelo de cor que representa as principais cores que compõe a face. Uma outra parte do módulo de visão faz uma busca pela cor da camisa. Esta busca é realizada em uma região retangular posicionada abaixo da região da face e nenhuma informação prévia sobre a cor da camisa é necessária, pois o próprio sistema se encarrega de detecta-lá na região retangular citada acima. Vale ressaltar que para o início do processo, a pessoa deve estar posicionada no centro do campo visual do robô. O *tracking* da pessoa é realizada pela busca das duas cores consideradas (face e camisa). Esta busca é realizada em um espaço determinado da imagem, relativamente próximos a posição no qual a face e a camisa foram anteriormente capturadas, como mostrado na Figura 2.12, onde uma região de busca está posicionada próxima a face e a outra próxima a camisa.

O segundo módulo implementado neste trabalho tem por finalidade a tarefa de reconhecimento dos gestos realizados pelo humano. Neste módulo duas abordagens foram utilizadas: uma baseada em correlação e uma baseada em redes neurais. Ambas foram combinadas com o algoritmo de Virtebi, utilizado no processo de reconhecimento de gestos dinâmicos, como por exemplo, o braço em movimento.

Em Medeiros & Romero (2002), uma rede neural foi MLP utilizada no desenvolvimento de um sistema que capacitou um robô do tipo Pioneer I a percorrer

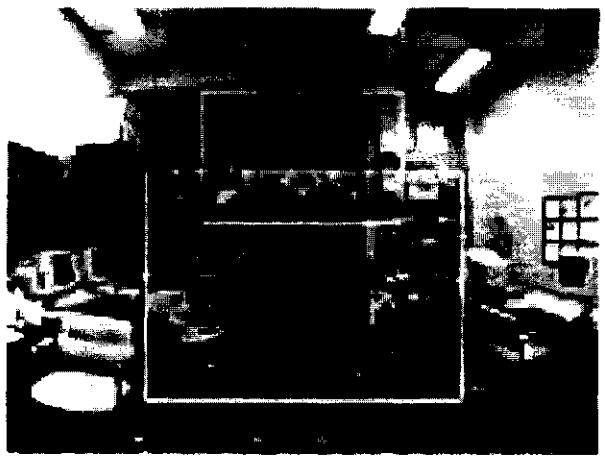


Figura 2.12: Região de busca pela face e camisa (Waldherr et al. 2000)

sobre um caminho delimitado por duas faixas laterais brancas. As faixas são capturadas por uma câmera posicionada à frente do robô. Cada quadro capturado passa por um pré-processamento, que consiste na redução e binarização (conversão da imagem em branco e preto) da imagem. Em seguida esta imagem é submetida a uma RNA com 7 elementos na camada de saída. As saídas da RNA são interpoladas por uma gaussiana, em que o pico da gaussiana indica qual a direção correta para o robô. A incorporação da gaussiana nas saídas da RNA teve como objetivo suavizar a movimentação do robô dentro das faixas. Este trabalho apresentou bons resultados, entretanto, a resposta do sistema ainda compromete o processamento em tempo real, tendo em vista que cada ciclo do sistema leva em torno de 319 milissegundos na plataforma em que o sistema foi implementado.

Os trabalhos apresentados acima relatam bons resultados para algumas tarefas. Entretanto, para a realização de tarefas mais complexas em visão computacional, como apresentado em Waldherr et al. (2000), a utilização de cores pode ser essencial, pois esta pode prover mais informações que imagens utilizando apenas níveis de cinza.

2.2 Segmentação de Imagens

Um dos primeiros passos que devem ser realizados na análise de imagens, no reconhecimento de padrões e outros problemas de visão computacional, é a segmentação. A segmentação é um componente crucial e essencial no desenvolvimento desses processos, sendo considerada uma tarefa de difícil realização e responsável pela determinação da qualidade do resultado final do trabalho (Cheng et al. 2001).

A tarefa de segmentação de imagens pode ser vista como um processo de

decisão, no qual cada ponto da imagem é classificado como pertencente ou não a um determinado objeto da cena que está sendo analisada (Jähne 1997). Segundo Nalwa (1993), o processo de segmentação consiste na divisão da imagem em fragmentos (segmentos), onde as regiões semelhantes são homogeneizadas a fim de separar (destacar) os objetos presentes na imagem.

Este processo de decisão pode ser realizado de diversas maneiras, sendo as principais apresentadas a seguir.

2.2.1 Métodos de Segmentação

A segmentação de imagens monocromáticas é geralmente baseada na descontinuidade ou na homogeneidade dos níveis de cinza presentes nas imagens (Gonzalez & Woods 2000).

Na segmentação baseada em descontinuidade dos níveis, o particionamento é realizado quando ocorrem variações bruscas nos níveis de cinza. Nesta categoria de segmentação, destacam-se a detecção de pontos isolados, detecção de linhas e bordas dos objetos presentes nas imagens. Já a segmentação baseada em similaridades visa aglomerar as regiões que possuem características semelhantes. As principais abordagens presentes nesta classe são a limiarização, crescimento de regiões e a divisão e/ou fusão de regiões.

A segmentação de imagens coloridas é geralmente caracterizada como uma extensão dos métodos de segmentação de imagens monocromáticas acrescidos de diferentes representações de cores (espaços de cores) (Cheng et al. 2001).

Segundo Skarbek & Koschan (1994), os métodos de segmentação de imagens coloridas podem ser classificados em quatro classes distintas:

- segmentação baseada em ponto[3],
- segmentação baseada em região[4],
- segmentação baseada em bordas[5],
- segmentação baseada em modelos físicos[6].

Na segmentação baseada em ponto (*pixel*), destacam-se a limiarização (*Histogram Thresholding*) e a clusterização do espaço de cores. Na limiarização, geralmente é realizado um estudo do histograma da imagem a fim de se estabelecer

[3] *Pixel based segmentation*
[4] *Area based segmentation*
[5] *Edge based segmentation*
[6] *Physics based segmentation*

Figura 2.13: Limiar de separação

limiares de separação (como a linha pontilhada apresentada na Figura 2.13) responsáveis por dividir a imagem em segmentos distintos pelo nível de cinza ou pela cor. Esta técnica foi primeiramente descrita para níveis de cinza e posteriormente generalizada para imagens coloridas. No caso de imagens coloridas, são estabelecidos limiares a cada um dos eixos que representam o espaço de cor utilizado. Com isso, determina-se uma região de classificação como pode ser observado na Figura 2.11. Nesta Figura, os pontos da imagens são classificados como pertencentes ou não a esta região (ou regiões). Um dos principais problemas desta abordagem é que nem sempre é possível realizar a separação de cores por intermédio de paralelepípedos como apresentado pela Figura 2.14.

A clusterização objetiva a criação de classes de cores distintas de tal forma que os pontos da imagem sejam classificados de acordo com os clusters formados. Deve ser observado que, nesta forma de segmentação, nenhuma informação adicional como nível de entropia, etc., é utilizada, ficando a segmentação restrita à análise do espaço de cores, geralmente utilizando-se o histograma da gerado. Tanto na limiarização quanto na clusterização de cores, o processo de segmentação fica restrito a um sistema de classificação de cores.

Os métodos de segmentação baseados em regiões incluem técnicas de crescimento, divisão e agrupamento de regiões, e combinações dessas técnicas com o objetivo de agrupar os pontos da imagem em regiões homogêneas (segmentos).

O crescimento de regiões caracteriza-se pela expansão (agrupamento) de pontos em determinadas regiões da imagem partindo-se de pontos denominados sementes. A partir da semente, os pontos vizinhos são agrupados a esta região considerando-se propriedades como a cor, textura, nível de entropia, etc.

Na divisão de regiões, ao contrário da técnica de crescimento de regiões, o processo de segmentação parte da imagem inteira e realiza-se divisões das regiões que são consideradas não homogêneas até que não existam mais regiões heterogêneas (com base em algum critério de análise).

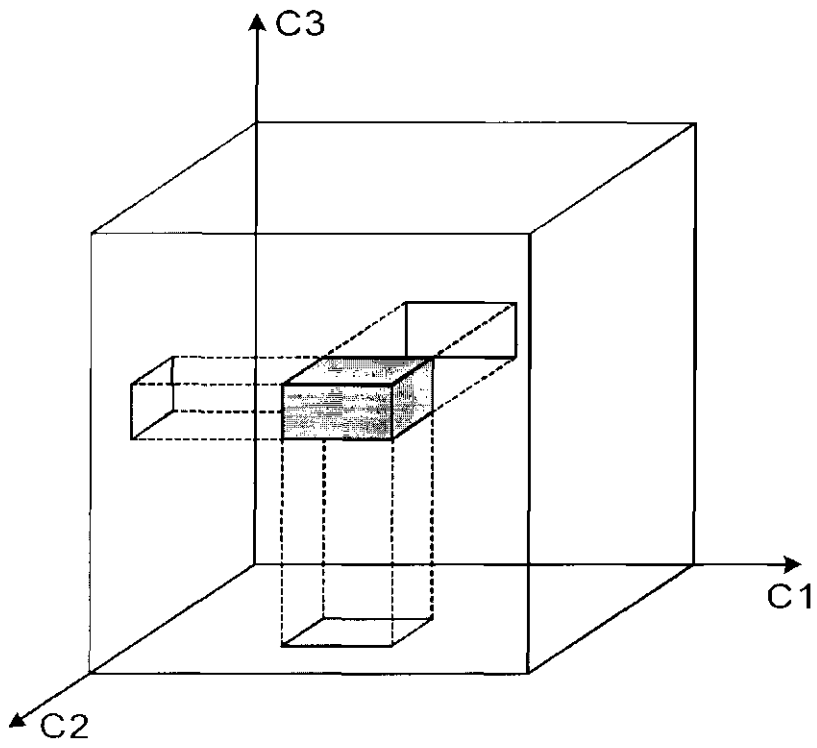


Figura 2.14: Região de separação por limiares

O agrupamento de regiões, geralmente combinado com a técnica de crescimento ou a de divisão de regiões, consiste na união de regiões vizinhas similares, com o objetivo de criar regiões homogêneas tão grande quanto possível. A técnica baseada em regiões é largamente utilizada em segmentação de imagens coloridas, pois, considera tanto as informações provenientes da cor do ponto quanto as informações espaciais envolvidas (Cheng et al. 2001).

A segmentação de imagens também pode ser estabelecida pela detecção de bordas entre as diversas regiões que a compõe. A detecção de bordas foi primeiramente utilizada na segmentação de imagens monocromáticas utilizando-se operadores como o Laplaciano (Gonzalez & Woods 2000). Para a deteção de bordas em imagens monocromáticas, apenas a diferença de intensidade luminosa é utilizada. Entretanto, em se tratando de imagens coloridas, mais informações estão disponíveis em cada ponto da imagem, permitindo assim, a detecção de bordas impossíveis de serem detectadas em imagens monocromáticas. Por exemplo, uma borda entre duas regiões com mesma intensidade luminosa mas com cores diferentes pode ser detectada quando o processamento considera a informação proveniente da cor (Cheng et al. 2001).

Uma outra técnica descrita na literatura sobre segmentação de imagens é baseada na utilização de modelos físicos. Esta técnica visa empregar modelos

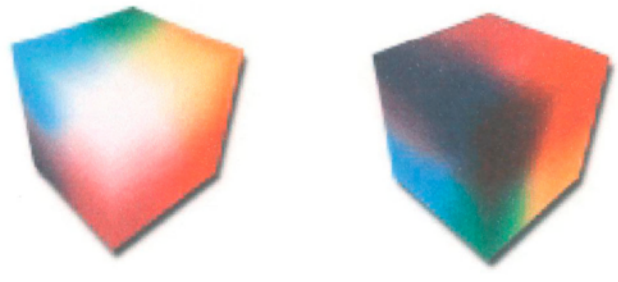


Figura 2.15: Duas visões do espaço de cores RGB

físicos que descrevem as propriedades de interação da luz com o material da superficie dos objetos presentes nas imagens. Isto possibilita o desenvolvimento de algoritmos de segmentação que solucionem problemas causados pelos reflexos luminosos, sombras, etc. (Lucchese & Mitra 2001).

Contudo, segundo Cheng et al. (2001), esses algoritmos baseados em modelos físicos são eficientes apenas para a segmentação de imagens cujos objetos presentes possuem suas propriedades de reflexão conhecidas e de fácil modelagem, limitando, desta forma, a utilização de algoritmos de segmentação que utilizem essas técnicas.

Além da técnica de segmentação escolhida, um outro fator muito importante está relacionado em como as cores estão representadas (espaço de cores) (Cheng et al. 2001). Abaixo estão descritos alguns dos principais espaços de cores encontrados na literatura.

2.2.2 Espaço de Cores

A percepção das cores pelos humanos é resultado da combinação de três estímulos de cores básicas: vermelho (*Red*), verde (*Green*) e azul (*Blue*). A partir do espaço de cores RGB, iniciais dos três estímulos, diversos outros espaços (representações de cores) podem ser derivados pelo uso de transformações lineares e não-lineares. A revisão bibliográfica apresentada por Cheng et al. (2001) mostra que na literatura são encontrados diversos espaços de cores utilizados em segmentação de imagens, tais como RGB, HSI, YUV, L*u*v*, Nrgb, dentre outros. Porém, nenhum deles é consagradamente o melhor, sendo sua escolha muito dependente da aplicação e do método de segmentação escolhido.

O RGB não é considerado um bom espaço de cores para realização do processo de segmentação, devido à alta correlação existente entre os componentes R, G e B, o que torna difícil a criação de limiares que permitam a separação das cores nessa forma de representação (o cubo RGB é apresentado na Figura 2.15). Essa

alta correlação é facilmente observada quando ocorrem variações da intensidade luminosa, acarretando na mudança de valores sobre os três componentes (RGB) do espaço de cor, o que dificulta a realização do processo de segmentação. Os atributos que geralmente são utilizados para realizar a distinção de cores são: cor (*hue*), brilho e saturação. Esses atributos podem ser obtidos a partir dos elementos R, G e B.

Transformações Lineares

Os espaços de cores obtidos a partir de transformações lineares (YIQ, YUV, dentre outros) possuem geralmente um custo computacional reduzido, podendo assim ser utilizado em processamento de imagens coloridas em tempo real. Porém, a correlação entre os componentes ainda permanece, embora não tão alta quanto o espaço de cores RGB.

A representação YIQ é utilizada na transmissão de sinais de televisão colorida, sua transformação é dada por:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.1}$$

onde R, G e B são valores entre 0 e 1. Nesta transformação, a componente Y do sistema representa a luminância e a informação sobre a cor é representada pelas componentes I e Q. Esta transformação para o espaço YIQ é realizada para garantir a compatibilidade com o sistema monocromático de televisão, pois Y fornece todas as informações necessárias para este tipo de televisor (Gonzalez & Woods 2000).

O espaço de cores YUV também é utilizado na transmissão de sinais de televisão colorida (sistema de televisão europeu), sua transformação é dada por:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & 0.515 & 0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.2}$$

onde R, G e B são valores entre 0 e 1.

Transformações Não-Lineares

Quando há a necessidade de decompor cada ponto da imagem nos atributos como brilho, saturação e cor, existe a necessidade de realizar uma transformação não-linear sobre os componentes R, G e B. Os espaços de cores obtidos via transformações não-lineares (HSI, $L^*u^*v^*$, etc.) são geralmente mais indicados no

processo de segmentação de imagens, pois uma vez obtidos os componentes como cor, saturação, pode se realizar a separação de cores indicando apenas limiares para esses componentes.

Além disso, os sistemas de segmentação de imagens que utilizam esses espaços de cores são mais robustos a variações de luminosidade presentes nos ambientes reais, como as sombras encontradas nas imagens etc. Entretanto, apresentam instabilidade quando aplicados a pontos com baixa saturação. Um outro fator agravante existente é o tempo computacional gasto nas transformações, o que ocasionalmente pode inviabilizar o uso desses espaços, principalmente em aplicações realizadas em tempo real.

Uma das representações mais utilizada no processo de segmentação de imagens é o HSI. Contudo, sua utilização é bastante restrita em sistemas de software em tempo real devido ao alto custo computacional envolvido na transformações. A transformação RGB para HSI é dada por:

$$I = \frac{1}{3}(R + G + B) \tag{2.3}$$

$$S = 1 - \frac{3}{(R + G + B)}[min(R, G, B)] \tag{2.4}$$

$$H = cos^{-1}\left(\frac{\frac{1}{2}[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{\frac{1}{2}}}\right) \tag{2.5}$$

onde R, G e B são valores entre 0 e 1, I é a luminância (níveis de cinza), S a saturação (grau de diluição de uma cor pela luz branca) e H representa a matiz (cor propriamente dita).

Espaços Híbridos

Uma outra forma de se representar o espaço de cor para o processo de segmentação de imagens é através da criação de espaços híbridos. Esses espaços são compostos por um conjunto de elementos (como o R, G e B), obtidos a partir de vários espaços de cores clássicos. Com isso, busca-se obter a menor correlação possível entre os componentes selecionados e o menor tempo computacional necessário à obtenção dos elementos do espaço de cor. Assim é possível extrair as principais vantagens de cada espaço evitando suas deficiências.

Em Vandenbroucke et al. (2003), um espaço de cor híbrido foi adotado para a análise de imagens de um jogo de futebol real. Os testes realizados com o espaço híbrido obtiveram uma separação bem mais precisa que as utilizando outros espaços de cores clássicos como o RGB e o L*a*b*. Ainda é ressaltado no artigo que

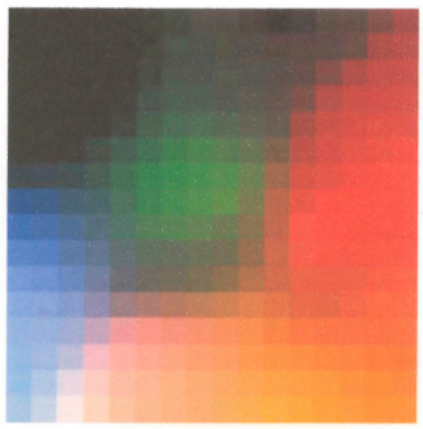


Figura 2.16: Cores mapeadas em uma rede neural SOM 2D (Ong et al. 2002)

nenhum espaço de cor clássico provê resultados satisfatórios para todos os tipos de imagens. Assim a construção de espaços híbridos pode ser uma boa alternativa para alguns domínios onde são encontradas deficiências com os espaços clássicos.

2.2.3 RNAs para Segmentação de Imagens

Para o desenvolvimento deste projeto, foi realizada uma busca de trabalhos que utilizaram redes neurais artificiais na implementação de sistemas de segmentação de imagens coloridas, ou mesmo trabalhos que utilizaram as RNAs no desenvolvimento de técnicas de classificação e/ou redução de cores. Essa busca teve por objetivo encontrar uma metodologia que pudesse ser ajustada ao sistema proposto por este projeto de mestrado.

Diversos trabalhos têm utilizado redes neurais artificiais no processamento de imagens (Egmont-Petersen et al. 2002). Para o processo de segmentação de imagens coloridas, as redes neurais têm sido aplicadas diretamente no processo de segmentação ou como ferramenta auxiliar.

A seguir, são apresentados os principais trabalhos estudados.

Em Ong et al. (2002) um método de segmentação de imagens coloridas, baseado em classificação de cores, foi proposto utilizando dois estágios de redes neurais auto-organizáveis (SOM - *Self-Organizing Maps*). No primeiro estágio, uma rede neural 2D foi utilizada para fazer um mapeamento das cores presentes nas imagens de treinamento formando *clusters* dessas cores. Na Figura 2.16 pode ser visto que a rede apresenta a formação de clusters de cores bem definidos. Já no segundo estágio foi utilizada uma rede 1D que tem como finalidade extrair elementos que representem os clusters formados na primeira rede. Assim, de uma forma não supervisionada, a quantidade de cores significativas nas imagens são obtidas. O tamanho da rede do segundo estágio não é fixo, podendo sofre

modificações, como agrupamentos e divisões, a fim de aglomerar cores muito próximas em um único cluster ou mesmo dividir um cluster em mais cores. A distância euclideana foi utilizada para medir a distância entre duas cores. O espaço de cores utilizado no processo de segmentação foi o L*u*v*, obtidos a partir das componentes RBG das imagens consideradas. A utilização de dois estágios de redes neurais foi justificada, pois experimentos realizados mostraram que é difícil obter a extração de cores diretamente em uma rede 1D e a utilização direta de uma única rede 2D não seria adequada para o controle do número de clusters de cores formados. Com a utilização dos dois estágios ambos os problemas citados foram resolvidos. Apesar deste trabalho apresentar bons resultados no processo de segmentação, ele não foi desenvolvido com o propósito de segmentar imagens em tempo real, necessitando assim de modificações para adaptá-lo à esta finalidade.

Em Simões & Costa (2000a, 2000b), um sistema de segmentação por classificação de cores foi desenvolvido. Uma rede neural do tipo MLP treinada via algoritmo de retropropagação foi utilizada no processo de segmentação. A rede neural recebe como entrada as três componentes RGB de cada ponto da imagem e as classifica em uma das n cores estabelecidas no processo de segmentação. Para o domínio do futebol de robôs, 7 classes cores foram estabelecidas: laranja, rosa, verde, amarelo, branco, cinza e preto, podendo também ser definidas outras classes de cores conforme a necessidade do problema em estudo.

Os resultados obtidos demonstraram que a classificação de cores por redes neurais é uma boa alternativa para o domínio de futebol de robôs, pois diferentemente da técnica de liniarização para a separação de cores, em que paralelepípedos de classificação são definidos, a rede MLP separa o espaço de cores por hiperplanos, formando-se assim regiões que se adequam melhor aos limites reais encontrados no cubo RGB. Além disso, ao contrário dos outros métodos de segmentação em que o espaço de cor RBG não apresenta bons resultados, nos testes comparativos apresentados em Simões & Costa (2001), o espaço de cor RGB obteve resultados melhores que os espaços de cores como o HSV e o YUV como mostra a Figura 2.18, onde são apresentados os resultados do processo de segmentação para a imagem da Figura 2.17.

Entretanto esse método apresentado foi utilizado na segmentação de imagens obtidas por uma câmera situada no teto do campo de futebol de robôs, diferentemente do presente trabalho, no qual a câmera está acoplada ao robô, tornando as imagens adquiridas mais expostas a variações de luminosidade e sombras presentes no ambiente.

Figura 2.17: Imagem do campo de futebol de robôs (Simões & Costa 2001)

(a) (b) (c)

Figura 2.18: Imagens resultantes de segmentação da Figura 2.17 utilizando os espaços de cores: a)RGB; b)HSV e c)YUV (Simões & Costa 2001)

Em Papamarkos (1999), um sistema para redução de cores em imagens foi proposto. Uma rede neural do tipo SOM foi utilizada para agrupar as classes de cores mais significativas da imagem. Ao contrário do trabalho proposto por Ong et al. (2002), este utiliza apenas uma rede SOM. No modelo apresentado, além das três componentes do espaço de cor, um maior conjunto de características podem ser utilizados na formação das entradas da rede neural. Por exemplo, algumas características da vizinhança do ponto, como média da vizinhança, entropia, etc. podem ser utilizadas. Em Papamarkos et al. (2002), foi proposto uma nova técnica adaptativa de redução de cores. Um sistema de clusterização adaptativo baseado em uma estrutura de árvore foi implementado onde cada nó da árvore possui um NNC (*Neural Network Classifier*) semelhante ao modelo utilizado em Papamarkos (1999). Contudo, uma rede do tipo PCA (*Principal Component Analysis*) (Haykin 2001) foi acrescentada ao modelo com o objetivo de diminuir a correlação entre as n características (entradas) que são apresentadas a rede SOM (Figura 2.19). Em cada nível da árvore um conjunto adicional de características são utilizadas de tal forma que novas classes de cores possam ser vistas (separadas). A técnica de redução de cores em cada nó da árvore é aplicada aos pontos da imagem inicial, contudo, diferentes característcias obtidas da imagem são utilizadas. Ao final, os componentes extraídos da imagem são

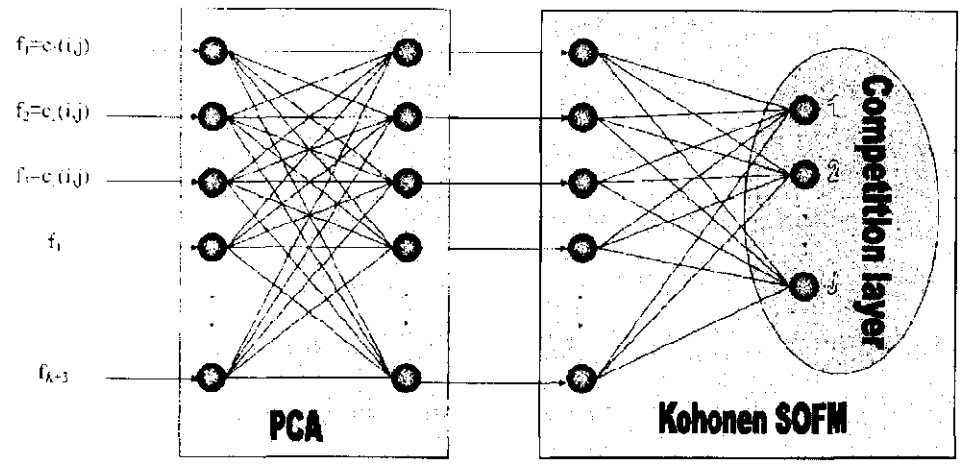


Figura 2.19: Modelo do sistema (PCA | SOM)

agrupados por uma técnica de *merging*.

Ambos os trabalhos propostos por Papamarkos apresentam bons resultados no processo de redução de cores utilizando o espaço de cor RGB. Entretanto, as metodologias apresentadass não foram designadas para atuarem em tempo real, pois o sistema (RNAs) deve ser treinado para cada imagem que terá suas cores reduzidas, limitando assim, a utilização destas metodologias para o propósito deste trabalho de mestrado.

Um outro trabalho está sendo desenvolvido pelo pesquisador Papamarkos, no qual uma rede neural do tipo GNG (*Growing Neural Gas* (Fritzke 1992; Fritzke 1995)) e uma rede derivada da GNG denominada SGONG (*Self-Growing Organized Neural Gas*) estão sendo utilizadas no processo de redução de cores em imagens. Os primeiros resultados apresentados em (http:// ipml.ee.duth.gr/ papamark/SGONG/start.html) demonstram que essa metodologia é muito promissora quando comparada com os resultados apresentados pelas redes do tipo SOM. A Figura 2.21 apresenta o resultado do processo de redução de cores para a imagem apresentada na Figura 2.20, utilizando três tipos de RNAs ((a) SOM, (b) GNG e (c) SGONG), onde é possível observar que as redes GNG e SGONG apresentaram um resultado superior ao obtido pela rede SOM.

Nestes trabalhos a RNA não foi utilizada diretamente no processo de segmentação, mas sim na tarefa de redução de cores das imagens. Entretanto, com base nos estudos realizados, os sistemas apresentados podem ser implementados como um sistema de segmentação baseado em classificação de cores.

Com base no levantamento bibliográfico realizado e em alguns experimentos empíricos, foi decido adotar um modelo de rede neural MLP para o desenvolvimento do sistema de segmentação desenvolvido. Dentre os principais motivos

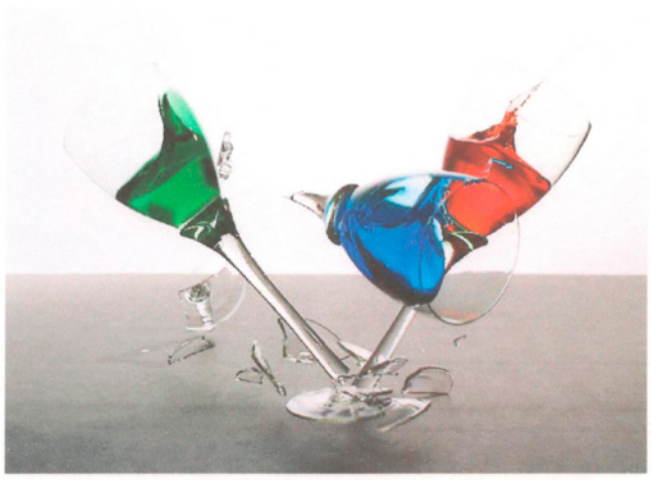


Figura 2.20: Imagem original [http:// ipml.ee.duth.gr/ papamark/ SGONG/ start.html]

para esta escolha podem ser destacados:

- A rede MLP se mostrou mais adequada ao processo de classificação de cores, principalmente no desenvolvimento de um sistema de binarização das imagens;
- Por se tratar de um sistema onde a base de dados pode ser facilmente rotulada (classificação das cores por um humano), o sistema de aprendizagem supervisionado se mostrou mais adequado.
- A rede MLP se mostrou adequada para realização do processo de classificação de cores utilizando-se o espaço de cor RGB;
- A rede MLP é reconhecida por ser capaz de lidar com dados ruidosos, o que é uma grande vantagem quando se trabalha com imagens;
- A rede neural MLP já está implementada em FPGAs pelo grupo de computação reconfigurável do ICMC, o que pode facilitar a implementação deste sistema em um hardware embarcado.

O sistama de segmentação proposto no presente trabalho será apresentado no Capítulo 4.

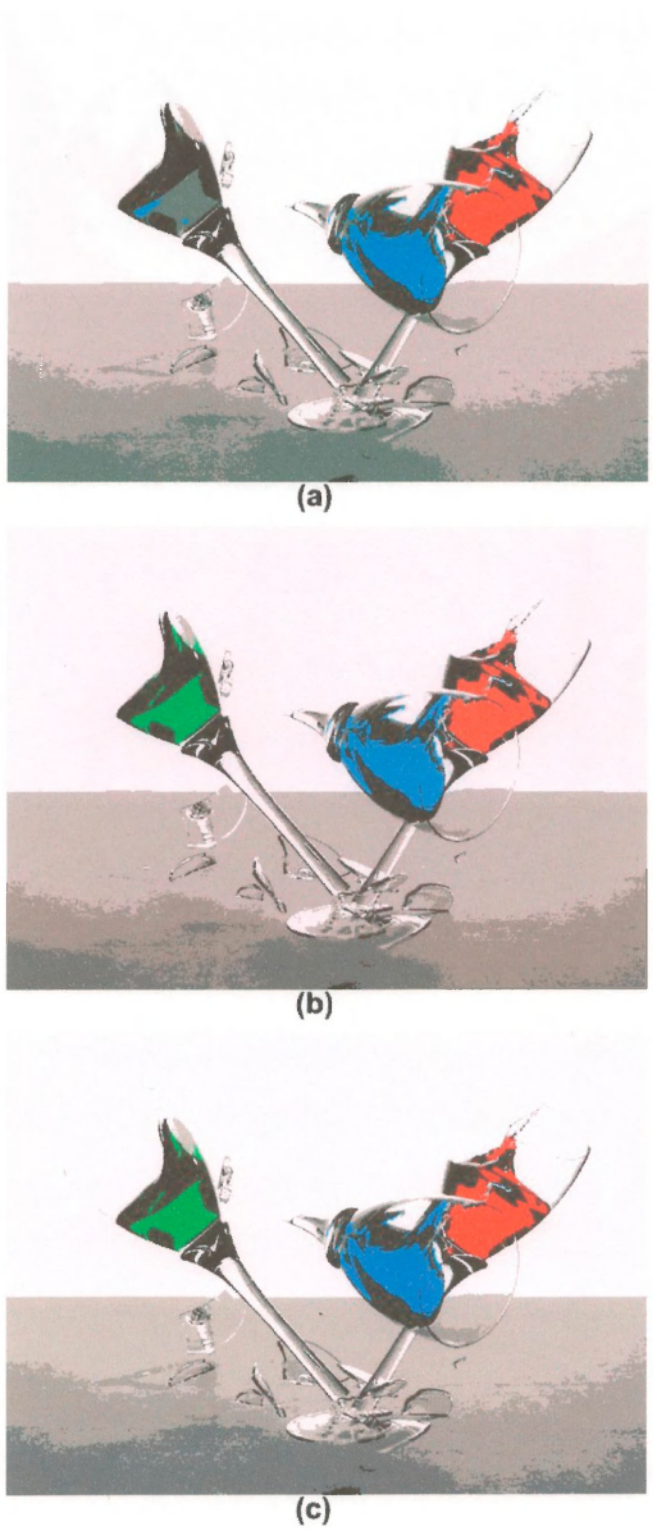


Figura 2.21: Imagens com cores reduzidas. (a) Rede SOM; (b) Rede GNG e (c) Rede SGONG [http:// ipml.ee.duth.gr/ papamark/ SGONG/ start.html]

Capítulo

3

Técnicas de Computação Bioinspirada

Desenvolver modelos computacionais dotados de inteligência ainda é um dos grandes desafios a serem decifrados pelos pesquisadores em Ciência da Computação. E na busca pela elaboração de modelos computacionais para resolver tarefas de difícil solução para a computação tradicional, como problemas de otimização e reconhecimento de padrões, os pesquisadores têm encontrado na natureza a inspiração para construção desses modelos (de Carvalho et al. 2004).

A computação bioinspirada visa criar modelos computacionais inspirados na organização e no funcionamento de sistemas biológicos, como: a colônia de formigas, sistema nervoso humano, evolução natural das espécies, sistema imunológico, etc. Algumas dessas técnicas, como as redes neurais artificiais e os algoritmos evolutivos tem sido, ao longo das últimas décadas, aplicadas na resolução de problemas reais e tem se mostrado eficientes a problemas de difícil solução para a computação tradicional (de Pádua Braga et al. 2000).

Nas seções abaixo são brevemente apresentadas três áreas fortemente inspiradas na natureza: as Redes Neurais Artificiais, os Algoritmos Evolutivos e a técnica de Campos Potenciais que foram utilizadas no desenvolvimento deste trabalho de mestrado.

3.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs), também referenciadas na literatura como sistema conexionista, sistema paralelo distribuído (PDP), redes adaptativas, são modelos computacionais inspirados na estrutura e no funcionamento do cérebro.

Segundo Kandel et al. (1997), a principal semelhança entre as RNAs e os circuitos formados no cérebro está no extenso processamento paralelo apresentado por ambos e uma outra semelhança é que as operações nos dois circuitos não

são dependentes de qualquer elemento (neurônio) isolado, mas sim uma função do conjunto dos elementos. Assim se um pequeno número de elementos forem danificados, a rede ainda poderia responder com uma mínima degradação de desempenho.

3.1.1 Histórico

Primeiramente identificado pelo cientista Santiago Ramón y Cajal em 1894, o neurônio constitui a base do sistema nervoso (Kandel et al. 1997).

Além da descoberta das estruturas anatômicas da célula neural, Ramón y Cajal também estabeleceu o princípio da polarização dinâmica. Este princípio propõe que os sinais elétricos em um neurônio fluem em uma determinada direção, percorrendo dos locais receptivos (dendritos e soma) até as terminações do axônio (sinapses) (Kandel et al. 1997).

Com base no que se conhecia até então, Warren McCulloch e Walter Pitts em 1943 propuseram o primeiro neurônio artificial (matemático) que pode ser visto como uma simplificação do que já havia sido descoberto a respeito do neurônio biológico (de Pádua Braga et al. 2000). Este neurônio ficou conhecido como nó MCP. Porém o neurônio desenvolvido por McCulloch e Pitts ficou restrito a descrição de um modelo artificial que representasse, computacionalmente, o neurônio biológico e não o processo de aprendizagem que ocorria no cérebro.

O primeiro trabalho sobre aprendizado foi descrito pelo neuropsicólogo Donald Hebb em 1949 em seu livro *The Organization of Behavior*. Hebb propôs que o aprendizado em neurônios biológicos é baseado no reforço das ligações sinápticas, de tal forma que se dois neurônios são ativados sincronamente, a ligação entre eles (sinapse) é reforçada por algum processo de crescimento ou metabólico (Haykin 2001).

Rosenblatt em 1958, apresentou um modelo de rede denominada *perceptron*, que consistia de uma rede formada por nós MCP acrescido de sinapses ajustáveis por uma regra de aprendizado. Em 1962, Rosenblatt demonstrou o teorema de convergência do *perceptron*, mostrando que uma rede *perceptron* sempre converge para padrões linearmente separáveis (de Pádua Braga et al. 2000).

Quase em paralelo com Rosenblatt, em 1960, Widrow e Hoff apresentaram uma regra de aprendizagem baseada no gradiente descendente δ (Equação 3.1) para a minimização do erro (Equação 3.2). Esta regra ficou conhecida como regra *delta*, sendo esta utilizada na formulação do modelo neural linear ADALINE (*Adaptative Linear Element*). Porteriormente, Widrow e seus alunos, em 1962, desenvolveram a rede MADALINE (Múltiplas Adaline) que é um modelo multidimensional da ADALINE (Haykin 2001).

$$\delta = - \frac{\partial E}{\partial w} \tag{3.1}$$

$$E = \frac{1}{2} \sum_{i} (desejado - obtido)^2 \tag{3.2}$$

Porém, em 1969 foi apresentado por Minsky e Pappert algumas limitações do *perceptron*, mostrando que este não era capaz de resolver problemas simples como a operação lógica XOR. Além disso, em seu livro publicado em 1969, Minsky e Pappert afirmaram que mesmo o *perceptron* multi-camadas, possivelmente, não seria capaz de superar as limitações apresentadas pelo *perceptron* de uma única camada (Haykin 2001). Devido a essas afirmações, a pesquisa no campo das redes neurais ficou adormecida por toda a década de 70 e início da década de 80, restando apenas um pequeno número de pesquisadores contribuindo com a área. Entre eles podem ser destacados: Teuvo Kohonen (mapas auto-organizáveis), Igor Aleksander (redes neurais sem pesos), Kunihiko Fukushima (cognitron e neocognitron) e Steven Grossberg (sistemas auto-adaptativos) (de Pádua Braga et al. 2000).

No início da década de 80, a publicação do trabalho, desenvolvido pelo pesquisador John Hopfield, utilizando redes recorrentes para construção de memórias associativas acarretou em um ressurgimento do interesse pela pesquisa em redes neurais. Este modelo, conhecido como redes de Hopfield, se caracterizou por sua capacidade de associar modelos de entrada com modelos previamente armazenados na memória da rede (Haykin 2001).

Em 1982, o pesquisador Teuvo Kohonen apresentou o modelo de rede neural denominado Mapas Auto-Organizáveis (SOM - *Self-Organizing Maps*). Este modelo se diferenciou dos demais por ser inspirado na neurofisiologia do córtex cerebral (Kohonen 2001).

Porém, apenas em 1986 com o desenvolvimento do algoritmo de retropropagação (*backpropagation*) pelos pesquisadores Rumelhart, Hinton e Williams, a visão pessimista de Minsky e Pappert sobre o *perceptron* multi-camadas foi desfeita, reabrindo-se então uma nova era nas pesquisas em redes neurais artificiais.

3.1.2 O Neurônio

O Neurônio Biológico

O sistema nervoso é composto por duas classes distintas de células. A primeira classe, chamada de células da glia (ou glias), possui função de sustentação, isolamento, modulação da atividade neural e defesa (Machado 2000). A segunda

classe, conhecida como células neurais (ou neurônios), possui como função principal receber, processar e transmitir informação (Machado 2000).

O neurônio, assim como todas as células do corpo, é dividido em três partes principais: o núcleo, o citoplasma e a membrana plasmática. A membrana plasmática atua como uma barreira separando o meio intracelular do meio extracelular. Ela é formada por uma camada bilipídica atravessada por proteínas que formam os canais iônicos. Os canais iônicos são responsáveis pelo transporte de íons entre o meio extracelular e o meio intracelular. Segundo Berne & Levy (1996), o transporte de íons pode ocorrer de duas maneiras: o transporte passivo e o transporte ativo. No transporte passivo (ou facilitado), o fluxo de íons é a favor do gradiente de concentração, agindo no sentido de equilibrar o potencial eletroquímico encontrado entre o citoplasma e o líquido extracelular. No transporte passivo não há consumo de energia metabólica. Já no transporte ativo, o fluxo de íons ocorre contra o grandiente de concentração e sua principal função está na manutenção do potencial de repouso da célula. Diferenciando do transporte passivo, o transporte ativo consome energia metabólica e, devido a essa dependência do metabolismo energético, este fluxo de íons pode ser inibido por substâncias que interfiram no metabolismo da célula.

As células que compõe o tecido neural possuem uma diversidade morfológica maior que qualquer outra célula do corpo. Apesar dos neurônios se diferenciarem entre si, eles compartilham algumas características que os diferenciam das demais células. Segundo Kandel et al. (1997), a célula neural é constituída de quatro regiões morfologicamente bem definidas, o corpo celular (ou soma), os dendritos, o axônio e suas terminações pré-sinápticas (Figura 3.1).

O corpo celular é a base da célula, nele está o núcleo responsável pelo armazenamento do código genético, além de diversas outras organelas citoplasmáticas. A partir do soma, geralmente são encontrados dois tipos de ramificações distintas, o axônio e os dendritos. Na maioria das células neurais, são encontrados vários dendritos que se ramificam em forma de ramos de árvore e sua principal função é na recepção de sinais provenientes de outras células neurais. O Axônio é um prolongamento único que se origina de uma região da célula denominada cone axônico. O diâmetro do axônio pode variar de $0{,}2\mu m$ até $20\mu m$ (algumas exceções são encontradas na natureza, como o axônio gigante da lula que pode atingir até $0{,}5mm$ de diâmetro). O axônio têm como função principal conduzir o sinal elétrico (informação) do corpo celular até as sinapses. O comprimento do axônio pode variar de comprimentos de $0{,}1mm$ até $2m$. A maioria dos axônios se ramificam com a finalidade de transmitir o sinal a diferentes células alvo (Kandel

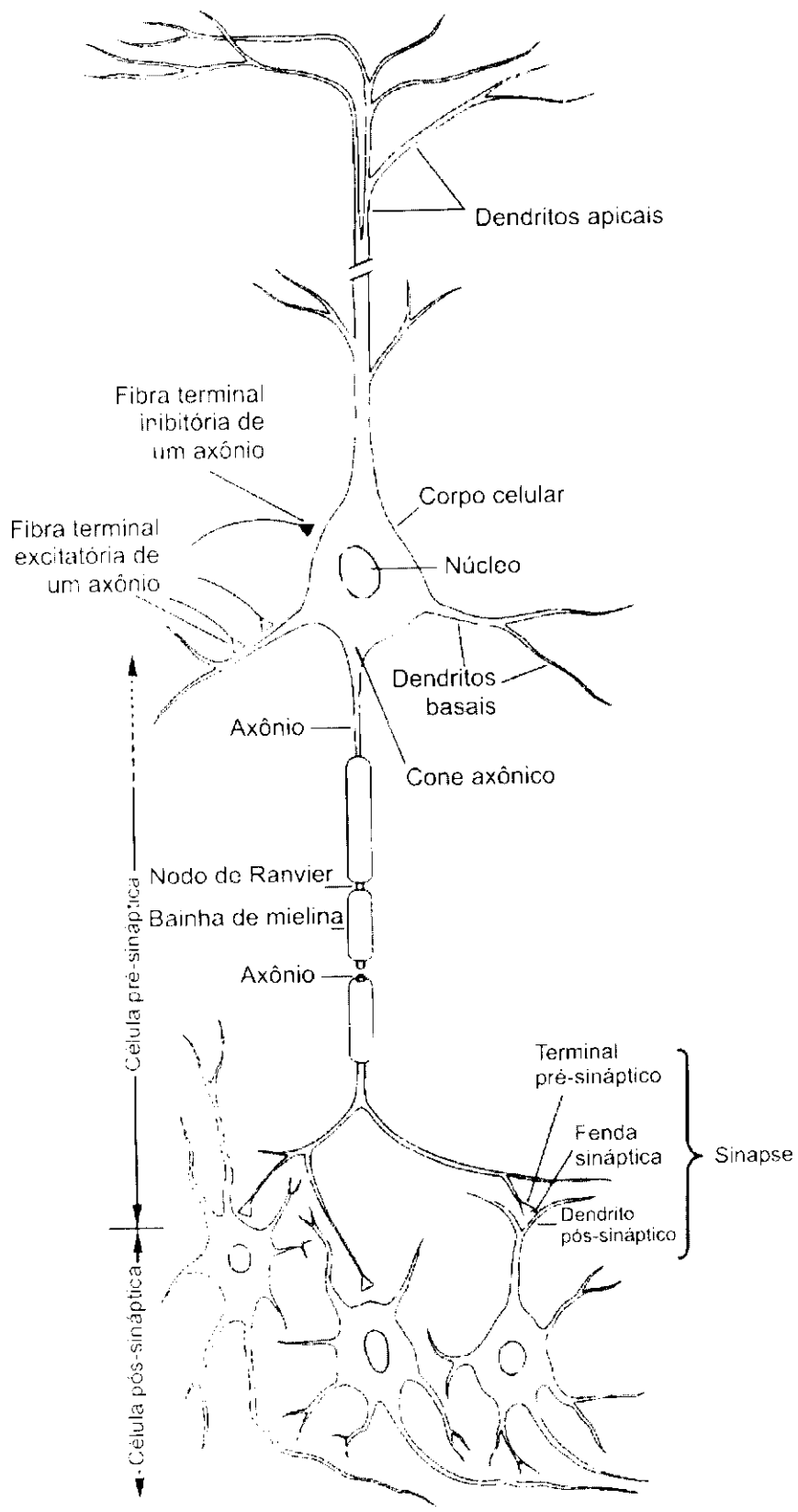


Figura 3.1: Morfologia do neurônio biológico (Kandel et al. 1997)

et al. 1997).

Os sinais elétricos que se propagam através do axônio são chamados de potencial de ação. Estes são sinais elétricos constantes com amplitudes na ordem de $100mV$ e com duração em torno de $1ms$, que se originam no cone axônico e se propagam através de todo comprimento do axônio, com velocidade entre 1 e 100 metros por segundo, sem interrupções e distorções. Os potenciais de ação são utilizados pelo cérebro para receber, analisar e transmitir informação (Kandel et al. 1997).

Nas extremidades do axônio, é encontrada uma região subdividida em segmentos finos que tem por finalidade realizar a comunicação com outros neurônios. Esse ponto de comunicação entre os neurônios é chamado sinapse. A célula que transmite o sinal é denominada célula pré-sináptica, a célula que recebe o sinal pela sinapse é chamada célula pós-sináptica e o espaço que as separam é nomeado fenda sináptica. Quando um potencial de ação atinge a região terminal do axônio (sinapse), ocorre a liberação de pacotes contendo um transmissor químico, também chamado de neurotransmissor. Existem diversos tipos de neurotransmissores, como exemplo, pode ser citado: acetilcolina, L-glutamato, encefalina, etc. Esses neurotransmissores funcionam como um sinal de saída do neurônio se difundindo pela fenda sináptica até atingir as moléculas receptoras localizadas na membrana das células pós-sinápticas. A fixação do neurotransmissor à molécula receptora pode fazer com que a célula pós-sináptica gere um potencial sináptico que pode ser inibitório ou excitatório (Kandel et al. 1997).

Diversos trabalhos têm sido realizados na tentativa de construir modelos matemáticos-computacionais dos neurônios biológicos. Pode-se dividir esses trabalhos em dois grandes grupos: no primeiro grupo estão os modelos com função de simular aspectos reais do neurônio biológico, como o modelo de Hodgkin-Huxley e a equação da membrana (Koch 1998). Além destes, outros modelos tentam reproduzir os aspectos morfológicos reais dos neurônios e das redes neurais, como os modelos apresentados em da Fontoura Costa et al. (1999) e Coelho & da Fontoura Costa (2002). No segundo grupo encontram-se os modelos que tentam apenas apresentar algumas propriedades semelhantes aos neurônios reais, porém, sem se preocupar com a plausibilidade biológica.

Este trabalho se restringirá ao detalhamento da segunda classe de modelos matemáticos do neurônio, os quais são utilizados na construção das redes neurais artificiais.

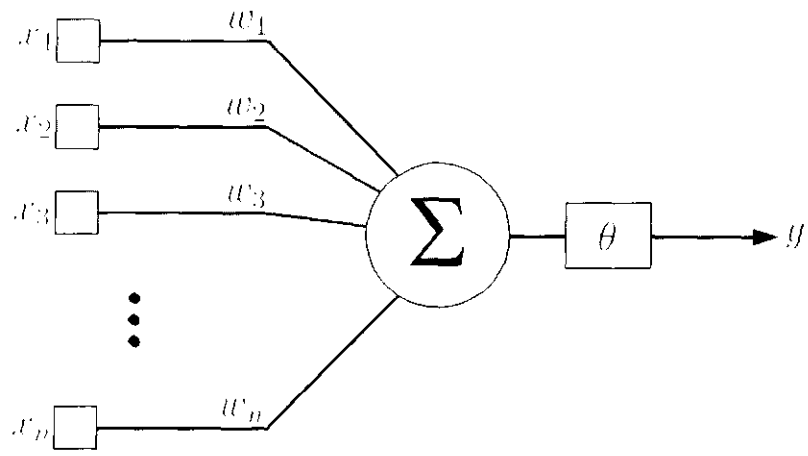


Figura 3.2: O neurônio de McCulloch e Pitts (Nó MCP)

O Neurônio Artificial

Como relatado anteriormente, o primeiro modelo matemático do neurônio foi desenvolvido em 1943 pelos pesquisadores Warren McCulloch e Walter Pitts. Este neurônio, conhecido como nó MCP (Figura 3.2), é composto por diversas entradas (dendritos) ponderadas por pesos (simulando o comportamento das sinapses), um somador (corpo celular) e uma saída (axônio). Na Figura 3.2, θ representa o limiar de ativação do neurônio. Matematicamente, a saída do neurônio MCP se torna ativa quando:

$$\sum_{i=1}^{n}(x_i w_i) \geq \theta \tag{3.3}$$

onde n representa o número de entradas (dendritos), x as entradas, w os pesos associados as sinapses (quando é w positivo a sinapse é excitatória, quando negativo é inibitória) e θ representa o limiar de ativação, isto é, quando o somatório atingir o limiar θ, o neurônio se torna ativo. Matematicamente, a Equação 3.3 é o produto escalar (também conhecido como produto interno $x.w$) do vetor de pesos w com o vetor de entradas x.

Contudo, em sua definição original, o nó MCP foi proposto com pesos não-ajustáveis (fixos) (de Pádua Braga et al. 2000). Assim, alguns outros modelos artificiais foram propostos com o objetivo de eliminar tal limitação, como exemplo pode ser citado o *perceptron* e o Adaline. Em 1986, com o desenvolvimento do algoritmo de retropropagação pelos pesquisadores Rumelhart, Hinton e Willians (Rumelhart 1896), foi definido um modelo de neurônio não-linear baseado no perceptron. Este neurônio, é atualmente o mais utilizado pela comunidade de

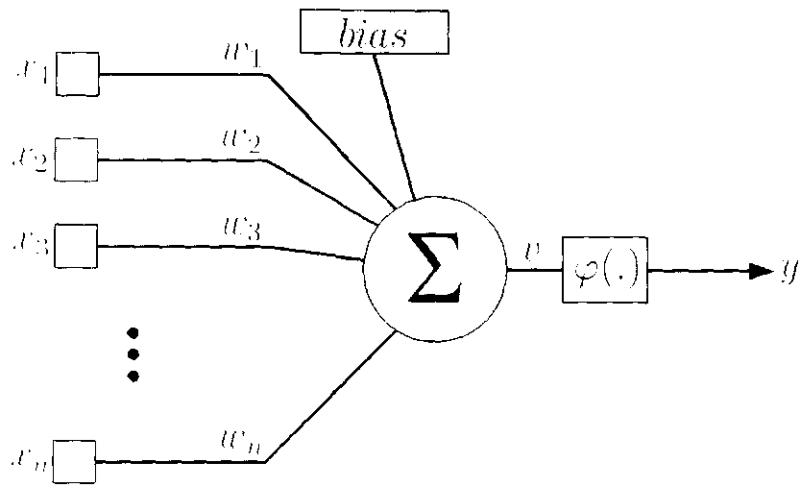


Figura 3.3: O neurônio não-linear

redes neurais (Figura 3.3).

No neurônio não-linear representado pela Figura 3.3, x_i representa a entrada presente na sinapse i do neurônio, w_i o peso associado a sinapse i, v o somatório das entradas ponderadas pelos pesos acrescido do termo *bias* (Equação 3.4), $\varphi(.)$ a função de ativação não-linear do neurônio e y a sua saída (Equação 3.5). As principais funções de ativação do neurônio serão apresentadas na próxima Subseção.

$$v = (\sum_{i=1}^{n} x_i w_i) + bias \tag{3.4}$$

$$y = \varphi(v) \tag{3.5}$$

Um outro tipo de neurônio, não inspirado no neurônio biológico, foi definido em 1966 pelo pesquisador Igor Aleksander. Este neurônio, denominado Nó RAM, diferencia dos neurônios como o MCP por não possuir pesos. Seu conhecimento está representado em funções booleanas armazenadas em uma memória de acesso randômico. Os nós RAM não são capazes de generalizar a nível de neurônio, contudo, a generalização se torna possível no nível de rede (denominadas Redes Neurais sem Pesos - RNSPs) (de Pádua Braga et al. 2000). Uma outra grande diferença entre os neurônios das RNSPs e os das RNAs é que o neurônio da RNSP é capaz de computar qualquer função booleana, enquanto o da RNA se restringe a funções linearmente separáveis. Entretanto, os nós das RNAs podem processar valores contínuos enquanto os nós RAM apenas valores discretos. O detalhamento deste tipo de rede neural (RNSP) está fora do escopo deste trabalho.

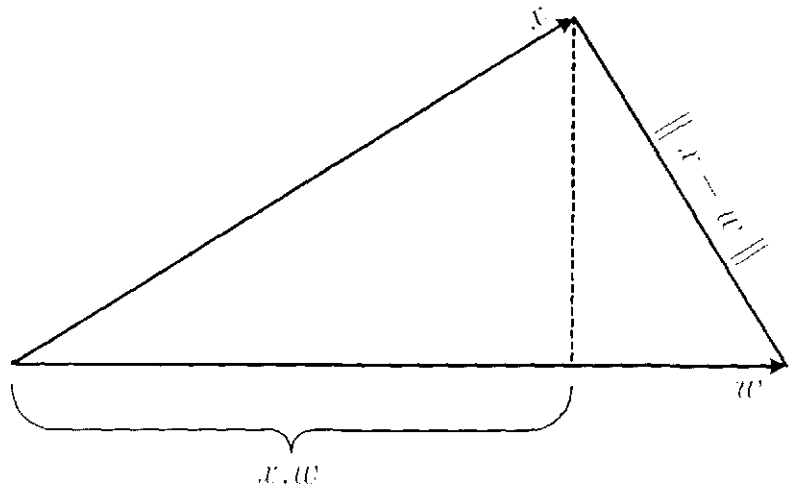


Figura 3.4: Comparação geométrica das funções de medida de similaridade utilizada pelos neurônios

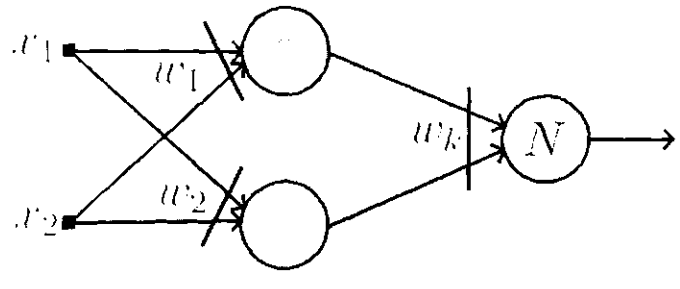
Conforme apresentado até o momento (Equações 3.3 e 3.4), a saída do neurônio, geralmente, é dependente de uma combinação linear das entradas submetida a uma função de ativação (3.5). Porém, para alguns modelos de redes neurais como: redes RBF (*Radial Basis Function* (Haykin 2001)), redes SOM (*Self-Organizing Maps* (Kohonen 2001)), etc., um outro tipo de função é utilizada no cálculo de v:

$$v = \sum_{i=1}^{n} (x_i - w_i)^2 \tag{3.6}$$

onde v representa a distância euclidiana entre os vetores w e x.

A principal diferença entre os dois tipos de funções apresentadas (Equação 3.4 e 3.6) pode ser representada geometricamente como ilustrado na Figura 3.4. Na qual é observada a relação antagônica entre as medidas: quanto mais próximos forem os vetores w e x menor será a distância euclidiana e maior será o produto interno entre eles.

Na Figura 3.5 é apresentado uma representação geométrica para a solução do problema XOR utilizando as duas abordagens (produto escalar (a) e distância euclidiana (b)). Na primeira abordagem (a) os pesos dos neurônios da camada oculta são representados pelos vetores w_1 e w_2 que são a base para os dois planos (perpendiculares aos vetores pesos). Na segunda abordagem (b) os pesos w_1 e w_2 representam os centros dos círculos. O neurônio de saída para ambas as abordagens pode ser um perceptron, sabendo-se que a função deste último neurônio é a implementação de uma porta lógica simples (AND para a abordagem (a) e uma porta OR para a abordagem (b)).



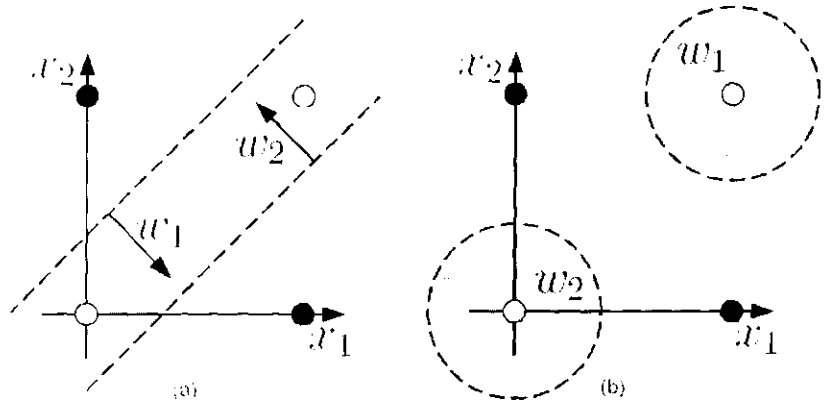


Figura 3.5: Duas soluções (RNA) para o problema XOR

3.1.3 Funções de Ativação

Quando a computação de v de cada neurônio é realizada pela distância euclidiana, geralmente, a seguinte função de ativação é utilizada:

$$y = exp(-v) \tag{3.7}$$

desta forma, quando a distância for próxima ou igual a zero a saída do neurônio será representado por 1 e para distâncias grandes, a saída da função será próxima à zero.

Para os neurônio que realizam a computação pelo produto escalar, as principais funções utilizadas são: liniar (Figura 3.6), linear por partes (Figura 3.7), logística (Figura 3.8) e tangente hiperbólica (Figura 3.9). Segundo Haykin (2001), a principal vantagem da utilização de funções sigmóides como a logística e a função tangente hiperbólica está na garantia de derivação dessas funções, permitindo assim, a construção de algoritmos, como o algoritmo de retropropagação (Seção 3.1.5), que dependem da derivada da função para o cálculo do gradiente. A Tabela 3.1 apresenta as funções citadas acima. Uma função de ativação linear também pode ser utilizada na saída do neurônio, porém, como apresentado em de Pádua Braga et al. (2000), uma rede de múltiplas camadas utilizando estas funções

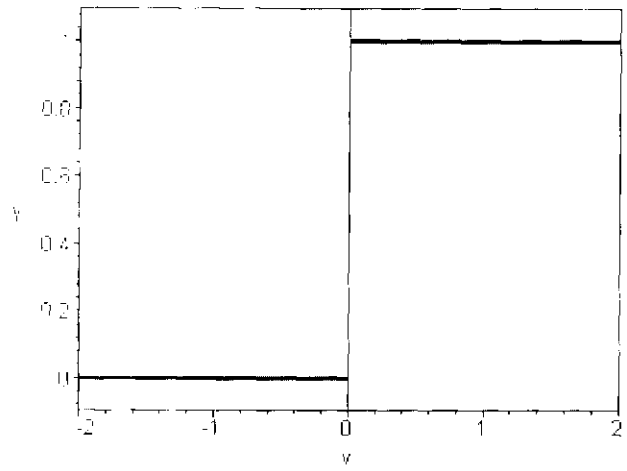


Figura 3.6: Função liniar

pode ser representada por uma rede neural de uma única camada equivalente.

Tabela 3.1: Funções de Ativação do Neurônio

Tipo de Função	**Função**
Liniar	$\varphi(v) = 1$ se $v \geq 0$
	0 se $v < 0$
Linear por Partes	$\varphi(v) = 1$ se $v \geq \frac{1}{2}$
	v se $-\frac{1}{2} < v < \frac{1}{2}$
	0 se $v \leq -\frac{1}{2}$
Sigmóide Logística	$\varphi(v) = (1 + exp(-av))^{-1}$
Tangente Hiperbólica	$\varphi(v) = \tanh(v)$

3.1.4 Classificação das Redes Neurais

As redes neurais artificiais podem ser classificadas segundo três parâmetros que as compõe: Arquitetura, tipo de neurônio e regra de aprendizagem. Nesta seção serão apresentadas as arquiteturas de redes encontradas na literatura e seus paradigmas de aprendizagem.

Arquitetura das RNAs

A arquitetura de uma rede neural é formada pelo número de camadas, número de neurônios em cada camada, pelas ligações entre os neurônios e pelo fluxo de informação. Segundo Haykin (2001), com base nas arquiteturas existentes, as redes neurais podem ser divididas em três classes principais: Redes *feedforward* (alimentadas adiante) com uma única camada (Figura 3.10), Redes *feedforward* com múltiplas camadas (Figura 3.11) e redes recorrentes (Figura 3.12).

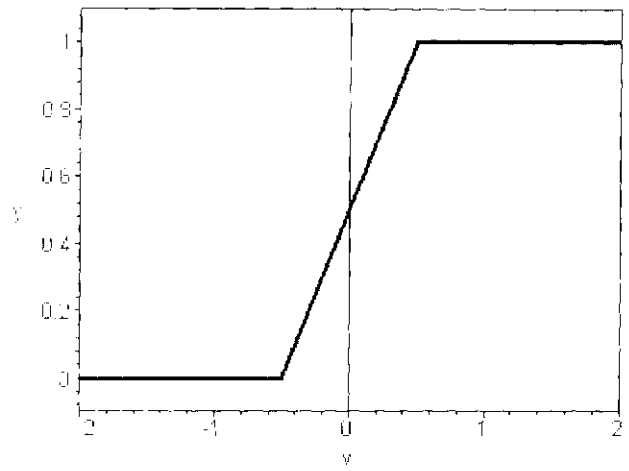


Figura 3.7: Função linear por Partes

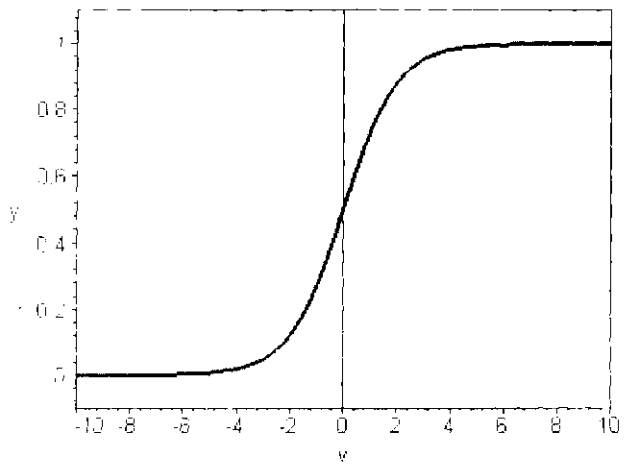


Figura 3.8: Função sigmóide logística

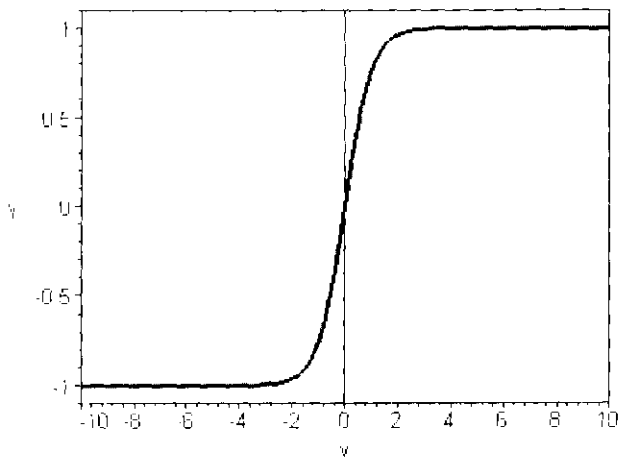


Figura 3.9: Função tangente hiperbólica

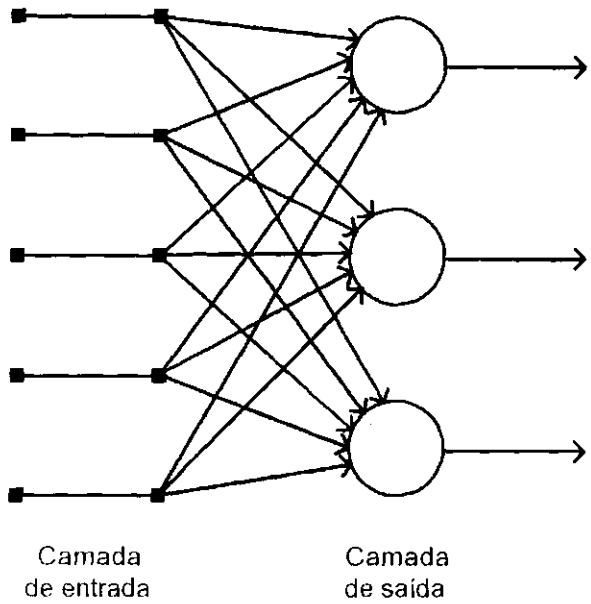


Figura 3.10: Rede neural com uma única camada alimentada adiante

As redes neurais *feedforward* possuem o fluxo de processamento organizado em uma única direção, no qual os sinais processados pelos neurônios de cada camada da rede são propagados para a próxima camada, isto é, o sinal se propaga da camada de entrada até a camada de saída de maneira unidirecional. A rede *perceptron* multi-camadas (MLP) e a rede Adaline são exemplos dessa forma de arquitetura.

As redes neurais recorrentes possuem laços de realimentação responsáveis por ligar neurônios das camadas posteriores (como a camada de saída da rede) a neurônios de camadas precedentes (por exemplo a camada de entrada). Essas ligações de realimentação possuem como característica principal a criação de um comportamento temporal na rede, como uma memória de curto prazo. Um exemplo clássico de rede neural recorrente é a Rede de Hopfield.

As redes neurais multi camadas ainda podem ser subdivididas em três categorias: redes completamente conectadas, parcialmente conectadas e localmente conectadas.

Nas redes totalmente conectadas todos os neurônios da camada l estão ligados a todos os neurônios da camada $l + 1$. Na parcialmente conectadas, alguns de neurônios da camada l estão ligados a alguns neurônios da camada $l + 1$. Já na rede localmente conectada, determinadas regiões (neurônios) de uma camada l estão ligados a determinados neurônios da camada seguinte $l + 1$.

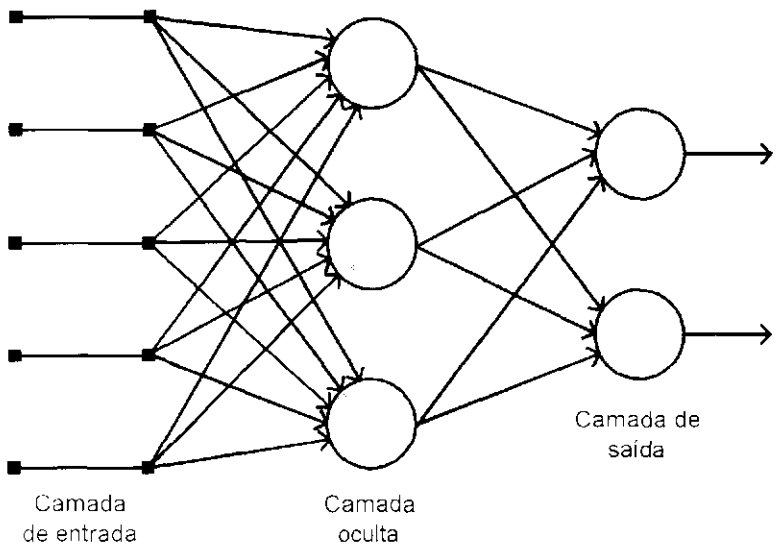


Figura 3.11: Rede neural com múltiplas camadas alimentada adiante

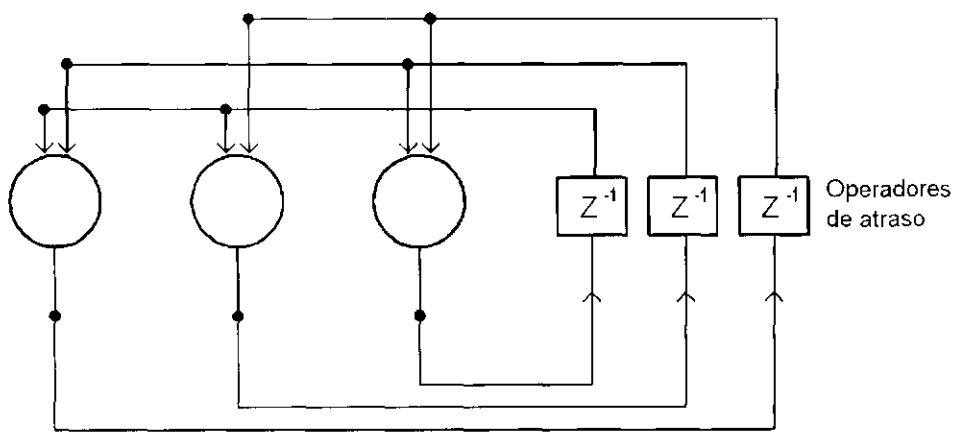


Figura 3.12: Rede neural recorrente

Tipos de Aprendizagem

Segundo Haykin (2001) a aprendizagem em redes neurais artificiais pode ser definida como:

> "*A aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre.*".

Na literatura são encontrados três paradigmas de aprendizagem: aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem por reforço. Vale ressaltar que alguns autores não consideram a aprendizagem por reforço como um paradigma independente, mas sim como um modo de aprendizagem pertencente ao paradigma não-supervisionado (sem um professor) (Haykin 2001), ou ainda como um caso particular do paradigma supervisionado (de Pádua Braga et al. 2000).

No aprendizado supervisionado um conjunto de treinamento composto de padrões de entradas associados aos seus respectivos rótulos são utilizados no treinamento da RNA. Durante o treinamento, cada padrão é apresentado à rede gerando um sinal de saída. Esse sinal gerado é comparado ao valor correspondente ao padrão apresentado (saída desejada) e com base nessa comparação é calculado um erro o qual é utilizado para a correção dos pesos da rede. Esse processo é repetido com todos os padrões do conjunto de treinamento até que o erro apresentado pela saída da rede esteja abaixo de um limiar aceitável. As redes neurais como a MLP, Adaline, perceptron, são exemplos que utilizam este paradigma de treinamento.

No aprendizado não-supervisionado não existe um crítico responsável por supervisionar o processo de aprendizagem. Desta forma, a propria rede neural deve extrair do conjunto de entradas as informações necessárias para a realização do mapeamento entrada-saída. São exemplos deste paradigma as redes neurais da família ART e os Mapas Auto-Organizáveis de Kohonen.

No aprendizado por reforço, uma função, definida a priori, é utilizada para indicar se a saída gerada pela rede é boa ou ruim, gerando assim um processo de recompensa ou penalização para a rede neural. O ajuste nos pesos da rede é realizado com base nesses valores de penalizações ou de recompensas obtidos a partir da função de avalição.

3.1.5 Perceptron Multi-camadas

Considerações Iniciais

As redes neurais perceptron multi-camadas (*Multi-Layer Perceptron* - MLP) podem ser vistas como uma generalização das redes *perceptron* de uma única camada, dado as devidas diferenças:

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

A arquitetura da rede MLP é apresentada na Figura 3.13, na qual é possível observar que a organização dos neurônios na rede é dada por camadas. A primeira camada (camada de entrada $l = 0$), é responsável por receber os sinais provenientes do ambiente (sensores, chaves, interrupções, etc.) e transmiti-lo à próxima camada ($l = 1$). Aquela não é propriamente uma camada da RNA, pois, não é constituída de neurônios como as demais camadas da rede (o sinal recebido pela camada $l = 0$ é o mesmo sinal transmitido à camada $l = 1$ ($y_j^0 = x_j$)). O fluxo de informação em uma rede MLP é propagado a partir da camada de entrada ($l = 0$), passando por todas as camadas ocultas até a camada de saída ($l = L$). Desta forma, as entradas de cada neurônio (N_i^l) das camadas ocultas e da camada de saída ($l = 1,2..L$) são as respectivas saídas (y_j) de todos os neurônios (N_j^{l-1}) da camada imediamente anterior ponderadas pelos respectivos pesos sinápticos (w_{ij}). A soma de todas as entradas (y_j) ponderadas pelos pesos (w_{ij}) é chamada de campo local induzido (v), representado pela equação 3.8.

$$v_j^l = (\sum_{i=1}^{m} w_{ji}^l y_i^{l-1}) + bias_j^l \tag{3.8}$$

onde o termo *bias* representa a polarização do neurônio j.

Por sua vez, o sinal de saída do neurônio (N_i^l) é computado pela função de ativação $\varphi(.)$ (Equação 3.9).

$$y_j^l = \varphi(v_j^l) \tag{3.9}$$

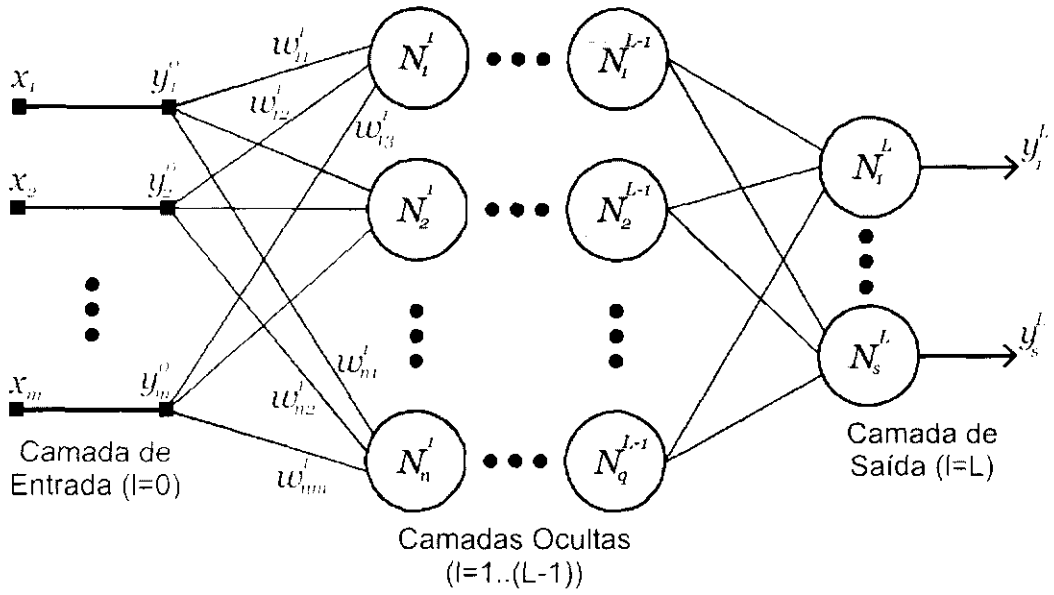


Figura 3.13: Arquitetura da rede neural MLP

onde a função $\varphi(.)$ é representada por uma função do tipo sigmóide, comumente a função logística ou a função tangente hiperbólica é utilizada (Tabela 3.1).

O principal problema de uma rede MLP esteve justamente em como realizar o ajuste dos pesos w_{ij} das camadas ocultas da rede. Este problema esteve presente por toda a década de 70. Apenas em meados da década de 80, com o desenvolvimento do algoritmo de retropropagação do erro (*backpropagation*) pelos pesquisadores Rumelhart, Hintno e Willians (Rumelhart 1896), foi possível realizar o treinamento de uma rede MLP. O Algoritmo de retropropagação do erro é descrito a seguir.

O Algoritmo de Retropropagação do Erro

O algoritmo de Retropropagação é sem dúvidas o mais importante algoritmo desenvolvido até hoje para o treinamento de redes neurais, tanto por fatores históricos (renascimento das RNAs) como pelo seu poder de treinamento para as redes MLPs.

Também conhecido como regra delta generalizada, o algoritmo de retropropagação, baseando-se no método de treinamento proposto por Widrow (regra delta), também utiliza o gradiente descendente da superfície de erro (Equação 3.1) com objetivo de minimizar o erro quadrático total apresentado pela saída da rede (Equação 3.2). Complementando a regra delta, por isso denominado regra delta generalizada, Rummelhart e sua equipe desenvolveram uma forma de calcular o erro em cada neurônio da camada oculta da rede. Por exemplo, o erro de um

Tabela 3.2: Tabela de Símbolos

Índice	Descrição
i, j, m, n, q, s	Índices gerais
l	Índice das camadas da rede
L	Índice da camada de saída da rede
P	Total de padrões do conjunto de treinamento
p	Índice de um padrão de entrada $X(p)$
t	Índice de iteração
N_j^l	Neurônio da rede
$X(p)$	Padrão (vetor) de entrada
x_j	Elemento de um vetor de entrada $X = \{x_1, x_2, ..., x_m\}$
$Y^l(p)$	Sinal (vetor) de saída da camada l da rede para o padrão p
y_i	i-ésimo elemento do vetor $Y = \{y_1, y_2, ..., y_n\}$
$D(p)$	Saída desejada para o padrão p
d_i	i-ésimo elemento do vetor $D = \{d_1, d_2, ..., d_n\}$
w_{ij}^l	Peso da j-ésima entrada do neurônio i da camada l
v_i^l	Campo local induzido do neurônio i da camada l (Eq. 3.8)
$\varphi(v)$	Função de ativação do neurônio (Tabela 3.1)
φ'	Derivada da função de ativação

neurônio da camada oculta $L-1$ é gerado pela soma dos erros apresentados pelos neurônio da camada L ponderado pelos respectivos pesos w. Assim, pode-se dizer que o erro na camada $L-1$ é uma estimativa do erro apresentado pela camada de saída. Sucessivamente, o erro de uma camada $L-2$ é uma estimativa da estimativa do erro da camada L. Por esse motivo, deve-se evitar um número excessivo de camada ocultas, pois quanto mais o erro se retropropaga pelas camadas da rede, menor é a sua precisão (de Pádua Braga et al. 2000).

O algoritmo de retropropagação do erro, na sua forma padrão, é apresentado no algoritmo 3.1. Pode ser observado que o algoritmo é composto por duas fases distintas: uma fase *forward* e uma fase *backward*. Na fase *forward* cada padrão p do conjunto de treinamento é apresentado na camada de entrada ($l = 0$) da rede (Equação 3.10), a qual por sua vez é responsável por progagar o sinal a próxima camada ($l = 1$). A saída dos neurônios da camada $l = 1$ é gerada pelas Equações 3.8 e 3.9, respectivamente. Camada a camada esse procedimento é executado até que o sinal atinja a camada de saída $l = L$ gerando o vetor $Y^L(p)$. Esse vetor é comparado ao vetor contendo os valores de saída desejados $D(p)$ para o padrão p apresentado (Equação 3.11) obtendo assim, o erro e_j para cada neurônio j da camada de saída L. O erro quadrático total apresentado pela rede para o padrão de entrada p é definido por $E(p)$ (Equação 3.12) e o erro médio quadrático para todos os padrões é definido pela Equação 3.13.

$$y_j^0 = x_j \tag{3.10}$$

$$e_j(p) = d_j(p) - y_j^L(p) \tag{3.11}$$

$$E(p) = \frac{1}{2}\sum_{j \in L} e_j^2(p) \tag{3.12}$$

$$E_{med} = \frac{1}{P}\sum_{p=1}^{P} E(p) \tag{3.13}$$

A segunda fase do algoritmo (*backward*) tem por objetivo ajustar os pesos w da rede a fim de diminuir o erro quadrático gerado (Equação 3.12). Para isto, camada a camada, partindo-se da camada de saída L o erro é retropropagado até a camada $l = 1$ calculando-se os gradientes δ para cada neurônio da rede. O cálculo do gradiente δ para cada neurônio da camada de saída é calculado pela Equação 3.14:

$$\delta_j^L = e_j^L \varphi_j'(v_j^L) \tag{3.14}$$

onde $\varphi'(.)$ é a derivada da função de ativação $\varphi(.)$. Os gradientes para os neurônios das camadas ocultas são calculados pela Equação 3.15:

$$\delta_j^l = \varphi_j'(v_j^l) \sum_k \delta_k^{l+1} w_{kj}^{l+1} \tag{3.15}$$

Após o cálculo dos gradientes δ, a Equação 3.16 é utilizada para ajustar os pesos da rede.

$$w_{ji}^l(t+1) = w_{ji}^l(t) + \eta \delta_j^l y_i^{l-1} \tag{3.16}$$

onde η representa o coeficiente de aprendizagem ($\eta \subset [0, 1]$), responsável pela velocidade de convergência (aprendizagem) da rede. Entretanto, vale ressaltar que um valor alto de η não significa garantia de aprendizagem, mas sim uma possibilidade de não-estabilidade no processo de treinamento.

Algoritmo 3.1 Algoritmo de Retropropagação Padrão

para todo i e j das camadas $l = 1..L$ **faça**
  $w_{ij}^l =$ **valor-aleatório()**
fim-para
enquanto não convergir **faça**
  para todo Padrão $X(n)$ **faça**
    Apresente $X(n)$ a RNA e obtenha a saída $Y(n)$
    Calcule o erro $e(n)$ (Equação 3.11)
    para todo Neurônio j da camada $l = L$ até a camada $l = 1$ **faça**
      Calcule o Gradiente δ_j^l da seguinte maneira:
      se Neurônio $j \in L$ **então**
        $\delta_j^l = e_j^L \varphi_j'(v_j^L)$
      senão
        $\delta_j^l = \varphi_j'(v_j^l) \sum_k \delta_k^{l+1} w_{kj}^{l+1}$
      fim-se
      $\Delta w_{ji}(t) = \eta \delta_j^l y_i^{l-1}$
      $w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t)$
    fim-para
  fim-para
fim-enquanto

O algoritmo de retropropagação apresentado até o momento é comumente referenciado na literatura como algoritmo de retropropagação padrão (*on-line*). Como descrito acima, para cada padrão do conjunto de treinamento apresentado, os pesos da rede neural são atualizados. Segundo de Pádua Braga et al. (2000),

apesar do erro quadrático médio total E_{med} descrever a soma de todos os erros $e_j(p)$ dos neurônios j de saída para todos os padrões p, a minimização do erro $E(p)$ para cada padrão conduzirá a minimização do erro total E_{med} (Equação 3.13).

O Algoritmo de Retropropagação em Lote

Uma outra forma de implementação do algoritmo de retropropagação é conhecida como retropropagação em lote. Na implementação em lote todos os padrões do conjunto de treinamento (uma época) são apresentados a rede gerando um gradiente médio δ_{med} (Equação 3.17) que é uma estimativa mais precisa do vetor gradiente para o problema dado quando comparado ao método padrão, que gera um vetor gradiente δ para cada padrão apresentado.

$$\delta_{med,j} = \frac{1}{P}\sum_{p=1}^{P}\delta_j(p) \tag{3.17}$$

Após o cálculo dos gradientes δ_{med} (Algoritmo 3.2), a atualização dos pesos é realizada da mesma forma que foi apresentada para o algoritmo de retropropagação padrão (Equação 3.16). Entretanto, a variação do peso Δw_{ji}^l é dada produto $\eta\delta_j^l y_i^{l-1}$, podendo ser observado que além do gradiente δ_{med} também é necessário, para a atualização dos pesos w_{ji}^l de cada neurônio N_j^l, o valor das saídas y_i^{l-1} de todos os neurônios N_i^{l-1}. Uma forma de realizar essa atualização dos pesos é, ao invés de armazenar apenas os vetores gradientes $\delta_{med,j}$ de cada neurônio N_j^l, armazenar o produto $\delta_j^l(p)y_i^{l-1}(p)$ para todo neurônio N_i^{l-1} na apresentação do padrão p (Equação 3.18).

$$\rho_{ji}^l = \sum_{p\in P}\delta_j^l(p)y_i^{l-1}(p) \tag{3.18}$$

onde ρ_{ji}^l é o somatório do produto do descrito acima para cada padrão $p \in P$. Desta forma a atualização de cada peso w_{ji}^l será dado pela seguinte Equação 3.19:

$$w_{ji}^l(t+1) = w_{ji}^l(t) + \eta\rho_{ji}^l \tag{3.19}$$

o algoritmo para o cálculo dos vetores de gradientes e dos acumuladores é apresentado no Algoritmo 3.2 e o algoritmo de retropropagação em lote é apresentado no Algoritmo 3.3.

Adição do Termo Momentum

Como relatado anteriormente, η representa a taxa de aprendizagem, isto é, com qual velocidade ocorrerão os ajustes dos pesos sinápticos w da rede neural.

Algoritmo 3.2 Cálculo dos Vetores Gradientes: Treinamento em Lote

para todo δ^l_{medj} e ρ^l_{ji} **faça**
  $\delta^l_{medj} = 0$
  $\rho^l_{ji} = 0$
fim-para
para todo padrão $p \subset P$ **faça**
  Apresente $X(p)$ a RNA e obtenha a saída $Y(p)$
  Calcule o erro $e(p)$ (Equação 3.11)
  para todo Neurônio j da camada $l = L$ até a camada $l = 1$ **faça**
    se Neurônio $j \subset L$ **então**
      $\delta^l_j = e^l_j \varphi'_j(v^l_j)$
      $\delta^l_{medj} = \delta^l_{medj} + \frac{1}{P}\delta^l_j$
    senão
      $\delta^l_j = \varphi'_j(v^l_j) \sum_k \delta^{l+1}_k w^{l-1}_{kj}$
      $\delta^l_{medj} = \delta^l_{medj} + \frac{1}{P}\delta^l_j$
    fim-se
    para todo Neurônio i da camada $l = L - 1$ até a camada $l = 0$ **faça**
      $\rho^l_{ji} = \rho^l_{ji} + \frac{1}{P}\delta^l_j y^{l-1}_i$
    fim-para
  fim-para
fim-para

Algoritmo 3.3 Algoritmo de Retropropagação em Lote

para todo i e j das camadas $l = 1..L$ **faça**
  w^l_{ij} = **valor-aleatório()**
fim-para
enquanto não convergir **faça**
  Calcule os Vetores Gradientes (Algoritmo 3.2)
  para todo w^l_{ji} das camadas $l = 1..L$ **faça**
    $\Delta w^l_{ji}(t) = \eta\rho^l_{ji}$
    $w^l_{ij}(t+1) = w^l_{ij}(t) - \Delta w^l_{ij}(t)$
  fim-para
fim-enquanto

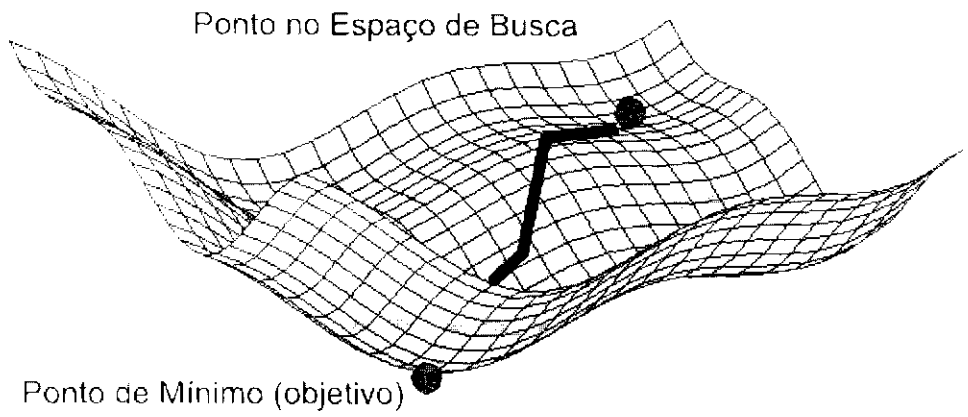


Figura 3.11: Representação do espaço de busca do problema

Segundo Haykin (2001), o algoritmo de retropropagação é responsável por fornecer um vetor gradiente δ cuja função é estabelecer uma aproximação para o trajetória que será percorrida pelo espaço de busca do problema (Figura 3.14). Desta forma, quando η é representado por um valor pequeno (próximo à zero) a trajetória no espaço de busca será suave, porém consumindo uma grande quantida de iterações para a realização do treinamento da RNA. Por outro lado, se η for um valor próximo a um, o treinamento da rede pode se tornar instável, impossibilitando que a busca pela superfície encontre um mínimo. Uma forma de acelerar o processo de treinamento de uma MLP reduzindo o perigo de instabilidade pode ser realizada adicionando-se o termo *momentum* (Rumelhart 1896). Assim, com a utilização do termo *momentum*, a atualização dos pesos w da rede MLP é dada por:

$$w_{ji}^{l}(t+1) = w_{ji}^{l}(t) + \eta\delta_{j}^{t}y_{i}^{l-1} + \alpha\Delta w_{ji}^{l}(t-1) \tag{3.20}$$

onde α é o termo momentum ($\alpha \in [0,1]$). A Equação 3.20 é utilizada na implementação padrão do algoritmo de retropropagação. Para a implementação em lote, a atualização dos pesos w é dada por:

$$w_{ji}^{l}(t+1) = w_{ji}^{l}(t) + \eta\rho_{ji}^{l} + \alpha\Delta w_{ji}^{l}(t-1) \tag{3.21}$$

O termo *momentum*, além de tornar o processo de treinamento mais estável, insere uma inercia no processo de descida na superfície de erro, evitando, em alguns casos, possíveis mínimos locais e também diminuindo as oscilações observadas quando a rede está em torno de um mínimo local.

3.1.6 Algoritmo Rprop

Um grande problema encontrado no algoritmo de retropropagação está na forma do uso do gradiente descendente. Neste algoritmo (retropropagação) o valor do gradiente é utilizado diretamente na etapa de atualização dos pesos da rede. Um problema que pode ser nitidamente notado quando a superfície de busca possui diversas regiões muito planas, resultando em um gradiente com valor próximo à zero.

Para tentar resolver este problema, os pesquisadores Martin Riedmiller e Heinrich Braun (Riedmiller & Braun 1992) propuseram um algoritmo de treinamento denominado Rprop (***R**esilient back**prop**agation*).

O algoritmo Rprop é um algoritmo supervisionado de treinamento em lote e, ao contrário do algoritmo de retropropagação que possui uma taxa de atualização fixa ou decrescente dos pesos, o Rprop realiza uma adaptação local da taxa atualização para cada peso da rede. Além disso, apenas o sinal do gradiente é utilizado para indicar a direção da atualização e não o seu valor como no algoritmo de retropropagação. A taxa de atualização, denotada por Δ_{ji}^{l}, é aumentada por um fator η^{+}, quando a derivada parcial do erro relativa a um peso w_{ji} não altera o seu sinal, indicando que o último reajuste realizou uma correção que diminuiu o erro apresentado pela rede. Quando o sinal da derivada parcial do erro apresenta uma mudança de sinal, significa que o último ajuste foi maior que o necessário. Assim, o valor da taxa de atualização Δ_{ji}^{l} é diminuído por um fator η^{-}.

As taxas η^{-} e η^{-} são dadas por:

$$0 < \eta_{-} < 1 < \eta_{+} \tag{3.22}$$

Em Riedmiller (1994b) são sugeridos os valores de $\eta^{-} = 0,5$ e de $\eta_{+} = 1,2$. Esses valores apresentaram bons resultados independente do problema a ser tratado pela RNA.

Como apresentado no Algoritmo 3.4, todos as taxas de atualização Δ_{ji}^{l} são configuradas em uma taxa inicial Δ_0. Geralmente este valor é configurado com uma valor pequeno, na ordem de $\Delta_0 = 0,1$. Outros dois parâmetros que devem ser estabelecidos são: Δ_{max} e Δ_{min}. Em Riedmiller (1994b) são sugeridos os valores de $\Delta_{max} = 50,0$ e $\Delta_{min} = 10^{-6}$.

Embora a convergência do algoritmo não seja tão sensível a taxa Δ_{max}, é alertado que para alguns problemas um valor menor que $50,0$ deve ser utilizado, evitando assim uma convergência prematura para um mínimo local subótimo.

O algoritmo Rprop é um dos métodos de treinamento de primeira ordem para

redes MLP que apresenta melhor performance em tempo de convergência (Igel & Hüsken 2003). Em Riedmiller (1994a) foi realizado um estudo de alguns algoritmos de primeira ordem utilizados no treinamento de redes MLP e os resultados demonstram que o Rprop apresenta uma convergência mais rápida que algoritmos como o algoritmo de retropropagação.

Algoritmo 3.4 Algoritmo Rprop (Riedmiller 1994b)

para todo i e j das camadas $l = 1..L$ **faça**
  $w_{ji}^{l} =$ **valor-aleatório()**
  $\Delta_{ji}^{l}(t) = \Delta_0$
  $\delta_{ji}^{l}(t-1) = 0$
fim-para
enquanto não convergir **faça**
  Calcule os Vetores Gradientes (Algoritmo 3.2)
  para todo w **faça**
    se $\delta_{ji}^{l}(t-1) * \delta_{ji}^{l}(t) > 0$ **então**
      $\Delta_{ij}^{l}(t) =$ **mínimo**$(\Delta_{ij}^{l}(t-1) * \eta^{+}, \Delta_{max})$
      $\Delta w_{ij}(t) =$ **−sinal**$(\delta_{ji}^{l}(t)) * \Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\delta_{ji}^{l}(t-1) = \delta_{ji}^{l}(t)$
    senãose $\delta_{ji}^{l}(t-1) * \delta_{ji}^{l}(t) < 0$ **então**
      $\Delta_{ij}^{l}(t) =$ **máximo**$(\Delta_{ij}^{l}(t-1) * \eta^{-}, \Delta_{min})$
      $\delta_{ji}^{l}(t-1) = 0$
    senãose $\delta_{ji}^{l}(t-1) * \delta_{ji}^{l}(t) = 0$ **então**
      $\Delta w_{ij}(t) =$ **-sinal**$(\delta_{ji}^{l}(t)) * \Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\delta_{ji}^{l}(t-1) = \delta_{ji}^{l}(t)$
    fim-se
  fim-para
fim-enquanto

3.1.7 Considerações Finais Sobre MLPs

As redes neurais MLPs são capazes de extrair informações até mesmo de dados ruidosos provenientes de fontes como sonares, microfones, câmeras, etc. (Mitchel 1997).

Apesar do tempo de treinamento de uma rede neural MLP ser relativamente lento, após treinada, a rede apresenta tempos de resposta pequenos, o que valida a sua aplicação no desenvolvimento de sistemas para atuarem em tempo real.

Devido a essas informações até aqui citadas, as rede neurais do tipo MLP foram adotadas para o desenvolvimento deste trabalho de mestrado.

3.2 Algoritmos Evolutivos

Primeiramente proposto por Holland (1975) (Algoritmos Genéticos) na Universidade de Michigan, os Algoritmos Evolutivos (AEs) foram idealizados sem planejar a criação de um projeto de algoritmo para solucionar determinados problemas. Holland pretendia estudar formalmente o fenômeno de adaptação que ocorre na natureza e, com isso, estabelecer formas para permitir que tais mecanismos que naturais fossem implementados em um sistema computacional. Entretanto, desde sua criação, os AEs têm sido aprimorados e aplicados as mais diversas áreas do conhecimento.

Os AEs são uma interessante estratégia computacional devido à sua habilidade de resolver problemas difíceis de otimização e sua versatilidade em aplicações envolvendo a área de aprendizado de máquina (Tomassini 1995). As metodologias de AEs têm sido aplicadas não somente na pesquisa acadêmica, mas também na resolução de problemas industriais. AEs são procedimentos de adaptação, busca e otimização inspirados na biologia, mais especificamente na Teoria da Evolução Darwiniana (Darwin 1859). Eles tentam abstrair e imitar algumas das características da evolução natural para produzir processos funcionais artificiais adaptativos. AEs consistem em mapear possíveis soluções em algum espaço do problema em cromossomos de indivíduos. As características de cada possível solução também sao chamadas de fenótipo, que é representada por um cromossomo, chamado genótipo. Assim, cada indivíduo, através de uma codificação adequada, representa um ponto (uma possível solução) em um espaço de busca de um dado problema.

Segundo Goldberg (1989), as principais diferenças entre os AEs e os métodos tradicionais de busca e otimização são:

- os AEs realizam a busca considerando uma codificação do conjunto de parâmetros do problema e não os próprios parâmetros;
- efetuam a busca utilizando uma população de possíveis soluções em paralelo e não uma única solução;
- utilizam informações como custo ou recompensa e não modelos matemáticos como a derivada.

3.2.1 Descrição Geral de um AE

Para a utilização de um AE em um dado problema, dois aspéctos principais devem ser levados em consideração:

1. o problema deve ser representado (codificado) em um cromossomo (genótipo) que o caracterize;

2. uma função de aptidão deve ser estabelecida, com o objetivo de se avaliar quão próximo cada indivíduo da população está de uma solução desejada ou satisfatória.

Considerado como o primeiro passo na utilização de um AE, a codificação do problema deve ser estabelecida para que uma população inicial randômica de possíveis soluções (representações) para o problema possa ser gerada. Uma vez gerada tal população, no qual cada indivíduo é representado por um cromossomo, deve-se decodificar cada um desses em seus respectivos fenótipos para que possam ser avaliados através da função de aptidão estabelecida.

Os cromossomos da população, tradicionalmente são representados por vetores binários, no qual cada posição do vetor denota a presença ou a ausência de determinada característica. Contudo, diversas representações podem ser utilizadas, como: números inteiros, números reais, etc.

Após a avaliação de todos os indivíduos da população em uma dada geração t, é realizado um processo de seleção reponsável por determinar quais dos indivíduos farão parte da fase de reprodução, a fim de se estabelecer uma nova população para a geração $t + 1$.

Para a fase de seleção, diversas técnicas são utilizadas:

- **Roleta**: os indivíduos da população são escolhidos de conforme sua aptidão (*fitness*), sendo que quanto maior for o valor de aptidão deste indivíduo, maior será a região da roleta que o representa, aumentando assim, sua probabilidade de ser selecionado.

- **Elitismo**: o indivíduo com maior aptidão é diretamente selecionado;

- **Seleção por Truncamento**: os t indivíduos com os maiores valores de aptidão são selecionados;

- dentre outras técnicas.

Após a seleção dos indivíduos, inicia-se a fase de reprodução. Nesta fase os indivíduos podem ser modificados e/ou combinados a fim de se gerar uma nova população.

Para realização da reprodução entre os indivíduos selecionados, alguns operadores de reprodução são utilizados, como o operador de *crossover*, o operador de mutação, etc.

O operador de *crossover* (cruzamento ou recombinação) é responsável pela realização da reprodução "sexuada", no qual dois indivíduos pais são selecionados e partes dos seus cromossomos são trocados ou combinados. O *crossover* pode ser realizado segundo diversas metodologias:

- **Um ponto**: é estabelecida um ponto de cruzamento e, a partir deste as informações genéticas dos pais são cruzadas (trocadas);
- **Multi-pontos**: da mesma forma que o *crossover* de um ponto, porém, neste diversos pontos de cruzamento são estabelecidos;
- **Uniforme**: é gerado uma mascara indicando quais genes dos pais serão trocados para formar os novos indivíduos;
- **Média**: quando o cromosso não é gerado apenas por números binários, uma possível forma de realização do *crossover* pode ser dada através da média entre os genes dos pais;
- dentre outras técnicas.

Um outro operador de reprodução muito utilizado é a mutação. Este é responsável pela realização da reprodução "assexuada", e tem por finalidade gerar uma maior diversidade genética na população. A mutação geralmente ocorre com uma pequena probabilidade, denominada taxa de mutação, no qual um ou mais genes de cada cromossomo são aleatoriamente modificados. Com a aplicação do operador de mutação, diversas possíveis soluções que antes não faziam parte da população de indivíduos podem ser alcançadas, realizando assim, uma maior cobertura do espaço de busca do problema.

No algoritmo 3.5 é apresentada uma possível forma de se implementar um algoritmo evolutivo, onde t representa a atual geração do processo de evolução, $P(t)$ é a população na geração t e n o número de indivíduos da população.

3.2.2 AEs e Redes Neurais

A determinação de uma rede neural aplicada à solução de uma determinada tarefa, como para a construção de um sistema de visão aplicado ao controle de um robô móvel, não é uma tarefa simples. Diversos parâmetros, como taxa de aprendizagem, topologia, função de ativação dos neurônios estão intimamente relacionados com o desempenho e capacidade de generalização que a rede pode apresentar (de Pádua Braga et al. 2000).

A busca por uma rede neural adequada no espaço de possibilidades do problema é computacionalmente custosa. Um outro fator importante apresentado

Algoritmo 3.5 Algoritmo Evolutivo

```
t ← 0 (geração 0)
Gerar uma população inicial randômica P(t) de n indivíduos
Avaliar, segundo a função de aptidão, os n indivíduos de (P(t))
enquanto não atingir condição de parada faça
    Selecione P(t + 1) a partir dos indivíduos de P(t)
    Reproduza P(t + 1)
    Avalie P(t + 1)
    t ← t + 1
fim-enquanto
```

por Miller et al. (1989) é que o espaço de busca é deceptivo e multimodal. Deceptivo porque duas redes muito semelhantes podem apresentar performances muito diferentes e multimodal pois, duas redes bem diferentes podem possuir performances muito semelhantes.

Algoritmos Evolutivos têm sido aplicados com sucesso na otimização de redes neurais. Além disso, uma possível explicação biológica para a utilização de AEs para a obtenção de uma melhor arquitetura para uma rede neural é que o cérebro humano é resultado de um processo de evolução natural (Murray 1994).

Os algoritmos evolutivos tem sido aplicados na evolução dos diversos parâmetros das RNAs como: topologia, pesos sinápticos, taxa de aprendizagem, etc. Contudo este trabalho, como será apresentado abaixo, restringe-se a utilização dos AEs com a finalidade de se obter uma topologia adequada para o problema em estudo.

Diversos trabalhos empregando AEs na evolução de redes neurais multi-camadas têm sido apresentados. Em (Castillo et al. 2000) o método G-Prop (Genetic Backpropagation) é apresentado. Neste trabalho, um algoritmo genético é utilizado na obtenção dos pesos iniciais de uma rede MLP e o número de elementos em sua camada oculta (apenas uma camada oculta é tratada no problema), em seguida o algoritmo de retropropagação do erro é utilizado no treinamento da rede. Com essa metodologia, a rede possui uma menor chance de ficar presa em um mínimo local, o que é mais provável de acontecer quando somente o algoritmo de retropropagação é utilizado. O critério de avaliação utilizado neste artigo é dado pelo número de elementos classificados corretamente, juntamente com o número de elementos na camada intermediária. Em Abraham (2004) foi desenvolvido um sistema evolutivo automático aplicado ao processo de otimização de redes neurais MLP. Neste sistema (MLEANN - *Meta Learning Evolutionary Artificial Neural Networks*), todos os parâmetros da rede neural como: arquitetura da rede, função de ativação, pesos, algoritmo de treinamento e seus parâmetros são evoluídos

a fim de otimizar a rede neural para uma determinada aplicação. Foi realizado um levantamento dos tradicionais algoritmos utilizados no treinamento de redes MLP e alguns experimentos foram realizados. Nos testes realizados o MLEANN foi mais rápido e obteve um maior grau de generalização quando comparado com as abordagens não evolutivas.

3.2.3 Algoritmo Evolutivo Desenvolvido

Neste trabalho, o algoritmo evolutivo é utilizado como uma ferramenta de busca por uma melhor topologia para redes MLP.

O processo evolutivo da topologia da rede neural MLP consiste na seleção da quantidade de camadas ocultas e da quantidade de neurônios em cada uma dessas camadas.

Cada rede neural gerada é treinada com uma base de dados de treinamento pelo algoritmo Rprop (Seção 3.1.6). Assim que a rede neural é treinada, uma outra base de dados é utilizada no processo de validação da rede. O erro quadrático médio (EQM) gerado a partir do conjunto de validação é utilizado como critério de avaliação da rede neural. O *fitness* de cada indivíduo da população é gerado a partir do erro quadrático médio (EQM) obtido com a base de validação (Equação 3.23.

$$fitness = 1 - EQM \tag{3.23}$$

No fluxograma do sistema evolutivo, apresentado na Figura 3.15, a população inicial é formada a partir da criação de indivíduos (RNAs) com cromossomos gerados aleatoriamente. Em seguida, os indivíduos são avaliados através do erro quadrático médio e recebem um valor *fitness*. Em seguida, é verificado se a condição de término do experimento foi alcançada. O critério de parada para o AE implementado é dado pela obtenção de um resultados estável mantido por várias gerações. Caso a condição de termino ainda não esteja satisfeita, o próximo passo é a seleção de indivíduos para o cruzamento genético. Em seguida, é utilizado o *crossover* entre os indivíduos para gerar a nova população. Os novos indivíduos podem vir a sofrer mutações em seus genes, o que pode vir a contribuir para aumentar a diversidade genética.

O cromosso que representa o genótipo da rede neural é formado por três genes (Figura 3.16): um representando a quantidade de camadas ocultas (zero, uma ou duas camadas) na RNA, considerando que uma MLP com duas camadas ocultas, teoricamente, é capaz de mapear qualquer função matemática. O segundo e o terceiro genes informam a quantidade de elementos na primeira e na segunda

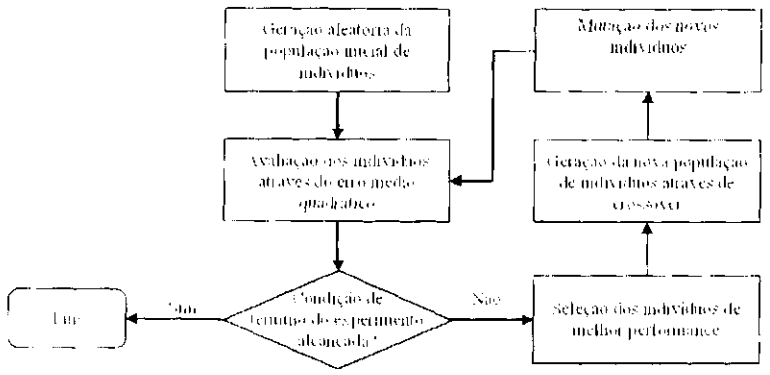


Figura 3.15: Fluxograma do sistema evolutivo implementado

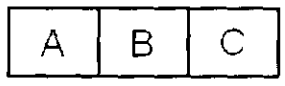


Figura 3.16: O cromossomo

camada ocultas da rede, caso existam, dependendo do primeiro gene.

O gene A possui um valor entre 0 e 29. Se este valor estiver entre 0 e 9, isto significa que não existirá nenhuma camada oculta na RNA. Se estiver entre 10 e 19, haverá apenas uma camada. E se estiver entre 20 e 29, duas camadas. Portanto, a quantidade de camadas ocultas dependerá do intervalo em que se encontra o valor em A. Os genes B e C são inicialmente gerados com valores aleatórios entre 1 e 150, determinando, assim, o número de neurônios nas camadas 1 e 2, respectivamente, caso existam. Na seleção dos indivíduos para o cruzamento, foi utilizado o elitismo (Tomassini 1995), onde é escolhido um indivíduo que tem a melhor aptidão (*fitness*) dentre um grupo de indivíduos da população. Essa técnica existe para garantir que os melhores cromossomos encontrados sejam preservados. Para a criação dos novos cromossomos (filhos) da próxima geração, é realizado o crossover entre dois indivíduos (pais). Neste processo, é feita a média entre cada gene correspondente nos pais para gerar o filho: faz-se a média aritmética entre os valores no gene A dos pais para obter o valor no gene A do cromossomo do filho; em seguida, repete-se o procedimento para os genes B e C.

Na mutação, optou-se por adotar a seguinte metodologia: inicialmente determinam-se quais genes serão mutados, ou seja, se só o gene A ou B ou C ou combinações, como A e B ou A, B e C. Se o gene A for selecionado para ser mutado, então deverá ser somado ou subtraído 3 de seu valor (constante definida empiricamente). No caso dos genes B e C, caso sejam mutados, o valor da variação (soma ou subtração) é determinado de acordo com os valores apresentados na Tabela 3.3. É gerado um número aleatório entre 1 e 100 que determinará qual será o valor de

Tabela 3.3: Valor das variações da mutação nos genes B e C

Intervalo	1-50	51-75	76-90	91-100
Percentual	50%	25%	15%	10%
Valor da variação	±1	±2	±5	±10

variação. Caso o número aleatório seja, por exemplo, 55, então a variação será de ±2. Assim, tem-se que a chance da variação ser ±01 é de 50%, ±2 de 25%, ±5 de 15% e ±10 de 10%. Também é escolhido aleatoriamente se o valor da variação será positivo ou negativo.

Modificações no Algoritmo Evolutivo (AE-II)

Uma modificação do algoritmo, descrito anteriormente, é proposta adicionando-se ao AE algumas novas funções. Essas modificações objetivam encontrar melhores resultados, tanto na topologia da MLP evoluída como no tempo para encontrar tal solução.

As diferenças entre o algoritmo anteriormente descrito e este modificado são:

- um indivíduo é selecionado para se reproduzir assexuadamenteem cada geração.
- acréscimo da predação randômica (Simões 2000): o indivídio com o menor *fitness* é selecionado e seus parâmetros (cromossomo) são reconfigurados de forma randomica. Esta função é executada a cada 20 gerações do processo evolutivo.
- acréscimo da predação por síntese: o indivíduo com menor *fitness* (indivíduo diferente do selecionado na predação randômica) é selecionado e seus parâmetros são reconfigurados com a média dos parâmetros dos outros indivíduos da população.
- uma função de mutação modificada: um valor entre -5 e $+5$ pode ser adicionado aos parâmetros B e C do cromossomo. Esta função é chamada 3 vezes para gerações cujo número seja múltiplo de 6, 2 vezes para gerações múltiplos de 2 e 1 vez para as demais gerações.

Os resultados dos testes realizados estão descritos no Capítulo 5.

A seguir, é apresentada uma outra técnica utilizada no desenvolvimento deste trabalho de mestrado: a técnica de Campos Potenciais.

3.3 Campos Potenciais

Esta técnica vem sendo muito utilizada para fazer com que robós móveis planejem trajetórias em um ambiente dinâmico evitando colisões com obstáculos.

A idéia de imaginar forças atuando sobre um robô foi sugerida por Khatib (1985). Neste método, obstáculos exercem forças repulsivas e a meta aplica uma força atrativa sobre o robô. A força resultante $\overline{F}$ é composta de uma força atrativa direcionada para a meta e forças repulsivas proveniente de obstáculos. Para cada nova posição do robô todas as forças deverão ser novamente calculadas. Com isto, é possível que o robô desvie dos obstáculos e chegue até a meta.

Krogh (1984) aprimorou este conceito considerando a velocidade do robô na vizinhança dos obstáculos. Thorpe (1984) aplicou o método de campos potenciais para planejamento *off-line*. Krogh & Thorpe (1986) sugeriram uma combinação do método para planejamento de caminhos locais e globais utilizando para isto uma abordagem denominada Campos Potenciais Generalizado. Newman & Hogan (1987) introduziram a construção de funções potenciais através da combinação de funções individuais de obstáculos com operadores lógicos.

Em todos os métodos citados foi assumido um modelo conhecido do mundo, com formas geométricas pré-definidas representando obstáculos e o caminho do robô foi gerado *off-line*. No entanto, Brooks (1986) e Arkin (1989) são os pioneiros em utilizar a técnica de campos potenciais sem utilizar informações sobre o ambiente, realizando o cálculo do campo potencial utilizando-se os dados recebidos pelos sensores ultrasônicos (sonar) em cada instante de tempo.

Brooks utilizou campos potenciais como um controlador reflexivo. O controlador simplesmente reage executando ações diretamente ligadas as percepções, resultando assim em respostas imediatas a estímulos externos.

Um método similar, denominado *Motor Schema*, foi utilizado por Arkin em seus robós. Este método consiste em ativar vários comportamentos simultâneos que produzem um vetor força (atração ou repulsão) como resposta. A direção a ser seguida pelo robo será o vetor resultante da soma de todos os vetores gerados. Esta técnica é fundamentada na Teoria de Esquemas que é considerada ideal para expressar funções do cérebro.

Outras abordagens utilizando campos potenciais para o planejamento de trajetórias utilizando robós móveis foram sugeridas por Faria & Romero (2000), Costa & Pegoraro (2000) e Gomes & Campos (2001).

Várias métodos podem ser escritos para descrever as forças que agem sobre o robô. É por meio da combinação destas forças (força resultante) que se monta os comportamentos esperados para um robô. Os mais utilizados correspondem

aos comportamentos de busca pelo alvo (força de atração) e desvio de obstáculos (força de repulsão). Contudo, nada impede que se criem novos comportamentos baseados em outros tipos de forças. A seguir, será mostrado como realizar os cálculos das forças de repulsão, atração e da força resultante.

3.3.1 Força de Repulsão

Na Figura 3.17(a) mostra-se que a força de repulsão decai ao se afastar de um obstáculo, pois esta força ($\vec{F_r}$) é um vetor cujo módulo é inversamente proporcional ao quadrado da distancia (d) entre o robô (R) e objeto observado (O) (Figura 3.17(b)).

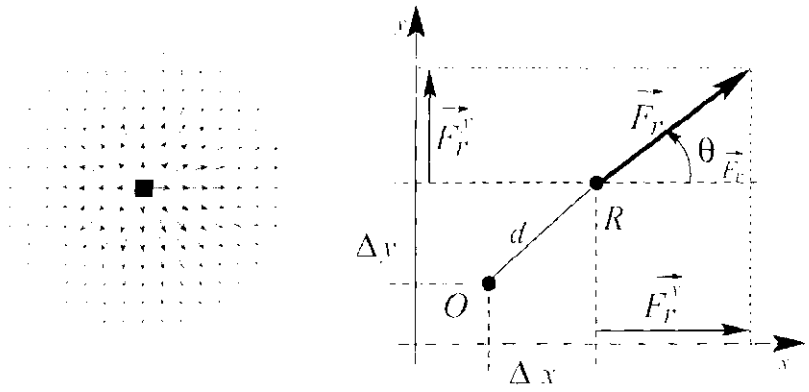


Figura 3.17: (a) Força de repulsão na vizinhança de um obstáculo; (b) Força de repulsão do objeto O sobre um robô localizado em R.

O vetor $\vec{F_r}$ também pode ser representado por suas componentes: módulo e direção, apresentados na Equação 3.24:

$$|\vec{F_r}| = \frac{Q}{d^2} \quad e \quad \theta_{\vec{F_r}} = \arctan(-\Delta y, -\Delta x) \tag{3.24}$$

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

Para calcular a força de repulsão para vários obstáculos, deve-se fazer o somatório dos vetores das forças de repulsão geradas, como mostrado na Equação 3.25:

$$\vec{F_R} = \sum \vec{F_r} \tag{3.25}$$

3.3.2 Força de Atração

Na Figura 3.18(a) e Figura 3.18(b) é apresentado o campo potencial de atração, no qual é observado que o valor de $|\vec{F_a}|$ deve ser considerado constante para que o agente seja atraído pela meta mesmo estando distante da mesma (Equação 3.26).

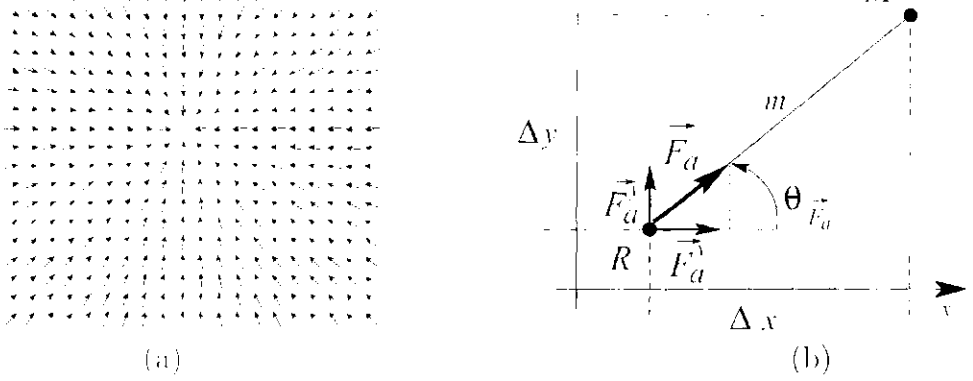


Figura 3.18: (a) Campo de atração constante; (b) Atração entre o robô R e a meta M.

$$|\vec{F_a}| = C \quad e \quad \theta_{\vec{F_a}} = \arctan(\Delta y, \Delta x) \tag{3.26}$$

no qual C representa um escalar constante de atração. Note que Δx e Δy diferem da Equação 3.24 por não estarem acompanhados do sinal negativo. Como resultado, a direção do vetor força de atração será na direção da meta.

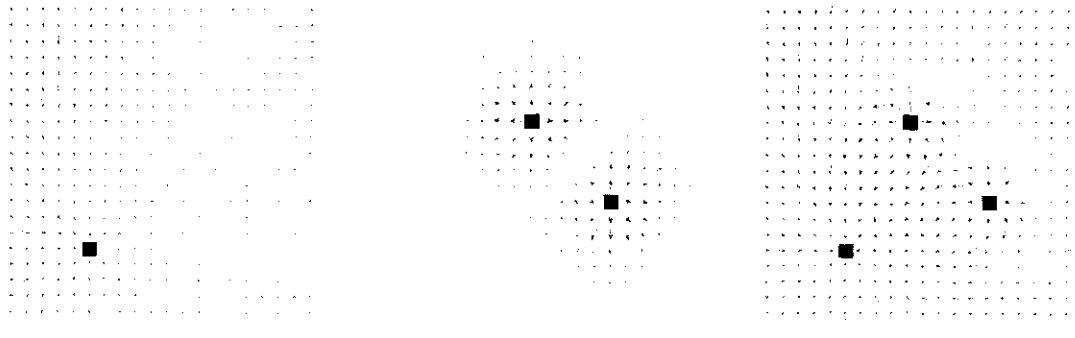
3.3.3 Força Resultante

Ao realizar a soma dos vetores força de atração e força de repulsão obtém-se a força resultante dada pela Equação 3.27. Este vetor de força é responsável por indicar a direção correta de navegação do robô, para que este seja capaz de navegar pelo ambiente desviando de possíveis obstáculos.

$$\vec{F} = \vec{F_R} + \vec{F_a} \tag{3.27}$$

Para ilustrar a superposição dos campos de atração e repulsão, é apresentado na Figura 3.19(a) o campo de atração gerado pela meta, na Figura 3.19(b) o campo de repulsão gerado por dois obstáculos e finalmente na Figura 3.19(c) mostra-se a soma dos campos de atração e de repulsão.

A técnica de Campos Potenciais, mesmo sendo muito utilizada para o planejamento de trajetórias de robôs móveis, demonstra algumas limitações.



(a) (b) (c)

Figura 3.19: (a) potencial atrativo da meta, (b) potencial repulsivo de dois obstáculos, (c) soma destes dois campos.

Para determinadas situações, quando a força resultante dos vetores de repulsão e de atração forem nulas, o robô irá parar. Uma outra possibilidade de falha é encontrada quando o robô está sendo atraído por uma meta que se encontra atrás de uma parede. Esses dois casos citados, geralmente são classificados como problemas de mínimos locais ocasionando uma falha que não pode ser contornada utilizando-se apenas a técnica de Campos Potenciais.

Os métodos Campo de Força Virtual ou VFF (*Virtual Force Field*) (Borenstein & Koren 1989) e Histograma de Campo Vetorial ou VFH (*Vector Field Histogram*) (Borenstein & Koren 1991) foram propostos por Borenstein & Koren (1989) e tem sido bastante utilizados em pesquisas na área de navegação de robôs móveis. Estes métodos diferenciam se dos acima apresentados pela utilização de um mapa inspirado na técnica de Grade de Ocupação (*Occupancy Grid*) (Elfes 1989) para representar o ambiente com seus obstáculos.

Essas técnicas são utilizadas para contornar os problemas gerados pela técnica de Campos Potenciais citada acima. Porém, o estudo desses métodos fogem ao escopo deste trabalho, considerando que nenhum mapa (*grid*) é tratado pelo sistema proposto.

Capítulo

4

O Sistema Computacional Desenvolvido

O Sistema de Visão Computacional (SVC) desenvolvido durante este projeto de mestrado foi constituído de três fases principais: a idealização do sistema, a implementação e os experimentos.

Durante a fase de idealização, optou-se por desenvolver um sistema de visão computacional que capacitasse um robô móvel a caminhar por um ambiente de características delimitadas e sendo capaz de distinguir determinadas cores presentes neste ambiente. Além disso, foi estabelecido que toda a parte de visão seria desenvolvida utilizando-se modelos de redes neuraias, mais especificamente o modelo MLP (Capítulo 3), o que facilitaria uma possível implementação deste sistema em FPGAs, considerando que o grupo já possui tal modelo implementado em hardware.

A fase de implementação e de experimentos tramitaram quase que em sua totalidade de forma paralela, no qual cada parte do sistema que estava sendo implementado era testado no ambiente em que o SVC atuaria.

Neste capítulo é apresentada uma descrição do sistema de visão desenvolvido. São detalhados os principais módulos que o compõe, como o módulo de Visão e o de Controle de Navegação. Também é apresentada uma breve descrição sobre o hardware robótico, a interface com o usuário e o ambiente Saphira.

4.1 Uma Descrição Geral do Sistema

Para o desenvolvimento deste projeto de mestrado diversos subsistemas foram implementados, desde um simples sistema para capacitar um robô a perseguir um objeto de cor determinada até um sistema mais complexo para capacitar o robô a navegar por um ambiente de forma autônoma desviando de obstáculos e sendo atraído por um objeto de cor determinada. Contudo, todos esses subsistemas implementados compartilham (em auto nível) das mesmas funcionalidades. Na

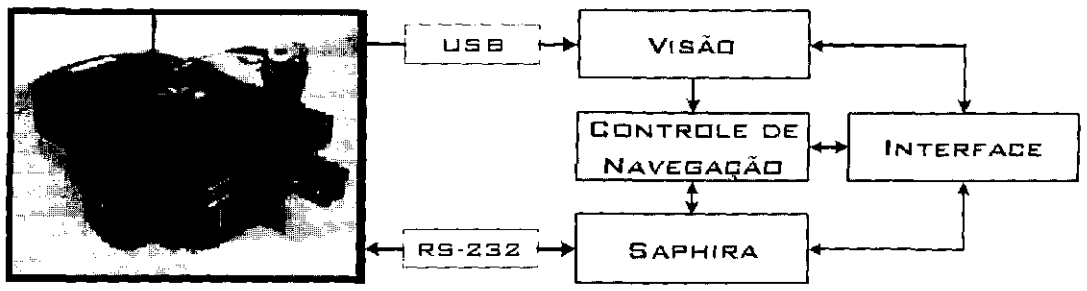


Figura 4.1: Arquitetura do sistema

Figura 4.1 é apresentada uma visão geral da arquitetura que compõe o sistema desenvolvido, no qual as setas indicam o fluxo de dados. A conexão RS232 é utilizada tanto para enviar dados ao robô (ex: configurar sua velocidade de translação) como para receber as informações das leituras realizadas pelos sonares. A interface USB foi utilizada para estabelecer a conexão entre o laptop e a câmera.

Todo o sistema implementado segue o paradigma reativo (Arkin 1998), no qual nenhuma informação prévia do ambiente é conhecida e nenhuma informação adquirida sobre o ambiente é armazenada na memória do robô. O paradigma reativo tem se mostrado bastante adequado quando aplicado ao controle de robôs móveis atuando em ambientes reais. Isso se deve ao fato de que todos os comportamentos que regem o controle do robô são relações claras e diretas de como este deve reagir sob determinadas condições (percepção) do ambiente. Além disso, uma vez definidos e implementados tais comportamentos reativos, estes podem servir como base para uma implementação de sistemas de controle baseados no paradigma híbrido (Arkin 1998). Isto possibilita o robô a realizar tarefas mais complexas, como por exemplo, explorar e criar um mapa do ambiente desviando de possíveis obstáculos.

Nas próximas Seções serão detalhados os módulos que compõe o SVC, bem como o hardware robótico utilizado e o ambiente Saphira.

4.2 O Hardware Robótico

Todo o desenvolvimento do sistema e experimentos foram realizados sob a plataforma do robô Pioneer I (Figura 4.1) da ActivMedia Robotics. Este robô é equipado com 7 sonares Polaroid 6500. Sua comunicação com dispositivos externos (como um computador) pode ser realizada através de rádio e da interface RS-232.

Para o desenvolvimento de um sistema de visão computacional foi necessário a aquisição e adequação de uma unidade de captura de imagens (câmera) ao robô, pois a única forma de percepção já disponível no hardware robótico são os sonares.

Para captura das imagens foi utilizada uma câmera WebCam Go Plus da Creative. Esta câmera foi posicionada na parte central do robô levemente direcionada para o chão. Dois níveis de inclinação foram utilizadados: no primeiro o alcance de captura foi configurado em $4.8m$ frontais ao robô e no segundo ajuste de inclinação o alcance foi de $1.65m$. O angulo de captura da câmera utilizada é em torno de $41°$.

4.3 O Ambiente Saphira

A comunicação entre o sistema desenvolvido e o robô Pioneer I é realizado através do ambiente Saphira.

O Saphira é uma arquitetura para o desenvolvimento de aplicações robóticas, desenvolvido e mantido pelo *Artificial Intelligence Center* do *Stanford Research Institute*. Ele possui uma biblioteca de rotinas que permite a construção de programas em linguagem C/C++ para o controle de robôs móveis como o Pioneer I, fornecendo uma abstração de alto nível do hardware robótico provendo informações sensoriais, como leituras de sonares e *encoders* e recebendo comandos motores como "vire 30 graus".

4.4 A Interface

A interface é responsável pela interação entre o usuário e o sistema. Nela estão disponíveis as funções necessárias para estabelecer a comunicação com o robô, seleção das cores para segmentação, seleção da forma do objeto (quando necessário). Além disso, também estão disponíveis parâmetros como velocidade máxima de translação e rotação.

4.5 Módulo de Visão

O módulo de visão é responsável por capturar, processar e enviar para os demais módulos do SVC as informações adquiridas e processadas do mundo real. De certa forma, o módulo de visão atua como um tradutor que recebe como entrada um *quadro* do ambiente (formado por um conjunto de pontos) e gera como saída uma interpretação de alto nível deste quadro, como por exemplo: objeto vermelho encontra-se na posição XY. De tal forma que os demais módulos possam, com base na informação passada pelo módulo de visão, gerar os comandos necessários para a navegação do robô.

O módulo de visão está dividido em três fases: fase de pré-processamento, fase de segmentação e fase de reconhecimento. Estas fases estão descritas abaixo.

4.5.1 Pré-Processamento

A primeira fase é responsável pela captura dos quadros (imagens) a partir da câmera à uma resolução de 320x240 pontos e pela redução dos mesmos à resolução de 80x60 pontos. A redução das imagens é realizada pela média dos pontos, no qual uma janela de 4X4 percorre a imagem capturada gerando um ponto com o valor médio dos 16 pontos presentes na janela.

A resolução de 80X60 pontos foi definida empiricamente com base em alguns testes realizados com imagens do ambiente real, sendo que esta foi a resolução mínima necessária para o reconhecimento dos objetos tratados por este projeto.

4.5.2 Segmentação

O processo de segmentação de imagens é realizado por um sistema de classificação de cores. O sistema classificador é composto por diversas redes neurais (Figura 4.2), no qual cada rede tem por objetivo classificar uma determinada cor, separando-a das outras cores presentes na imagem. Por exemplo, quando a cor vermelha for solicitada, cada ponto da imagem é classificado em vermelho e não vermelho (demais cores). A classificação é feita com base nos componentes RGB de cada ponto da imagem. Neste caso, a rede neural solicitada recebe os três componentes (RGB) e apresenta na saída da rede o resultado de classificação que pode assumir dois valores: classificado como a cor selecionada ou classificado como demais cores (Ex: cor vermelha e cores não vermelhas). Como o objetivo do processo de segmentação implementado neste trabalho é de isolar a cor de interesse, a imagem segmentada resultante é composta por apenas duas cores: cor de interesse (representada pela cor branca) e demais cores (representada pela cor preta).

Também foi implementado junto com o processo de segmentação, um filtro com o objetivo de eliminar os ruídos, como pontos brancos isolados, presentes nas imagens já segmentadas. Este filtro faz uma varredura pelo quadro com uma janela de tamanho 4X4, verificando a quantidade de pontos pretos e brancos presentes nesta janela. Quando a quantidade de pontos pretos é igual ou superior a quantidade de pontos brancos, todos os pontos da janela são setados como pretos, caso contrário, são setados como pontos brancos.

4.5.3 Reconhecimento

A terceira fase do módulo de visão tem por finalidade realizar o reconhecimento das imagens já segmentadas. Para isto, uma rede neural do tipo MLP recebe todos os pontos que constituem a imagem segmentada em suas entradas e informa como

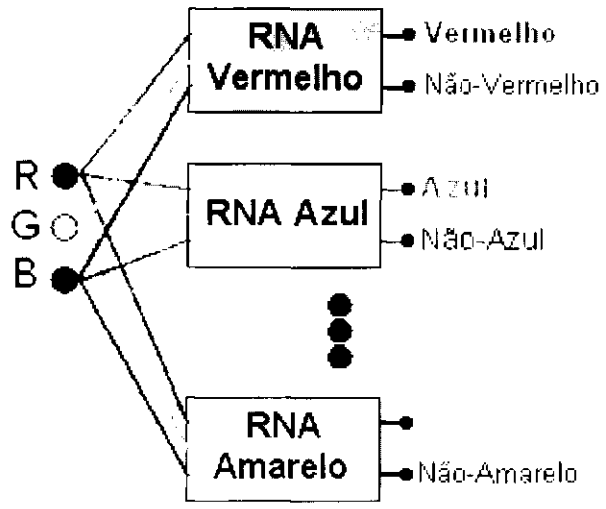


Figura 4.2: Sistema de classificação de cores composto por um conjunto de redes neurais do tipo MLP

saída o resultado analisado. Duas tarefas de reconhecimento foram testadas: reconhecimento da posição do objeto e reconhecimento da posição e forma do objeto.

Em relação à primeira tarefa, a rede MLP teve por finalidade identificar, com base em todos os pontos da imagem, qual o posicionamento do objeto no campo visual. Esta tarefa foi implementada seguindo duas metodologias diferenciadas entre sí pela forma de resposta apresentada na saída da rede neural. Foi utilizada a inclinação que garantiu um alcance de $4.8m$ a frente do robô (Seção 1.2).

Na primeira metodologia foi utilizada uma rede neural com 14 saídas onde as primeiras 13 saídas representam, respectivamente, cada um dos 13 blocos presentes na imagem como é mostrado na Figura 4.3. A última saída indica se o objeto está ou não presente no campo visual do robô. Desta forma, a saída da rede com o maior valor (≈ 1) indica em qual posição do campo visual o objeto se encontra com base nas áreas estabelecidas (Figura 4.3). Os 13 blocos que correspondem à imagem foram obtidos empiricamente com base em análises realizadas em algumas imagens com os objetos no ambiente real de trabalho onde o robô está inserido.

A segunda metodologia utilizada na primeira tarefa consiste na implementação de uma rede neural com apenas 3 saídas, em que as duas primeiras representam as respectivas as coordenadas X e Y do objeto na imagem e a terceira saída (P) indica se o objeto está ou não presente na imagem (Figura 4.5). As saídas X e Y foram discretizadas empiricamente com base em estudos realizados em campo, no qual as imagens foram rotuladas com base nos seguintes valores:

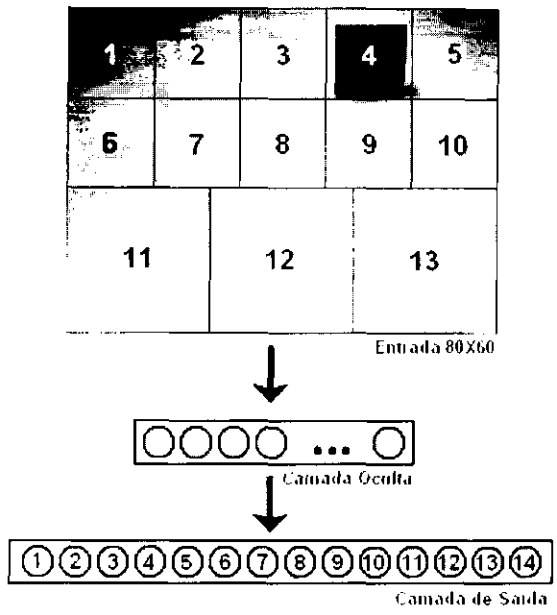


Figura 4.3: RNA para identificação da posição do objeto (1)

- X : 0.0, 0.1, 0.2, ... , 1.0
- Y : 0.0, 0.1, 0.2, ... , 1.0

onde X e Y representam os valores medidos com base na disposição do objeto no campo de visão como apresentado pela Figura 4.4. Contudo, nos testes realizados no ambiente real, com o auxílio de uma trena, observou-se que os valores: 0.0, 0.4, 0.7, 0.8, 0.85, 0.88, 0.9 e 0.91 para Y, representavam aproximadamente intervalos $0.5m$ no espaço real, isto é, quando Y assume 0.0, significa que o objeto está encostado ou muito próximo ao robô, quando Y assume 0.1 o objeto encontra-se a aproximadamente $0.5m$ do robô, e assim sucessivamente em incrementos de $0.5m$. Este sistema de estimativa do posicionamento real do objeto no campo visual do robô foi desenvolvido considerando a impossibilidade de se obter a posição 3D real do objeto sem o auxílio de uma Visão Estereoscópica, cujo desenvolvimento foge ao escopo deste trabalho de mestrado.

Vale ressaltar que as coordenadas X e Y não são as coordenadas X e Y do mundo real, mas sim a posição do objeto no quadro que está em análise.

A segunda tarefa (Figura 4.6), responsável pelo reconhecimento da posição e forma do objeto, consiste em uma rede neural do tipo MLP com 10 saídas. Cada objeto é representado por três saídas da rede. Da mesma forma que o sistema com 3 saídas descrito anteriormente, uma saída (P) informa se o objeto está ou não presente na imagem e em caso positivo as saídas X e Y informam as coordenadas do objeto no quadro. Esta análise é realizada para cada um dos três objetos:

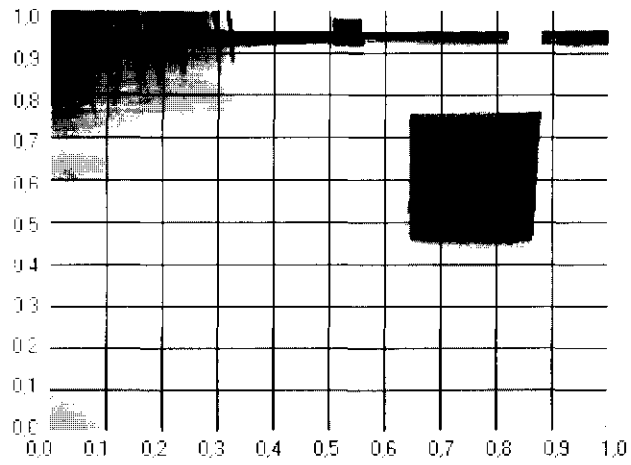


Figura 4.4: Discretização dos quadros capturados

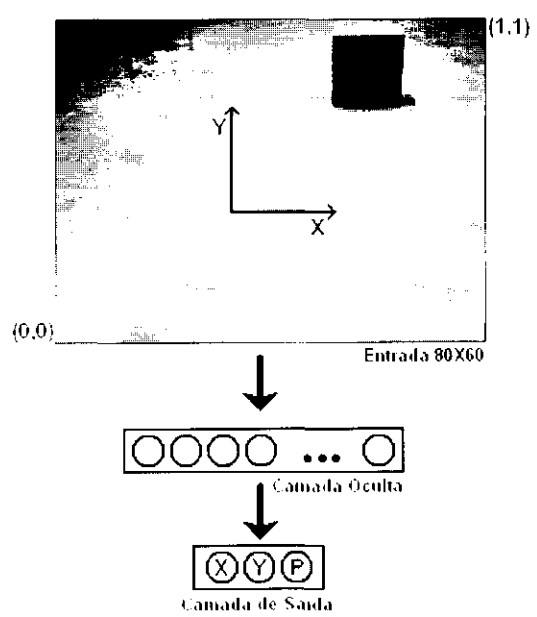


Figura 4.5: RNA para identificação da posição do objeto (2)

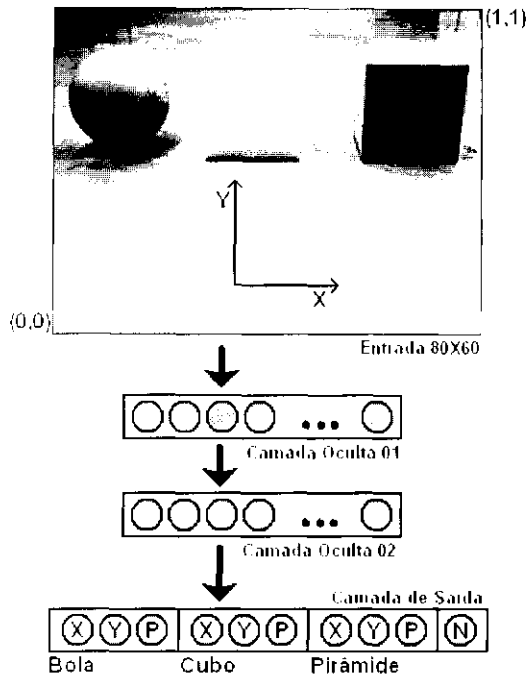


Figura 4.6: RNA para identificação da posição e da forma do objeto

para a bola, cubo e pirâmide, respectivamente (ver Figura 4.6). A última saída (N) é ativada quando não existe nenhum objeto no campo de visão do robô. Vale ressaltar que nenhum pré-processamento adicional é realizado na tarefa de identificação da forma do objeto, como por exemplo: centralização do objeto, rotação, escala, etc. Estabelecendo-se a responsabilidade de reconhecimento da forma e da posição do objeto apenas a rede neural.

Nesta tarefa foi utilizada a inclinação para um alcance de $1.65m$ a frente do robô (Seção 4.2), considerando que, com a qualidade da imagem que está sendo tratada, ficaria impossível reconhecer um objeto a distância maior que a definida ($1.65m$).

4.6 Módulo de Controle de Navegação

Com o objetivo de testar o módulo de visão criado, alguns sistemas de controle baseado em comportamento reativo foram desenvolvidos. Para cada uma das abordagens descrita acima um conjunto de comportamentos foram estabelecidos, de tal forma que o controlador desenvolvido capacitasse o robô a perseguir um objeto pelo chão de uma sala.

Um outro controlador baseado na técnica de campos potenciais também foi desenvolvido com o objetivo de capacitar um robô a navegar de forma autônoma por um ambiente, desviando de obstáculos como paredes e sendo atraído por uma

meta (objeto) de cor determinada.

4.6.1 Controlador I

O primeiro controlador desenvolvido foi baseado no módulo de visão implementado na primeira metodologia da primeira tarefa, onde as saídas da rede neural da fase de reconhecimento estavam discretizada em 13 regiões definidas (Figura 4.3). Este controlador é constituído de dois comportamentos:

COMPORTAMENTO 01: **Follow()**
SE OBJETO ESTÁ PRESENTE NO QUADRANTE Q ENTÃO
VELOCIDADE DE TRANSLAÇÃO = V
VELOCIDADE DE ROTAÇÃO = R

onde Q representa o quadrante onde o objeto está presente na imagem (saídas da rede neural: 1-13) (Figura 4.3), V é a velocidade de translação do Robô e R é a sua velocidade de rotação. A velocidade de translação no robô é configurada por alta, média e baixa. A velocidade de rotação é definida como "vire fortemente para a esquerda ou para a direita" ou "vire moderadamente para a esquerda ou para a direita" ou igual a zero, de acordo com o posicionamento do objeto:

- Translação Alta: quadrantes 1-5;
- Translação Média: quadrantes 6-10;
- Translação Baixa: quadrantes 11-13;
- Rotação à Esquerda Forte: quadrantes: 1 e 6;
- Rotação à Esquerda Moderada: quadrantes: 2, 7 e 11;
- Rotação à Direita Moderada: quadrantes: 4, 9 e 13;
- Rotação à Direita Forte: quadrantes: 5 e 10;
- Rotação Zero: quadrantes 3, 8 e 12.

O segundo comportamento, que toma a decisão de parar o robô quando o objeto não está presente no campo visual do robô, é dado por:

COMPORTAMENTO 02: **Stop()**
SE OBJETO NÃO ESTÁ VISÍVEL (SAÍDA 14=0) ENTÃO
VELOCIDADE DE TRANSLAÇÃO = 0

Velocidade de Rotação = 0

Os parâmetros: velocidade alta, média e baixa e vire fortemente e moderadamente foram configurados com base em experimentos realizados no ambiente real e representam:

- Velocidade Alta: $15cm/s$
- Velocidade Média: $10cm/s$
- Velocidade Baixa: $3cm/s$
- Vire Fortemente: $\pm 100graus/s$
- Vire Moderadamente: $\pm 50graus/s$

4.6.2 Controlador II

O segundo controlador foi implementado com base nos dados recebidos do sistema de visão desenvolvido utilizando a metodologia da primeira tarefa, na qual apenas 3 saídas na rede neural compõe a saída do módulo de visão (Figura 4.5). Diferenciando do controlador apresentado anteriormente, este recebe parâmetros não binários (X e Y) permitindo que diversos níveis de velocidade sejam configurados no robô. Dois comportamentos foram implementados neste controlador. O primeiro deles é dado pela seguinte regra:

Comportamento 01: **Follow()**
SE objeto está presente na imagem (P=1) ENTÃO
Velocidade de Translação = C1*Y
Velocidade de Rotação = (2*C2)*(0.5-X)

e o segundo é dado pela regra:

Comportamento 02: **Stop()**
SE objeto não está visível (P=0) ENTÃO
Velocidade de Translação = 0
Velocidade de Rotação = 0

onde C1 e C2 são constantes empíricas que definem as velocidades máximas ($15cm/s$) de translação e rotação ($100graus/s$), respectivamente, e X, Y e P são as saídas do módulo de visão (fase de reconhecimento).

4.6.3 Controlador III

O terceiro controlador, responsável pelo reconhecimento da posição e identificação da forma o objeto, foi implementado utilizando a segunda abordagem desenvolvida no módulo de visão (Figura 4.6). Neste controlador o usuário deve informar qual o objeto que deve ser perseguido (bola, cubo ou pirâmide). Considerando a variável OBJ como sendo a opção do usuário e, podendo assumir um de três valores 1,2,3, onde 1 representa o objeto bola, 2 o objeto cubo e 3 representa uma pirâmide. Quatro comportamentos foram implementados:

COMPORTAMENTO 01: **Stop()**
SE NENHUM OBJETO ESTÁ PRESENTE NA IMAGEM (N=1) ENTÃO
VELOCIDADE DE TRANSLAÇÃO = 0
VELOCIDADE DE ROTAÇÃO = 0

COMPORTAMENTO 02: **FollowBall()**
SE (OBJ = 1) E (P[BOLA] = 1) ENTÃO
VELOCIDADE DE TRANSLAÇÃO = C1*Y[BOLA]
VELOCIDADE DE ROTAÇÃO = (2*C2)*(0.5-X[BOLA])
SENÃO
VELOCIDADE DE TRANSLAÇÃO = 0
VELOCIDADE DE ROTAÇÃO = 0

COMPORTAMENTO 03: **FollowCube()**
SE (OBJ = 2) E (P[CUBO] = 1) ENTÃO
VELOCIDADE DE TRANSLAÇÃO = C1*Y[CUBO]
VELOCIDADE DE ROTAÇÃO = (2*C2)*(0.5-X[CUBO])
SENÃO
VELOCIDADE DE TRANSLAÇÃO = 0
VELOCIDADE DE ROTAÇÃO = 0

COMPORTAMENTO 04: **FollowPyramid()**
SE (OBJ = 3) E (P[PIRAMIDE] = 1) ENTÃO
VELOCIDADE DE TRANSLAÇÃO = C1*Y[PIRAMIDE]
VELOCIDADE DE ROTAÇÃO = (2*C2)*(0.5-X[PIRAMIDE])

SENÃO

VELOCIDADE DE TRANSLAÇÃO = 0

VELOCIDADE DE ROTAÇÃO = 0

onde as variáveis X[Objeto], Y[Objeto] e P[Objeto] representam as saídas da rede neural X, Y e P, respectivamente para cada um dos objetos reconhecidos pelo módulo de visão. Quando P[Objeto] está ativo (P[Objeto]=1), significa que o objeto de interesse está presente no campo visual. C1 e C2 são constantes empíricas que definem as velocidades máximas de translação ($10cm/s$) e rotação ($60graus/s$), respectivamente.

4.6.4 Controlador IV - Aplicação de Campos Potenciais

Este último controlador foi desenvolvido com o objetivo de capacitar o robô a navegar pelo ambiente desviando de possíveis obstáculos e buscando um objeto de uma dada cor. Ele se diferencia dos controladores apresentados anteriormente que apenas seguiam o objeto de interesse. Com este controlador o robô é capaz de permanecer navegando pelo ambiente de forma aleatória, buscando o objeto da cor requisitada. Caso visualize o objeto (cor) de interesse, caminha em sua direção.

Para a implementação deste controlador, três comportamentos foram desenvolvidos: GoAhead(), Follow() e AvoidCollision().

A técnica de Campos Potenciais (Seção 3.3) foi utilizada para gerar a trajetória que o robô deve seguir com base nas informações geradas pelos três comportamentos.

O primeiro comportamento, GoAhead(), tem por finalidade manter o robô em movimento constante para frente. O segundo comportamento, Follow(), é responsável por conduzir o robô em direção ao alvo utilizando o sistema de visão. O último comportamento, AvoidCollision(), encarrega-se de direcionar o robô no sentido contrário ao de possíveis obstáculos, por exemplo uma parede.

Cada um dos comportamentos citados possuem como forma de resposta uma saída vetorial de duas dimensões, representando o ângulo de rotação (direção - θ_F) e magnitude ($|\overrightarrow{F}|$).

O comportamento GoAhead() não possui nenhuma forma de percepção do mundo, considerando que seu único objetivo é manter o robô navegando pelo ambiente, mesmo quando não estiver sendo atraído por nenhum objeto. Assim, um vetor com direção e magnitude constante (C_g) pode ser utilizado para representar o comportamento (campo uniforme).

$$|\vec{F_g}| = C_g \tag{4.1}$$

$$\theta_{\vec{F_g}} = 0 \tag{4.2}$$

O comportamento Follow(), responsável por conduzir o robô em direção ao alvo, possui como percepção do mundo o sistema de visão. Mais especificamente, foi utilizada a segunda metodologia correspondente a realização primeira tarefa da fase de reconhecimento do módulo de visão, na qual a rede MLP informa como saída as posições X e Y do objeto, quando este está no campo visual ($P = 1$).

Uma vez que a rede neural informa na saída as posições X e Y e como descrito anteriormente Y foi discretizada com valores que aproximam medidas de incremento de $0.5m$, no espaço real (3D), é possível calcular um valor aproximado para X_{3D} e Y_{3D} no espaço real através de uma função definida por:

$$FuncaoEspaco(X,Y) = \begin{cases} ([(0.5-X)*200|,[0000]), & \text{se } Y \approx 0.0 \\ ([(0.5-X)*400|,[0500]), & \text{se } Y \approx 0.4 \\ ([(0.5-X)*750|,[1000]), & \text{se } Y \approx 0.7 \\ ([(0.5-X)*1024|,[1500]), & \text{se } Y \approx 0.8 \\ ([(0.5-X)*1500|,[2000]), & \text{se } Y \approx 0.85 \\ ([(0.5-X)*1875|,[2500]), & \text{se } Y \approx 0.88 \\ ([(0.5-X)*2250|,[3000]), & \text{se } Y \approx 0.9 \\ ([(0.5-X)*2625|,[3500]), & \text{se } Y \approx 0.91 \end{cases} \tag{4.3}$$

onde esta função recebe como parâmetro os valores X e Y, obtidos a partir da rede neural e retorna $([X_{3D}|,[Y_{3D}])$ (expressos em mm) como valores aproximados do posicionamento do objeto no ambiente real.

Uma vez obtidas as posições X_{3D} e Y_{3D} pode-se gerar o vetor de saída do comportamento Follow().

Semelhante ao comportamento GoAhead(), o comportamento Follow() possui magnitude constante (C_a). Porém, ele possui um ângulo de rotação que indica quanto o robô deve girar para direcionar sua navegação rumo ao alvo (força de atração - campo uniforme):

$$|\vec{F_a}| = C_a \tag{4.4}$$

$$\theta_{\overrightarrow{F_a}} = \arctan(X_{3D}, Y_{3D}) \tag{4.5}$$

O comportamento AvoidCollision() é responsável por afastar o robô de possíveis obstáculos. A percepção do mundo para este comportamento é obtida através dos sonares do robô. Considerando que o robô está equipado com 7 sonares, cada um gerando um vetor distinto (Forças de Repulsão), a resposta do comportamento AvoidCollision() será uma composição de 7 vetores.

A força de repulsão do comportamento AvoidCollision() é gerada por um campo radial repulsivo cuja função de decaimento adotada foi:

$$|\overrightarrow{F_{ri}}| = \exp((-D + L)/T) \tag{4.6}$$

onde D é a distância entre o obstáculo e o robô, L é o limite de proximidade (quando a distância D for igual a L o módulo do vetor atinge o valor 1) e T é a constante que define o grau de decaimento da função exponencial. A direção do vetor de repulsão é dada por:

$$\theta_{\overrightarrow{F_{ri}}} = \arctan(-XS_i, -YS_i) \tag{4.7}$$

onde XS_i e YS_i são as coordenadas X e Y, expressas em *mm*, fornecidas por cada sonar i, i=1.....7, presente no robô. O sinal $-$ é utilizado para se inverter o θ do vetor $\overrightarrow{F_{ri}}$, considerando que o objetivo deste comportamento é afastar o robô do alvo detectado.

Para o cálculo da força de repulsão resultante para os 7 sonares, deve-se fazer o somatório dos vetores das forças de repulsão geradas, como mostrado na Equação 4.8.

$$\overrightarrow{F_R} = \sum_i \overrightarrow{F_{ri}} \tag{4.8}$$

Uma vez calculada a força para cada um dos comportamentos definidos, a força resultante (trajetória do robô) é definida por:

$$\overrightarrow{F} = P_1\overrightarrow{F_G} + P_2\overrightarrow{F_A} + P_3\overrightarrow{F_R} \tag{4.9}$$

onde os P_i são os pesos atribuídos a cada uma das forças envolvidas no sistema. Durante os experimentos (Apresentados no Capítulo 5) os valores de P_i foram definidor empiricamente por:

- $P_1 = 0.5$

- $P_1 = 1.0$
- $P_2 = 0.7$

Após calculado a força resultante da integração dos comportamentos o robô deve navegar baseando-se nos seguintes valores:

$$VelT = |\overrightarrow{F}| \, VMax \tag{4.10}$$

e,

$$\theta_{\overrightarrow{F}} = \arctan(F_X, F_Y) \tag{4.11}$$

onde $VelT$ é a velocidade de translação, definida pelo módulo do vetor Força e a direção do robô é dado pela $\arctan(F_X, F_Y)$ das componentes F_X e F_Y do vetor Força. Para $VelT$ foi definido durante os experimentos um valor de $15cm/s$

Esta técnica mostrou ser bastante promissora quando combinada com o módulo de visão desenvolvido. Os experimentos e os resultados obtidos em ambientes reais serão apresentados no próximo capítulo.

Capítulo

5

Experimentos

Neste capítulo são apresentados os resultados obtidos durante todo o processo de desenvolvimento deste trabalho. Na primeira Seção são apresentados os resultados do processo de ajustes do sistema e na segunda Seção estão descritos os resultados obtidos a partir de experimentos com o robô Pioneer I (Seção 1.2) em um ambiente real.

5.1 Definição dos Parâmetros do Sistema

Nesta primeira seção estão descritos os experimentos realizados e os resultados obtidos durante a fase de ajustes do sistema. Mais especificamente, os resultados referentes ao processo de aprendizagem das RNAs utilizadas: segmentação, reconhecimento da posição e reconhecimento da posição e forma do objeto. Também é apresentado o processo evolutivo utilizado na obtenção da topologia da rede MLP responsável pela tarefa de reconhecimento da forma e posicionamento do objeto.

Para o treinamento das RNAs dois algoritmos foram utilizados: o algoritmo de retroprogação do erro (Seção 3.1.5) e o algoritmo Rprop (Seção 3.1.6). Para o algoritmo de retropropagação duas metodologias foram implementadas: a implementação padrão[1] e a implementação em lote[2].

Para todas as bases de dados do SVC, diversos testes, que serão apresentados abaixo, foram realizados com o objetivo de encontrar os parâmetros adequados para o treinamento das RNAs. Entre esses parâmetros encontram-se: número de camadas intermediárias, número de neurônios em cada camada intermediária, taxa de aprendizagem (η) e coeficiente do termo *momentum* (α), para o algoritmo

[1] do Inglês *on line*
[2] do Inglês *batch*

de retropropagação, e as constantes Δ_{max} e Δ_{min} e as taxas η^{+} e η^{-}, para o algoritmo Rprop.

Durante esses testes foi observado que os valores:

- $\eta = 0.2$
- $\alpha = 0.8$
- $\eta^{+} = 1.2$
- $\eta^{-} = 0.5$
- $\Delta_{max} = 10.0$
- $\Delta_{min} = 0.00001$

se mostraram *adequados* para o treinamento de todas as bases de dados do sistema. Sendo assim, todos os resultados referentes ao desempenho das RNAs serão descritos com base na utilização desses valores.

Nos testes de precisão das RNAs foi utilizada a técnica de Validação Cruzada *n-folds* (Rezende 2003), que consiste na divisão da base de dados em n partes das quais $n - 1$ partes são utilizadas no processo de treinamento e uma parte é utilizada na fase de validação. Esta técnica é repetida n vezes sempre considerando uma parte diferente para a validação.

No presente projeto, a base de dados foi divida em 10 partes [3].

O critério de parada utilizada nos algoritmos foi a convergência do erro quadrático médio para o conjunto de treinamento a um limiar inferior a 10^{-2}.

5.1.1 RNA para Segmentação

Para o treinamento das redes neurais da fase de segmentação foram montadas bases de dados contendo aproximadamente 100 exemplos em cada base. Por exemplo, para o treinamento da rede responsável pela cor vermelha, uma base de dados com 50 exemplos de pontos vermelhos e 50 pontos não vermelhos foram rotulados. Três cores foram consideradas para a tarefa de segmentação: vermelha, azul e amarela.

Com o objetivo de se encontrar uma topologia adequada para as rede neurais alguns testes dividindo se a base de dados em conjunto de treinamento (80%) e conjunto de validação (20%) foram realizados. Nestes testes foram observados que a topologia 3X3X2, isto é, uma rede constituída de três neurônios na camada de entrada (RGB), 3 neurônios na camada oculta (intermediária) e dois neurônios

[3] do Inglês *10-fold Cross Validation*

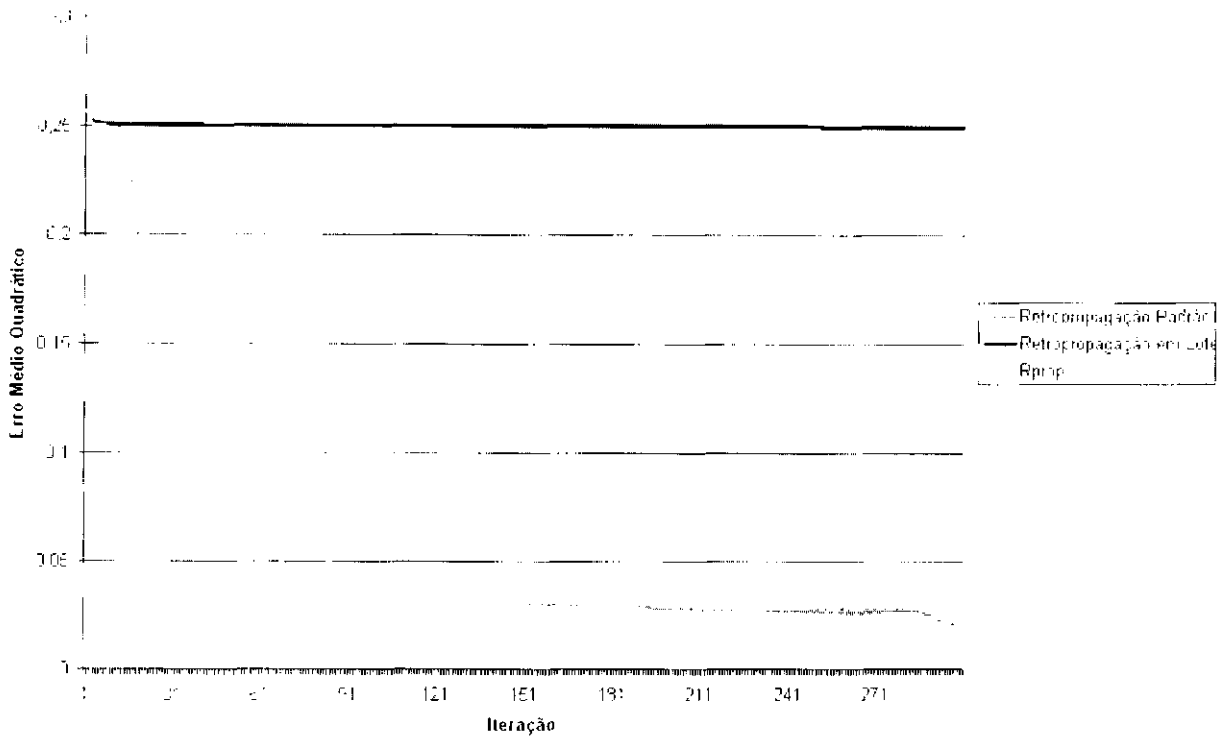


Figura 5.1: Treinamento da rede MLP para a segmentação da cor vermelha

na camada de saída (responsáveis por indicar se a entrada RGB pertence ou não a determinada cor), foi adequada para as cores azul e vermelha. Para a cor amarela, a topologia 3X5X2 foi a mais indicada.

Nas Figuras 5.1, 5.2 e 5.3 podem ser observadas a evolução do processo de aprendizagem utilizando-se os algoritmos Rprop, retropropagação em lote e retropropagação padrão para as cores vermelha, azul e amarela, respectivamente, considerando a topologia 3X3X2 para as cores vermelha e azul e a topologia 3X5X2 para a cor amarela.

O gráfico demonstra claramente que o algoritmo de Retropropagação em Lote apresenta uma convergência bem mais lenta que os demais algoritmos, necessitando desta forma um número maior de iterações (épocas) para que a rede aprenda a tarefa solicitada. Também pode ser observado que a convergência do processo de aprendizagem do algoritmo Rprop e do algoritmo retropropagação padrão foram semelhantes considerando o número de iterações necessárias.

Pelo fato do algoritmo de retropropagação em lote apresentar uma convergência mais lenta (número de iterações) que os demais algoritmos citados, foram realizados experimentos utilizando-se um número maior de iterações, pois, o número máximo de iterações adotado anteriormente não foi suficiente para sua

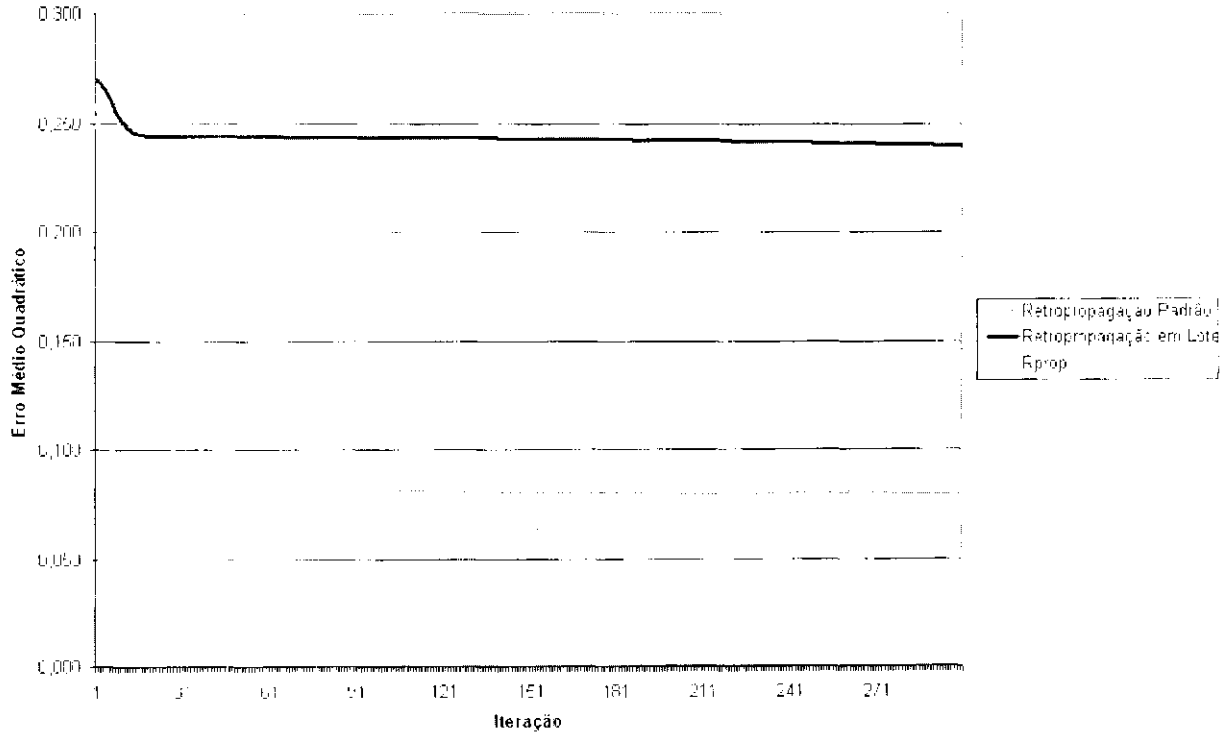


Figura 5.2: Treinamento da rede MLP para a segmentação da cor azul

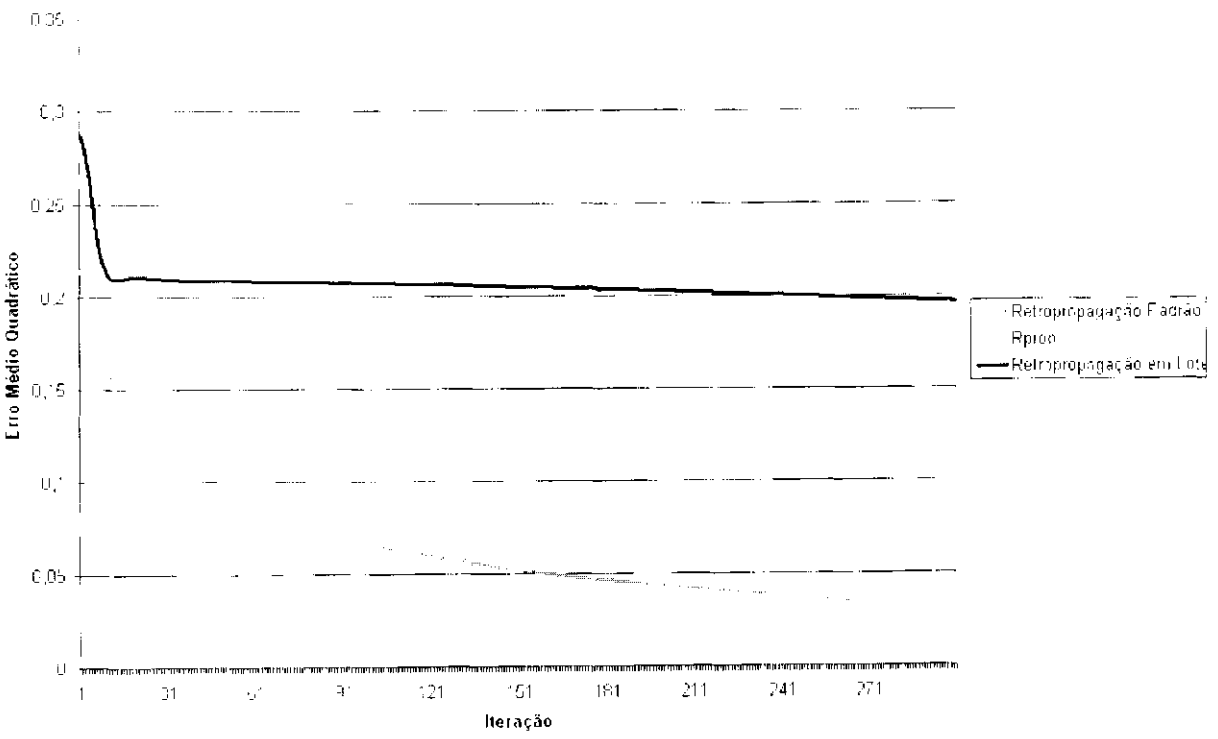


Figura 5.3: Treinamento da rede MLP para a segmentação da cor amarela

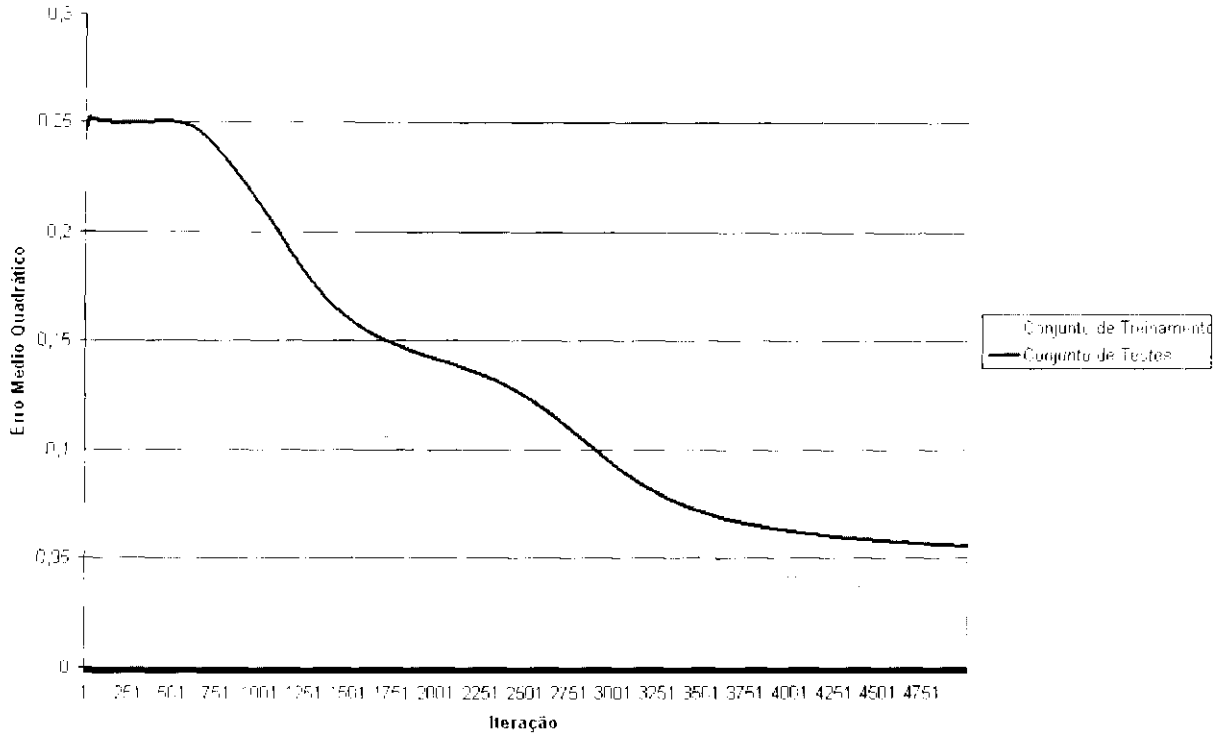


Figura 5.4: Treinamento da rede MLP para a segmentação da cor vermelha - algoritmo de retropropagação em lote

convergência. Esses experimentos podem ser visualizados nas Figuras 5.4, 5.5 e 5.6, respectivamente, para as cores vermelha, azul e amarela.

Nesses três gráficos são apresentados o processo de convergência considerando o erro quadrático médio para o conjunto de treinamento e o erro quadrático médio para o conjunto de validação. Uma clara diferença que pode ser observada entre o algoritmo de retropropagação em lote e os demais algoritmos está na suavidade da curva do processo de aprendizagem. Isto se deve ao fato de que o o gradiente obtido pelo processamento em lote é uma estimativa mais aproximada do gradiente real na superfície de erro do problema, resultando em menos oscilações, ou melhor, em uma busca menos estocástica.

Após o ajuste de uma topologia adequada ao problema, foram realizados testes utilizando-se a técnica Validação Cruzada com divisão do conjunto em 10 partes (*10-Fold Cross-Validation*). As Tabelas 5.1, 5.2 e 5.3 apresentam, respectivamente, os resultados obtidos a partir do treinamento das redes neurais MLP, utilizando-se os algoritmos: Rprop, Retropropagação em Lote (BP em Lote) e o algoritmo de Retropropagação Padrão (BP Padrão). Nestas tabelas, são apresentados o número médio de iterações necessários a convergência do processo de aprendizagem, a média do erro quadrático médio para o conjunto de validação

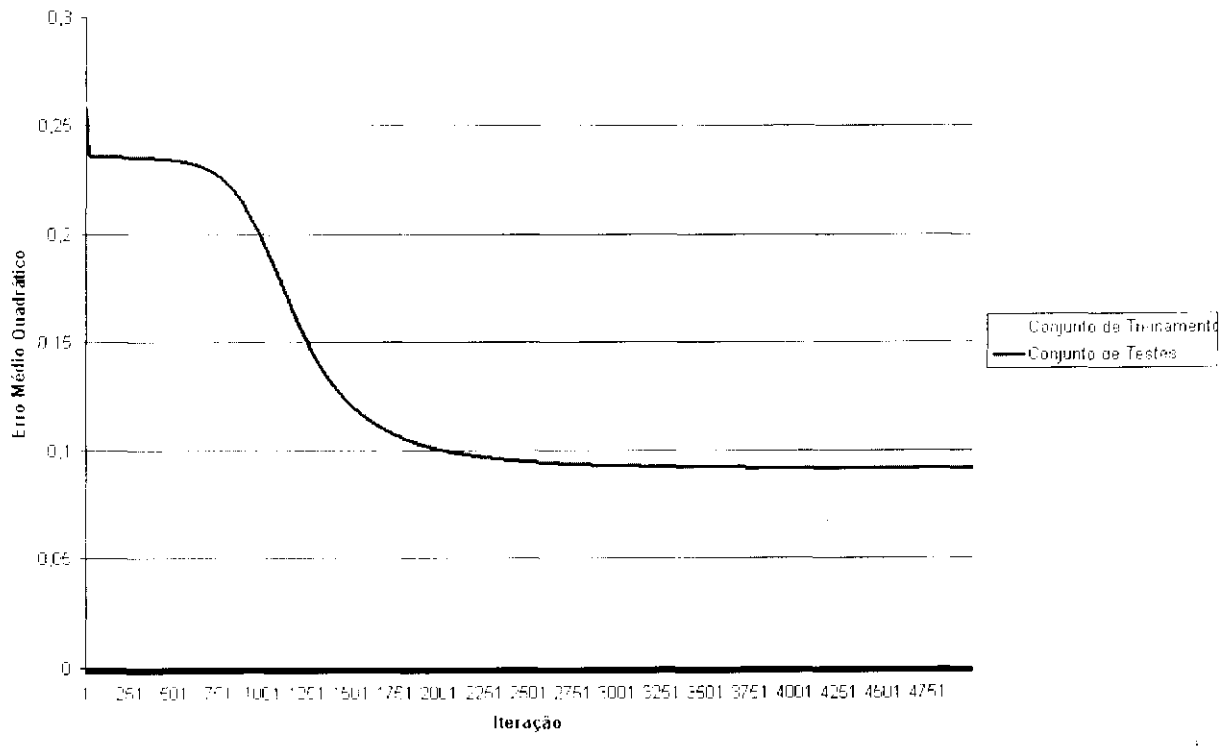


Figura 5.5: Treinamento da rede MLP para a segmentação da cor azul - algoritmo de retropropagação em lote

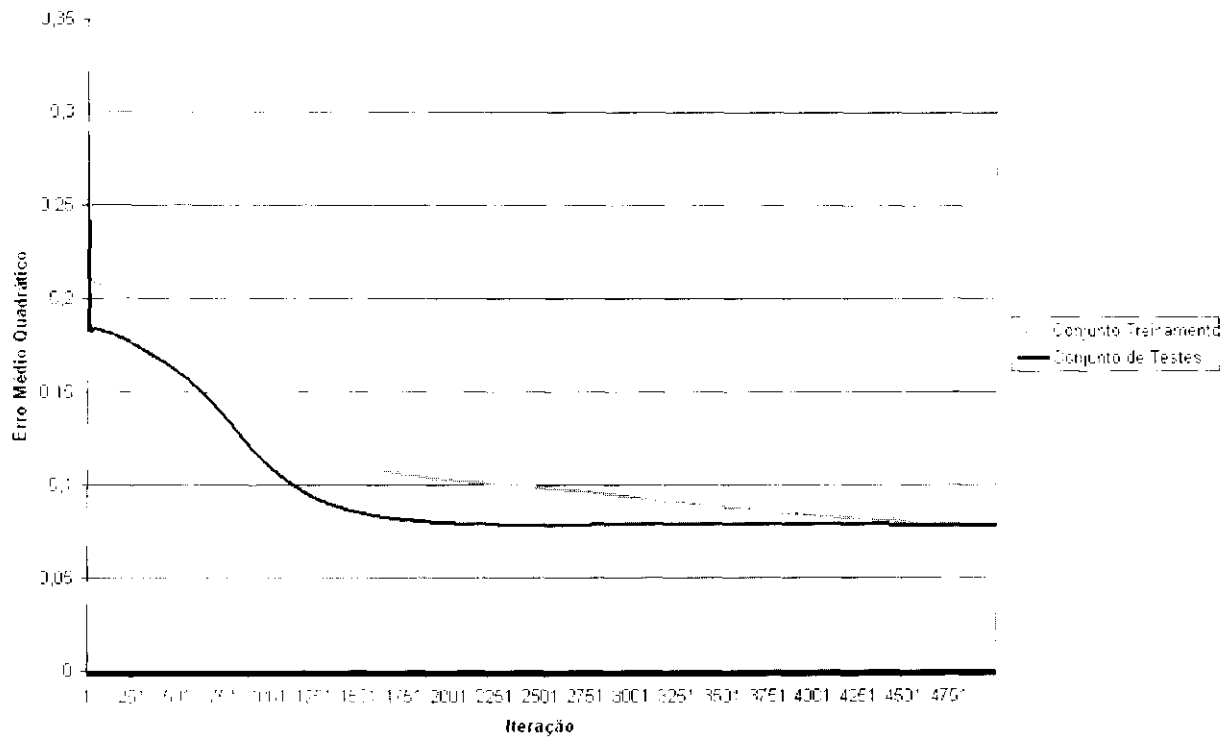


Figura 5.6: Treinamento da rede MLP para a segmentação da cor amarela - algoritmo de retropropagação em lote

(EQM) e o desvio padrão do erro quadrático médio para as 10 execuções dos algoritmos. Pode ser observado nestas tabelas que os resultados considerando a precisão do classificador neural, treinado com os algoritmos citados, foi praticamente a mesma, diferindo apenas no número de iterações necessárias à convergência da rede MLP.

Com relação ao tempo computacional gasto no treinamento, também não houve diferença significativa entre os algoritmos. Isto se deve ao fato de tanto a base de dados quanto a topologia das redes neurais serem pequenas.

Nos testes realizados em um computador *Pentium* IV 2.26GHz o tempo de segmentação para cada quadro 80X60 pontos foi de 0.0125s para as cores azul e

Tabela 5.1: Resultado do Processo de Treinamento da Rede Neural de Segmentação para a Cor Vermelha

	Iterações	EQM	Desvio Padrão
Rprop	298	0,039053	0.03100381
BP em Lote	5000	0.069356	0.02955651
BP Padrão	477	0.044432	0.05420535

Tabela 5.2: Resultado do Processo de Treinamento da Rede Neural de Segmentação para a Cor Azul

	Iterações	EQM	Desvio Padrão
Rprop	300	0.085364	0.05286000
BP em Lote	5000	0.086744	0.05250048
BP Padrão	513	0.083830	0.06681052

Tabela 5.3: Resultado do Processo de Treinamento da Rede Neural de Segmentação para a Cor Amarela

	Iterações	EQM	Desvio Padrão
Rprop	300	0.066133	0.04166800
BP em Lote	5000	0.099955	0.03892008
BP Padrão	491	0.067659	0.04531617

vermelha e de $0.0187s$ para a cor amarela. Com o objetivo de melhorar o tempo de resposta da fase de segmentação, também foi testada uma função linear por partes (Figura 5.7 - Equação 5.1) com o objetivo de aproximar a função de ativação dos neurônios (função logística) da rede MLP, reduzindo o tempo computacional gasto no cálculo desta função. Esta função foi definida empiricamente, selecionando alguns pontos da função logística e traçando retas entre esses pontos selecionados.

$$SigmoideLinear(x) = \begin{cases} 0, & \text{se } x < -6 \\ 0.054 + 0.009 * x, & \text{se } x \geq -6 \text{ e } x < -4 \\ 0.222 + 0.051 * x, & \text{se } x \geq -4 \text{ e } x < -2 \\ 0.420 - 0.150 * x, & \text{se } x \geq -2 \text{ e } x < -1 \\ 0.500 + 0.230 * x, & \text{se } x \geq -1 \text{ e } x < 1 \\ 0.580 + 0.150 * x, & \text{se } x \geq 1 \text{ e } x < 2 \\ 0.778 + 0.051 * x, & \text{se } x \geq 2 \text{ e } x < 4 \\ 0.946 - 0.009 * x, & \text{se } x \geq 4 \text{ e } x < 6 \\ 1, & \text{se } x \geq 6 \end{cases} \quad (5.1)$$

Os resultados obtidos com a função linearizada por partes foram de $0.0062s$ para a segmentação dos quadros para as cores vermelha e azul e de $0.0125s$ para a cor amarela. Isto demonstra a vantagem de se aproximar a função sigmóide por uma função linearizada com um custo computacional reduzido.

A qualidade do processo de segmentação não foi visualmente prejudicada

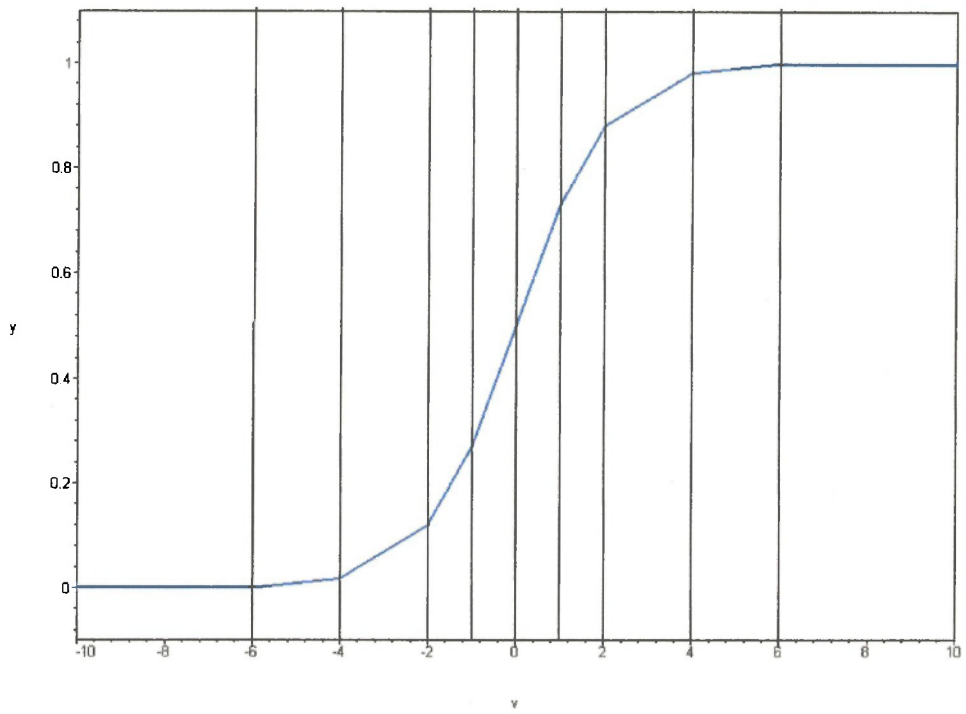


Figura 5.7: Função sigmóide linearizada

quando se utilizou a função linearizada por partes. As Figuras 5.8, 5.9 e 5.10 apresentam o resultado do processo de segmentação para as cores vermelha, azul e amarela respectivamente. Vale observar que esses resultados apresentados por estas Figuras são os mesmos tanto utilizando a função logística, quanto utilizando a função linearizada por partes.

5.1.2 RNA para Reconhecimento da Posição do Objeto

A fase de reconhecimento da posição do objeto, como citado na Seção 4.5.3, foi dividida em duas metodologias: a primeira utilizando uma rede MLP com 14 saídas e a segunda metodologia utilizando uma MLP com 3 saídas.

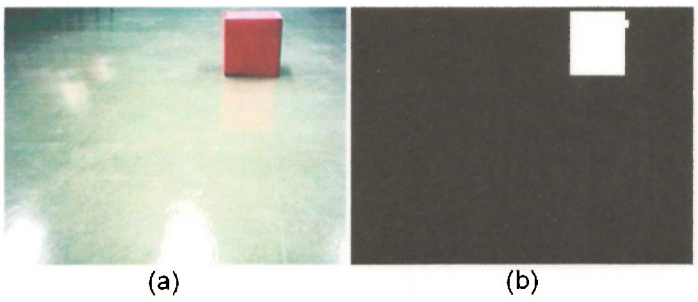


Figura 5.8: Resultado do processo de segmentação por classificação de cores utilizando a representação de cores RGB. (a) Imagem original, (b) Imagem pré-processada e segmentada para a cor vermelha.

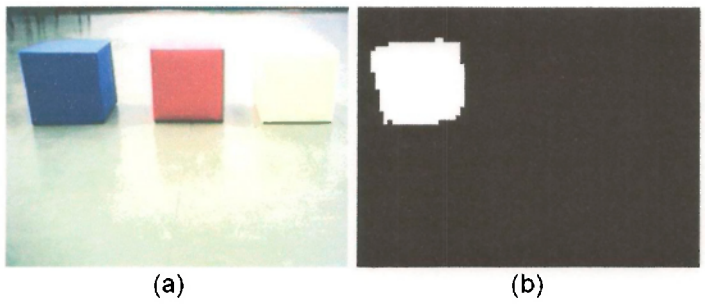


Figura 5.9: Resultado do processo de segmentação por classificação de cores utilizando a representação de cores RGB. (a) Imagem original, (b) Imagem pré-processada e segmentada para a cor azul.

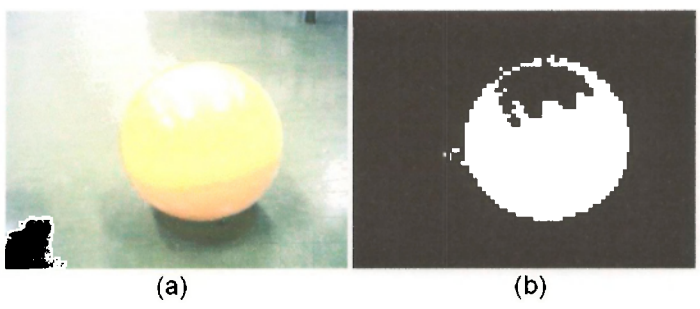


Figura 5.10: Resultado do processo de segmentação por classificação de cores utilizando a representação de cores RGB. (a) Imagem original, (b) Imagem pré-processada e segmentada para a cor amarela.

Para o treinamento dessas redes MLPs, duas bases de dados foram geradas com 700 padrões em cada base. Cada padrão da base é composto por um quadro pré-processado e segmentado (imagem binarizada de 80X60 pontos) e seu respectivo rótulo contendo a posição do objeto no campo visual. A única diferença entre as duas bases está no rótulo de cada padrão, pois para a primeira base, utilizada na primeira metodologia, a imagem foi rotulada segundo as 14 saídas possíveis da rede MLP e na segunda base a rotulação foi realizada com base na rede MLP com 3 saídas, ambas descritas na Seção 4.5.3.

Da mesma forma que foi apresentado na Seção 5.1.1, para se estabelecer a topologia dessas redes neurais responsáveis pelo reconhecimento da posição, testes foram feitos utilizando a técnica de validação cruzada, isto é, dividindo-se a base de dados em duas partes: 80% treinamento e 20% validados.

Os algoritmos de retropropagação do erro em lote e padrão e o algoritmo Rprop foram utilizados nos experimentos onde as topologias 4800X10X14 e 4800X10X3 se mostraram adequadas para toda a série de testes realizados. A topologia 4800X10XN significa que foi considerada uma rede constituída por 4800 neurônios de entrada (os 4800 pontos dos quadros 80X60), 10 neurônios na camada oculta e N o número de neurônios na camada de saída, sendo 14 para a primeira metodologia e 3 para a segunda metodologia.

Nos testes realizados foi observado que para ambas as redes neurais MLPs e com qualquer um dos algoritmos utilizados a rede convergiu facilmente para a solução, sendo que apenas o algoritmo de retropropagação em lote exigiu um pouco mais de tempo computacional no processo de aprendizagem. Além disso, durante os testes também foi constatado que uma rede perceptron com uma única camada seria suficiente para resolver a tarefa.

Contudo, uma rede MLP com uma camada oculta (citada anteriormente) foi utilizada por apresentar uma maior precisão na realização da tarefa de reconhecimento.

As Tabelas 5.4 e 5.5 apresentam os resultados da aplicação da técnica de Validação Cruzada (*10-folds*) para as redes neurais da primeira e da segunda metodologia, respectivamente, considerando-se as redes com 10 neurônios na camada oculta. Nestas tabelas constam o número médio de iterações, a média do erro quadrático médio para cada rede neural a partir da aplicação dos algoritmos citados e o desvio padrão obtido.

5.1.3 RNA para Reconhecimento da Posição e Forma do Objeto

Para o treinamento desta rede neural, responsável pelo reconhecimento da posição e da forma dos objetos (bola, cubo e pirâmide), uma base com 2861

Tabela 5.4: Resultado do Processo de Treinamento para a Rede Neural de Reconhecimento da Posição do Objeto - Metodologia 01

	Iterações	EQM	Desvio Padrão
Rprop	18	0.011675	0.003698
BP em Lote	1529	0.014443	0.006344
BP Padrão	6	0.010448	0.003499

Tabela 5.5: Resultado do Processo de Treinamento para a Rede Neural de Reconhecimento da Posição do Objeto - Metodologia 02

	Iterações	EQM	Desvio Padrão
Rprop	6	0.004809	0.000883
BP em Lote	503	0.012650	0.003577
BP Padrão	2	0.005850	0.001403

imagens (padrões) foi criada, sendo constituída de 700 imagens do objeto bola, 726 imagens do objeto cubo, 732 imagens do objeto pirâmide, todos capturados na tentativa de cobrir todo campo visual do robô além de 703 imagens não contendo nenhum dos objetos.

Vale ressaltar que essa base de dados é constituída dos valores dos 4800 pontos (80X60) resultantes do processo de pré-processamento e de segmentação das imagens adquiridas do ambiente seguido pelos seus respectivos rótulos. A rotulação dessas imagens foi gerada manualmente com o auxílio de um sistema de rotulação desenvolvido, no qual cada rótulo foi gerado com um vetor de 10 componentes, correspondentes as 10 saídas da rede neural MLP responsável pela realização desta tarefa (Seção 4.5.3).

Novamente como realizado para as redes neurais descritas anteriormente (segmentação e reconhecimento da posição do objeto), esta base de dados foi divida em dois conjuntos: 80% conjunto de treinamento e 20% conjunto de validação. Esta divisão foi realizada com a finalidade de se estabelecer uma topologia adequada para a realização da tarefa. Entretanto, mesmo com os diversos testes realizados considerando diversas topologias compostas por uma ou mais camadas ocultas, o erro quadrático médio para o conjunto de validação foi sempre superior a 0.11, no qual a melhor topologia obtida através de diversos testes empíricos foi a 4800X150X150X10.

Devido a dificuldade de se estabelecer uma melhor topologia para a rede neural responsável pelo reconhecimento da posição e da forma dos objetos, optou-se pela

implementação de um Algoritmo Evolutivo (Seção 3.2) que realizasse tal busca por uma melhor topologia que a obtida através de testes empíricos e também considerando que uma busca exaustiva pela melhor topologia seria inviabilizada pelo grande tempo computacional gasto para sua realização.

Conforme descrito na Seção 3.2, dois Algoritmos Evolutivos foram implementados, sendo que o segundo é composto do primeiro algoritmo acrescido de algumas melhorias na fase de reprodução do primeiro. Para a execução do processo evolutivo, foi gerada uma população de 8 cromossomos (RNAs) (Seção 3.2). Ao todo, o processo de evolução levou 130 gerações consumindo em torno de $54h$ de processamento para cada um dos algoritmos implementados.

O resultado obtido a partir do primeiro algoritmo (AE-I) foi uma rede com topologia 4800X60X52X10 obtendo um erro quadrático médio de 0.092 (*fitness*: 0.907605) e o resultado obtido através do processo evolutivo implementado pelo segundo algoritmo (AE-II) foi uma topologia 4800X69X43X10 com um erro quadrático médio de 0.091 (*fitness*: 0.908319). Isto representa resultados semelhantes considerando o erro quadrático médio obtido por ambas as topologias evoluídas pelo AE.

Contudo, observando as Figuras 5.11 e 5.12 que apresentam os gráficos dos processos evolutivos gerados pelos algoritmos AE-I e AE-II, respectivamente, pode ser notado que o segundo algoritmo apresentou uma evolução mais rápida que o primeiro, tendo consumido apenas 33 gerações para obter um *fitness* maior que 0.9, enquanto o primeiro algoritmo levou 65 gerações para atingir tal resultado. Considerando que o tempo é um fator crucial para a execução dos experimentos, o AE-II se mostrou mais adequado na tarefa de evolução da topologia da rede neural MLP.

Uma vez obtida uma topologia adequada ao problema (de reconhecer a posição e a forma do objeto), testes utilizando a técnica de validação cruzada (já descrita acima) foi utilizada para obter um resultado mais real sobre a precisão das topologias encontradas pelos AEs. As Tabelas 5.6 e 5.7 apresentam os resultados obtidos a partir da aplicação da técnica de validação cruzada para as duas topologias evoluídas. São apresentados o número médio de iterações necessárias a convergência, a média do erro quadrático médio, o desvio padrão do erro quadrático médio e o tempo computacional gasto na realização dos experimentos.

Pelos resultados obtidos pode-se observar que a precisão das redes neurais, treinadas pelos algoritmos descritos, é praticamente a mesma. Entretanto, o tempo de convergência de cada algoritmo é bem distinto. Claramente, pode ser observado que o algoritmo Rprop, como já descrito na literatura apresenta um

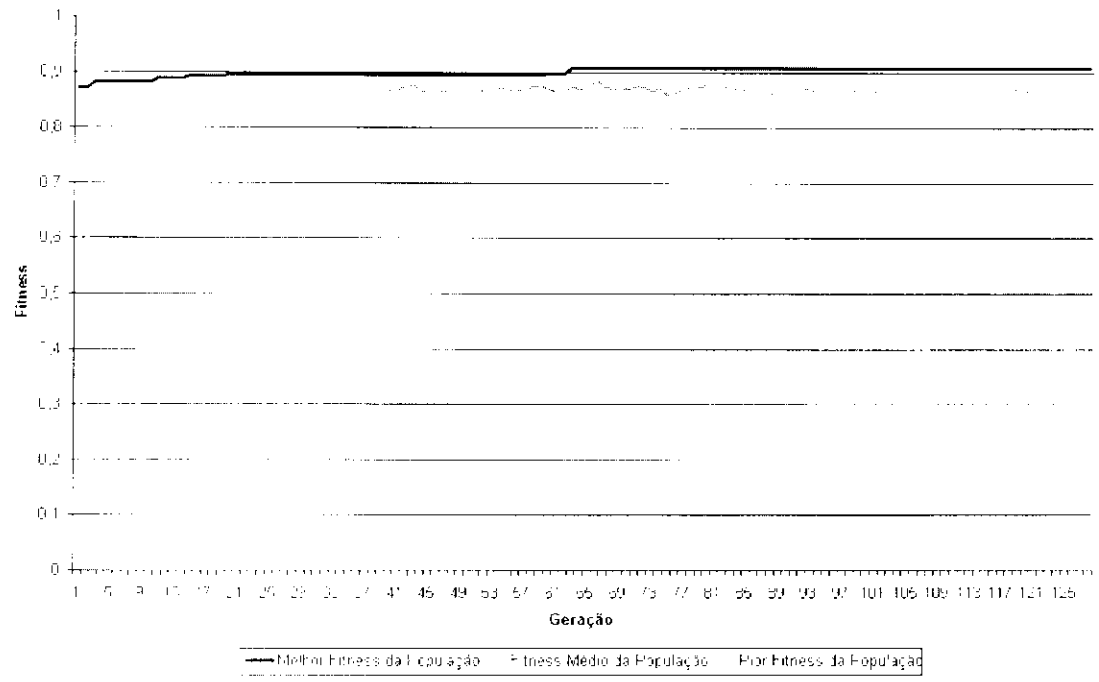


Figura 5.11: Resultado do processo evolutivo gerado a partir do algoritmo I

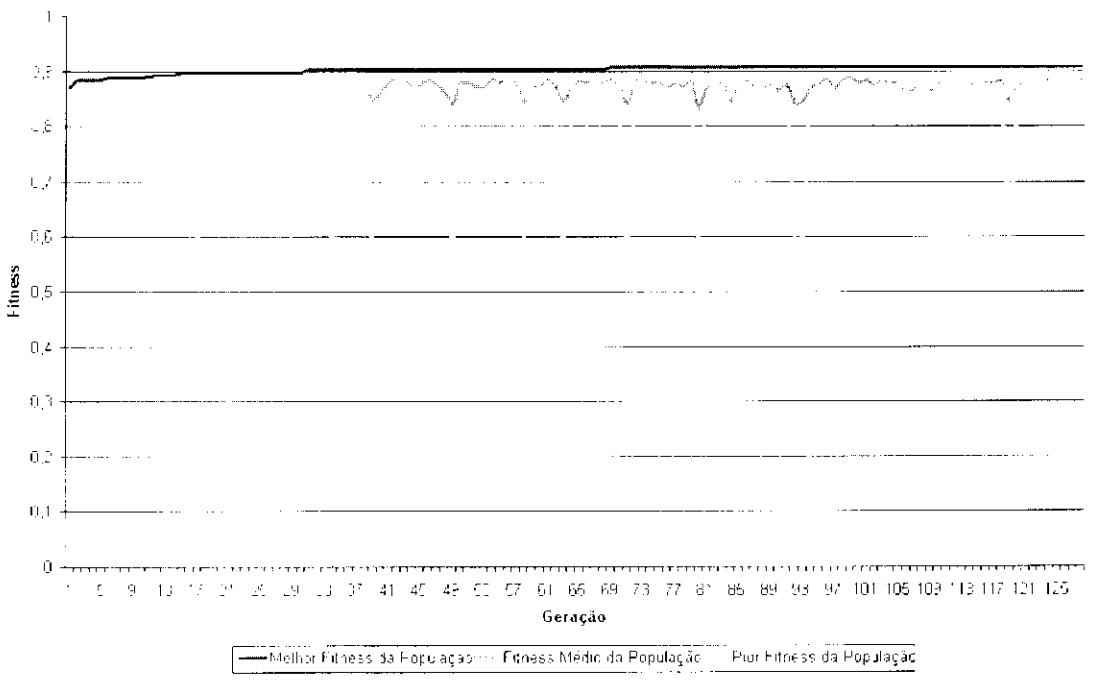


Figura 5.12: Resultado do processo evolutivo gerado a partir do algoritmo II

Tabela 5.6: Resultado do Processo de Treinamento para a Rede Neural de Reconhecimento da Posição e Forma dos Objetos - Topologia 4800X60X52X10 (AE-I)

	Iterações	EQM	Desvio Padrão	Tempo Treinamento
Rprop	38	0,093670	0.02153000	$\approx 50 minutos$
BP em Lote	2134	0.102270	0.01125000	$\approx 35 horas$
BP Padrão	500	0.099153	0.02271200	$\approx 14 dias$

Tabela 5.7: Resultado do Processo de Treinamento para a Rede Neural de Reconhecimento da Posição e Forma dos Objetos - Topologia 4800X69X43X10 (AE-II)

	Iterações	EQM	Desvio Padrão	Tempo Treinamento
Rprop	36	0.091108	0.00993700	$\approx 55 minutos$
BP em Lote	2092	0.100966	0.01218600	$\approx 40 horas$
BP Padrão	500	0.098701	0.01354000	$\approx 16 dias$

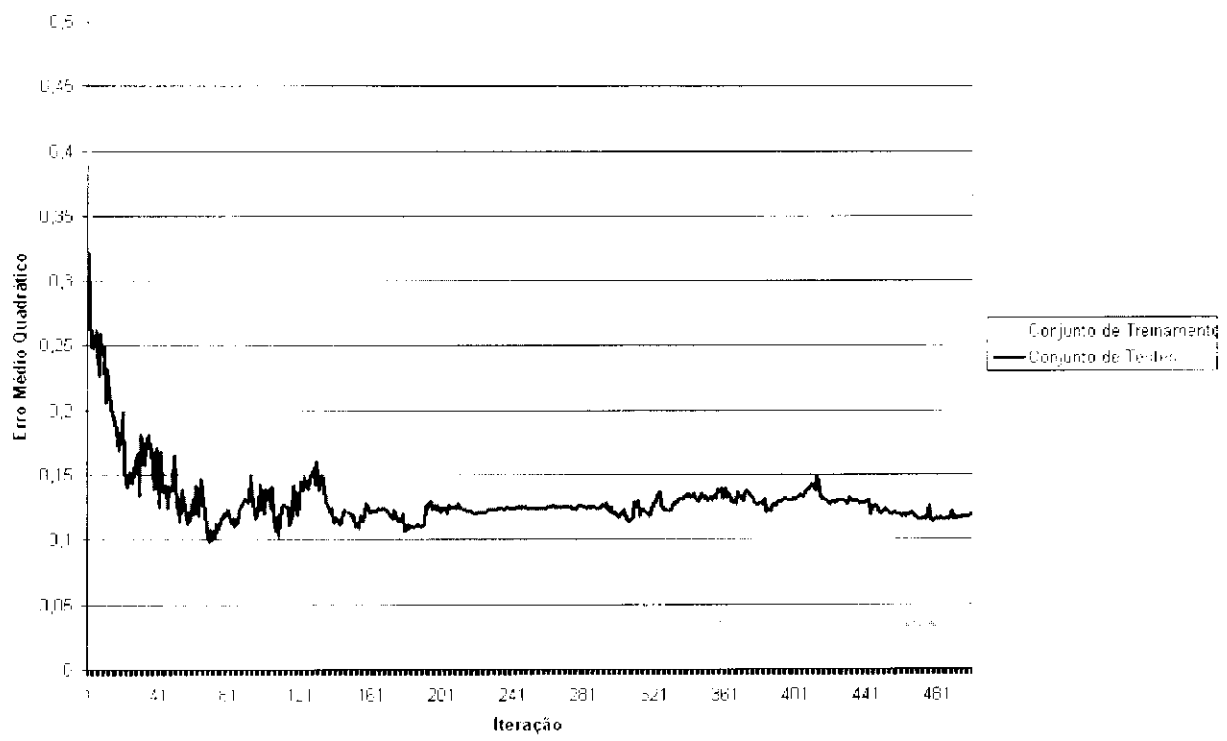


Figura 5.13: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 1800X60X52X10 - algoritmo de retropropagação padrão

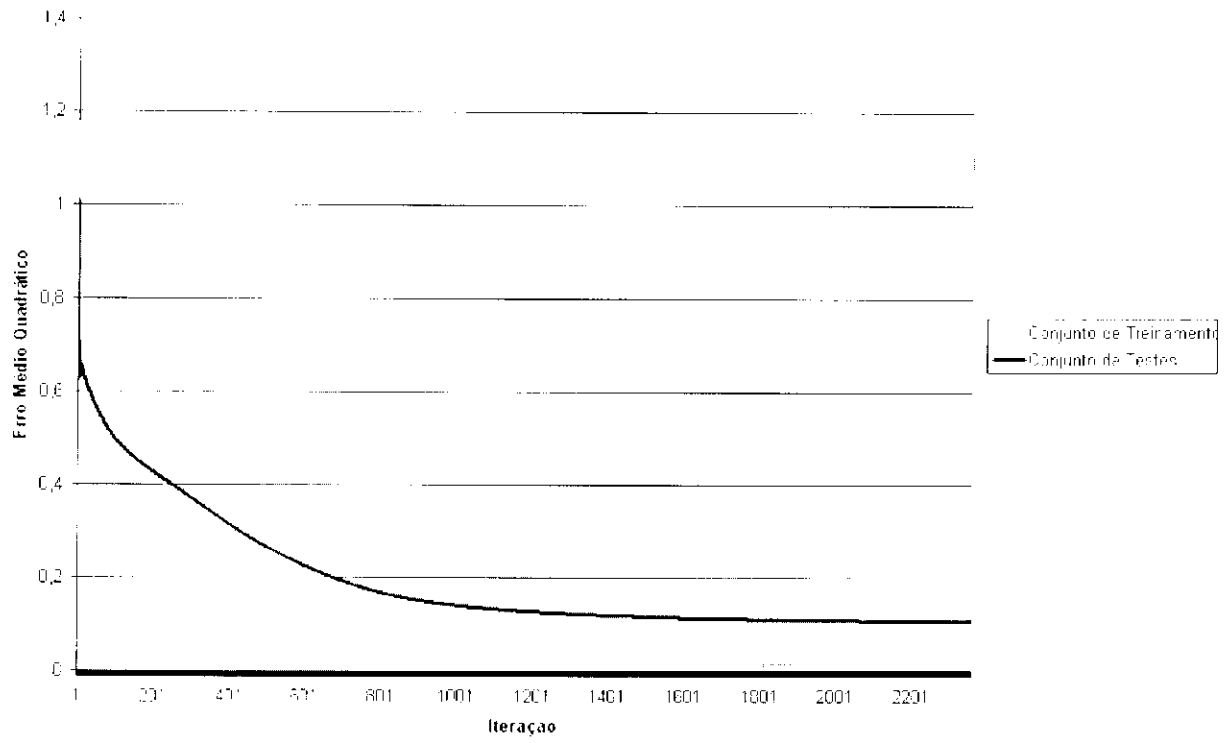


Figura 5.14: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 1800X60X52X10 - algoritmo de retropropagação em lote

tempo de convergência extremamente mais rápido que o algoritmo de retropropagação do erro, principalmente quando a tarefa a ser aprendida pela rede for de difícil convergência e a base de dados grande.

Um outro fator interessante também observado é que o algoritmo de retropropagação do erro em lote embora leve um número bem maior de iterações (épocas) para convergir, no quesito tempo computacional gasto, ele se mostrou muito mais eficiente que o algoritmo de retropropagação implementado na sua forma padrão. Isto se deve ao fato de que na implementação padrão, para a apresentação de cada estímulo (padrão) à rede MLP é realizado o ajuste dos pesos sinápticos da rede enquanto que na implementação em lote, o ajuste dos pesos só é realizado uma vez após a apresentação de todos os padrões do conjunto de treinamento, o que resulta em um menor consumo de tempo para o processo de treinamento.

Exemplos da convergencia para as redes neurais cuja topologia foi evoluída pelos AEs (Topologia 1800X60X52X10 e 4800X69X43X10) podem ser visualizadas nas Figuras 5.13, 5.14 e 5.15 para a primeira topologia e Figuras 5.16, 5.17 e 5.18 para a segunda topologia. Nesses gráficos pode ser claramente destacado o tipo de convergência realizada por cada algoritmo. Para o algoritmo de retropro-

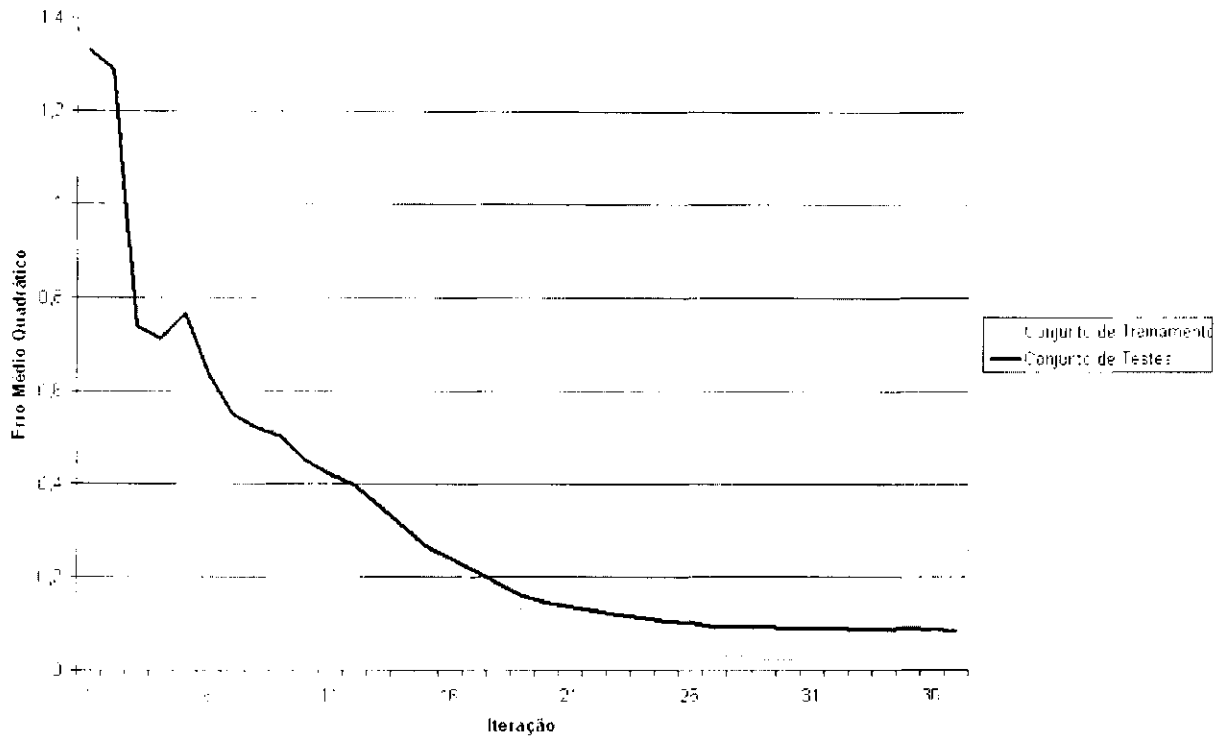


Figura 5.15: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 4800X60X52X10 - algoritmo Rprop

pagação do erro implementado na sua forma padrão, a descida pela superfície de erro acontece de forma estocástica, causando as oscilações observadas nos gráficos (Figuras 5.13 e 5.16). Quando implementado em lote, o algoritmo de retropropagação faz uma descida mais suave pela superfície, porém, consumindo mais iterações para garantir a convergência. Já o Rprop faz uma rápida descida pela superfície por utilizar apenas o sinal da derivada (gradiente do erro) e não o seu valor, o que garante uma descida mais rápida.

Nas Seção abaixo são apresentados os resultados dos experimentos realizados em campo com o robo real.

5.2 Experimentos com o Robô

Com o objetivo de validar o SVC desenvolvido, diversos experimentos com o robô real foram realizados:

1. O robô perseguindo um objeto de cor determinada;
2. O robô perseguindo um objeto de cor e forma determinada e;
3. O robo navegando pelo ambiente perseguindo um objeto de cor determinada

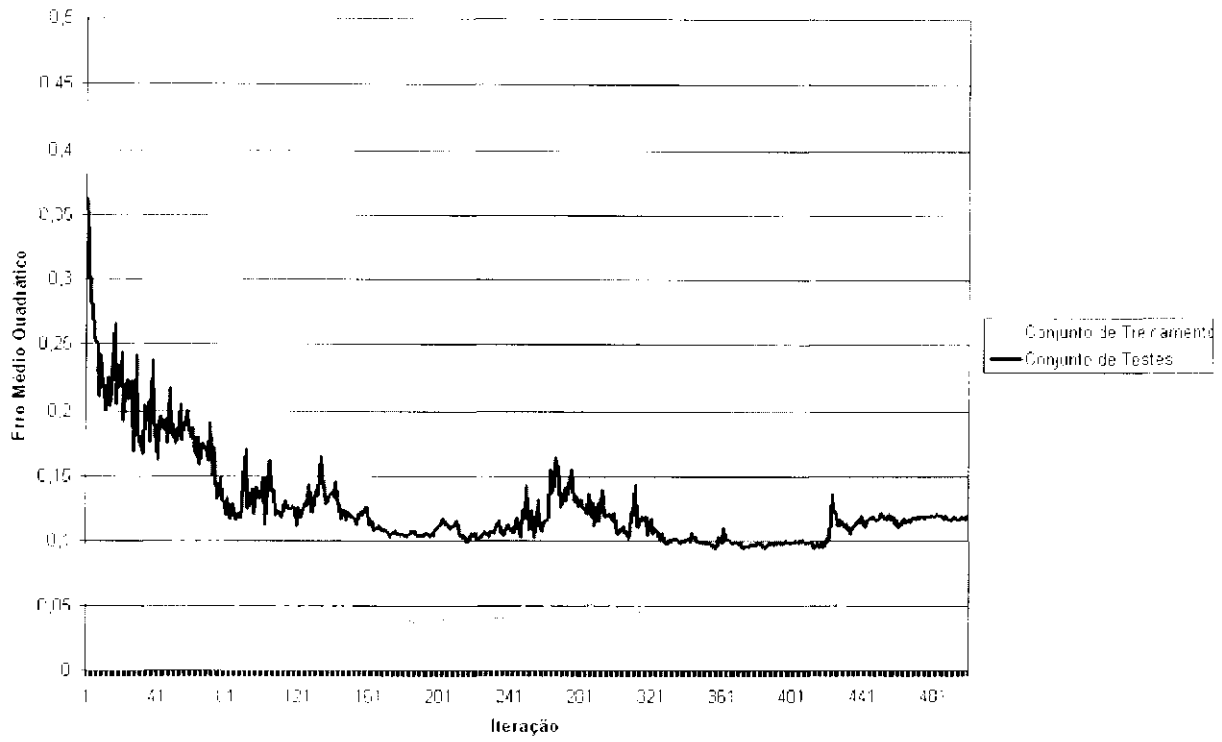


Figura 5.16: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 4800X69X43X10 - algoritmo de retropropagação padrão

e desviando de obstáculos.

Todos os testes foram desenvolvidos na plataforma do robô Pioneer I (Seção 4.2) utilizando um *laptop Pentium* III $500MHz$ com $64MB$ de memória RAM.

5.2.1 O Robô Perseguindo Cores

Conforme apresentado na Seção 4.5.3, duas metodologias foram implementadas para o sistema de visão responsável por reconhecer a posição do objeto no campo visual. Para essas duas metodologias foram desenvolvidos dois controladores (Seção 4.6) que com base nas saídas da rede neural (módulo de visão) deveriam capacitar o robô a perseguir o objeto de cor indicada.

Para a realização dos testes, um usuário é responsável por selecionar a cor que o robô deve perseguir. Uma vez selecionada a cor de interesse é estabelecida a comunicação com o robô através do ambiente Saphira, o qual é responsável por enviar os comandos ao robô. Após conectado (*laptop*/robô), objetos devem ser colocados no campo de visão do robô, sendo que este deve apenas reconhecer e perseguir o objeto de cor indicada.

Vários testes foram realizados com as duas metodologias descritas na Seção

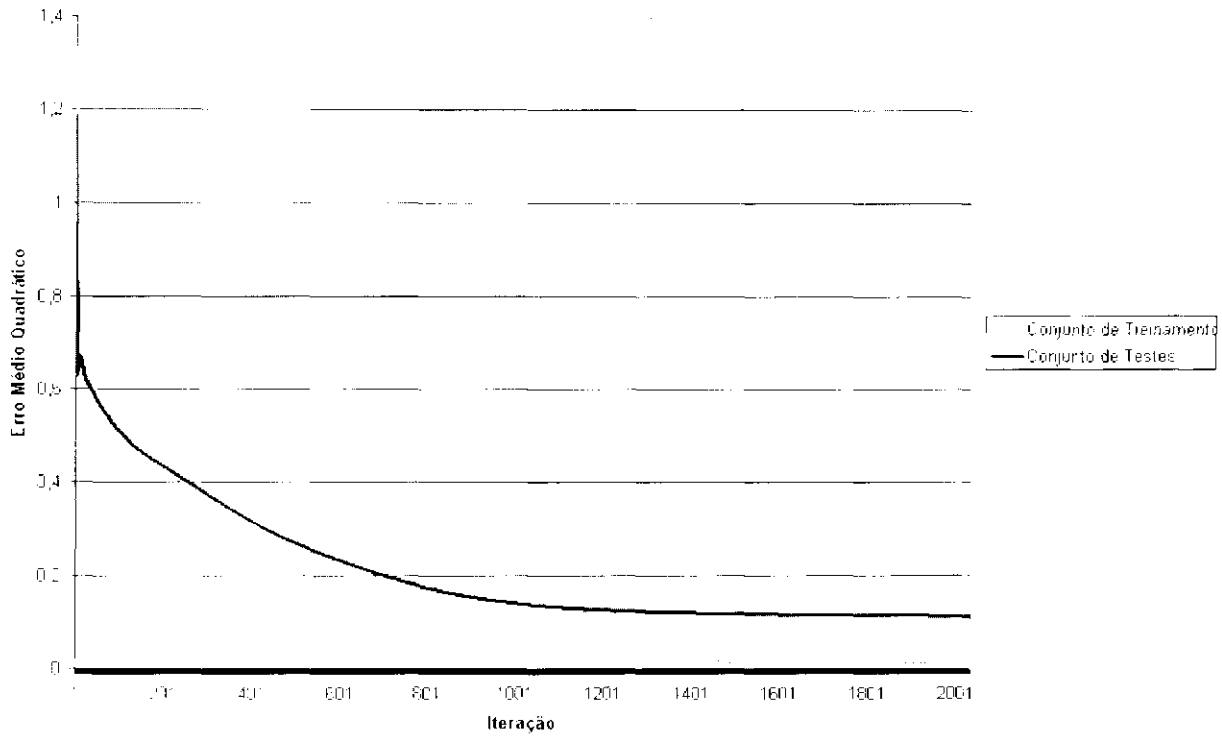


Figura 5.17: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 4800X69X43X10 - algoritmo de retropropagação em lote

4.6 e em todos esses testes com as três cores consideradas (Seção 5.1.1), o robô foi capaz de perseguir o objeto atuando em tempo real. Se objeto estiver fora do campo de visão do robô, o robô pára.

5.2.2 O Robô Perseguindo Cores e Formas

Embora nos testes de precisão apresentados acima a rede neural responsável pelo reconhecimento da forma e da posição do objeto tenha obtido bons resultados, nos experimentos reais com o robô, a rede não foi capaz de realizar as duas tarefas solicitadas: reconhecer a forma do objeto e indicar sua posição no campo visual.

Nesses testes foi observado que a rede neural conseguiu distinguir bem as formas dos objetos, contudo, não foi capaz de informar sua posição com precisão, gerando valores aleatórios para as saídas X e Y da rede neural para cada objeto. Também houve problemas no reconhecimento da forma, quando o robô realizou movimentos rápidos, distorcendo a imagem adquirida pela câmera e impossibilitando o reconhecimento dessas imagens pela RNA. Isto se deve ao fato de que o hardware utilizado é bastante limitado, tanto a câmera quanto o poder de processamento do *laptop* utilizado possuem restrições, o que degrada de uma forma

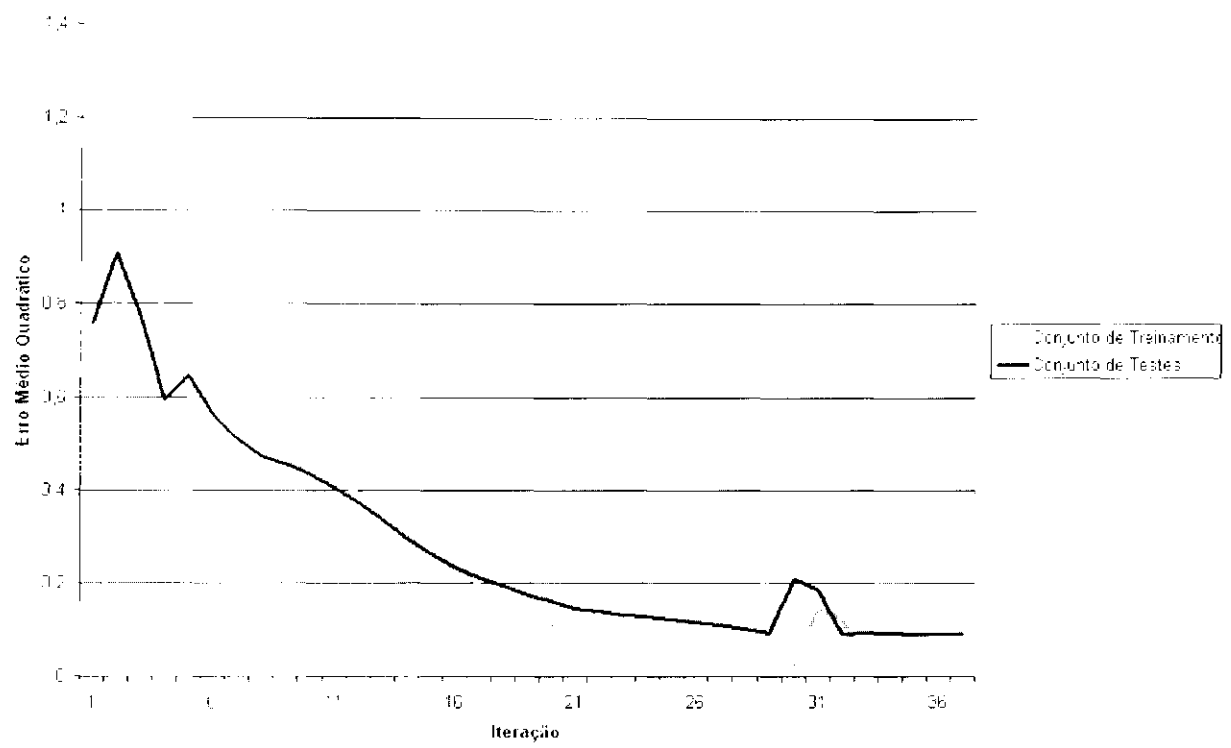


Figura 5.18: Treinamento da rede MLP para o reconhecimento da posição e forma do objeto - topologia 4800X69X43X10 - algoritmo Rprop

global o desempenho esperado do sistema.

Para solucionar o problema do posicionamento do objeto no campo de visão, foram realizados alguns experimentos aglomerando a tarefa de reconhecimento da posição a tarefa para o reconhecimento da forma e posição do objeto, sendo que desta segunda tarefa utilizou-se apenas as saídas da rede neural referentes a identificação do objeto (3 saídas - P"Objeto") e deixando o posicionamento deste a cargo da rede neural responsável pelo reconhecimento da posição. As duas redes operam em paralelo, isto é, recebem as mesmas entradas e as suas saídas são combinadas para informar a posição e a forma do objeto.

Com essas modificações, o robô foi capaz de perseguir, em tempo real, um objeto de forma e cor determinada. Contudo, como a rede de reconhecimento da forma não apresenta 100% de precisão, para algumas imagens capturadas, onde ocorreram erros de classificação da forma do objeto, o robô parou de se locomover (comportamento Stop()), necessitando da interação do usuário para continuar a perseguir o objeto. Esse problema é identificado apenas quando o robô está em uma velocidade muito baixa impedindo que este mude sua posição e automaticamente capture um quadro diferente do ambiente, retirando-se assim de um "mínimo local" formado no campo de visão onde ocorreu o erro de classificação.

5.2.3 Aplicação de Campos Potenciais

Para dar um caracter mais genérico ao SVC implementado, resolveu-se incorporar uma técnica de navegação de robós móveis ao sistema. Como já foi dito no Capítulo 4, a técnica escolhida foi a de Campos Potenciais, por ser uma técnica simples e já estudada por alguns membros do grupo de Robótica do ICMC-USP.

Com a incorporação desta técnica foi possível capacitar o robô Pioneer a navegar em ambientes buscando por um objeto de cor definida utilizando-se o sistema de visão e, desviar de obstáculos pela utilização das informações provenientes dos sonares do robô.

Em todos os testes realizados, o robô foi capaz de navegar pelo ambiente desviando dos obstáculos e, sendo atraído por uma cor (objeto) determinado pelo usuário (quando está se encontrava no campo visual do robô).

Considerando todos os testes realizados em tempo real com o robô Pioneer I, pôde-se observar que a metodologia adotada foi adequada na realização das tarefas designadas. Sendo que apenas a abordagem utilizada para o reconhecimento de forma e posição dos objetos apresentou algumas limitações, contudo, nenhuma limitação que inviabilizasse o sistema desenvolvido.

No capítulo seguinte são apresentadas as conclusões do trabalho e os possíveis trabalhos futuros.

Capítulo

6

Conclusões

Nesta dissertação foi proposto um sistema de visão computacional para o controle de robôs móveis, desenvolvido utilizando-se modelos de redes neurais do tipo MLP.

Dois módulos principais foram implementados: módulo de visão e módulo de controle de navegação.

No módulo de visão, o processamento se dá em em três fases: pré-processamento, segmentação e reconhecimento. Nesta última, diversas abordagens foram elaboradas.

Dentro do módulo de controle foram testados quatro controladores. Dois deles são responsáveis por capacitar o robô a perseguir um objeto de cor determinada. O terceiro tem a finalidade de permitir que o robô persiga um objeto de cor e forma estabelecidas. O último controlador, tem o objetivo de estabelecer um sistema de navegação autonomo, habilitando o robô a caminhar pelo ambiente e, ao mesmo tempo, desviar de obstáculos e buscar um objeto de cor determinada. Para integração dos comportamentos implementados neste último controlador foi utilizada a técnica de Campos Potenciais.

Para a determinação dos parâmetros do sistema, tais como o número de camadas em cada rede neural, o número de neurônios em cada camada e demais parâmetros, diversos testes empíricos foram realizados utilizando-se dois algoritmos de treinamento para redes neurais MLPs: o algoritmo Rprop e o algoritmo de retropropagação. Este último foi implementado tanto na sua forma padrão como na forma em lote.

Além desses testes empíricos, para se estabelecer a topologia da rede neural utilizada no reconhecimento da forma e posição do objeto, foi utilizado um Algoritmo Evolutivo (AE). Pode ser notado que as soluções encontradas com o AE superaram aquela encontrada manualmente por tentativa e erro. Assim, fica

demonstrado o potencial dos AEs para otimização de topologias de RNAs, considerando que uma busca exaustiva pela melhor topologia poderia ser inviabilizada pela grande quantidade de tempo necessário para sua realização.

Um outro resultado interessante observado durante o treinamento da rede MLP para reconhecimento de forma e posição dos objetos, foi o ganho de tempo computacional obtido utilizando-se o algoritmo Rprop. A utiliação deste algoritmo resultou em uma diminuição aproximada de 99,75% no tempo gasto, quando comparado com o algoritmo de retropropagação do erro padrão, e de 89,6%, se comparado ao algoritmo de retropropagação em lote.

Além disso, também pode ser notado que, embora o algoritmo de retropropagação do erro implementado em lote exija mais iterações para convergir, no quesito tempo computacional gasto ele foi mais rápido que o algoritmo de retroprogação padrão. Isto se deve ao fato de que, para cada elemento da base de dados utilizada no treinamento, o algoritmo de retropropagação padrão realiza o ajuste dos pesos sinápticos da rede, ao contrário do algoritmo implementado em lote, que realiza o ajuste dos pesos apenas uma vez a cada época de treinamento.

O sistema de segmentação adotado se mostrou adequado ao domínio da aplicação, atuando em tempo real e produzindo uma boa qualidade de segmentação para as tres cores utilizadas: vermelha, azul e amarela.

Além disso, para melhorar o tempo de resposta do sistema de segmentação adotado, também foi proposta neste trabalho uma linearização da função logística. Com isso, buscou-se reduzir o tempo computacional gasto com o cálculo de tal função. Os resultados demonstraram que esta mudança proporcionou um ganho de mais de 30% na velocidade de segmentação. Além disso, não ocorreram degradações na qualidade final da imagem segmentada.

Nos testes realizados em ambiente real pode ser observado que, mesmo com a utilização de recursos limitados, como o *laptop* e a câmera utilizados, foi comprovada a eficiência da metodologia proposta. Considerando o SVC capacitou o robô a perseguir um objeto e desviar de obstáculos ao navegar pelo ambiente atuando em tempo real.

Também observou se, durante os experimentos, que a metodologia utilizando apenas uma rede neural MLP para reconhecer a forma e posicionamento do objeto não se mostrou totalmente adequada. Isto se deve ao fato de a rede neural não ter sido capaz de realizar as duas tarefas a ela atribuídas: classificar o objeto e informar sua posição no campo visual com uma precisão aceitável.

Tal problema foi solucionado ao utilizar duas redes neurais, uma para o reconhecimento da forma e outra para informar a posição do objeto no campo visual.

Contudo, mesmo estando o problema aparentemente solucionado, foi verificado que a metodologia para reconhecimento dos objetos utilizando-se apenas a rede neural MLP possui limitações. Isto se deve ao fato de que, para cada objeto tratado pela rede neural, todas as possibilidades de posicionamento no campo visual devem ser cobertas. Isto torna o sistema muito propício a falhas e, além disso, o número de objetos que a rede neural pode aprender também é bastante limitado.

Como citado ao longo do texto, este trabalho de mestrado teve como objetivo investigar e desenvolver um sistema de visão baseado apenas em redes neurais do tipo MLP, pois uma possível implementação em FPGAs é dada como trabalho futuro, considerando-se que este modelo de rede já se encontra implementando em tal hardware pelo grupo de Computação Reconfigurável do ICMC.

Como trabalhos futuros pretende-se implementar o sistema desenvolvido em FPGAs, além de investigar possíveis ampliações ainda não tratadas por este SVC, como por exemplo, realizar o tratamento das imagens quando mais de um objeto da mesma cor estiver presente no campo visual.

Para tentar resolver o problema encontrada na determinação da posição e forma do objeto, utilizando uma única rede neural, pretende-se realizar uma codificação gaussiana da saída, uma vez que esta tem se mostrado eficiente e tem sido utilizada em muitos trabalhos da literatura.

Além disso, está previsto também a ampliação do sistema desenvolvido realizando a fusão com outros módulos já implementados e em desenvolvimento pelo grupo de robótica do ICMC, tais como, o módulo de mapeamento do ambiente, sistema de localização e controle de trajetória e o módulo de controle de robô via Web.

Referências

Abraham, A. (2001). Meta learning evolutionary artificial neural networks. *Neurocomputing* (56), 1–38.

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

Arkin, R. (1998). *Behavior-Based Robotics*. MIT Press.

Berne, R. M. & Levy, M. N. (1996). *Fisiologia* (3.Ed. ed.). Rio de Janeiro: Guanabara Koogan.

Blank, D. S. & Ross, J. O. (1997). Incorporating a connectionist vision module into a fuzzy, behavior-based robot controller. In *Proceedings of the 1997 Midwest Artificial Intelligence and Cognitive Science Society Conference*.

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

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

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

Castillo, P. A., Merelo, J. J., Prieto, A., Rivas, V. & Romero, G. (2000). G-prop: Global optimization of multilayers perceptrons using gas. *Neurocomputing* (35), 149–163.

Cheng, H. D., Jiang, X. H., Sun, Y. & Wang, J. (2001). Color image segmentation: advances and prospects. *Pattern Recognition* (34), 2259–2281.

Chen, T. Q. & Lu, Y. (2002). Color image segmentation - an innovative approach. *Pattern Recognition* (35), 395–405.

Coelho, R. C. & da Fontoura Costa, L. (2002). Realistic neuromorphic models and their application to neural reorganization simulations. *Neurocomputing* (48), 555–571.

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

Cybenko, G. (1988). Continous valued neural network with two hidden layers are sufficient. Technical report, Department of Computer Science - Tufts University.

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

Darwin, C. (1859). *On The Origin Of Species By Means Of Natural Selection*. London, John Murray.

da Fontoura Costa, L., Jr., R. M. C., Coelho, R. C. & Tanaka, J. S. (1999). *Modeling in the Neurosciences: From Ionic Channels to Neural Networks*, Chapter Analysis and Synthesis of Morphologically Realistic Neural Networks, pp. 505–527. Harwood Academic Publishers.

de Carvalho, A. C. P. L. F., Delbem, A. C. B., Romero, R. A. F., Simões, E. V. & Telles, G. P. (2001). *Computação Bioinspirada*. SBC.

de Pádua Braga, A., Ludermir, T. B. & de Leon Ferreira Carvalho, A. C. P. (2000). *Redes Neurais Artificiais: Teoria e aplicações*. LTC.

Egmont-Petersen, M., de Ridder, D. & Handels, H. (2002). Image processing with neural networks - a review. *Patter Recognition* (35), 2279–2301.

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

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

Fritzke, B. (1992). In , I. Aleksander, *Artificial Neural Networks, 2*, Volume 2, Amsterdam, Netherlands, pp. 1051–1056. North-Holland.

Fritzke, B. (1995). In , G. Tesauro, D. S. Touretzky, *Advances in Neural Information Processing Systems 7*, pp. 625–632. Cambridge MA: MIT Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.

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

Gonzalez, R. C. & Woods, R. E. (2000). *Processamento de Imagens Digitais*. São Paulo-SP: Edgard Blücher LTDA.

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

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

Igel, C. & Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing* (50), 105–123.

Jähne, B. (1997). *Pratical Handbook on Image Processing for Scientific Applications*. CRC Press.

Jonsson, M., Wiberg, P.-A. & Wickström, N. (1997). Vision-based low-level navigation using a feed-forward neural network. In *International Workshop on Mechatronical Computer Systems for Perception and Action*, Pisa - Italy.

Kandel, E. R., Schwartz, J. H. & Jessell, T. M. (1997). *Fundamentos da Neurociência e do Comportamento*. Guanabara Koogan.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, pp. 500–505.

Koch, C. (1998). *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press.

Kohonen, T. (2001). *Self-Organizing Maps* (3th Edition ed.). Information Science. Springer.

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

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

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

Li, B., Hu, H. & Spacek, L. (2003). An adaptive color segmentation algorithm for sony legged robots. In *21st IASTED International Conference*, Applied Informatics, Innsbruck, Austria, pp. 126–131.

Lucchese, L. & Mitra, S. K. (2001). Color image segmentation: A state-of-the-art survey. In *Proc. of the Indian National Science Academy (INSA-A)*, Volume 67 of *Image Processing, Vision, and Pattern Recognition*, New Delhi, India, pp. 207–221.

Machado, A. (2000). *Neuroanatomia Funcional* (2.Ed. ed.). Rio de Janeiro: Atheneu.

Medeiros, D. M. R. & Romero, R. A. F. (2002). Sistema de controle autônomo de robôs móveis com utilização de redes neurais artificiais. In *Workcomp'2002 - V Workshop de Computação*.

Miller, G., Todd, P. & Hedge, S. (1989). Designing neural networks using genetic algorithms. In *3rd International Conference on Genetic Algorithms*, pp. 379–384.

Mitchel, T. M. (1997). *Machine Learning*. Boston: McGraw Hill.

Murray, D. (1994). Tuning neural networks with genetic algorithms. *IA Expert* (9), 27–31.

Nalwa, V. S. (1993). *A Guided Tour of Computer Vision*. Addison-Wesley.

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

Ong, S. H., Yeo, N. C., Lee, K. H., Venkatesh, Y. V. & Cao, D. M. (2002). Segmentation of color images using a two-stage self-organizing network. *Image and Vision Computing* (20), 279–289.

Pagello, E., Arai, T., Dillmann, R. & Stentz, A. (2002). Towards the intelligent autonomous systems of the third millenium. *Robotics and Autonomous Systems* (40), 63–68.

Papamarkos, N., Atsalakis, A. & Strouthopoulos, C. (2002). Adaptive color reduction. *IEEE Trans. on Systems, Man, and Cybernetics-Part B 32*(1), 44–56.

Papamarkos, N., Strouthopoulos, C. & Andreadis, I. (2000). Multithresholding of color and gray-level images through a neural network technique. *Image and Vision Computing* (18), 213–222.

Papamarkos, N. (1999). Color reduction using local features and a sofm neural network. *Int. Journal of Imaging Systems and Technology 10*, 404–409.

Pomerleau, D., Gowdy, J. & Thorpe, C. (1991). Combining artificial neural networks and symbolic processing for autonomous robot guidance. *Engineering Applications of Artificial Intelligence 4*(4), 279 – 285.

Pomerleau, D. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation 3*(1), 88–97.

Pomerleau, D. (1992). Progress in neural network-based vision for autonomous robot driving. In *Proceedings of the 1992 Intelligent Vehicles Conference*, pp. 391 – 396.

Pomerleau, D. (1993a). In . J. Connell, *Robot Learning*.

Pomerleau, D. (1993b). Neural networks for intelligent vehicles. In *Proceedings of the 1993 Intelligent Vehicles Conference*, pp. 19 – 24.

Pomerleau, D. (1995). Neural network vision for robot driving. In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*.

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Editora Manole.

Riedmiller, M. & Braun, H. (1992). Rprop- a fast adaptive learning algorithm. In *Proc. of ISCIS VII*.

Riedmiller, M. (1994a). Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms. *Int. Journal of Computer Standards and Interfaces* (16), 265–278.

Riedmiller, M. (1994b). Rprop - description and implementation details. Technical report, Universitat Karlsruhe - Institut für Logic, Komplexität und Deduktionssyteme.

Rumelhart, D. E. (1896). *Parallel Distributed Processing: Foundations*. Volume 1. Mit Press.

Simões, A. S. & Costa, A. H. R. (2000a). Segmentação de imagens por classificação de cores: uma abordagem neural para a representação rgb. In C. H. C. R. , M. T. S. Sakude (Ed.), *Workshop de Computação WORKCOMP'2000*, ITA - São José dos Campos, pp. 25–31.

Simões, A. S. & Costa, A. H. R. (2000b). Using neural color classification in robotic soccer domain. In *International Joint Conference IBERAMIA'2000*.

Simões, A. S. & Costa, A. H. R. (2001). Classificação de cores por redes neurais artificiais: Um estudo do uso de diferentes sistemas de representação de cores no futebol de robôs móveis autônomos. In *ENIA*.

Simões, E. D. V. (2000). *Development of An Embedded Evolutionary Controller to Enable Collision-Free Navigation of a Population of Autonomous Mobile Robots*. Ph. D. thesis, The University of Kent at Canterbury, UK.

Skarbek, W. & Koschan, A. (1994). Colour image segmentation - a survey. Technical report, Universidade de Berlin.

Thorpe, C. (1984). Path relaxation: Path planning for a mobile robot. Technical Report CMU-RI-TR-84-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Tomassini, M. (1995). A survey of genetic algorithms. *Annual Reviews of Computational Physics*, *World Scientific*, 87–118.

Tyler, L. G. & Czarnecki, C. A. (1999). A neural vision based controller for a robot footballer. *Image Processing and its Application* (465), 77–81.

Vandenbroucke, N., Macaire, L. & Postaire, J.-G. (2003). Color image segmentation by pixel classification in an adapted hybrid color space. application to soccer image analysis. *Computer Vision and Image Understanding* (90), 190–216.

Waldherr, S., Thrun, S. & Romero, R. (2000). A gesture-based interface for human-robot interaction. *Autonomous Robots 9*(2), 151–173.