

Class
Cat
Temp

FIXAÇÃO DE OBJETOS DE FORMATO GEOMÉTRICO DESCONHECIDO UTILIZANDO REDES NEURAS ARTIFICIAIS

Carlos Magno de Oliveira Valente



Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica

ORIENTADOR: Prof. Dr. Glauco A. P. Caurin

São Carlos
1999

Class.	TESE-EEEC
Curr.	68430
Tombo	196/99

31100049951

S/S 1057869

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

V154f Valente, Carlos Magno de Oliveira
Fixação de objetos de formato geométrico
desconhecido utilizando redes neurais artificiais /
Carlos Magno de Oliveira Valente. -- São Carlos, 1999.

Dissertação (Mestrado) -- Escola de Engenharia de
São Carlos-Universidade de São Paulo, 1999.

Área: Engenharia Mecânica.

Orientador: Prof. Dr. Glauco Augusto de Paula
Caurin.

1. Garras de robo. 2. Redes neurais artificiais.
3. Processamento de imagem. I. Título.

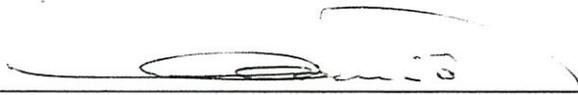
FOLHA DE APROVAÇÃO

Candidato: Engenheiro **CARLOS MAGNO DE OLIVEIRA VALENTE**

Dissertação defendida e aprovada em 11.05.1999
pela Comissão Julgadora:



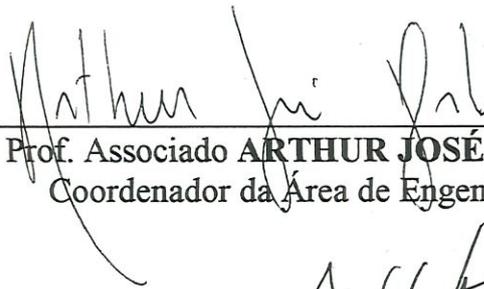
Prof. Dr. **GLAUCO AUGUSTO DE PAULA CAURIN (Orientador)**
(Universidade de Mogi das Cruzes)



Prof. Dr. **MARIO FRANCISCO MUCHERONI**
(Escola de Engenharia de São Carlos – Universidade de São Paulo)



Prof. Dr. **ALUÍZIO FAUSTO RIBEIRO ARAÚJO**
(Escola de Engenharia de São Carlos – Universidade de São Paulo)



Prof. Associado **ARTHUR JOSÉ VIEIRA PORTO**
Coordenador da Área de Engenharia Mecânica



JOSÉ CARLOS A. CINTRA
Presidente da Comissão de Pós-Graduação da EESC

Ao meu pai, minha mãe, minha família e à Érica,
que são as pessoas que eu amo.

Ao Prof. Glauco, pela amizade e pela orientação fornecida durante a elaboração deste trabalho.

À Fundação de Amparo à Pesquisa do Estado de São Paulo, pela excelente estrutura de pesquisa fornecida através de bolsa de pesquisa e reserva técnica, que permitiu a realização deste trabalho.

Ao Prof. Mário Mucheroni e Prof. Aluizio, que colaboraram de forma decisiva para o desenvolvimento da pesquisa.

À Cristina, Suzete, Xina e Cláudio (Joãozinho), pelo suporte técnico e administrativo junto às atividades do Laboratório de Dinâmica.

Aos amigos Schammas, Ivo Antunes e Paulo, pelo agradável convívio dentro do Laboratório de Mecatrônica.

Aos amigos David, Jairo e Bonito, pelas horas de “muxibada” dentro de casa.

E aos demais colegas, professores e funcionários do Departamento de Engenharia Mecânica da EESC/USP, pela colaboração.

SUMÁRIO

LISTA DE FIGURAS	vii
LISTA DE TABELAS	x
RESUMO	xi
<i>ABSTRACT</i>	xii
1. INTRODUÇÃO	1
<hr/>	
1.1. Motivação	1
1.2. Garras industriais	2
1.3. Garras antropomórficas	4
1.4. Planejamento dos parâmetros de fixação	8
1.5. Processos analíticos de programação <i>off-line</i>	9
1.6. Processos analíticos de programação <i>on-line</i>	12
1.7. Processos baseados no comportamento da mão humana	13
1.8. Definição dos pontos de contato utilizando redes neurais	16
1.9. Formalização do problema	18
1.10. Algoritmo proposto	19
1.11. Estrutura da dissertação	20
2. PROCESSAMENTO DE IMAGEM	21
<hr/>	
2.1. Introdução ao processamento de imagem	21
2.2. Detecção do contorno do objeto	24
2.3. Processos de filtragem	25
2.4. Realçamento e detecção de borda	25
2.4.1. Operador de Sobel	27
2.4.2. Operador de Prewitt	27
2.4.3. Operador de Johnson	28

2.4.4. Operador de Laplace	29
2.4.5. Operador laplaciano sobre uma gaussiana	31
2.4.6. Método OU-Exclusivo	33
2.4.7. Método do vizinho mais próximo (<i>Nearest Neighbor</i>)	34
2.5. Comparação dos operador de ênfatização e detecção de borda	35
2.6. Análise final dos operadores de ênfatização de borda	39
3. REDE COMPETITIVA DE HOPFIELD	40

3.1. Rede Clássica de Hopfield	40
3.2. Formalização do problema de aproximação poligonal	42
3.3. Implementação da Rede Competitiva de Hopfield	44
3.4. Resultados do processo de aproximação poligonal	46
4. REDES DE FUNÇÕES DE BASE RADIAL	48

4.1. Arquitetura das Redes de Funções de Base Radial (RBF)	48
4.2. Estratégias de treinamento	50
4.2.1. Seleção de centros e raios fixos	50
4.2.2. Seleção de centros fixos e raios ajustáveis	50
4.2.3. Seleção supervisionada dos centros e raios	51
4.3. <i>Forward Selection</i>	51
4.4. <i>Global Ridge Regression</i>	53
4.5. <i>Local Ridge Regression</i>	55
4.6. <i>Dynamic Decay Adjustment</i> (DDA)	56
4.7. Seleção supervisionada de centros e raios	58
4.8. Comparação entre os diversos algoritmos de treinamento	60
4.9. Comparação de performance entre RBF e MLP	64
5. METODOLOGIA E RESULTADOS	67

5.1. Sistema de fixação	67
5.2. Projeto mecânico da garra	68
5.3. Sistema de processamento de imagem	71
5.4. Sistema de definição dos pontos de contato	73
5.4.1. Aproximação poligonal (Rede Competitiva de Hopfield)	74

5.4.2. Rotação e normalização da figura	74
5.4.3. Definição dos pontos de contato (Rede de Funções de Base Radial)	75
5.4.4. Aproximação do contato para pontos do contorno do objeto	76
5.5. Cálculo das forças estimadas de contato	77
5.6. Resposta do sistema para objetos de formato desconhecido	79

6. CONCLUSÕES E PROPOSTAS FUTURAS	85
--	-----------

ANEXO 1 – ALGORITMO DE BUSCA PELOS PONTOS ÓTIMOS DE CONTATO	88
--	-----------

REFERÊNCIAS BIBLIOGRÁFICAS	92
-----------------------------------	-----------

APÊNDICE 1 – INTRODUÇÃO ÀS REDES NEURAIS ARTIFICIAIS

APÊNDICE 2 – PADRÕES DE TREINAMENTO DA RBF

APÊNDICE 3 – DEFINIÇÃO DO PONTO CENTRAL DA FIXAÇÃO

LISTA DE FIGURAS

FIGURA 1 - Tipos de garras industriais	2
FIGURA 2 – Dispositivo especial de montagem	3
FIGURA 3 – Processo de montagem	3
FIGURA 4 – Exemplos de garras acionadas por cabo	4
FIGURA 5 – Exemplo de garras de acionamento hidráulico e pneumático	5
FIGURA 6 – Foto de duas das mais avançadas garras antropomórficas já construídas	6
FIGURA 7 – Diversas configurações de fixação permitidas pela BarrettHand	7
FIGURA 8 – Garras de formato simplificado porém sem grande perda de funcionalidade	8
FIGURA 9 – Representação geométrica do cone de atrito	9
FIGURA 10 – Algoritmo de fixação proposto por NGUYEN	10
FIGURA 11 – Algoritmo de fixação proposto por PONCE	11
FIGURA 12 – Classificação dos tipo de fixação adotados pela mão humana	14
FIGURA 13 – Configurações primitivas da mão humana para fixações envolventes	15
FIGURA 14 – Algoritmo de fixação proposto por SCHERRER	17
FIGURA 15 – Falha do algoritmo de SCHERRER para um objeto diferente dos padrões utilizados para treinamento da rede MLP	17
FIGURA 16 – Ilustração do sistema de fixação proposto	18
FIGURA 17 – Etapas de processamento do algoritmo proposto	19
FIGURA 18 – Sistema de visão utilizado para captura e processamento da imagem do objeto	21
FIGURA 19 – Arquitetura de uma câmera CCD	22
FIGURA 20 – <i>Grid</i> utilizado para discretização da imagem em uma matriz de pixels	22
FIGURA 21 – Padrão RGB para representação das cores dos pixels	23
FIGURA 22 - Representação hexadecimal das cores primárias	23
FIGURA 23 – Representação hexadecimal para combinações das cores primárias	23

FIGURA 24 – Representação hexadecimal para padrões de cinza	24
FIGURA 25 – Detecção de borda do operador de segunda ordem	29
FIGURA 26 – Distribuição de pixels de uma cena, identificando uma forma quadrática	31
FIGURA 27 – Detecção de borda utilizando o operador de Laplace	31
FIGURA 28 – Função (invertida) resultante da aplicação de Laplace ao filtro gaussiano	32
FIGURA 29 – Detecção da borda do objeto utilizando o método OU-Exclusivo	33
FIGURA 30 – Definição de vizinhança ₄ e vizinhança ₈	34
FIGURA 31 – Algoritmo do vizinho mais próximo	35
FIGURA 32 – Resultado da aplicação do algoritmo do vizinho mais próximo	35
FIGURA 33 – Imagem padrão utilizada para exemplificação dos métodos de detecção de borda	35
FIGURA 34 – Comparação entre os algoritmos de Sobel e Prewitt para diversos valores de <i>threshold</i>	36
FIGURA 35 – Resultados do operador de Johnson para diferentes valores de <i>threshold</i> e <i>d</i>	37
FIGURA 36 – Resultados de baixa qualidade do método OU-Exclusivo	37
FIGURA 37 – Resultado do método do vizinho mais próximo	38
FIGURA 38 – Variação do tempo de processamento do algoritmo do vizinho mais próximo em função da posição do objeto na cena.	39
FIGURA 39 - Representação vetorial da rede de Hopfield	40
FIGURA 40 - Cálculo do desvio ponto-corda $h_{x,y}$.	42
FIGURA 41 – Representação matricial da Rede Clássica de Hopfield	43
FIGURA 42 - Arquitetura da Rede Competitiva de Hopfield	45
FIGURA 43 - Exemplos de aproximação poligonal da Rede Competitiva de Hopfield	47
FIGURA 44 - Arquitetura da rede RBF	48
FIGURA 45 – Comparação entre três funções de ativação das unidades escondidas	49
FIGURA 46 – Montagem da camada intermediária através do algoritmo <i>Forward Selection</i>	52
FIGURA 47 – Processo de ajuste do parâmetro regulador λ	54
FIGURA 48 – Definição do vetor erro (vermelho) à partir da projeção da saída desejada \hat{y} (azul) na direção perpendicular ao modelo	54

FIGURA 49 – Definição dos <i>thresholds</i> positivo θ^+ e negativo θ^- .	56
FIGURA 50 – Montagem da camada intermediária à partir do algoritmo DDA	57
FIGURA 51 - Montagem da camada intermediária a partir do algoritmo DDA	57
FIGURA 52 - Montagem da camada intermediária à partir do algoritmo DDA	58
FIGURA 53: Resultados típicos dos diversos algoritmos durante teste de recuperação de padrões	62
FIGURA 54: Exemplos de padrões não treinados utilizados para o teste de generalização	63
FIGURA 55 – Resultados típicos do teste de generalização	64
FIGURA 56 - Formas de divisão do espaço de entrada	65
FIGURA 57 – Diagrama mostrando a sequência de operação do sistema de fixação	67
FIGURA 58 – Foto do sistema de fixação implementado	68
FIGURA 59 – Projeto mecânico do dedo, composto por uma base e duas falanges	68
FIGURA 60 – Sistema de acionamento da garra. Protótipo da garra	69
FIGURA 61 – Configuração dos tendões e das polias (vista lateral simplificada).	69
FIGURA 62 – Projeto mecânico do dispositivo de acionamento da garra	70
FIGURA 63 – Dispositivo de acionamento da garra	70
FIGURA 64 – Sistema dedicado VME	71
FIGURA 65 – Tela de controle do sistema de captura e processamento da imagem	72
FIGURA 66 – Resultados do processamento de imagem	73
FIGURA 67 – Processo de aproximação poligonal, com a definição de um polígono de dez lados sobre os pontos do contorno da peça	74
FIGURA 68 – Rotação e normalização das figuras, a fim de reduzir o conjunto de treinamento da rede RBF	75
FIGURA 69 – Definição dos três pontos de fixação pela rede RBF	76
FIGURA 70 – Determinação dos pontos efetivos de contato	77
FIGURA 71 - Modelo físico da fixação	78
FIGURA 72 – Definição das forças estimadas de contato para cada um dos dedos da garra	78
FIGURA 73 – Resposta do sistema para objetos desconhecidos	83
FIGURA 74 – Padrão regular da classe Retângulo4	84
FIGURA 75 – Algoritmo de busca pelos pontos ótimos de contato	90

LISTA DE TABELAS

TABELA 1 – Processo de otimização para definição dos pontos e das forças de contato	12
TABELA 2 – Comparação dos algoritmo quanto ao tempo de processamento	38
TABELA 3 – Teste de recuperação de padrões	61
TABELA 4 – Resultados do teste de generalização	63

RESUMO

Desenvolvimento de uma garra robótica de três dedos, capaz de fixar objetos de formato arbitrário. Para manipular estes objetos, propõe-se um sistema composto por dois estágios: processamento de imagem e cálculo neural dos pontos de contato do objeto. O sistema de visão captura imagens de topo da cena e utiliza o algoritmo do vizinho mais próximo para identificar os pontos que definem o contorno do objeto. No segundo estágio, dois modelos de redes neurais foram implementados para planejar a fixação, definindo os pontos de contato entre a garra e a peça. A primeira rede neural (Rede Competitiva de Hopfield) realiza uma aproximação poligonal sobre o conjunto de pontos do contorno, simplificando a representação deste. O segundo modelo de rede é responsável pelo cálculo efetivo dos três pontos de contato. Diversas configurações de redes Multi-layer Perceptron (MLP) e Redes de Funções de Base Radiais (RBF) foram testadas a fim de definir o método mais adequado. Através desta análise, a rede RBF treinada pelo algoritmo *Global Ridge Regression* apresentou uma maior qualidade de resposta e um desempenho compatível com aplicações em tempo real.

ABSTRACT

This work presents a robot gripper with three fingers which is able to capture objects of arbitrary shape. To handle arbitrary shaped objects, we propose a solution in two stages - image processing and object contact points definition. The vision system captures the top image and uses the Nearest-Neighbor Method to define a set of points representing the object outline. In the second stage, two neural network architectures work together selecting three contact points for the gripper on the outline. A first neural network (Competitive Hopfield Network) realizes a polygonal approximation over the set of points, reducing the number of points to be analyzed. A second supervised neural network computes the three contact points from the approximated polygon. Several configurations of Multi-layer Perceptron (MLP) and Radial Basis Function (RBF) networks were tested to define the most suitable configuration. Through this analysis, the RBF network trained by the Global Ridge Regression method presented the best response accuracy and a performance compatible with on-line applications.

1. INTRODUÇÃO

1.1. Motivação

A mão é uma das partes mais evoluídas do corpo humano, um diferencial na evolução do homem. Possibilita uma interação extremamente versátil com o meio-ambiente, demonstrando movimentos, captando sensações e muitas outras habilidades que os pesquisadores gostariam de emular em garras robóticas. A proposta de construção de uma mão artificial é bastante pretensiosa, porém, a comunidade científica lançou o desafio de se aproximar ao máximo do padrão humano, pois o alcance deste objetivo teria grandes aplicações.

As garras comerciais, hoje existentes, atendem somente a certas tarefas específicas. Isto explica suas aplicações em ambientes industriais, onde as operações são totalmente estruturadas e repetitivas. Nestas situações, elas funcionam mais como uma ferramenta do que como uma mão. Porém, algumas tarefas já exigem uma certa inteligência das garras, como trabalhos em ambientes não-estruturados ou mesmo em ambientes hostis. Nestas situações, é necessária uma maior flexibilidade do sistema e a capacidade de executar tarefas de maneira autônoma. Para isto, estão sendo pesquisadas garras de múltiplos dedos para serem aplicadas em usinas nucleares (JONGKIND, 1993), em operações no fundo do mar (AMADEUS PROJECT) e também no espaço (DLR ARTICULATED HAND). Outra importante aplicação das garras de múltiplos dedos está no ramo das próteses (SOUTHAMPTON ARTIFICIAL HAND), onde elas devem definitivamente assumir uma postura de mão e não de ferramenta.

Embora as garras antropomórficas sejam ainda pouco utilizadas comercialmente, existe uma tendência natural de que elas ganhem espaço mesmo em áreas estruturadas como nas indústrias. Seria o mesmo caminho seguido pelos robôs, que no início eram aplicados apenas em operações especiais e hoje estão espalhados pelas mais diversas fábricas. Isto ocorreu porque flexibilidade e eficiência são requisitos importantes em qualquer ramo da automação. Estes requisitos aumentam a importância das garras de múltiplos dedos, que poderiam aperfeiçoar os processos de fixação, manipulação e montagem, sendo este último

ainda um dos principais desafios da robótica atual. Contudo, estas garras são ainda sistemas complexos, difíceis de controlar, pouco confiáveis e requerem um grande esforço computacional. O desafio está em desenvolver atuadores, sensores e metodologias de controle que reduzam estes problemas, o que resultaria na substituição das garras simplificadas e específicas por sistemas mais flexíveis e eficientes.

A grande capacidade de fixação da mão humana é resultado em parte das suas ricas características anatômicas e em parte das complexas atividades cognitivas, que definem o tipo de fixação a ser utilizada e as forças a serem aplicadas. Da mesma forma, as pesquisas se dividem na busca por projetos mecânicos cada vez mais eficientes e no planejamento e controle de uma fixação cada vez mais inteligente.

1.2. Garras Industriais

A grande maioria das garras industriais possuem acionamento hidráulico ou pneumático e as seguintes configuração básicas:

- dois dedos paralelos ou angulares
- três dedos radiais na forma de castanha.

A FIGURA 1 mostra algumas destas configurações. Existem desde garras em miniatura com 23mm de comprimento, força máxima por dedo de 1,7N e carga máxima recomendada de 18g (FIGURA 1A), até garra de potência com 280mm de largura, força máxima por dedo de 5300N e carga máxima recomendada de 50Kg (FIGURA 1C).

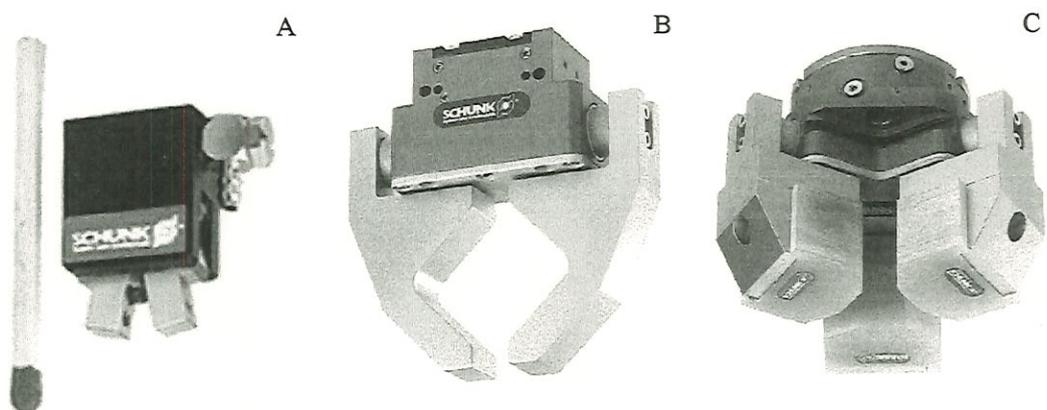


FIGURA 1- Tipos de garras industriais: (A) garra miniatura angular de dois dedos; (B) garra paralela de dois dedos; (C) garra de três dedos radiais em forma de castanha. Estas fotos foram retiradas do catálogo de dispositivos de fixação da SCHUNK PRECISION WORKHOLDING SYSTEMS.

Alguns dispositivos possuem sensores de força posicionados na base dos dedos, que permitem a detecção de colisões e sobrecargas, porém não permitem o controle de força individual por dedo. Sem este controle, as garras industriais normalmente trabalham em malha aberta, utilizando a potência máxima de seus atuadores.

Existem também dispositivos especiais para montagem (FIGURA 2). Eles se caracterizam por uma baixa rigidez a esforços transversais, o que permite um certo deslocamento ou uma certa rotação da peça a ser montada, que desta forma se acomoda no alojamento (FIGURA 3). O ponto negativo é que cada dispositivo deve ser construído segundo dimensões específicas da peça.



FIGURA 2 – Dispositivo especial de montagem, caracterizado por uma pequena rigidez a esforços laterais.

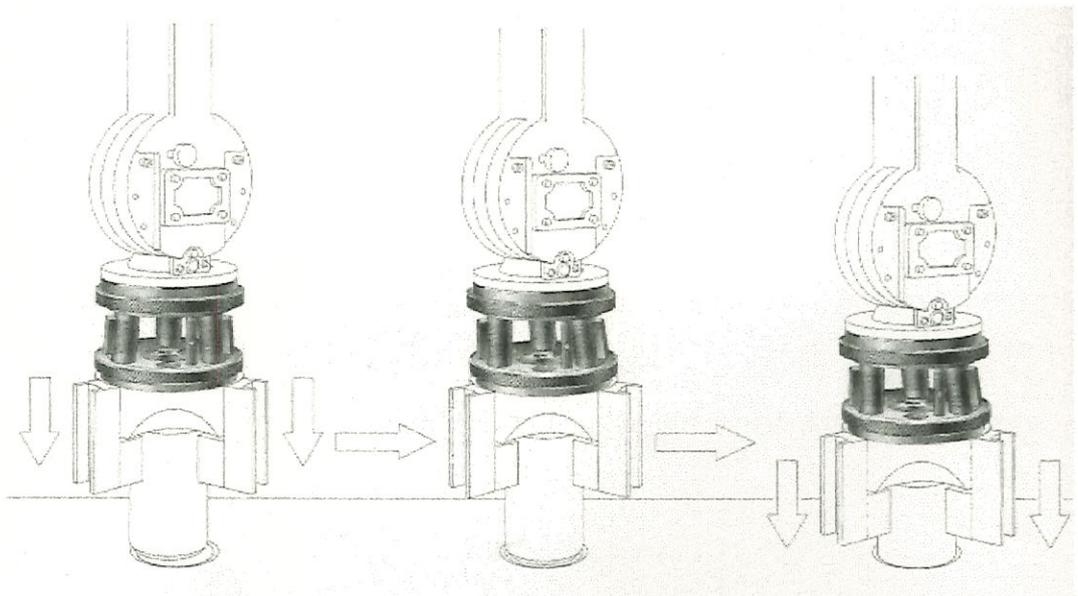


FIGURA 3 – Processo de montagem, no qual é permitido um certo deslocamento ou uma certa rotação da peça a ser montada, que desta forma se acomoda no alojamento

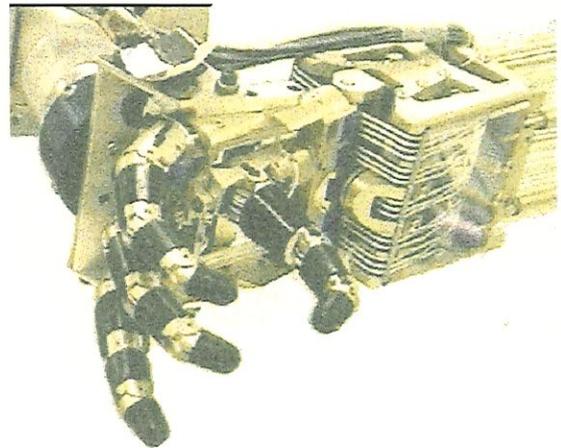
1.3. Garras antropomórficas

As garras industriais são dispositivos restritos a certas operações ou peças. As garras antropomórficas, ou garras de múltiplos dedos, estão sendo desenvolvidas para aumentar a flexibilidade e a eficiência das operações de manipulação. As principais garras foram desenvolvidas no Japão, Estados Unidos e na Europa, apresentando diversas configurações (número de dedos e graus de liberdade por dedo) e tipos de acionamento (elétrico, pneumático e hidráulico). A seguir é realizada uma revisão dos principais trabalhos de pesquisa envolvendo garras de múltiplos dedos.

As primeiras garras foram desenvolvidas utilizando o princípio de atuação por cabo. Este princípio apresenta boa acuracidade de posicionamento, movimentos rápidos e possibilita uma redução do peso próprio da garra. Porém, as garras acionadas por cabos não são muito fáceis de controlar e podem apresentar folgas com o tempo. Os principais exemplos são as garras de STANFORD/JPL e UTAH/MIT, apresentados na figura abaixo.



(A) Garra de STANFORD/JPL
3 dedos, 3 graus de liberdade por dedo
Acionada por 12 cabos revestidos com teflon
12 sensores de tensão dos cabos



(B) Garra de UTAH/MIT
4 dedos, 4 graus de liberdade por dedo
Acionada por 32 tendões poliméricos
32 sensores de tensão dos cabos

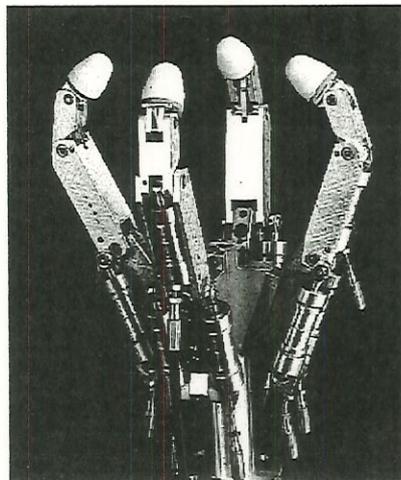
FIGURA 4 – Exemplos de garras acionadas por cabo

Buscando manter o baixo peso próprio e evitar os problemas de montagem dos cabos, foram desenvolvidas garras hidráulicas e pneumáticas, sendo que esta última apresenta uma melhor relação peso/potência de atuação. Os compressores são colocados na base do robô e apenas as linhas de pressão são levadas até a garra.

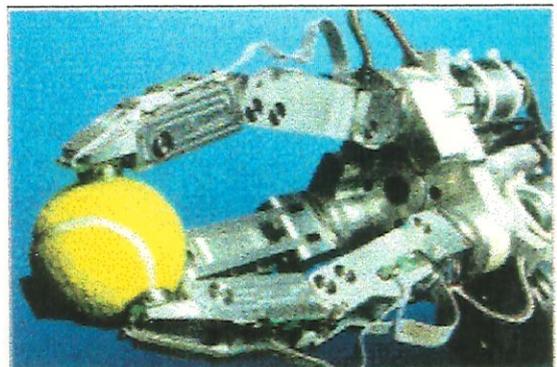
Para a garra do projeto *Advanced Manipulation for Underwater Sampling – AMADEUS*, destinada a operações no fundo do mar, foi desenvolvido um novo sistema pneumático a fim de suportar as condições do ambiente proposto. Os sistemas de pistões convencionais estariam sujeitos a problemas de vedação, corrosão e sujeira.

A garra desenvolvida na Technical University of Munich (MENZEL, 1994; PFEIFFER, 1996) é acionada hidráulicamente, através de quatro pistões montados no dedo (FIGURA 5A).

A garra desenvolvida na TU-Delft (JONGKIND, 1993; OVERDIJK, 1997) (FIGURA 5B) apresenta um acionamento misto: pneumático para o movimento das falanges, elétrico para o movimento longitudinal dos dedos e hidráulico para a rotação da palma. Esta garra fazia parte do projeto TELEMAN, que visava desenvolver um sistema de manipulação capaz de operar em centrais nucleares. O desafio era vencer as condições de radioatividade, que inviabilizava a utilização de sensores com eletrônica ativa incorporada. Tornou-se necessário o desenvolvimento de sensores passivos¹ de posição e sensores de tato que suportassem tais condições.



(A) LBM – Munique
4 dedos, 3 graus de liberdade por dedo
4 atuadores hidráulicos por dedo
Estrutura elástica na ponta dos dedos



(B) DELFT
3 dedos, 3 graus de liberdade por dedo
Acionamento hidráulico e elétrico do dedo
Acionamento pneumático da palma giratória
6 sensores passivos de posição (LVDTs)
3 sensores matriciais de tato

FIGURA 5 – Exemplo de garras de acionamento hidráulico e pneumático.

¹ Sensores passivos são aqueles que não imitem sinal, apenas recebem a informação. Já os sensores ativos dependem de uma emissão e medem a reflexão do sinal. Exemplo: sensor ultra-som (ativo), microfone (passivo).

A FIGURA 6A apresenta a garra desenvolvida no DLR (HIRZINGER, 1993; DLR ARTICULATED HAND) que representa um dos mais complexos sistemas de manipulação já construídos. Esta garra foi implementada para substituir as limitadas garras de dois dedos ainda utilizadas nos ônibus espaciais americanos. Com o objetivo de permitir manipulações mais complexas, ela impressiona pela configuração de quatro dedos, doze graus de liberdade e principalmente vinte e cinco sensores por dedo, incluindo sensores de força, posição e temperatura, além de uma pequena câmera instalada na palma da mão. Outra importante característica é que todo o sistema de acionamento está integrado junto à palma ou diretamente aos dedos, sem aumentar demasiadamente o peso próprio da garra (1,8 Kg).



(A) Garra DLR
4 dedos, 3 graus de liberdade por dedo
Acionamento (acoplado à palma)
25 sensores por dedo (tato, temperatura, força)



(B) Prótese de mão de Southampton
5 dedos, 4 graus de liberdade totais
Atuadores, sensores e eletrônica incorporada
Baixo consumo de energia

FIGURA 6 – Foto de duas das mais avançadas garras antropomórficas já construídas

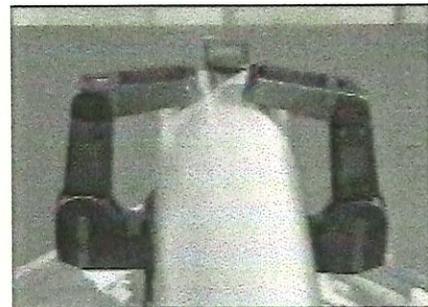
A FIGURA 6B apresenta a garra desenvolvida na Universidade de Southampton (SOUTHAMPTON ARTIFICIAL HAND), que segue uma outra filosofia de aplicação: o ramo de próteses de mão. Este ramo apresenta alguns pontos críticos como tamanho, peso, consumo dos atuadores e principalmente interface homem-máquina. O cérebro utiliza uma vasta quantidade de informações providas da mão (tato, temperatura, força, etc) para controlar os reflexos musculares e permitir uma fixação segura. As mãos artificiais não conseguem estabelecer esta interface e trabalham sem qualquer realimentação. Devido a estes empecilhos, as atuais próteses comerciais são extremamente limitadas, se restringindo ao movimento de pinça entre os dedos e o polegar e apresentando baixa capacidade de evitar o escorregamento do objeto fixado. Neste contexto, a mais avançada prótese de mão foi desenvolvida em Southampton para aumentar a capacidade de manipulação dos sistemas

artificiais. Dotada de cinco dedos e quatro graus de liberdade, a prótese inteligente de Southampton utiliza sensores de força e tecnologia microprocessada para se auto-ajustar aos parâmetros de fixação e evitar escorregamentos.

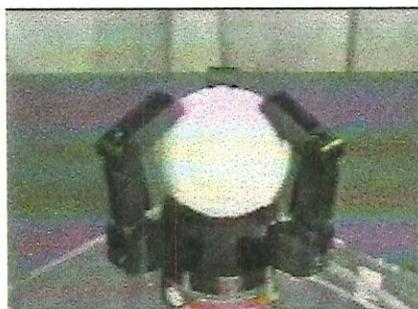
Mais recentemente, a empresa americana Barrett Technology Inc. desenvolveu uma das primeiras garras com definitivo potencial para aplicações industriais. A BarrettHand (TOWNSEND, 1999) apresenta uma arquitetura compacta, leve (1,18Kg) e totalmente integrada, de forma que todos os dispositivos mecânicos e eletrônicos são montados na base da mesma. Esta garra foi implementada numa configuração de três dedos (dois graus de liberdade cada), sendo que dois destes dedos apresentam um grau de liberdade extra, responsável pela sua movimentação lateral (FIGURA 7). Tal mobilidade adicional possibilita que a garra assuma diversas configurações de fixação. Contudo, apenas quatro motores são responsáveis pelo acionamento de todos os oito graus de liberdade, utilizando um mecanismo sub-atuado por cabos. Tal tecnologia permite reduzir consideravelmente o esforço computacional exigido, bem como compactar o sistema de acionamento.



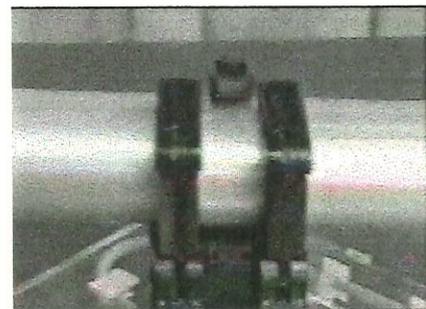
(A) Configuração a 90°



(B) Configuração a 90°



(C) Configuração a 120°

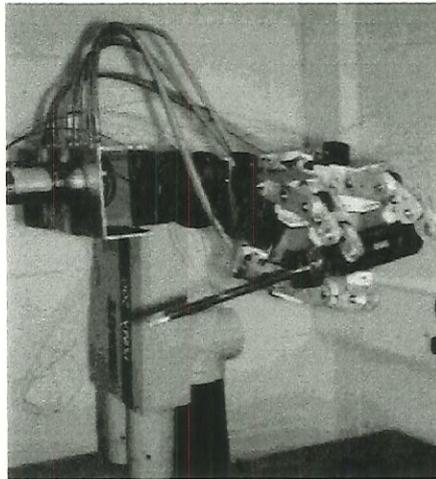


(D) Configuração a 180°

FIGURA 7 – Diversas configurações de fixação permitidas pela BarrettHand. Estas figuras demonstram a versatilidade adquirida através da mobilidade lateral de dois dos três dedos.

Por fim, vale ressaltar que alguns sistemas foram desenvolvidos seguindo um padrão mais simplificado, porém sem grande perda de funcionalidade (FIGURA 8). A garra

GRASPAR, desenvolvida na *Northeastern University* – Boston (CRISMAN, 1996; GRASPAR HAND), representa um projeto extremamente simplificado e também sub-atuado por cabo. Assim, utilizando-se apenas um motor, são acionados todos os três graus de liberdade de cada dedo. Já a garra desenvolvida no ETHZ – ZURIQUE apresenta uma configuração bastante simplificada de dedos paralelos e acionamento por fuso e motor elétrico. Porém, o sistema neural utilizado para controle possibilita um funcionamento inteligente.



(A) Garra GRASPAR
 2 dedos, 3 graus de liberdade cada
 1 polegar com 2 graus de liberdade
 Acionamento por 6 cabos
 8 sensores de contato (1 em cada falange)



(B) Garra ETHZ – Zurich
 3 dedos paralelos, 3 graus de liberdade
 Acionamento por fuso e motor elétrico
 3 sensores de força capacitivos
 Controle por redes neurais

FIGURA 8 – Garras de formato simplificado porém sem grande perda de funcionalidade.

Estas duas últimas garras, apesar da configuração simplificada, permitem também manipulações complexas. Assim, elas serão utilizadas como ponto de partida para o trabalho proposto nesta dissertação.

1.4. Planejamento dos parâmetros de fixação

O processo de fixação de uma peça envolve parâmetros físicos e geométricos: formato da peça, equilíbrio das forças, contato com atrito, evitar escorregamentos e colisões. Diversas pesquisas estão focadas em alguns destes aspectos, porém uma solução completa ainda está muito longe, principalmente quando comparada com o comportamento da mão humana. Assim, definir uma estratégia eficiente de fixação não envolve somente o cálculo das forças de contato para cada dedo, mas também determinar os pontos onde ocorrerão o

contato e a forma com que estes pontos serão atingidos. Deve-se conseguir posicionar a garra para fixar o objeto de forma eficiente.

Diversas abordagens foram propostas. Algumas eram baseadas em processos analíticos que buscavam uma representação matemática do problema de fixação. Outras eram baseadas em estudos sobre o comportamento da mão humana e buscavam algoritmos para prever o tipo de fixação a ser adotado por um homem.

A seguir é apresentada uma revisão de diversas abordagens do problema de fixação.

1.5. Processos analíticos de programação *off-line*

Estes métodos são baseados em uma prévia determinação matemática dos pontos de contato, para, só então, ser conduzido o processo de fixação. Para uma melhor compreensão destes métodos, é essencial a definição de alguns conceitos:

Fixação do tipo *force-closure*: ocorre quando qualquer carregamento externo pode ser balanceado pela combinação linear de forças aplicadas nos respectivos pontos de contato.

Cone de atrito: definido por um cone centrado no ponto de contato e com ângulo θ expresso pela tangente do coeficiente de atrito μ (FIGURA 9). Toda força de contato localizada dentro do cone de atrito garante uma fixação sem escorregamento.

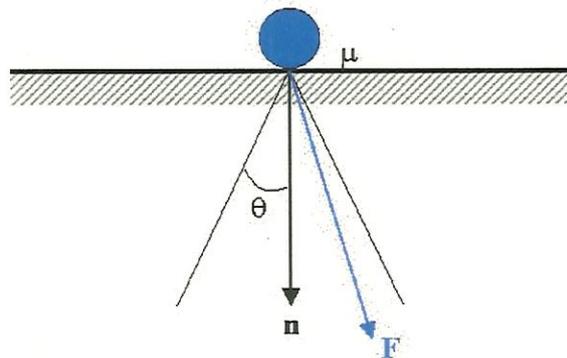


FIGURA 9 – Representação geométrica do cone de atrito. O ângulo de inclinação θ é igual à tangente do coeficiente de atrito.

Neste contexto, NGUYEN (1987, 1988) desenvolveu um dos primeiros trabalhos para definição dos pontos de contato de forma a realizar uma fixação tipo *force-closure*. Através de uma análise geométrica, definem-se duas regiões do contorno da peça, sendo que o contato nestes pontos resulta em uma *force-closure*. Desta forma, ao invés de estipular

pontos de contato, o algoritmo determina regiões de contato, de forma a reduzir a acuracidade necessária para o posicionamento do robô, tendo como alvo o ponto médio destas regiões. O algoritmo proposto era baseado na seguinte preposição: dois pontos de contato resultam numa *force-closure* se a força aplicada em um ponto estiver dentro do seu cone de atrito e aponta para o cone de atrito do outro ponto (FIGURA 10). Assim, o método de NGUYEN representa uma solução simples para uma garra de dois dedos e para uma análise bidimensional.

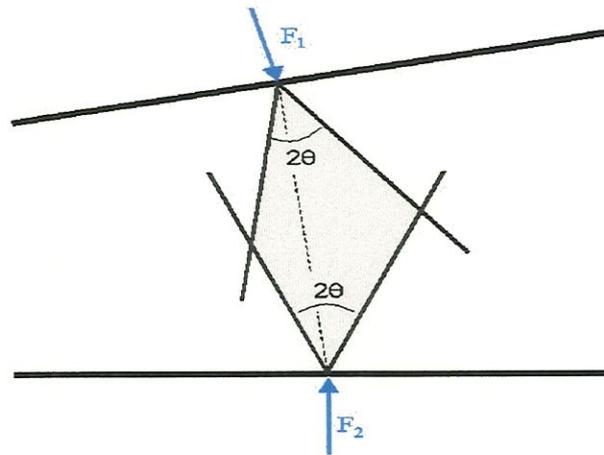


FIGURA 10 – Algoritmo de fixação proposto por NGUYEN. Os cones de atrito estão sempre posicionados frente a frente ou de costas um para o outro.

Contudo, o algoritmo proposto por NGUYEN é lento, computacionalmente pesado e incapaz de responder a certos tipos de polígono. Neste contexto, TUNG & KAK (1996) propuseram um método mais eficiente para definição de regiões de contato para polígonos arbitrários (inclusive polígonos côncavos) e com propriedades arbitrárias de materiais (diferentes coeficientes de atrito). O algoritmo baseia-se em uma busca exaustiva por pares de regiões que conduzem a uma fixação *force-closure*.

PONCE & FAVERJON (1995) propuseram uma extensão do algoritmo de NGUYEN para garras de três dedos. As condições geométricas do algoritmo foram equacionadas e foi proposto um novo processo de otimização linear para determinar as três regiões de contato, substituindo assim os processos de busca heurística anteriores. Através da análise das normais (devem abranger positivamente todo o espaço) e dos cones de atrito (intercessão dos cones não deve ser vazia), pode-se definir se a fixação é do tipo *force-closure*. O algoritmo estabeleceu boas respostas para objetos de formato complexo (FIGURA 11A), porém, apresentou algumas falhas em condições relativamente simples (FIGURA 11C).

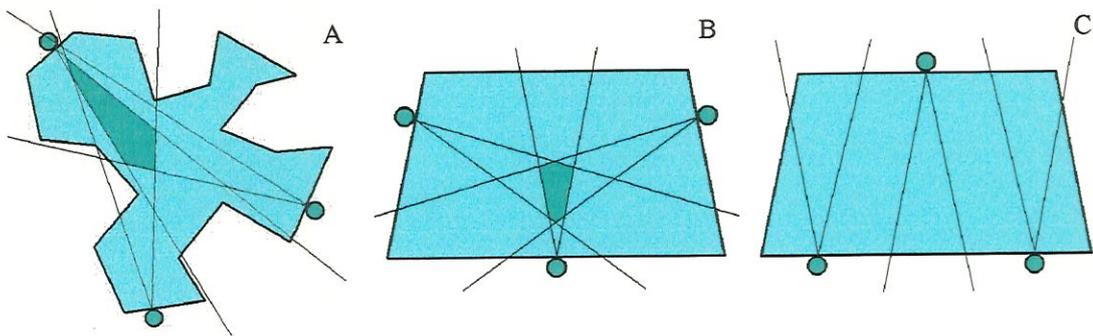


FIGURA 11 – Algoritmo de fixação proposto por PONCE: (A) Fixação proposta para uma figura de formato complexo; (B) Fixação proposta para o objeto de formato losangular; (C) Fixação considerada instável pelo algoritmo, uma vez que não ocorre intercessão entre os cones de atrito.

Todos os algoritmos anteriores apontavam pares (NGUYEN e TUNK) ou trios (PONCE & FAVERJON) de regiões que conduziam a uma *force-closure*, mas não indicavam qual era a melhor. Porém, FERRARI & CANNY (1992) já havia desenvolvido certos critérios para classificar a melhor fixação. Um dos critérios considerava como melhor fixação aquela que possuía as menores forças. O objetivo era minimizar as deformações dos dedos e da peça devido aos esforços aplicados. Outro critério escolhia aquela fixação que possuía a menor soma das forças aplicadas. O objetivo era reduzir a potência requerida dos atuadores, uma vez que a corrente nos motores é proporcional à magnitude das forças. Desta forma, os critérios propostos por FERRARI & CANNY determinavam as melhores fixações dentro de um conjunto de fixações estáveis.

WOELFL (1994a, 1994b) e PFEIFFER (1996) propuseram representar o problema de configuração da fixação como um problema clássico de otimização (TABELA 1). Como função objetivo foi utilizado um outro critério de menor diferença entre as magnitudes das forças. As condições de contorno buscavam garantir uma *force-closure*: condição de equilíbrio (somatório das forças e dos momentos é igual às forças e aos momentos resultantes), condição de contato (forças normais devem empurrar e não puxar o objeto), condição de não-escorregamento (forças aplicadas por cada dedo devem estar dentro do respectivo cone de atrito), condições geométrica da garra (manter uma distância mínima entre os pontos de fixação) e condição de estabilidade (quanto menor o somatório dos vetores normais mais estável é a fixação). O resultado do algoritmo é a definição dos pontos e das forças de contato que conduzem a uma fixação estável. O ponto negativo desta abordagem é que ela assume como dado de entrada a descrição matemática da superfície do objeto, que em alguns casos pode requerer uma definição analítica complexa.

TABELA 1 – Processo de otimização para definição dos pontos e das forças de contato

Função objetivo	$G = \sum_{i=1}^n \sum_{j=1(j \neq i)}^n \left(f_i ^2 - f_j ^2 \right)^2 \rightarrow \text{mínimo}$
Condições de Contorno	
Equilíbrio de força	$\sum_{i=1}^n (f_{n_i} + f_{t_i}) - F_e = 0$
Equilíbrio de momento	$\sum_{i=1}^n r_i \cdot (f_{n_i} + f_{t_i}) - M_e = 0$
Contato	$f_{n_i} \cdot n_i < 0$
Escorregamento	$ f_{t_i} ^2 - \mu^2 f_{n_i} ^2 < 0$
Estabilidade	$\left \sum_{i=1}^n n_i \right \leq S$
Geometria da garra	$ r_i - r_j - \varepsilon_{\min} \geq 0 \quad i \neq j$

Na TABELA 1, f_n e f_t representam as forças normais e tangenciais de contato, F_e e M_e as forças e momentos resultantes, S uma medida de estabilidade, r_i o vetor posição de cada um dos contatos e ε_{\min} a distância mínima entre os dedos.

Todos os algoritmos apresentados acima baseiam-se numa programação *off-line* da fixação, ou seja, os pontos e forças de contato são determinados *a priori*. Estes dados são repassados para a garra e inicia-se o processo de fixação. Porém, existem métodos que definem a configuração do contato à partir de tentativas *on-line* baseadas em informações provindas de sensores posicionados na garra. Estes métodos são destacados na próxima seção.

1.6. Processos analíticos de programação *on-line*

KAMON, FLASH & EDELMAN (1994) propuseram um algoritmo de busca pela melhor forma de fixação à partir de informações visuais. Dado um objeto, o sistema extrai uma imagem estacionária usando uma câmera de topo. Após uma etapa de processamento da imagem, comparam-se as informações extraídas (ângulo entre os dedos e as normais, distância entre o centro de gravidade e a linha que conecta os pontos de contato, distância entre os pontos de contato e o eixo de simetria, etc) com as informações armazenadas em um banco de dados, definem-se dois pontos iniciais de contato e executa-se a primeira tentativa de fixação da peça. Para analisar a qualidade desta fixação, uma segunda câmera lateral

fornece informações relativas ao comportamento da peça (ocorrência de escorregamento ou rotação). Se a fixação for considerada boa, suas informações são incorporadas ao banco de dados. Caso contrário, novos pontos de contato são tentados até se atingir uma fixação aceitável. Assim, o algoritmo peca por sua lentidão e também pela probabilidade de falhas geradas pelas imprecisões dos dados de visão.

COELHO & GRUPEN (1993, 1994, 1997) propuseram outro processo *on-line* baseado em sensores de tato, que informam os pontos de contato e as direções normais nestes pontos. À partir destas informações, é possível calcular os coeficientes de atrito necessários para uma fixação sem escorregamento. Desta forma, são realizadas sucessivas tentativas de fixação, através de constantes ajustes das coordenadas de contato, em busca de uma configuração de pontos que exija os menores coeficientes de atrito. O ponto negativo deste método é que ele requer um número elevado de tentativas para se alcançar a convergência.

1.7. Processos baseados no comportamento da mão humana

Enquanto o ser humano é capaz de fixar objetos de inúmeros formatos, a fim de realizar inúmeras tarefas e sob diferentes condições ambiente, os atuais sistemas robóticos permitem apenas manipulações de objetos simples sob condições especiais de laboratório. Embora as pesquisas no ramo de garras robóticas tenham obtido grande avanço, principalmente nas áreas de estratégia de fixação, sensoreamento e controle de força, elas estão longe da tarefa de construir um sistema capaz de decidir automaticamente qual estratégia utilizar para a fixação de objetos do dia-a-dia. Diversas simplificações e suposições são adotadas pelos métodos analíticos de forma a possibilitar uma representação matemática do problema, porém estas simplificações distanciam o sistema da solução real.

CUTKOSKY (1989) propôs um estudo do comportamento da mão humana para melhor entender como os requisitos da fixação (força, precisão, segurança) e a geometria do objeto influenciam na escolha do tipo de fixação. Assim, a fixação utilizada para pegar um lápis é diferente da fixação utilizada para escrever, embora o objeto seja o mesmo. Ainda, a fixação utilizada para pegar um objeto plano é diferente da utilizada para pegar um objeto redondo, mesmo que eles tenham o mesmo peso. Face a estas diferenças, foi montado um diagrama (FIGURA 12) com os principais tipos de fixação adotados, nas mais diversas situações.

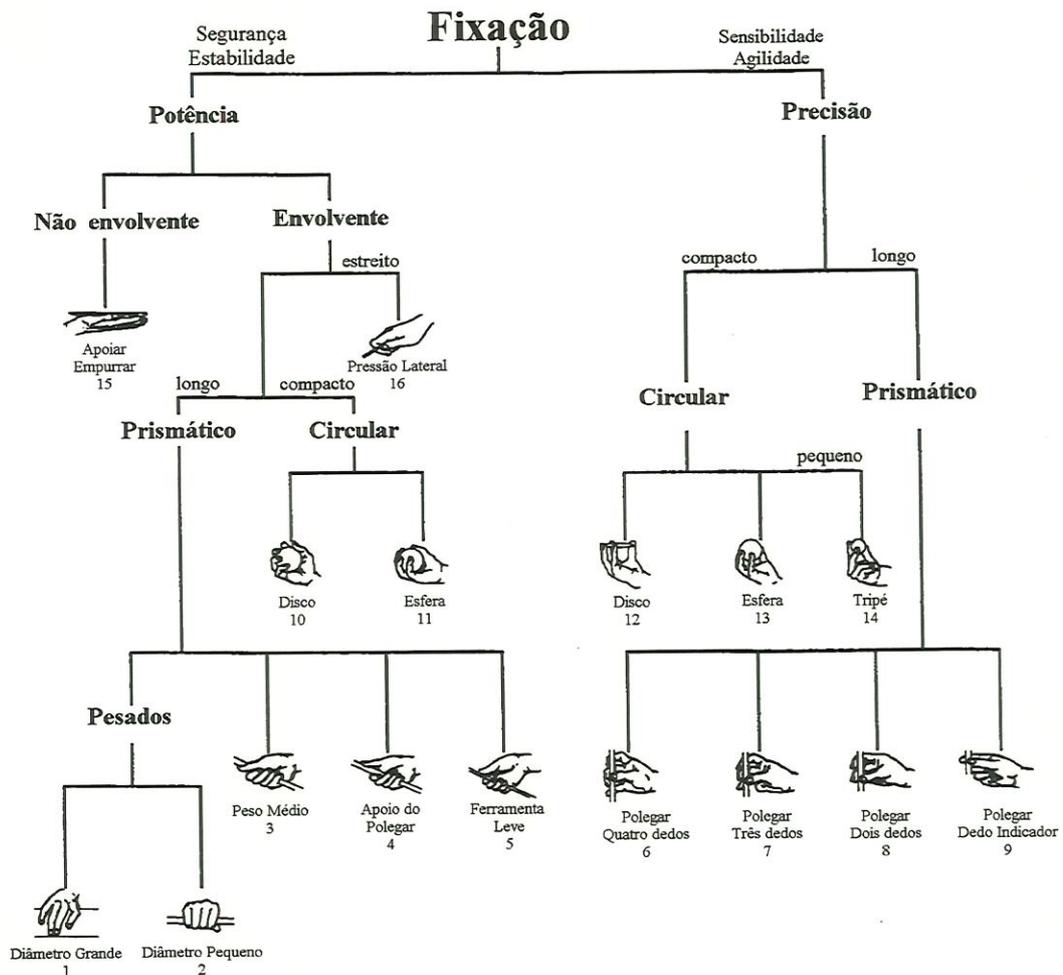


FIGURA 12 – Classificação dos tipos de fixação adotados pela mão humana. Estas fixações são basicamente divididas em fixações de potência (referentes a estabilidade e segurança) e fixações de precisão (referentes a sensibilidade e agilidade).

Neste diagrama, as fixações são basicamente divididas em dois grupos: potência e precisão. Analisando da esquerda para a direita, as fixações vão ficando cada vez mais precisas e os objetos cada vez menores. Analisando de cima para baixo, tanto as tarefas como a geometria dos objetos vão ficando cada vez mais detalhadas. Porém, durante a execução de uma tarefa, pode ocorrer uma alteração da fixação devido à mudança das condições exigidas. Considerando a tarefa de desrosquear uma lâmpada, a mesma é iniciada com a fixação 11 (FIGURA 12) e finalizada com a fixação 13.

Utilizando este diagrama, CUTKOSKY desenvolveu um método para identificar o tipo ideal de fixação, segundo a tarefa a ser realizada. Através de um questionário composto por cinquenta perguntas (se a fixação requer grandes forças, quanto de estabilidade requer o sistema, quão importante é a segurança da fixação, etc), são identificadas uma ou mais fixações que atendem aos requisitos da tarefa. Sob certas circunstâncias, o sistema consegue prever o tipo de fixação que as pessoas irão adotar.

CUTKOSKY propôs ainda uma nova filosofia para construção de garras industriais. Ao invés de serem projetadas para se ajustarem à geometria da peça a ser fixada, o que resulta em inúmeras trocas de garras, elas deveriam ser projetadas segundo o tipo de fixação a ser desempenhada. Assim, uma mesma garra fixaria todas as peças pertencentes a uma mesma classe de fixação.

IBERALL (1997) propôs que a análise da fixação fosse realizada sob a ótica de dedos virtuais, que são representações abstratas para cada conjunto de dedos e palma que exercem força em uma determinada direção. Assim, para uma fixação do tipo envolvente, pelo menos dois dedos virtuais devem ser aplicadas em direções opostas. Como resultado destas análises, foram definidas três configurações básicas ou primitivas da mão humana para uma fixação envolvente: bloco, palma e oposição lateral. A FIGURA 1.13A ilustra uma configuração do tipo bloco, com dois dedos virtuais representados pelo dedo indicador e pelo polegar. A FIGURA 1.13B ilustra uma configuração do tipo palma, com dois dedos virtuais: um representado pelo conjunto de quatro dedos e outro representado pela palma da mão. A FIGURA 1.13C ilustra uma configuração do tipo oposição lateral, onde os dedos virtuais são representados pelo polegar e pela lateral do dedo indicador. Todas as demais configurações de fixação são combinações destas três primitivas. Tal como CUTKOSKY, IBERALL buscava prever o tipo primitivo ou quais combinações de tipos primitivos de fixação seriam utilizadas por um homem em fixações do dia-a-dia. Para isto, era apresentada uma seqüência de quarenta perguntas que buscavam levantar os principais requisitos da fixação.

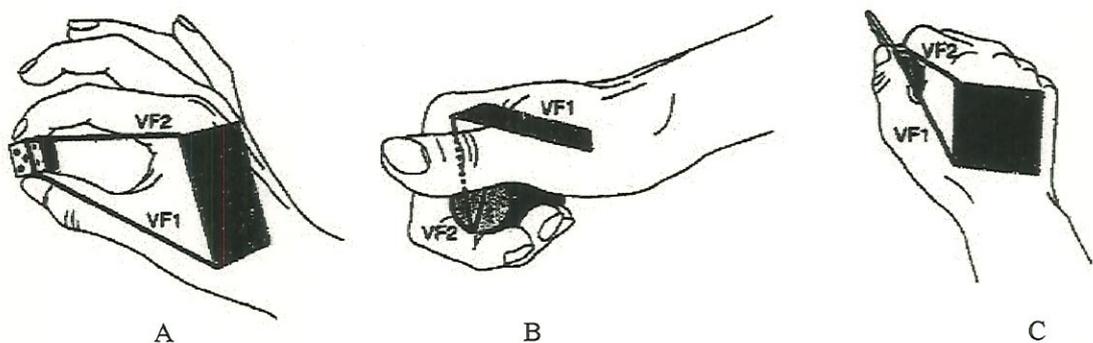


FIGURA 13 – Configurações primitivas da mão humana para fixações envolventes:
 (A) configuração do tipo bloco; (B) configuração do tipo palma;
 (C) configuração do tipo oposição lateral.

Desta forma, os métodos anteriores baseavam-se em um questionário para definir o tipo de fixação a ser utilizada. Porém, existe uma grande dificuldade de responder adequadamente a estas perguntas, uma vez que é complicado quantificar parâmetros como

sensibilidade e precisão. Uma maneira de representar estes parâmetros seria armazenar exemplos específicos de cada conceito em particular, para depois compará-los com as características da fixação a ser analisada. Esta solução abre um espaço para a aplicação de redes neurais que, uma vez treinadas por padrões específicos, podem distinguir características semelhantes e apontar o tipo de fixação mais adequado.

Partindo-se desta idéia, MOUSSA & KAMEN (1995) desenvolveram um sistema neural para identificar o melhor tipo de fixação a ser adotado. A entrada do sistema, composto por uma rede neural MLP, é baseada em três informações: um parâmetro que especifica a habilidade requerida para a fixação, além de outros dois parâmetros que informam o tamanho e o formato da peça a ser capturada. Assim, o ponto negativo deste algoritmo é que este ainda depende de um parâmetro difícil de quantificar: a habilidade da fixação.

A idéia de utilizar redes neurais para identificar não só o tipo de fixação como também os pontos de contato continuou sendo desenvolvida. A próxima seção destaca o algoritmo utilizado para controle da garra desenvolvida no ETHZ – ZURIQUE (FIGURA 8B).

1.8. Definição dos pontos de contato utilizando redes neurais

A garra desenvolvida no ETHZ – Zurique compensava a sua simplicidade com a utilização de sistema neural de controle. Através de um modelo formado por duas redes neurais distintas, eram definidos três pontos de contato para um objeto de formato desconhecido.

Assim, SCHERRER (1993) propôs dois estágios de processamento (FIGURA 14). Um estágio *off-line* calcula os pontos de fixação através de um processo de otimização, utilizando para isto uma rede neural de Hopfield (SEÇÃO 5). Esta rede gera um conjunto de soluções para diferentes formatos de objeto: quadrados, triângulos, círculos, etc. Este conjunto de soluções é utilizado para o treinamento de uma outra rede neural: rede *Multilayer Perceptron* – MLP (APÊNDICE A1.5). Através deste treinamento, também *off-line*, a rede MLP “assimila” as configurações de fixação calculadas pela Hopfield. Todo o processo de cálculo dos pontos de contato e treinamento da rede MLP é bastante lento (cerca de 20 horas) e, portanto, é realizado anteriormente a qualquer operação da garra. Uma vez treinada, a rede MLP está pronta para a tarefa em questão. Fornecido o contorno do objeto a ser capturado, a rede determina, em tempo real, os três pontos de contato.

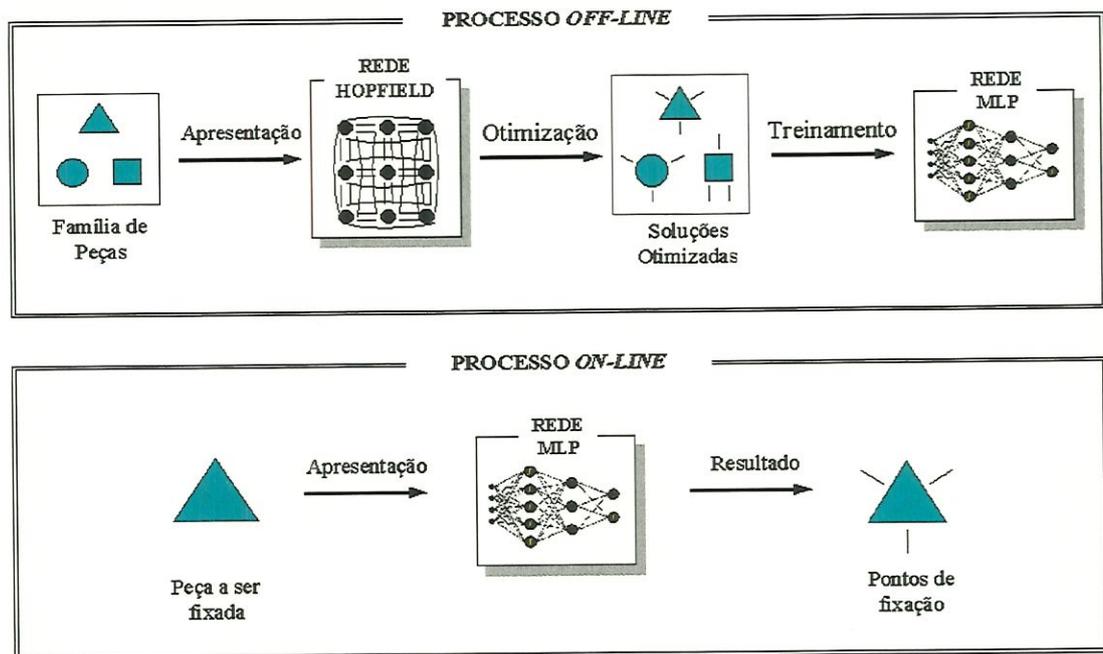


FIGURA 14 – Algoritmo de fixação proposto por SCHERRER, envolvendo uma etapa *off-line* de treinamento da rede MLP e uma etapa *on-line* já com a rede treinada.

Contudo, o algoritmo de SCHERRER ainda apresenta algumas falhas. A FIGURA 15 mostra a solução inadequada definida para um objeto próximo de um quadrado. Este exemplo demonstra a baixa capacidade de generalização adquirida pela rede neural.

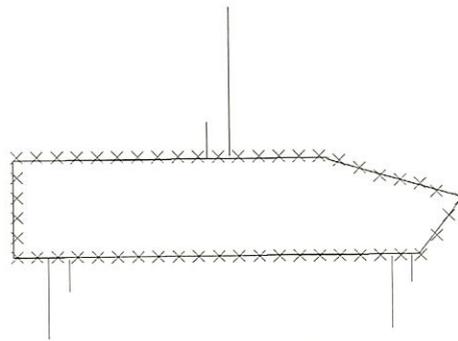


FIGURA 15 – Falha do algoritmo de SCHERRER para um objeto diferente dos padrões utilizados para treinamento da rede MLP.

1.9. Formalização do problema



FIGURA 16 – Ilustração do sistema de fixação proposto. O sistema deverá ser capaz de capturar objetos arbitrários dentro do espaço de trabalho do robô.

Como ilustrado na FIGURA 16, pretende-se implementar um sistema de manipulação capaz de utilizar uma garra, acoplada ao punho de um robô, para capturar objetos de formato arbitrário. O sistema utiliza uma câmera CCD como única fonte de informação dos dados da cena. Estes dados são processados por um microcomputador, que, através de algoritmos de processamento de imagem e de planejamento da fixação, determina os melhores pontos de contato entre os dedos da garra e a peça alvo. Considerando a utilização de uma garra de três dedos, este sistema deve determinar três pontos de contato que garantam uma fixação estável. Por fim, o sistema deve estimar as forças a serem aplicadas em cada ponto de contato e detectar possíveis escorregamentos do objeto, corrigindo estas forças.

Dentro deste contexto, este trabalho tem o objetivo de implementar as etapas de processamento de imagem e planejamento da fixação. Um importante requisito deste sistema é que ele apresente um funcionamento *on-line*, capturando objetos assim que estes sejam posicionados dentro do espaço de trabalho do robô. Tal objetivo impõe a utilização de eficientes algoritmos de processamento. Outro requisito é que a garra deverá fixar objetos de diferentes formas, materiais e pesos, independente de sua posição e orientação. A etapa de controle das forças aplicadas pela garra será desenvolvida em trabalhos futuros.

Nas seções anteriores, foram apresentados diversos algoritmos para planejamento dos parâmetros de fixação. Porém, nenhum atende às exigências de processamento em tempo real e manipulação de objetos de formato arbitrário. Os algoritmos analíticos de processamento *off-line*, na sua maioria baseados em processos iterativos de busca pela melhor solução, necessitam da definição matemática do contorno do objeto e de um grande esforço computacional. Desta forma, estes algoritmos não atendem à exigência de

processamento em tempo real. Os métodos analíticos *on-line* são igualmente lentos, uma vez que necessitam de um grande número de tentativas para atingir uma fixação estável. Os métodos baseados no comportamento da mão humana pecam pela necessidade de quantificar parâmetros abstratos, como precisão e sensibilidade. Finalmente, o método neural proposto por SCHERRER apresenta repostas rápidas, porém não é capaz de processar informações diferentes daquelas utilizadas para treinamento da rede neural.

Desta forma, propõe-se um algoritmo que seja realmente capaz de planejar fixações estáveis, em tempo real e que seja aplicado a objetos de diferentes formatos.

1.10. Algoritmo proposto

O algoritmo proposto neste trabalho é composto por três etapas sequenciais de processamento, conforme ilustra a figura abaixo.

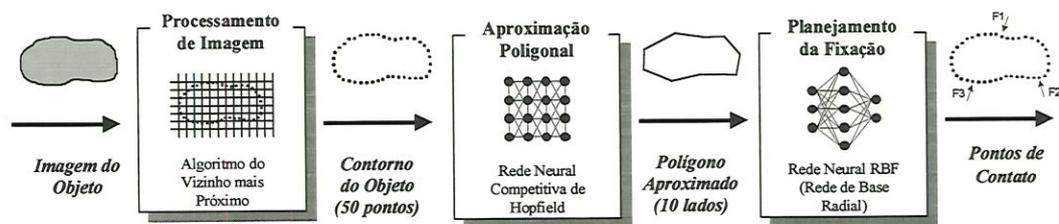


FIGURA 17 – Etapas de processamento do algoritmo proposto. Uma primeira etapa é responsável pelo processamento da imagem e extração do contorno do objeto. Uma segunda etapa define um polígono aproximado sobre os pontos deste contorno. Por fim, a etapa de planejamento da fixação determina os três melhores pontos de contato.

A primeira etapa representa o sistema de visão, que parte de uma imagem de topo capturada por uma câmera CCD. Utilizando o método do vizinho mais próximo, extrai-se o conjunto de pontos que definem o contorno do objeto. Este contorno é formado por um número finito de pontos que depende da resolução da imagem fornecida pela câmera. Um algoritmo de segmentação escolhe cinquenta pontos deste conjunto, suficientes para representar o formato do objeto sem perder nenhuma informação relevante.

Estes cinquenta pontos são utilizados como entrada da segunda etapa de processamento: uma Rede Neural Competitiva de Hopfield. Esta rede é responsável pela definição de um polígono aproximado sobre os pontos do contorno do objeto. Com isto, procura-se simplificar a análise e facilitar o treinamento da segunda rede neural.

Por fim, este polígono é fornecido para uma Rede de Funções de Base Radial (RBF), responsável pela definição dos três pontos de contato. Esta rede é caracterizada por um rápido processo de treinamento, o que possibilita, inclusive, a sua implementação *on-line*.

1.11. Estrutura da dissertação

O algoritmo proposto é detalhado nos capítulos seguintes da dissertação. No Capítulo 2, é introduzido o conceito de processamento de imagem. São comentadas diversas aplicações deste processamento, porém, é destacada a tarefa de detecção das bordas de objetos em uma cena. Diversos métodos de detecção de borda são apresentados e comparados, sendo o método do vizinho mais próximo (*nearest neighbor*) o mais adequado às condições da operação proposta: considerando apenas um objeto na cena, extrair o contorno do mesmo num tempo reduzido de processamento.

O Capítulo 3 inicialmente apresenta o modelo de rede neural proposto por Hopfield para operações de otimização. Este modelo é modificado a fim de executar a tarefa de aproximação poligonal sobre os pontos do contorno da peça. Define-se assim o modelo Competitivo de Hopfield, cujas respostas são apresentadas no final do capítulo.

O Capítulo 4 apresenta a rede de Funções de Base Radial. Esta rede será responsável pela geração dos três pontos de fixação da garra. São ainda apresentados e comparados os diversos algoritmos de treinamento, sendo o algoritmo Global Ridge Regression o mais adequado para a tarefa em questão.

O Capítulo 5 expõe as diversas etapas de processamento para a definição dos pontos de fixação da garra, partindo-se da imagem digitalizada do objeto a ser capturado. Os resultados parciais são destacados após cada etapa. São ainda calculados pontos de fixação para objetos de manipulação do dia-a-dia, demonstrando a habilidade do sistema em lidar com elementos de formato desconhecido.

Finalmente, o Capítulo 6 apresenta uma discussão dos resultados apresentados tanto pelo módulo de processamento de imagem como pelo módulo neural. São ainda apresentadas propostas futuras para o aprimoramento do trabalho.

Em anexo, é apresentado um algoritmo global de otimização utilizado para cálculo dos pontos ótimos de contato. Este algoritmo analisa todas as possíveis soluções do problema e aponta para um conjunto de soluções otimizadas, que representam as melhores configurações de fixação.

Em apêndice, é apresentada uma introdução às redes neurais artificiais. Esta inicia-se com o estudo do funcionamento do cérebro biológico. A partir da compreensão de algumas atividades cognitivas básicas, é introduzido o conceito de rede neural artificial e o processo de treinamento destas. Neste contexto, é apresentada a rede *Multilayer Perceptron* (MLP) e o algoritmo de treinamento supervisionado *back-propagation*.

2. PROCESSAMENTO DE IMAGEM

2.1. Introdução ao processamento de imagem

O uso de sistemas de visão artificial está se estendendo hoje às mais diversas atividades. Seu principal objetivo é criar um modelo do mundo real através de imagens, obtendo informações da cena à partir de sua projeção bidimensional. Com o processamento destas imagens, pode-se analisar tumores através de uma tomografia computadorizada, realizar inspeções de qualidade nas áreas agrícolas e industriais ou mesmo mapear uma região para a movimentação de um AGV (*Autonomous Guided Vehicle*).

As técnicas de processamento normalmente transformam uma imagem capturada em uma outra imagem processada (JAIN et al., 1995). Este processamento pode ser uma filtragem da imagem capturada fornecendo uma imagem sem ruído; uma correção de uma imagem borrada originária de uma filmagem fora de foco; um *zoom* em uma determinada parte da cena; ou mesmo a obtenção de informações como tamanho, posicionamento, orientação ou a extração das bordas dos objetos de uma cena.

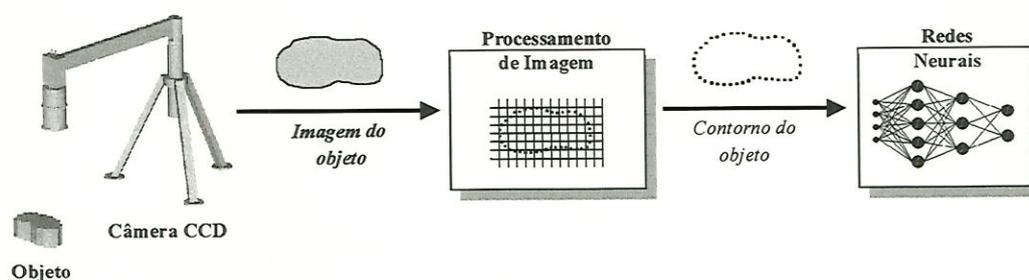


FIGURA 18 – Sistema de visão utilizado para captura e processamento da imagem do objeto. Uma câmera CCD obtém a imagem do objeto, que é processada para extração do contorno do mesmo e discretização deste contorno em cinquenta pontos.

A FIGURA 18 apresenta o sistema de visão implementado neste trabalho. Este sistema é responsável pela obtenção e pelo processamento da imagem do objeto a ser fixado

pela garra. Deste processamento resultam cinquenta pontos do contorno do objeto que representam o conjunto de possíveis locais de fixação da garra.

A captura da imagem é realizada por uma câmera digital CCD (*Digital Camera Charge-Couple Devices*). A câmera CCD é constituída por um sistema de lentes e uma matriz digitalizadora (FIGURA 19). As lentes projetam a imagem do objeto sobre esta matriz digitalizadora, que a discretiza em um número finito de pontos, cada qual representando um elemento da imagem (*PICTure Element = PIXEL*). Estes pontos (pixels) são enviados de forma binária para uma placa de aquisição de imagens acoplada ao microcomputador.

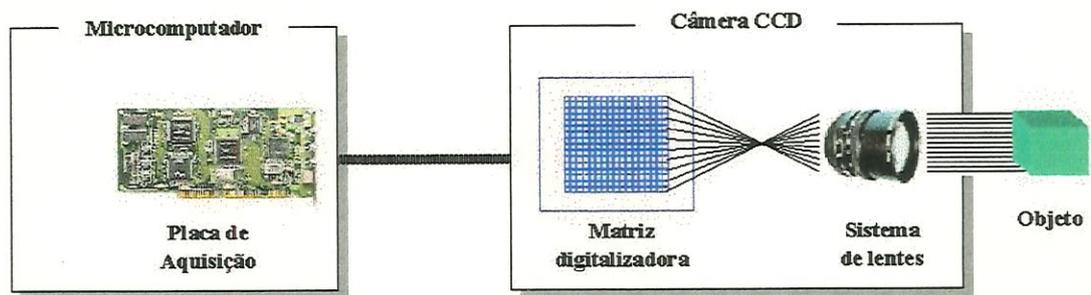


FIGURA 19 – Arquitetura de uma câmera CCD. O sistema de lentes focaliza a imagem sobre a matriz digitalizadora. A imagem é capturada por uma placa de aquisição acoplada ao computador.

No processo de discretização, a imagem é quadriculada por um *grid* regular (FIGURA 20). Cada pixel (célula) deste *grid* é descrito por um par de coordenadas (x,y) relativas a um referencial posicionado no canto superior esquerdo na imagem.

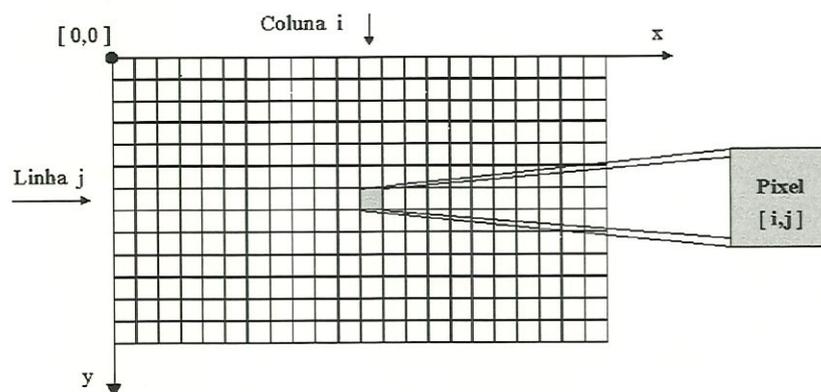


FIGURA 20 – *Grid* utilizado para discretização da imagem em uma matriz de pixels, cada um associado a uma coordenada (i,j) .

Desta forma, a imagem pode ser descrita como uma matriz, onde cada elemento está associado a um pixel e assume um valor que descreve a cor média da área da imagem por ele representado. Seguindo o padrão RGB, este valor representa uma combinação das cores primárias: vermelho (*Red*), verde (*Green*) e azul (*Blue*).

Assim, um pixel é descrito por três parcelas hexadecimais que representam a contribuição de cada cor primária. Uma parcela 00h representa a inexistência de determinada cor ao passo que uma parcela FFh representa a sua pigmentação máxima. Sendo cada parcela ocupada por um byte, um pixel é representado por três bytes conforme figura abaixo:



FIGURA 21 – Padrão RGB para representação das cores dos pixels.

Um pixel puramente vermelho, ou puramente verde ou puramente azul é respectivamente representado pelos seguintes valores:



FIGURA 22 - Representação hexadecimal das cores primárias: vermelho, verde, azul.

Outras combinações são demonstradas abaixo:

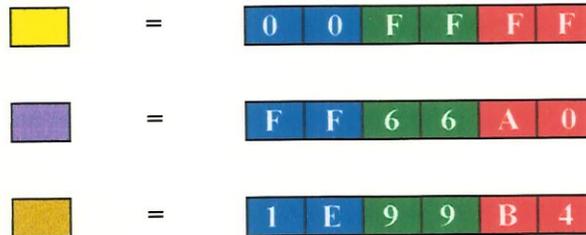


FIGURA 23 – Representação hexadecimal para combinações das cores primárias.

Numa imagem preto/branco, as cores são substituídas por uma escala de cinza. Nesta escala, todas as parcelas primárias assumem o mesmo valor, originando os vários tons de cinza. Já que as parcelas RGB são iguais, os tons de cinza são representados apenas por um byte hexadecimal (FIGURA 24). O valor 00h representa agora a inexistência de qualquer cor primária (preto); o valor FFh representa a existência de todas as cores (branco) e os valores intermediários representam as diversas tonalidades de cinza.

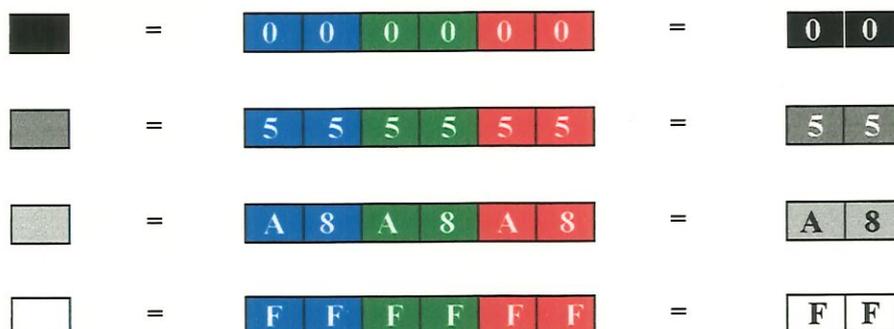


FIGURA 24 – Representação hexadecimal para padrões de cinza.

Já que a escala de cinza pode ser representada por apenas um valor, torna-se cômodo a utilização de valores decimais. Assim, para as cores acima, poder-se-ia utilizar os valores 0 para o preto, 255 para o branco e 80 e 158 para os padrões de cinza.

O sistema de visão (FIGURA 18) foi implementado utilizando uma câmera digital CCD preto/branco, permitindo a aquisição apenas de imagens na escala de cinza. Esta simplificação em nada reduz a funcionalidade do sistema, uma vez que ela é suficiente para a solução da tarefa proposta: definição do contorno de um objeto em destaque na cena.

2.2. Detecção do contorno do objeto

Como comentado anteriormente, o processamento digital visa estimar propriedades de objetos em uma cena. O contorno ou borda de um objeto pode ser uma destas propriedades, sendo identificado à partir de uma mudança brusca no valor dos pixels locais.

Os algoritmos de detecção de borda visam exatamente levantar o conjunto de pixels presentes nestas regiões de alteração brusca de intensidade (SCHALKOFF, 1989). Estes algoritmos apresentam três passos básicos:

Filtragem: Como os algoritmos são baseados em diferenças calculadas sobre apenas dois pontos, eles são muito sensíveis a ruído. Neste caso, uma prévia filtragem da cena é recomendável para amenizar este efeito e aumentar a performance do algoritmo. Porém, os filtros provocam uma suavização da imagem como um todo, o que também reduz o contraste das bordas. A aplicação de um filtro muito forte pode eliminar os ruídos mas também pode disfarçar as bordas.

Realçamento: Para facilitar a detecção da borda, é essencial que se enfatize as mudanças bruscas de intensidade na vizinhança de um pixel. Através do cálculo de gradientes locais, a etapa de realçamento enfatiza estes pontos pertencentes ao contorno do objeto.

Detecção: Nesta etapa, são determinados apenas os pontos que apresentam uma forte indicação para borda. Esta indicação é medida pela intensidade do gradiente calculado na etapa de realce. Porém, nem todos os pontos de gradiente não-zero são pontos de borda. Assim, são aplicados métodos para se determinar quais pontos são realmente interessantes. O método mais aplicado é o *threshold*, em que são escolhidos apenas os pontos de gradiente acima de um limite fixado.

2.3. Processos de filtragem

Uma imagem adquirida por uma câmera ou outro sistema pode apresentar ruído, variações de intensidade e iluminação e pobreza de contraste. Estes fatos prejudicam um eventual processo de detecção de borda. A aplicação dos filtros visa remover estes ruídos, preparando a imagem para um processamento posterior.

Destaca-se mais uma vez que, neste trabalho, as técnicas de processamento de imagem são utilizadas com o objetivo de extrair o contorno de uma peça a ser capturada por uma garra robótica, estando esta peça em destaque com relação ao restante da cena. Partindo-se desta característica particular da cena e tendo em vista o requisito de rapidez de processamento, propõe-se eliminar a etapa de filtragem e aplicar as etapas de realçamento e detecção de borda diretamente à cena original. Os resultados irão comprovar a viabilidade desta solução simplificada. Vale ressaltar que tal procedimento só foi possível devido à particularidade da cena.

2.4. Realçamento e detecção de borda

O processo de detecção de borda é, essencialmente, uma operação de detecção das mudanças locais significativas da imagem. A borda está associada a um pico na primeira derivada da função intensidade da imagem, assumindo esta como uma variação contínua de cores ou intensidade de cinza. Pode-se definir um gradiente G como a variação bidimensional da primeira derivada da imagem nas duas direções x,y :

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (2.1)$$

A magnitude do gradiente é dada por:

$$|G[f(x,y)]| = \sqrt{G_x^2 + G_y^2} \quad (2.2)$$

Para uma análise de imagens digitais, a definição de gradiente pode ser substituída por uma simples aproximação numérica:

$$\begin{aligned} G_x &\approx f[i, j+1] - f[i, j] \\ G_y &\approx f[i+1, j] - f[i, j] \end{aligned} \quad (2.3)$$

relembrando ser i e j coordenadas discretas da imagem. As equações acima também podem ser implementadas como máscaras simples, que representam os coeficientes para computar as derivadas parciais da imagem $f(x,y)$ entre pixels adjacentes.

$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Porém, quando calculado uma aproximação numérica do gradiente, é crucial que as derivadas parciais G_x e G_y sejam calculadas exatamente no mesmo ponto da imagem. Nas máscaras acima, G_x representa a aproximação do gradiente calculado no ponto $\left[i, j + \frac{1}{2} \right]$ e G_y representa a aproximação no ponto $\left[i + \frac{1}{2}, j \right]$. Assim, as máscaras 2×1 e 1×2 são substituídas pelas máscaras 2×2 abaixo.

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

onde os gradientes são aproximados em torno do ponto central $\left[i + \frac{1}{2}, j + \frac{1}{2} \right]$.

Este ponto ainda gera alguma confusão, uma vez que os pixels são representados por coordenadas inteiras. Uma solução foi a aplicação de máscaras 3×3 , com o cálculo do gradiente realizado em torno do ponto central. Diversos tipos de máscaras ou operadores 3×3 são discutidos a seguir.

2.4.1. Operador de Sobel

Assumindo a máscara hipotética abaixo:

a_0	a_1	a_2
a_7	$[i,j]$	a_3
a_6	a_5	a_4

O operador de Sobel é definido como a magnitude do gradiente calculado por:

$$M_S = \sqrt{S_x^2 + S_y^2} \quad (2.4)$$

sendo as derivadas parciais calculadas por:

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \quad (2.5)$$

$$S_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \quad (2.6)$$

Com a constante $c = 2$. Estas equações podem ser expressas pelas seguintes máscaras:

$$S_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$S_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Note que este operador enfatiza os pixels que se encontram mais próximos do centro da máscara. O operador de Sobel é aplicado a todos os pixels da imagem, sendo um dos processos de ênfase de borda mais utilizados.

Uma vez enfatizado pelo operador acima, o contorno é extraído pelo processo de *threshold*. Neste processo de detecção, assume-se como borda todo pixel cujo operador de Sobel for superior a um determinado valor ou *threshold*. Quanto maior o *threshold*, maior deverá ser o contraste entre os pixels para se reconhecer uma borda.

2.4.2. Operador de Prewitt

O operador de Prewitt utiliza as mesmas equações de Sobel (EQUAÇÕES 2.5 e 2.6) para o cálculo de gradiente, exceto que a constante $c = 1$. Assim:

$$P_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad P_y = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$M_P = \sqrt{P_x^2 + P_y^2} \quad (2.7)$$

Note que o operador de Prewitt não enfatiza nenhum pixel das proximidades do centro, aplicando o mesmo peso para todos os pixels da vizinhança.

Para a detecção da borda, utiliza-se o mesmo processo de *threshold*.

2.4.3. Operador de Johnson

O grande problema dos operadores de primeira ordem é que eles sofrem grande influência da iluminação ambiente. Por exemplo, um operador pode ser eficiente na determinação das bordas de um objeto em uma região bem iluminada da cena, mas pode ser incapaz de encontrar as bordas do mesmo objeto posicionado em uma região escura. Esta limitação pode ser aceitável nos casos em que haja um bom controle da iluminação, podendo-se evitar o aparecimento de sombras. Porém, em outros casos, a incapacidade de se prever uma boa iluminação pode comprometer o funcionamento dos algoritmos de primeira ordem.

Este problema pode ser parcialmente solucionado com a utilização de operadores de segunda ordem (veja SEÇÃO 2.4.4), que são sensíveis aos máximos locais da primeira derivada. Porém, a presença de ruído ou outras irregularidades da cena podem levar ao aparecimento de bordas falsas. A aplicação de um *threshold* poderia eliminar estas bordas indesejáveis, mas também eliminaria as bordas das regiões de iluminação pobre.

JOHNSON (1990) propõe uma modificação do operador de Sobel a fim de melhorar o seu desempenho em regiões escuras da cena. Neste novo operador, o resultado de Sobel $M_S(i,j)$ é dividido pela média da vizinhança 3x3 somada a uma constante d .

$$M_j(i,j) = \frac{M_S(i,j)}{\left(\frac{1}{9} \sum_{K=i-1}^{i+1} \sum_{L=j-1}^{j+1} f[K,L] \right) + d} \quad (2.8)$$

Como nos demais algoritmos, o operador de Johnson é utilizado como método de ênfase das bordas e o processo de *threshold* é utilizado para a detecção destas bordas.

2.4.4. Operador de Laplace

Os operadores discutidos anteriormente baseavam-se no cálculo da primeira derivada e, se este resultado fosse superior a um *threshold*, assumia-se o pixel em questão como ponto de borda. Este processo pode resultar em uma borda muito espessa. Uma proposta diferente seria considerar apenas os pontos que apresentam máximo local no valor do gradiente. Este ponto seria, então, um pico na primeira derivada e conseqüentemente um zero no cálculo da segunda derivada. Assim, os pontos de borda passam a ser detectados pelo cruzamento pelo zero da segunda derivada da imagem.

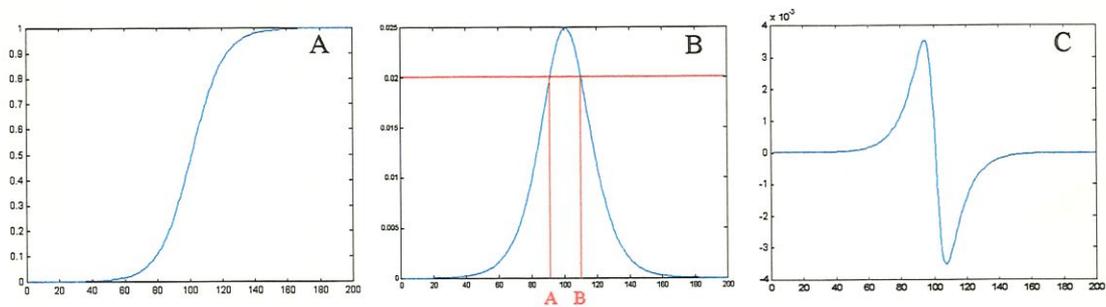


FIGURA 25 – Detecção de borda do operador de segunda ordem: (A) distribuição de tonalidade dos pixels próximo à borda; (B) primeira derivada da função; (C) segunda derivada da função.

A FIGURA 25B demonstra que a detecção de borda por *threshold* (indicado pela linha vermelha) define como borda todos os pontos entre *A* e *B*. Entretanto, analisando a segunda derivada da função $f(x,y)$, a borda é detectada com maior precisão através do ponto de cruzamento com o eixo das abscissas (FIGURA 25C). Este ponto corresponde ao máximo local da primeira derivada, ou seja, ao ponto que apresenta o maior gradiente da função.

O operador de Laplace é o equivalente bidimensional da segunda derivada da função.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (2.9)$$

A segunda derivada ao longo das direções x e y usando a aproximação numérica é expressa nas coordenadas discretas:

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} \\ &= \frac{\partial(f[i,j+1] - f[i,j])}{\partial x} \end{aligned}$$

$$\begin{aligned}
 &= \frac{\partial f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x} \\
 &= (f[i, j+2] - f[i, j+1]) - (f[i, j+1] - f[i, j]) \\
 &= f[i, j+2] - 2f[i, j+1] + f[i, j]
 \end{aligned} \tag{2.10}$$

Entretanto, esta aproximação está centrada no ponto $[i, j+1]$. Substituindo a coordenada j por $j-1$ obtém-se:

$$\frac{\partial^2 f}{\partial x^2} = f[i, j+1] - 2f[i, j] + f[i, j-1] \tag{2.11}$$

onde a aproximação está centrada no ponto desejado: $[i, j]$. Seguindo o mesmo processo, pode-se determinar a expressão para a segunda derivada na direção y :

$$\frac{\partial^2 f}{\partial y^2} = f[i+1, j] - 2f[i, j] + f[i-1, j] \tag{2.12}$$

Combinando estas duas equações num mesmo operador, a seguinte máscara é usada para uma aproximação laplaciana:

$$\nabla^2 \approx \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Algumas vezes, torna-se interessante dar ênfase aos pixels mais centrais da vizinhança. Neste sentido, a aproximação laplaciana pode ser assim modificada:

$$\nabla^2 \approx \begin{array}{|c|c|c|} \hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline \end{array}$$

A seqüência abaixo mostra o processo de detecção de borda utilizando o operador de Laplace. Considera-se uma imagem com a distribuição de intensidades mapeada na FIGURA 27A.

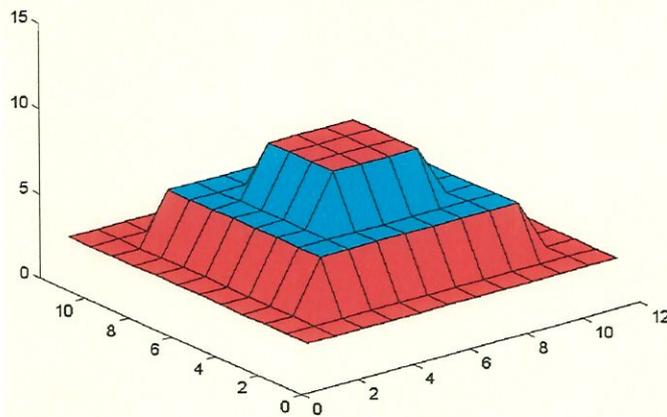


FIGURA 26 – Distribuição de pixels de uma cena, identificando uma forma quadrática.

2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	5	5	5	5	5	5	2	2	2
2	2	2	5	8	8	8	8	5	2	2	2
2	2	2	5	8	8	8	8	5	2	2	2
2	2	2	5	8	8	8	8	5	2	2	2
2	2	2	5	8	8	8	8	5	2	2	2
2	2	2	5	8	8	8	8	5	2	2	2
2	2	2	5	5	5	5	5	5	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2

(A)

	0	0	0	0	0	0	0	0	0	0	
	0	0	3	3	3	3	3	3	0	0	
	0	3	-6	0	0	0	0	-6	3	0	
	0	3	0	-6	-3	-3	-6	0	3	0	
	0	3	0	-3	0	0	-3	0	3	0	
	0	3	0	-3	0	0	-3	0	3	0	
	0	3	0	-6	-3	-3	-6	0	3	0	
	0	3	-6	0	0	0	0	-6	3	0	
	0	0	3	3	3	3	3	3	0	0	
	0	0	0	0	0	0	0	0	0	0	

(B)

FIGURA 27 – Detecção de borda utilizando o operador de Laplace: (A) mapeamento das intensidades dos pixels; (B) resultado da aplicação do operador de Laplace à imagem.

A FIGURA 27B mostra os resultados do operador de Laplace aplicado a cada um dos pixels. O processo de detecção desta borda leva em consideração o ponto em que este operador cruza o valor zero.

2.4.5. Operador Laplaciano sobre uma Gaussiana

Os operadores de segunda ordem descritos anteriormente são caracterizados por uma grande sensibilidade à ruídos. Portanto, torna-se sempre necessário a aplicação de um filtro anterior ao processo de ênfase da borda. Entretanto, um novo algoritmo foi desenvolvido combinando o filtro gaussiano e o operador de Laplace. Este processo foi denominado operador Laplaciano sobre uma Gaussiana (LoG – *Laplacian of Gaussian*).

O resultado do operador LoG é expresso pelo cálculo da convolução (\otimes) da função gaussiana $g(x,y)$ com a imagem $f(x,y)$ e posterior aplicação do operador laplaciano ∇^2 . A

convolução é aproximada pela aplicação discreta da máscara gaussiana aos pixels da imagem.

$$h(x, y) = \nabla^2 [g(x, y) \otimes f(x, y)] \quad (2.13)$$

A expressão acima é matematicamente equivalente a convolução da imagem com o filtro resultante da aplicação do operador de Laplace ao filtro gaussiano.

$$h(x, y) = [\nabla^2 g(x, y)] \otimes f(x, y) \quad (2.14)$$

onde,

$$\nabla^2 g(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{(x^2 + y^2)}{2\sigma^2}} \quad (2.15)$$

Esta última equação representa o filtro mostrado abaixo:

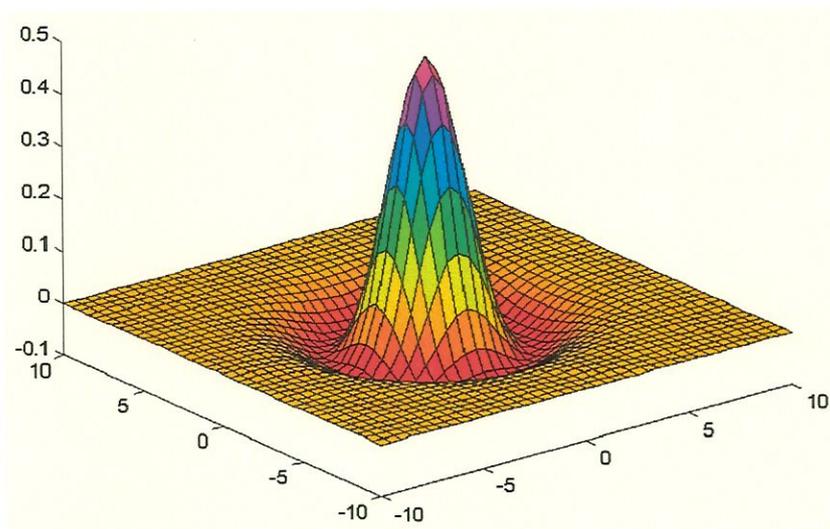


FIGURA 28 – Função (invertida) resultante da aplicação de Laplace ao filtro gaussiano

Pode-se montar uma máscara semelhante àquelas dos métodos vistos anteriormente. Uma máscara típica para aplicação do operador LoG é mostrada abaixo, considerando uma janela 5x5:

$$\text{LoG} \cong \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & -1 & 0 & 0 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline -1 & -2 & 16 & -2 & -1 \\ \hline 0 & -1 & -2 & -1 & 0 \\ \hline 0 & 0 & -1 & 0 & 0 \\ \hline \end{array}$$

A aplicação desta máscara resulta na ênfase dos pontos de borda, detectados pelo cruzamento do operador com o ponto zero.

2.4.6. Método OU-Exclusivo

O método OU-Exclusivo (GONZAGA & FRANÇA, 1996) difere dos métodos anteriores já que não depende do cálculo de gradientes. Este é um dos métodos mais rápidos para a detecção de borda, uma vez que exige um baixo esforço computacional. Sua aplicação envolve duas fases bem claras:

1. Quantização da imagem através da aplicação de um *threshold*. Nesta etapa, enfatiza-se não só a borda como toda a região compreendida pelo objeto.
2. Aplicação do OU-Exclusivo a cada um dos pixels da imagem, sendo que cada pixel é comparado com o pixel imediatamente à direita e imediatamente abaixo.

$$O(i, j) = [f(i, j) \oplus f(i, j+1)] + [f(i, j) \oplus f(i+1, j)] \quad (2.16)$$

sendo (\oplus) uma operação OU-Exclusivo e (+) uma operação OU.

A aplicação do algoritmo é exemplificada abaixo:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(A)

0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(B)

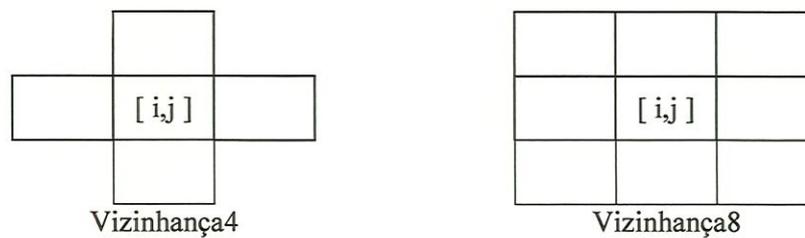
FIGURA 29 – Detecção da borda do objeto utilizando o método OU-Exclusivo: (A) Cena após aplicação do *threshold*; (B) Detecção dos pixels de borda $O(i, j) = 1$

Apesar de rápido, o método OU-Exclusivo apresenta uma baixa tolerância a ruídos. Pode-se ajustar o *threshold* para a eliminação dos ruídos, porém esta mesma solução pode eliminar regiões do próprio objeto. Percebe-se também um erro no posicionamento da borda, estendida para cima e para a esquerda.

2.4.7. Método do Vizinho mais Próximo (*Nearest Neighbor*)

Este é o método mais simples e mais rápido para a detecção de borda. Diferente dos demais métodos que percorrem toda a imagem, este algoritmo percorre a imagem da esquerda para a direita e de cima para baixo até encontrar o primeiro ponto que satisfaça as condições imposta para borda. Encontrado este ponto, o método passa a acompanhar a borda do objeto até definir todos os demais pontos pertencentes a ela. É empregado em cenas compostas por apenas um objeto de tonalidade diferente do fundo.

Para melhor compreender o método e a condição de borda, devemos introduzir dois conceitos: vizinhança4 e vizinhança8. A vizinhança4 compreende os pontos da vizinhança localizados imediatamente acima, abaixo, à esquerda e à direita. Já a vizinhança8 compreende estes quatro pontos mais todos os outros quatro pontos das diagonais. Assim:



$$\text{Vizinhança4}[i,j]: [i+1,j], [i,j+1], [i-1,j], [i,j-1]$$

$$\text{Vizinhança8}[i,j]: [i+1,j], [i+1,j+1], [i,j+1], [i-1,j+1], [i-1,j], [i-1,j-1], [i,j-1], [i+1,j-1]$$

FIGURA 30 – Definição de vizinhança4 e vizinhança8.

A imagem é inicialmente quantizada através da aplicação de um *threshold*. Todos os pixels pertencentes ao fundo da cena recebem o valor “0” e os pixels pertencentes ao objeto recebem o valor “1”. A imagem é então rastreada até que um pixel $h[i,j]$ satisfaça as condições de borda:

$$h[i,j] = 1 \quad (2.17)$$

$$\sum \text{vizinhança4}[i,j] \neq 4 \quad (2.18)$$

O ponto deve pertencer ao objeto (EQUAÇÃO 2.17) e ao mesmo tempo não deve estar localizado em seu interior (EQUAÇÃO 2.18). Determinado o primeiro ponto que satisfaça a condição de borda, percorre-se a sua vizinhança8 no sentido horário até que se defina o próximo ponto de borda.

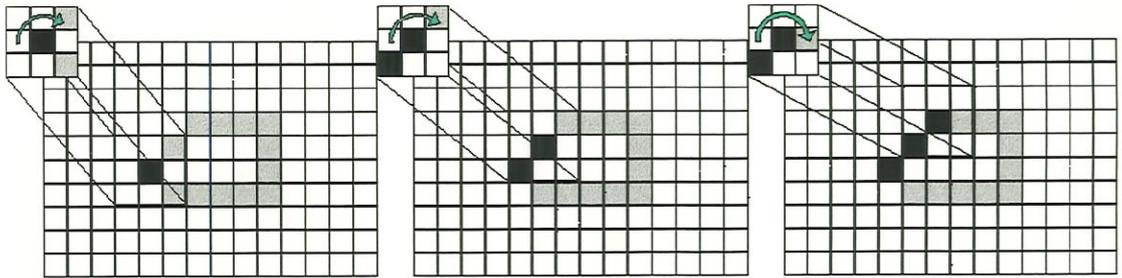


FIGURA 31 – Algoritmo do vizinho mais próximo A vizinhança⁸ é percorrida no sentido horário até que se defina o próximo ponto de borda.

Este algoritmo se repete até que se volte ao ponto inicial. O resultado é apresentado abaixo:

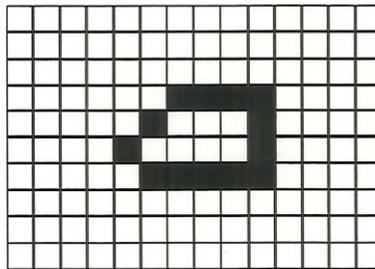


FIGURA 32 – Resultado da aplicação do algoritmo do vizinho mais próximo/

2.5. Comparação dos operadores de ênfase e detecção de borda

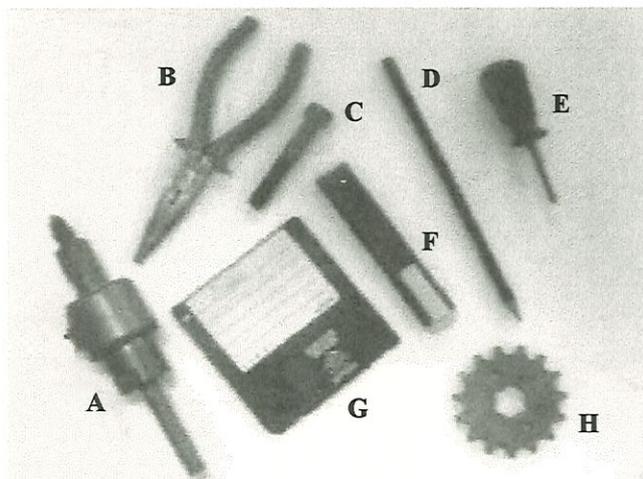


FIGURA 33 – Imagem padrão utilizada para exemplificação dos métodos de detecção de borda. A imagem é composta pelos seguintes objetos: (A) Eixo com diversos ressaltos; (B) alicate de bico; (C) Parafuso; (D) Lápis; (E) Chave de fenda; (F) Caneta de marcação; (G) Disquete; (H) Engrenagem.

Todos os métodos anteriormente apresentados para detecção e ênfase de borda foram implementados e serão comparados nesta seção. Inicialmente, destaca-se a imagem padrão (FIGURA 33) utilizada para a exemplificação dos algoritmos.

Esta imagem apresenta algumas características interessantes. Primeiro, é uma imagem preto/branco, no mesmo padrão das imagens capturadas pela câmera CCD. Os objetos espalhados pela cena, em diferentes escalas de cinza, são objetos comuns de um ambiente de trabalho. Outra característica interessante é que a iluminação da cena foi propositalmente reduzida, a fim de testar os algoritmos em condições adversas.

Sobre esta imagem padrão são aplicados os algoritmos de Sobel e Prewitt, com diferentes valores de *threshold* (FIGURA 34). Para ambos os algoritmos, um baixo valor de *threshold* ajuda a enfatizar todas as bordas dos objetos, já que um maior número de pontos passa a satisfazer a baixa condição de borda. Porém, ocorre a definição de uma borda muito espessa e portanto sem muita precisão de posicionamento. Para valores cada vez maiores de *threshold*, observa-se a definição de um menor número de pontos, uma vez que a condição imposta é mais exigente.

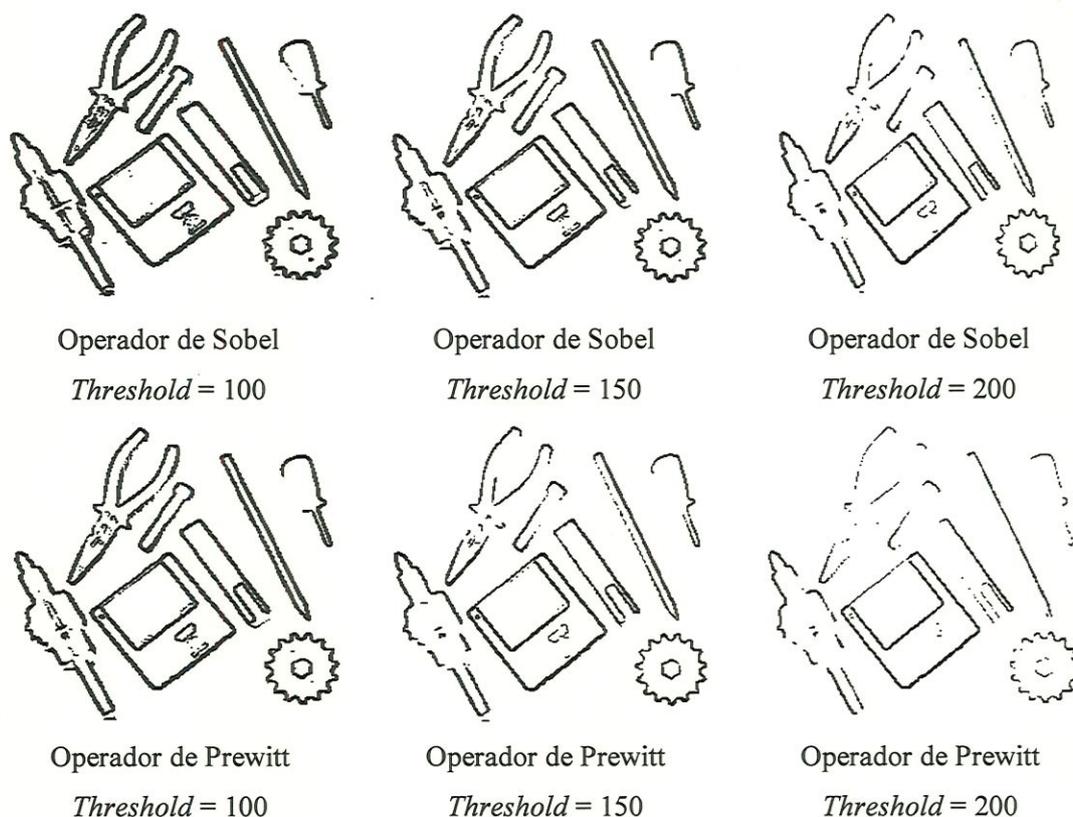


FIGURA 34 – Comparação entre os algoritmos de Sobel e Prewitt para diversos valores de *threshold*

Comparando-se os resultados obtidos pelos algoritmos de Sobel e Prewitt, percebe-se um maior detalhamento do primeiro. Isto é explicado através da análise dos operadores utilizados por cada um (SEÇÕES 2.4.1 e 2.4.2). No operador de Sobel, nota-se um maior enfoque dado aos pixels próximos do centro, o que lhe confere uma maior sensibilidade na definição das regiões de contraste.

Analisando agora o algoritmo de Johnson, destacam-se os resultados da FIGURA 35. Além da influência do *threshold*, percebe-se ainda a influência do parâmetro *d*. O ajuste deste parâmetro permite a ênfase de bordas mesmo em regiões escuras da cena. Uma vez que os demais algoritmos não conseguem tal definição, percebe-se a maior capacidade do operador de Johnson em analisar regiões com deficiência de iluminação.

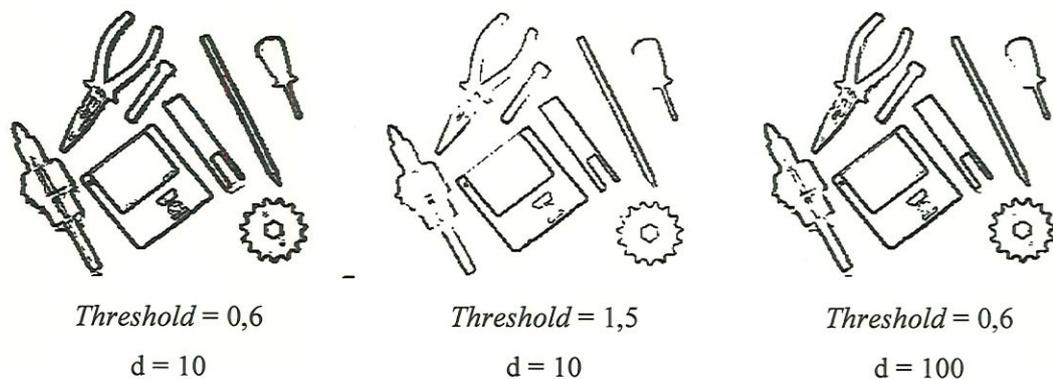


FIGURA 35 – Resultados do operador de Johnson para diferentes valores de *threshold* e *d*.

Aplica-se agora o método OU-Exclusivo. Apesar de este método ser muito mais rápido que os anteriores, os resultados deixam um pouco a desejar (conforme figura abaixo).

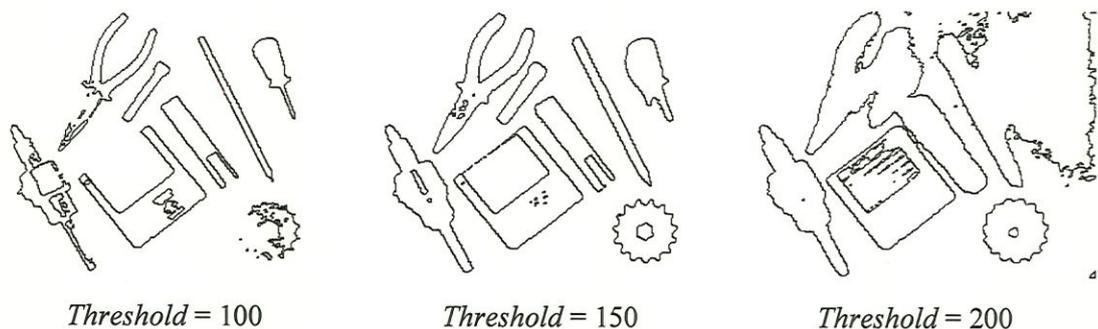


FIGURA 36 – Resultados de baixa qualidade do método OU-Exclusivo

Para um baixo valor de *threshold*, ocorrem falhas na detecção das bordas para formatos mais elaborados. Para altos valores de *threshold*, os resultados são bastante

influenciados pela presença de sombras. Conclui-se que o método OU-Exclusivo é mais adequado para situações de melhor luminosidade e maior contraste entre as bordas.

Por fim, é analisado o método do vizinho mais próximo. O resultado (FIGURA 37) mostra a detecção de apenas um dos objetos da cena, exatamente aquele mais a esquerda e mais acima. Isto ocorre devido a própria metodologia do algoritmo (SEÇÃO 2.4.7).

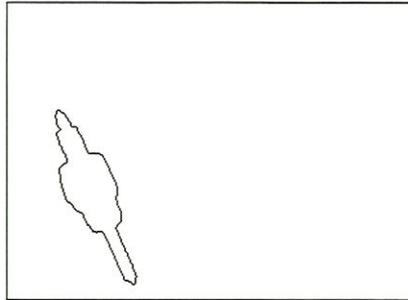


FIGURA 37 – Resultado do método do vizinho mais próximo, com a detecção apenas da figura mais a esquerda e mais abaixo.

A TABELA 2 compara o tempo de processamento para os algoritmos destacados nesta seção, considerando a utilização de um microcomputador Pentium II 233 MHz.

TABELA 2 – Comparação dos algoritmos quanto ao tempo de processamento (tempo médio após vinte análises)

Algoritmo	Tempo de processamento (segundos)	Variância (σ^2)
Sobel	20,7	0.139
Prewitt	20,4	0.131
Johnson	38,6	0.245
OU-Exclusivo	2,8	0.117
Vizinho mais próximo	1,3	0.093

Vale ressaltar ainda que o tempo de processamento do algoritmo do vizinho mais próximo é dependente da posição do objeto na cena: quanto mais a esquerda e mais acima estiver o objeto, mais rápido é o processo. A FIGURA 38 demonstra este fato:



FIGURA 38 – Variação do tempo de processamento do algoritmo do vizinho mais próximo em função da posição do objeto na cena.

2.6. Análise final dos operadores de ênfase e detecção de borda

Analisando os resultados anteriores, o algoritmo de JOHNSON apresentou a melhor extração de borda, principalmente devido ao seu desempenho nas regiões escuras da cena. Os algoritmo de Sobel e Prewitt demonstraram certa sensibilidade às condições de iluminação.

Contudo, para a tarefa de detecção do contorno de um objeto a ser fixado por uma garra robótica, o tempo de processamento é um fator importante. Neste ponto, destaca-se o algoritmo do vizinho mais próximo (TABELA 2). Como a cena é parcialmente rastreada, até a detecção do primeiro ponto de borda, este método é um pouco mais rápido que o método OU-Exclusivo e muito mais rápido que os demais métodos. Embora o rastreamento parcial limite o algoritmo à extração da borda de apenas um objeto na cena, tal limitação em nada prejudica o processo. Como a cena será sempre composta apenas pelo objeto a ser capturado, não há necessidade de analisar o seu restante.

Neste contexto, devido à grande vantagem de rapidez de processamento do algoritmo do vizinho mais próximo, este será responsável pela tarefa de processamento de imagem proposto neste trabalho.

3. REDE COMPETITIVA DE HOPFIELD

Neste trabalho, o modelo de rede Competitiva de Hopfield tem o objetivo de aproximar um polígono sobre os pontos do contorno do objeto, sendo estes pontos definidos pelo sistema de visão. As principais características deste modelo neural são apresentadas neste capítulo.

3.1. Rede Clássica de Hopfield

HOPFIELD & TANK (1985) propuseram um modelo de rede recursiva diferente do modelo *feed-forward* (APÊNDICE 1)². Neste novo modelo, cada neurônio é conectado a todos os demais (FIGURA 39). A rede é ainda caracterizada por uma matriz de pesos simétrica ($w_{ij} = w_{ji}$), com diagonal principal igual a zero ($w_{ii} = 0$). Assim, o peso da conexão entre um neurônio e outro é o mesmo nas duas direções.

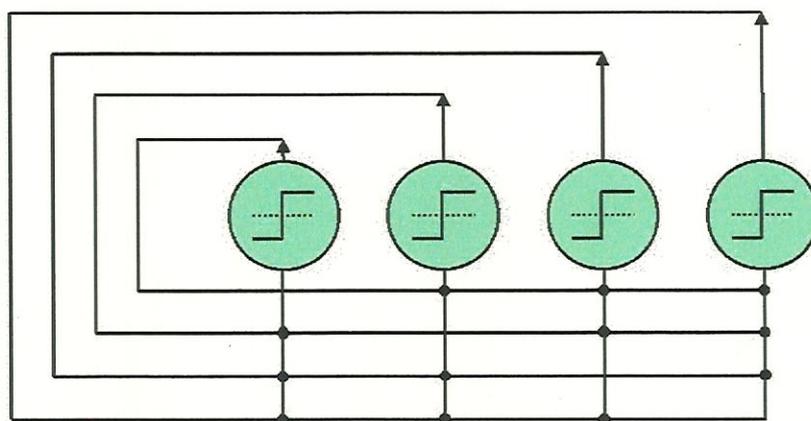


FIGURA 39 - Representação vetorial da rede de Hopfield. Cada unidade é conectada a todas as demais.

² Durante a elaboração desta dissertação, foi considerado que o leitor já esteja familiarizado com a teoria de redes neurais. Porém, para uma revisão dos conceitos básicos ou mesmo para um primeiro contato com a teoria, sugere-se uma leitura preliminar do Apêndice 1 – Introdução às Redes Neurais Artificiais.

A ativação de cada unidade é dada pela aplicação de uma função degrau f_h sobre o somatório ponderado dos valores de entrada subtraído do *threshold*. Estes valores de entrada representam as ativações das outras unidades no instante anterior:

$$x_i(t+1) = f_h \left[\sum_{j=1}^N w_{ij} x_j(t) - T_i \right] \quad i = 1, 2, \dots, N \quad (3.1)$$

sendo N o número total de neurônios; x_j a ativação do j ésimo neurônio; w_{ij} o peso da conexão entre o i ésimo e o j ésimo neurônio; T_i o *threshold* do i ésimo neurônio. A função degrau f_h é uma função não-linear que determina uma ativação binária (0,1) ou bipolar (-1,+1). A ativação bipolar é mais adequada sob o ponto de vista computacional, uma vez que torna a representação matemática mais simples.

A FIGURA 39 mostra que a rede de Hopfield não apresenta unidades de entrada e saída típicas de uma rede *feed-forward*. Numa rede *feed-forward*, as entradas são aplicadas em unidades específicas, o sinal é conduzido pela rede e é gerada uma saída representando a solução. A arquitetura diferenciada da rede Hopfield indica que a rede apresenta uma nova forma de operação. Como dados de entradas, são atribuídas ativações iniciais a todos os neurônios. Para o caso bipolar, todos os neurônios assumem valores -1 ou $+1$. Uma vez que todos os neurônios estão mutuamente conectados, cada ativação afeta as demais. Assim, enquanto uma unidade pode estar tentando ativar outra, uma terceira unidade pode estar tentando desativá-la. A rede então passa por diversos estados intermediários até que converge para um estado estável, que ocorre quando não há mais mudança nas ativações das unidades.

Para cada passo, a energia E do sistema é calculada através da seguinte equação:

$$E = -\frac{1}{2} \sum_i \sum_{j \neq i} w_{ij} x_i x_j + \sum_i x_i T_i \quad (3.2)$$

O ponto de estabilidade corresponde a um “ponto de mínimo” desta equação de energia. A busca por um ponto de mínima energia possibilita às redes de Hopfield uma aplicação em problemas de otimização. PROTZEL (1993) discute esta aplicação: se for possível associar o equacionamento do problema de otimização com o equacionamento da função energia, então a rede encontra a solução otimizada à medida que converge naturalmente de uma posição inicial para uma posição de mínima energia. Assim, a chave está na devida definição da função energia. Esta deverá incorporar as condições de contorno do problema a ser representado.

Um dos problemas clássicos de otimização é o problema de aproximação poligonal. Descrevendo-o através de uma função energia, este problema poderá ser solucionado por uma rede de Hopfield. Ao atingir o ponto de mínima energia, a rede estará apontando um

polígono que minimiza o desvio entre a curva original e a figura aproximada. Parte-se, então, para a definição dos termos do problema de aproximação poligonal.

3.2. Formalização do problema de aproximação poligonal

Na formalização do problema de aproximação poligonal, é considerada uma curva fechada $P = \{p_1, p_2, p_3, \dots, p_n\}$ consistindo de n pontos ordenados no sentido horário (FIGURA 40). Nesta definição, $p_x p_y$ e $\overline{p_x p_y}$ representam respectivamente o trecho da curva e a corda do ponto p_x ao ponto p_y , no sentido horário. ARAÚJO & TANAKA (1995) definem o desvio entre o polígono e a curva como o somatório das distâncias perpendiculares dos ponto pertencentes ao trecho da curva $p_x p_y$ até a corda $\overline{p_x p_y}$:

$$h_{xy} = \sum_{p \in p_x p_y} d(p, \overline{p_x p_y}) \quad (3.3)$$

A figura abaixo exemplifica o desvio calculado pelo somatório $h_{x,y} = d_r + d_s + d_t + d_u$.

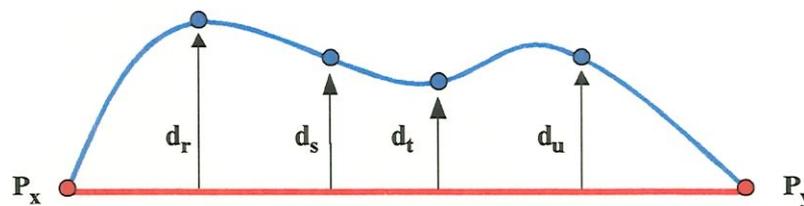


FIGURA 40 - Cálculo do desvio ponto-corda $h_{x,y}$. O erro de aproximação da corda $\overline{p_x p_y}$ (vermelha) com relação à curva $P_x P_y$ (azul) é definido pelo somatório $d_r + d_s + d_t + d_u$.

Define-se H como sendo a matriz de conexão formada por tais desvios. Nota-se que esta matriz não é simétrica, ou seja, $h_{x,y} \neq h_{y,x}$.

$$H = \{h_{xy}\}, \quad x, y = 1, 2, \dots, n \quad (3.4)$$

Baseado neste critério, desenvolve-se uma aplicação das redes de Hopfield no problema de aproximação poligonal. Para uma dada curva definida por n pontos e para um número determinado de pontos de aproximação m , o objetivo do método proposto é encontrar um polígono de aproximação $Q^{(m)}$ definido por m pontos dentre aqueles da curva P de tal forma a minimizar o desvio ponto-corda.

Uma Rede Clássica de Hopfield pode ser implementada na configuração bidimensional (FIGURA 41) com n linhas e m colunas, consistindo então de $n \cdot m$ neurônios mutuamente conectados.

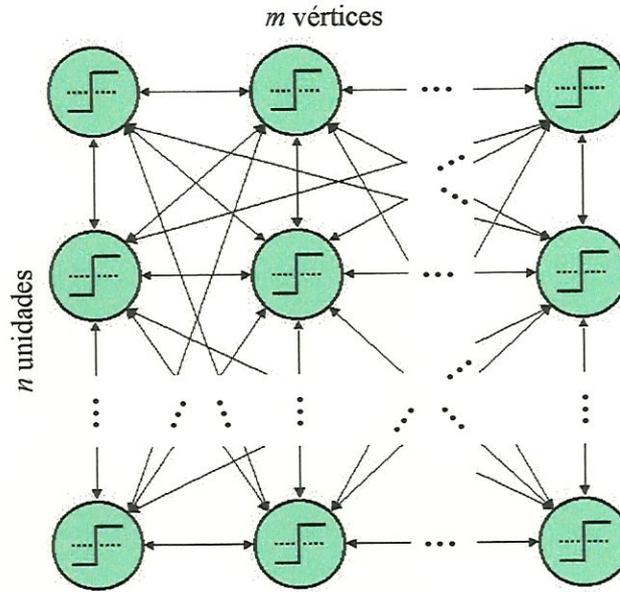


FIGURA 41 – Representação matricial da Rede Clássica de Hopfield. Cada unidade é conectada a todos as demais.

Sendo $V_{x,i}$ o estado binário do neurônio (x,i) e $W_{x,i;y,j}$ o valor de ponderação entre a unidade (x,i) e a unidade (y,j) , cada neurônio recebe um sinal $W_{x,i;y,j}V_{y,j}$ de cada unidade (y,j) além de um *bias* $I_{x,i}$ definido pelo meio externo. A equação (3.1) pode ser reescrita como:

$$U_{x,i} = \sum_{y=1}^n \sum_{j=1}^m W_{x,i;y,j} V_{y,j} + I_{x,i} \quad (3.5)$$

As linhas da rede matricial de Hopfield representam os pontos da curva, enquanto que as colunas representam as posições dos vértices do polígono de aproximação. Por exemplo, um neurônio $V_{x,i}$ no estado ativado indica que o x ésimo ponto da curva foi escolhido como o i ésimo vértice do polígono.

A estabilidade da rede de Hopfield de duas dimensões é definida pela seguinte função de Lyapunov:

$$E = - \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^m \sum_{j=1}^m W_{x,i;y,j} V_{x,i} V_{y,j} - 2 \sum_{x=1}^n \sum_{i=1}^m I_{x,i} V_{x,i} \quad (3.6)$$

A convergência da rede é atingida quando a energia definida pela função de Lyapunov é minimizada.

Uma possível função objetivo para a tarefa de aproximação poligonal é dada por:

$$E = A \cdot \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^m (h_{y,x} V_{x,i} V_{y,i-1} + h_{x,y} V_{x,i} V_{y,i+1}) + B \cdot \sum_{x=1}^n \sum_{i=1}^m \sum_{\substack{j=1 \\ j \neq i}}^m V_{x,i} V_{x,j} + \\ C \cdot \sum_{i=1}^m \left(1 - \sum_{x=1}^n V_{x,i} \right)^2 + D \cdot \left(m - \sum_{x=1}^n \sum_{i=1}^m V_{x,i} \right) \quad (3.7)$$

sendo os índices cíclicos, de tal forma que: $V_{0,i} = V_{n,i}$, $V_{n+1,i} = V_{1,i}$, $V_{x,0} = V_{x,m}$ e $V_{x,m+1} = V_{x,1}$.

Esta função incorpora as condições de contorno do problema de aproximação poligonal. O primeiro termo da função objetivo determina a soma total dos desvios entre o arco e a corda definidos pelos pontos adjacentes do polígono. O segundo termo determina que nenhum ponto p_x da curva ocorra em dois ou mais vértices do polígono. O terceiro termo penaliza a escolha de dois pontos p_x e p_y para um mesmo vértice. O quarto termo certifica que foram escolhidos um total de m vértices de aproximação. Deve-se notar que os três últimos termos são termos de penalização, que tentam manter a viabilidade da solução, enquanto o primeiro termo minimiza o desvio total entre a curva e o polígono. A qualidade da solução está diretamente ligada à definição dos fatores de ponderação A, B, C e D. Porém, a definição apropriada destes fatores é uma tarefa um tanto dispendiosa.

3.3. Implementação da Rede Competitiva de Hopfield

A Rede Competitiva de Hopfield é proposta para aliviar o problema de definição dos fatores de ponderação, uma vez que os termos de penalização serão definidos de maneira implícita.

A regra *winner-take-all* (BISHOP, 1995) é empregada para a atualização dos neurônios. Os neurônios de uma mesma camada competem entre si para definir qual receberá o maior estado de ativação. Assim, a unidade vencedora assume o valor 1 e os demais neurônios da camada assumem o valor 0:

$$V_{x,i} = \begin{cases} 1 & \text{se } U_{x,i} = \max\{U_{1,i}, U_{2,i}, \dots, U_{n,i}\} \\ 0 & \text{nos demais casos} \end{cases} \quad (3.8)$$

A regra *winner-take-all* evita que dois pontos p_x e p_y sejam definidos para o mesmo vértice. Ainda garante que sejam escolhidos um total de m pontos para os vértices do polígono.

Para manter a penalização imposta pelo segundo termo, uma nova definição da matriz de conexão foi adotada (EQUAÇÃO 3.9). A modificação imposta ao desvio arco-corda garante que nenhum ponto p_x ocorra em mais de um vértice do polígono.

$$h_{x,y} = \begin{cases} \infty & \text{se } x = y \\ \sum_{p \in P_x, P_y} d(p, \overline{p_x p_y}) & \text{se } x \neq y \end{cases} \quad (3.9)$$

As simplificações impostas pelas equações (3.8) e (3.9) restringem a função objetivo (3.7) ao primeiro termo de minimização e elimina a necessidade de determinar os fatores de ponderação. A equação final é apresentada abaixo e representa a função energia da rede.

$$E = \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^m (h_{y,x} V_{x,i} V_{y,i-1} + h_{x,y} V_{x,i} V_{y,i+1}) \quad (3.10)$$

A convergência da rede é provada por CHUNG et al. (1994).

Comparando a equação de energia (3.10) com a função de Lyapunov (3.6), pode-se obter os valores de ponderação e os *bias*:

$$W_{x,i; y,j} = -(\delta_{i-1,j} h_{y,x} + \delta_{i+1,j} h_{x,y}) \quad (3.11)$$

$$I_{x,i} = 0 \quad (3.12)$$

onde $\delta_{i,j}$ é a função delta de Kronecker, definida como: $\delta_{i,j} = 1$ se $x = j$ e $\delta_{i,j} = 0$ nos demais casos. Aplicando as equações (3.11) e (3.12) à equação (3.5), a entrada total da unidade (x,i) é obtida por:

$$U_{x,i} = \sum_{y=1}^n -(h_{y,x} V_{y,i-1} + h_{x,y} V_{y,i+1}) \quad (3.13)$$

Analisando a equação acima, percebe-se que a rede não é totalmente conectada. Os neurônios de uma camada específica recebem sinais provenientes dos neurônios da camada anterior e posterior (FIGURA 42). Esta nova regra de conexão reduz significativamente a complexidade da rede.

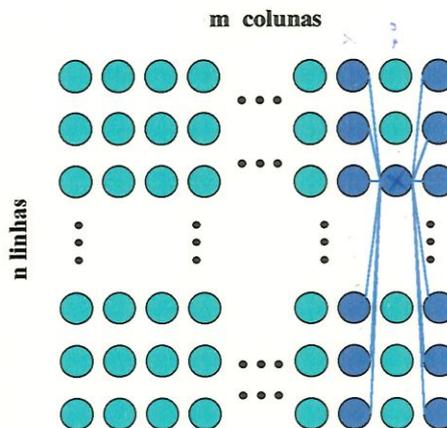


FIGURA 42 - Arquitetura da Rede Competitiva de Hopfield. As unidades de uma camada estão conectada a todas as unidades da camada imediatamente anterior e posterior.

Resume-se o algoritmo de aproximação poligonal empregando a Rede Competitiva de Hopfield da seguinte maneira:

1. Regra de aprendizagem: construção da matriz de conexão H , formada pelos desvios ponto-corda definidos pela equação (3.3);
2. Definição do estado inicial da rede: ajuste dos estados iniciais das unidades como m pontos, o mais equidistante possível, ao longo da curva;
3. Regra de propagação: cálculo da soma ponderada para cada unidade (x,i) ao longo da coluna i , usando a equação (3.13);
4. Regra de ativação: aplicação da regra *winner-take-all* (EQUAÇÃO 3.8) para se obter os novos estados para cada unidade da coluna;
5. Repetição dos passos 3. e 4. até não haver mudanças no estado da rede.
6. Montagem do polígono aproximado, cujos vértices q_1, q_2, \dots, q_m são apontados pelas unidades ativas da rede.

3.4. Resultados do processo de aproximação poligonal

A FIGURA 43 ilustra os resultados da aplicação da Rede Competitiva de Hopfield no processo de aproximação poligonal. Uma pistola de solda é utilizada como padrão, sendo seu contorno definido por cinquenta pontos ($n = 50$). Para demonstrar o poder do algoritmo, são realizadas três aproximações: inicialmente por um polígono de vinte e cinco lados ($m=25$), depois por um polígono de dez lados ($m = 10$) e finalmente por um polígono de cinco lados ($m = 5$).

Após a etapa de aproximação poligonal, segue uma etapa de rotação e normalização da figura (SEÇÃO 5.4.2). Este pós-processamento garante que a informação de formato do objeto, fornecida para a rede RBF através dos vértices do polígono aproximado, seja independente do tamanho e da orientação deste.

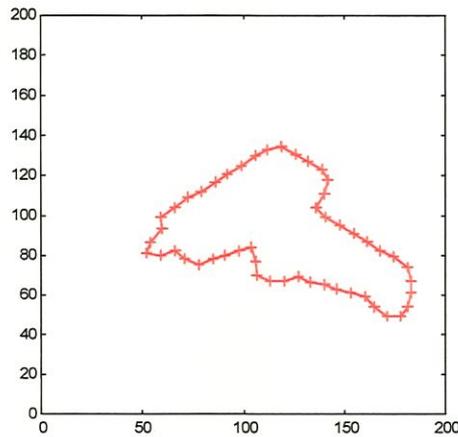


Figura original (n=50)

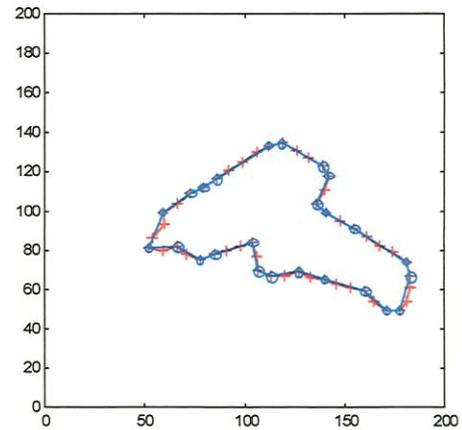
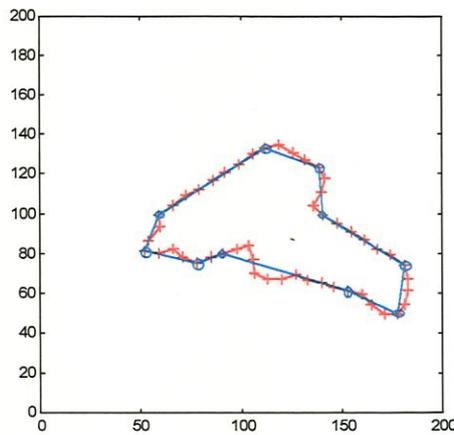
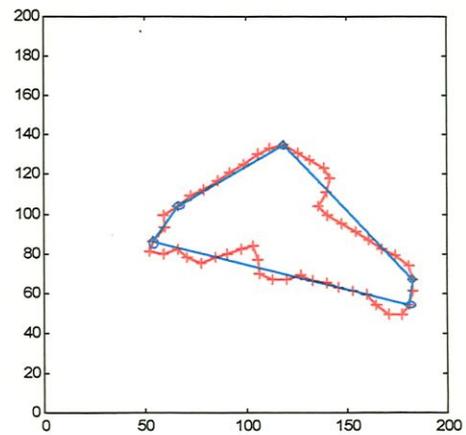
Aproximação em azul para $m = 25$ Aproximação em azul para $m = 10$ Aproximação em azul para $m = 5$

FIGURA 43 - Exemplos de aproximação poligonal da Rede Competitiva de Hopfield, considerando a imagem de uma pistola de solda capturada pelo sistema de visão. As linhas vermelhas representam a figura original e as linhas azuis representam o polígono aproximado.



4. REDES DE FUNÇÕES DE BASE RADIAL

4.1. Arquitetura das Redes de Funções de Base Radial (RBF)

Esta arquitetura de rede foi proposta por Broomhead e Lowe em 1988. Segundo HASSOUN (1995), as redes RBF apresentam estrutura *feed-forward*, na sua maioria constituídas de uma única camada escondida, totalmente conectada à camada de saída, como mostra a figura abaixo.

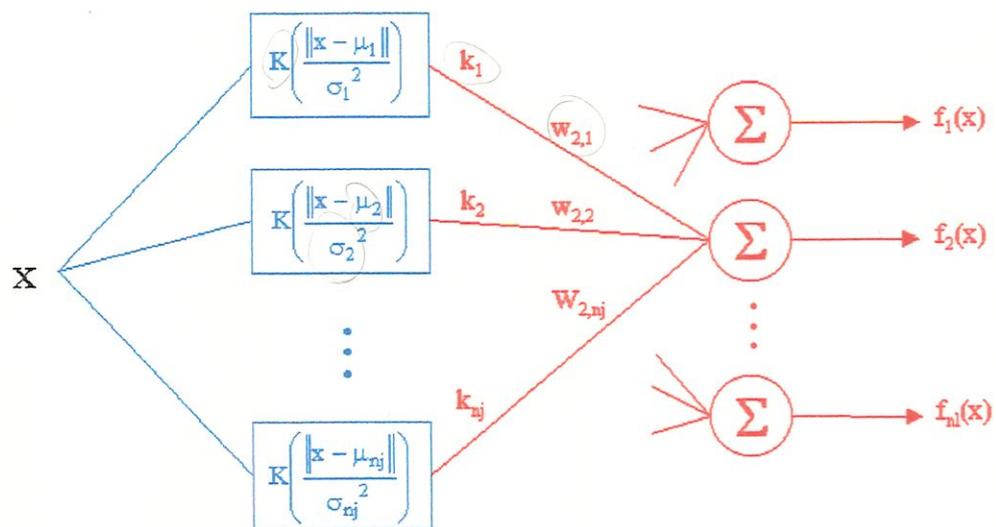


FIGURA 44 - Arquitetura da rede RBF. O vetor de entrada \underline{x} é apresentado a todas as unidades da camada intermediária. Cada unidade j calcula um valor k_j correspondente ao quão próximo se encontra o vetor \underline{x} do seu centro $\underline{\mu}_j$. As unidades da camada de saída realizam o somatório ponderado dos valores k_j , resultando na saída $f(\underline{x})$ da rede.

Todas as unidades escondidas (funções de base radiais) recebem simultaneamente um vetor de entrada \underline{x} n -dimensional. O processamento destas unidades é a principal característica das redes RBF. Cada uma fornece um sinal de saída k_j correspondente ao quão próximo se encontra o vetor de entrada de um vetor também n -dimensional $\underline{\mu}_j$ associado ao centro desta unidade. Sua função de ativação determina a distância entre o vetor de entrada e o centro da unidade. Esta função pode ser definida matematicamente por:

$$k_j(x) = K\left(\frac{\|x - \mu_j\|}{\sigma_j}\right) \quad (4.1)$$

sendo K uma função radialmente simétrica e estritamente positiva que atinge seu máximo no valor central e rapidamente cai a zero para qualquer posição longe deste centro. O parâmetro σ_j representa o raio da unidade. Assim, a saída k_j tem valor expressivo somente quando a distância $\|x - \mu_j\|$ (norma euclidiana) for menor que o raio σ_j . A figura abaixo apresenta algumas destas funções K .

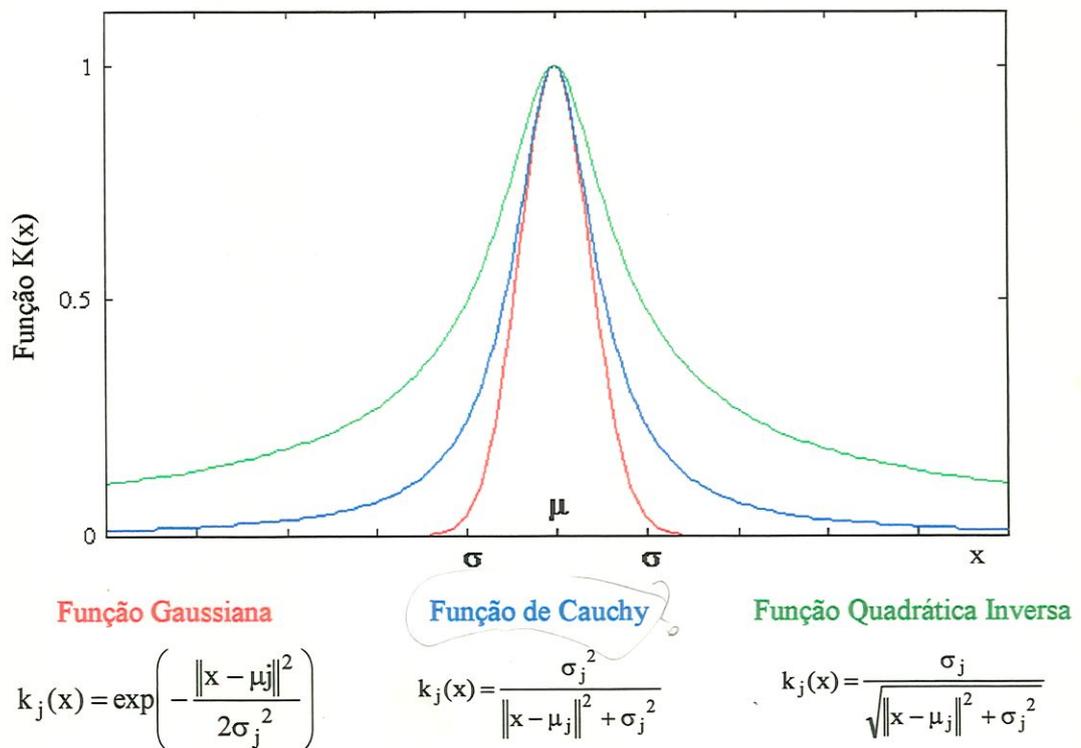


FIGURA 45 – Comparação entre três funções de ativação das unidades escondidas. A função gaussiana define uma saída mais restrita, enquanto a função quadrática inversa apresenta uma faixa de resposta bem mais aberta.

Analisando estas três funções, percebe-se que elas possuem comportamentos diferentes. A função gaussiana apresenta uma saída mais restrita, com valores significativos apenas para vetores de entrada \underline{x} bem próximos do centro $\underline{\mu}_j$, ou seja, efetivamente para os casos onde $\|x - \mu_j\| < \sigma_j$. A função de Cauchy, por sua vez, apresenta uma distribuição mais aberta, possibilitando valores de saída ainda significativos para os casos onde a distância entre os vetores \underline{x} e $\underline{\mu}_j$ seja pouco superior a σ_j . Já na função quadrática inversa, esta faixa de significância é ainda mais aberta. O uso de uma ou outra função depende da classificação

requerida. Emprega-se a função gaussiana nos casos onde é interessante utilizar funções de base radial mais fechadas, com faixas de significância bem definidas. Nas demais funções, principalmente para o caso quadrático inverso, ocorre uma iteração maior entre estas faixas, podendo uma entrada \underline{x} apresentar valores significativos em mais de uma função de base radial.

Assim, a acuracidade da camada escondida das redes RBF pode ser controlada por vários fatores: número de funções de base radial, localização do centro e raio de cada unidade e função de ativação.

Finalmente, a saída de uma rede RBF é o vetor 1-dimensional \underline{y} cuja l ésima coordenada é dada por:

$$f_l(\underline{x}) = \sum_{j=1}^{n_j} w_{lj} k_j(\underline{x}) \quad (4.2)$$

4.2. Estratégias de treinamento

As redes RBF apresentam dois estágios distintos: um estágio não-linear correspondente às operações da camada intermediária (EQUAÇÃO 4.1) e um estágio linear correspondente às operações da camada de saída (EQUAÇÃO 4.2). Já que estes dois estágios desempenham tarefas totalmente diferentes, é interessante separar o treinamento da camada intermediária do treinamento da camada de saída. Existem diversas estratégias de treinamento supervisionado e não-supervisionado (HAYKIN, 1994), que diferem pela forma como os parâmetros das funções de base radial são ajustados.

4.2.1. Seleção de centros e raios fixos

Através da definição de raios constantes para cada um dos padrões do conjunto de treinamento, esta abordagem escolhe alguns destes padrões para serem utilizados como centro das unidades intermediárias. Existem diversas metodologias para se determinar esta escolha. Posteriormente, um treinamento supervisionado é utilizado para se ajustar os pesos da camada de saída. Exemplos desta abordagem são os métodos: *Forward Selection* (SEÇÃO 4.3), *Global Ridge Regression* (SEÇÃO 4.4), *Local Ridge Regression* (SEÇÃO 4.5).

4.2.2. Seleção de centros fixos e raios ajustáveis

Esta abordagem também seleciona os centros das funções de base radial à partir do conjunto de treinamento. Porém, é proposto um ajuste não-supervisionado dos raios a fim de

controlar a interferência entre centros e minimizar as áreas de conflito. O ajuste dos pesos da camada de saída é realizado por um treinamento supervisionado. Um exemplo desta abordagem é o Dynamic Decay Adjustment - DDA (SEÇÃO 4.6).

4.2.3. Seleção supervisionada dos centros e raios

Uma abordagem mais genérica é o treinamento supervisionado de todos os parâmetros livres da rede. Após a montagem de uma função custo, são definidos os valores dos pesos da camada de saída e dos centros e raios da camada intermediária que minimizam esta função. Uma indicação natural para esta tarefa é o método do gradiente descendente (SEÇÃO 4.7).

4.3. Forward Selection

O método *Subset Selection* (RAWLINGS, 1988) é usado no ramo de estatística como um método de busca pelas melhores soluções à partir da comparação entre diferentes subconjuntos de um mesmo conjunto de candidatos. Porém, para um conjunto de M elementos, uma busca simples torna-se inviável, pois existem 2^{M-1} possíveis soluções. Algum método heurístico deve ser empregado para encontrar pequenas regiões de interesse dentro do espaço de subconjuntos. Um destes métodos heurísticos é o algoritmo *Forward Selection*, que parte de um subconjunto-solução vazio e vai acrescentando um elemento por vez. O elemento escolhido é aquele que provoca a maior redução do somatório do erro quadrático (SSE, *Sum-Squared-Error*). Outro processo heurístico é o *Backward Elimination*, que parte de um subconjunto-solução cheio e vai removendo um elemento por vez. O elemento a ser removido é aquele que menos reduz o SSE.

Assim, o algoritmo *Forward Selection* (ORR, 1996) é utilizado para treinamento não-supervisionado da camada intermediária das redes RBF, fazendo parte do grupo de seleção de centros e raios fixos (SEÇÃO 4.2.1). Ele determina, dentre o conjunto de treinamento, os padrões que irão formar a camada intermediária.

O algoritmo *Forward Selection* compreende as seguintes etapas:

1. Inicializar a rede com a camada intermediária vazia ($n_j = 0$);
2. Definir o raio r_j ($1 \leq j \leq np$) para cada padrão de treinamento, sendo np o número total de padrões;
3. Escolher o centro (padrão de treinamento) a ser adicionado à camada intermediária. Este centro é definido como aquele que mais reduz o SSE.

$$SSE = \sum_{i=1}^{np} (\hat{Y}^i - f(X^i))^2 \tag{4.3}$$

sendo \hat{Y} a matriz de respostas desejadas para todos os padrões.

4. Para cada nova unidade, estimar o erro médio seguindo o critério *Bayesian Information* (BIC);

$$\sigma_{BIC}^2 = \frac{SSE}{np \frac{np - \gamma}{np + (\ln(np) - 1)}} \tag{4.4}$$

5. Parar o processo quando um novo centro provocar o aumento do BIC (FIGURA 46). Para verificar a possibilidade de um mínimo local, acrescenta-se um nova unidade. Se ocorrer um novo aumento do BIC, as duas últimas unidades devem ser desconsideradas. Caso contrário, o processo continua.

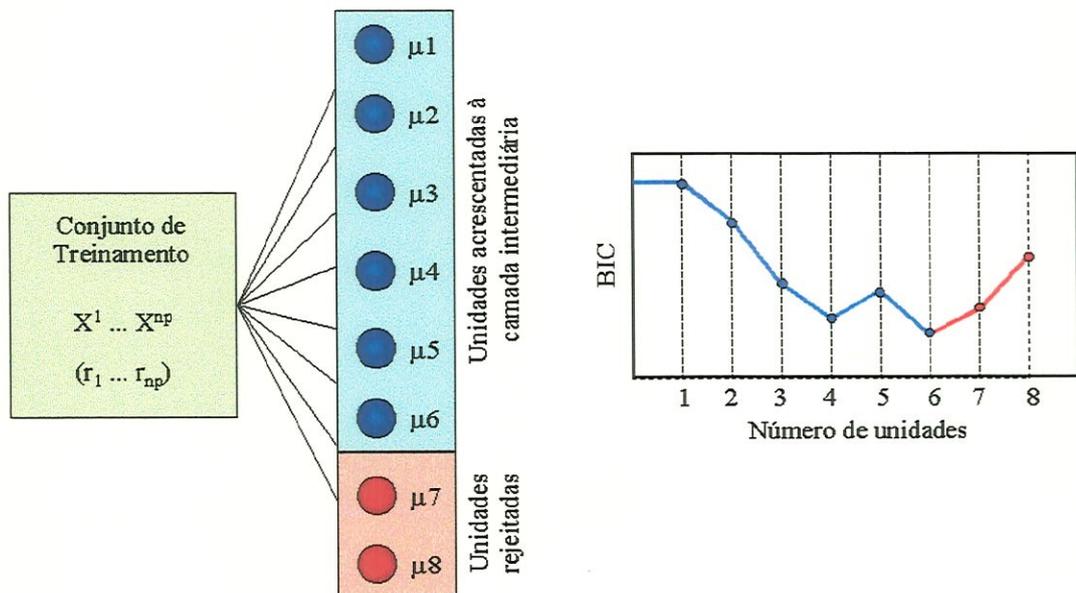


FIGURA 46 – Montagem da camada intermediária através do algoritmo *Forward Selection*. São escolhidos os padrões de treinamento que mais reduzem o SSE

Uma vez definida a camada intermediária, segue um treinamento supervisionado dos pesos w_{ij} da camada de saída através da equação (4.5). A prova desta equação parte da definição do SSE (EQUAÇÃO 4.3) e é demonstrada em (ORR, 1996). As linhas da matriz K ($np \times nj$) representam as respostas das funções de base radial para cada um dos np padrões de treinamento.

$$\hat{W} = (K^T K)^{-1} K^T \hat{Y}$$

$$K = \begin{bmatrix} k_1(X^1) & k_2(X^1) & \dots & k_{nj}(X^1) \\ \vdots & \vdots & & \vdots \\ k_1(X^{np}) & k_2(X^{np}) & \dots & k_{nj}(X^{np}) \end{bmatrix} \quad (4.5)$$

4.4. Global Ridge Regression

O método *Ridge Regression* (RAWLINGS, 1988) pode ser aplicado às redes neurais para punir conexões de neurônios consideradas irrelevantes. Isto se faz através da adição de um termo de penalização $\lambda \sum_{j=1}^{nj} w_j^2$, ponderado por um parâmetro regulador λ , ao somatório do erro quadrático (SSE). O vetor w_j reúne os pesos entre a unidade j da camada intermediária e as unidades da camada de saída. Assumindo um único parâmetro regulador para todos os pesos, define-se a seguinte função custo:

$$C = \sum_{i=1}^{np} (\hat{Y}^i - f(X^i))^2 + \lambda \sum_{j=1}^{nj} w_j^2 \quad (4.6)$$

Este parâmetro global caracteriza o algoritmo *Global Ridge Regression* [ORR, 1996, 1998], que é utilizado para treinamento não-supervisionado da camada intermediária. Todos os padrões são escolhidos como centros das funções de base radial e é definido um parâmetro regulador ótimo (EQUAÇÃO 4.7) que estabeleça um compromisso entre o ajuste e a penalização dos pesos da camada de saída. Para pequenos valores de λ , ocorre um ajuste livre de pesos, enquanto que para grandes valores de λ , ocorre um maior controle no sentido de evitar a definição de pesos altos.

$$\hat{\lambda} = \frac{\hat{Y}^T P^2 \hat{Y} \cdot \text{trace}(A^{-1} - \hat{\lambda} A^{-2})}{\hat{W}^T A^{-1} \hat{W} \cdot \text{trace}(P)} \quad (4.7)$$

Esta equação estabelece um processo iterativo, onde o parâmetro regulador é re-estimado até a convergência. No presente trabalho, foi utilizada uma estimativa inicial $\lambda = 0,01$. A FIGURA 47A mostra a convergência deste parâmetro até um valor ajustado de 10^{-11} . Neste ponto, o erro BIC também convergiu para um valor de 10^{-8} (FIGURA 47B). Uma análise do valor final de λ sugere que os pesos poderão ser ajustados praticamente sem nenhuma penalização.

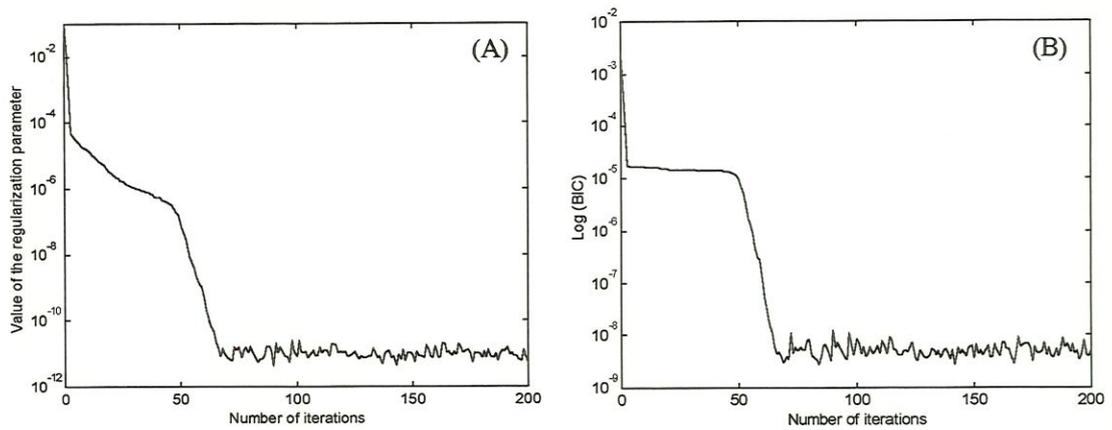


FIGURA 47 – Processo de ajuste do parâmetro regulador λ , comparado com a convergência do erro estimado BIC.

A equação (4.7) depende da matriz de variância (**A**) e da matriz de projeção (**P**). A matriz **A** indica uma aproximação para o cálculo da variância dos pesos ótimos definidos pela equação (4.11), uma vez que não existe nenhuma relação linear para esta definição.

$$A = Z^T Z + \lambda I_{nj} \quad (4.8)$$

sendo I_{nj} a matriz identidade de dimensão n_j . A matriz **P** projeta o valor de uma saída desejada (\hat{y}) na direção perpendicular ao espaço abrangido pelo modelo. Esta projeção representa o erro entre a saída da rede e a resposta desejada.

$$P = I_{np} - Z A^{-1} Z^T \quad (4.9)$$

Para uma visualização do processo, é interessante considerar um modelo bidimensional (plano verde da FIGURA 48). Um valor alvo \hat{y} (azul) é representado no modelo 2D através do vetor $(I_3 - P)\hat{y}$ e gera um erro representado pelo vetor $P\hat{y}$ (vermelho).

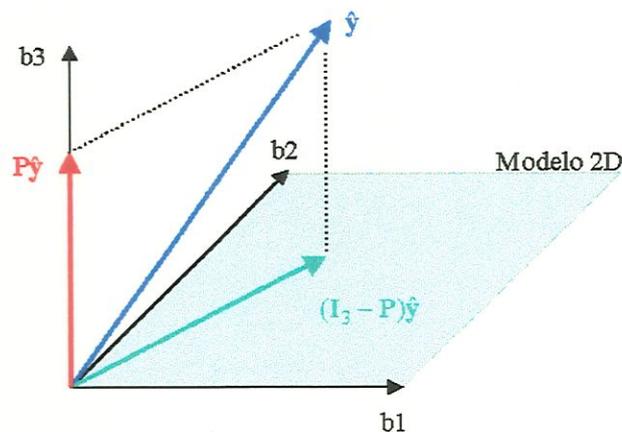


FIGURA 48 – Definição do vetor erro (vermelho) à partir da projeção da saída desejada \hat{y} (azul) na direção perpendicular ao modelo

A função custo (EQUAÇÃO 4.6) pode ser reescrita em função da matriz de projeção e das saídas desejadas:

$$C = \hat{y}^T P \hat{y} \quad (4.10)$$

Após o ajuste do parâmetro regulador, segue um treinamento supervisionado dos pesos da camada de saída, levando em consideração o parâmetro regulador.

$$\hat{W} = (Z^T Z + \lambda I_{n_j})^{-1} Z^T \hat{Y} \quad (4.11)$$

As equações (4.7) e (4.11) são demonstradas em (ORR, 1996).

4.5. Local Ridge Regression

O algoritmo *Local Ridge Regression* (ORR, 1996, 1998) difere do algoritmo anterior na definição do termo de penalização. Ao invés de tratar igualmente todos os pesos, utilizando o termo de penalização $\lambda \sum_{j=1}^{n_j} w_j^2$, eles são tratados separadamente e um parâmetro

regulador específico é associado a cada unidade intermediária: $\sum_{j=1}^{n_j} \lambda_j w_j^2$. A nova função custo é:

$$C = \sum_{i=1}^{np} (\hat{Y}^i - f(X^i))^2 + \sum_{j=1}^{n_j} \lambda_j w_j^2 \quad (4.12)$$

O parâmetro regulador agora é definido por um vetor $\lambda = [\lambda_1 \dots \lambda_{n_j}]$. Esta alteração afeta a matriz de variância (EQUAÇÃO 4.13) e conseqüentemente a matriz de projeção que é definida em função desta:

$$A = Z^T Z + \lambda^T I_{n_j} \quad (4.13)$$

A otimização destes parâmetros reguladores adota o mesmo processo anterior (EQUAÇÃO 4.7), com o agravante de que, agora, todos os n_j parâmetros devem ser otimizados simultaneamente. Assim, a cada otimização de um parâmetro λ_j , mudam-se os valores otimizados dos demais. A otimização segue um processo iterativo até que todos os padrões convirjam.

Durante esta convergência, alguns parâmetros podem atingir valores muito grandes ($\lambda_j = \infty$). Esta penalização máxima indica que estes parâmetros devem ser removidos da rede, o que significa a eliminação de unidades intermediárias desnecessárias.

Finalmente, a otimização dos pesos da camada de saída é realizado pela equação:

$$\hat{W} = A^{-1}Z^T\hat{Y} \quad (4.14)$$

4.6. Dynamic Decay Adjustment (DDA)

Todos os algoritmos apresentados anteriormente utilizam a estratégia de seleção de centros e raios fixos. O algoritmo DDA (BERTHOLD & DIAMOND, 1995), porém, propõe um ajuste dos raios das unidades. Utilizando informações sobre o posicionamento dos vizinhos, este método busca controlar a interferência entre centros e minimizar as áreas de conflito.

Dois *thresholds* são introduzidos ao processo: um positivo (θ^+) e outro negativo (θ^-). O *threshold* positivo representa um limite mínimo para uma função de base radial responder a um padrão da mesma classe. Caso este padrão gere a uma resposta menor que o *threshold*, uma nova unidade intermediária é criada. O *threshold* negativo representa o limite máximo de resposta para padrões de outras classes.

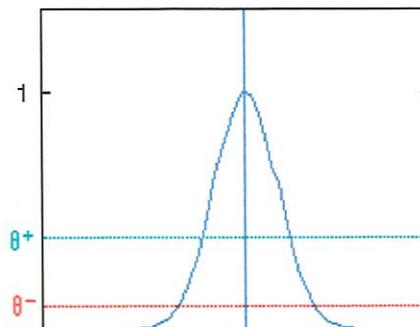


FIGURA 49 – Definição dos *thresholds* positivo θ^+ e negativo θ^- .

O algoritmo DDA monta a camada intermediária de forma a atender duas exigências fundamentais. Primeiro, que deve existir pelo menos uma função de base radial (protótipo) da classe c com ativação maior do que θ^+ quando for apresentado à rede um padrão \vec{X} da mesma classe (EQUAÇÃO 4.15). Segundo, que todos os padrões \vec{X} das demais classes devem ativar os protótipos da classe c com valores menores que θ^- (EQUAÇÃO 4.16).

$$\exists i : R_i^c(\vec{X}) \geq \theta^+ \quad (4.15)$$

$$\forall k \neq c, 1 \leq j \leq m_k : R_j^k(\vec{X}) < \theta^- \quad (4.16)$$

O algoritmo para montagem da camada intermediária, respeitando estas exigências, é sintetizado nas três figuras seguintes. As linhas horizontais determinam os *thresholds* positivos e negativos. Na FIGURA 50A, é inicialmente apresentado à rede um padrão da

classe vermelha. Este padrão é, então, acrescentado à camada intermediária como primeiro protótipo. Na FIGURA 50B, é apresentado à rede um padrão da classe verde, que também é acrescentado à rede. Seu raio é ajustado de forma que os protótipos das demais classes não respondam acima de θ^- .

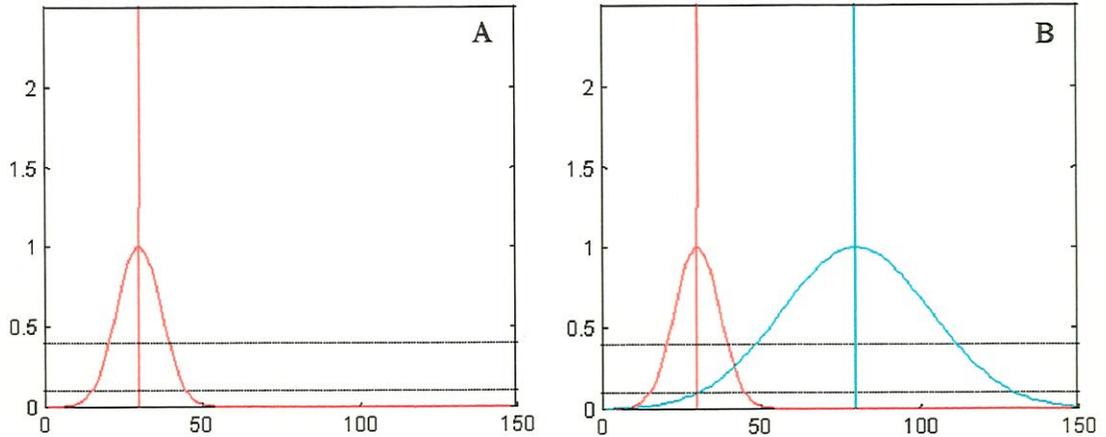


FIGURA 50 – Montagem da camada intermediária a partir do algoritmo DDA: (A) Definição do protótipo da classe vermelha; (B) Definição do protótipo da classe verde e ajuste do seu raio para evitar influência do protótipo vermelho.

Um novo padrão pertencente a classe azul é apresentado à rede (FIGURA 51A). Ele é acrescentado à camada intermediária e seu raio é ajustado de forma a minimizar a influência das classes vermelha e verde. Porém, nota-se uma interferência entre as classes verde e azul. O protótipo da classe verde apresentou uma ativação superior a θ^- em resposta ao novo protótipo da classe azul. O raio do primeiro deve então ser reduzido (FIGURA 51B) de forma a atender às exigências anteriores.

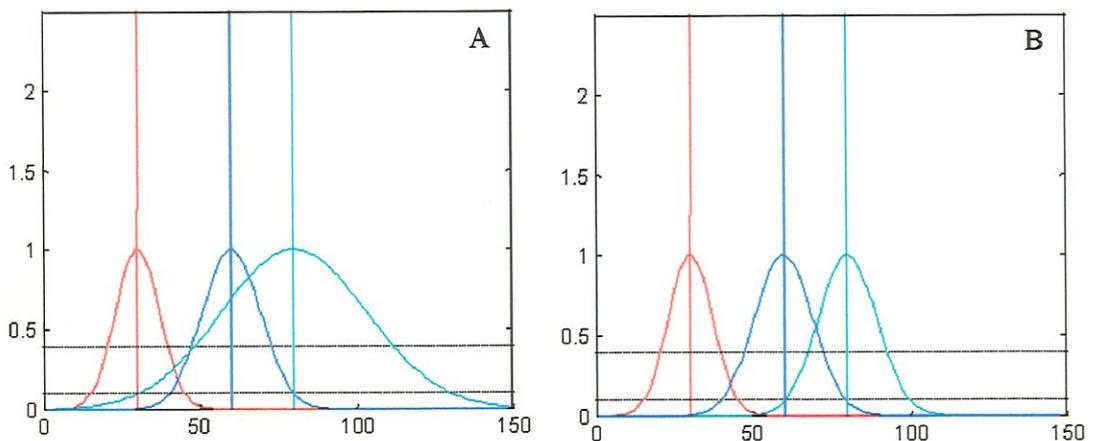


FIGURA 51 - Montagem da camada intermediária a partir do algoritmo DDA: (A) Definição de um novo protótipo da classe azul e ajuste de seu raio; (B) Ajuste do raio do protótipo verde a fim de reduzir a influências do protótipo azul.

Na FIGURA 52A, um outro padrão da classe azul é apresentado à rede. A ativação do protótipo azul é superior à θ^+ , o que significa que este centro abrange o padrão apresentado. O protótipo azul é então fortalecido. A FIGURA 52B apresenta um novo padrão da classe verde que não é abrangido pelo protótipo correspondente, o que provoca a sua adição como um segundo protótipo da classe. Seu raio é ajustado de forma a minimizar a influência das demais classes.

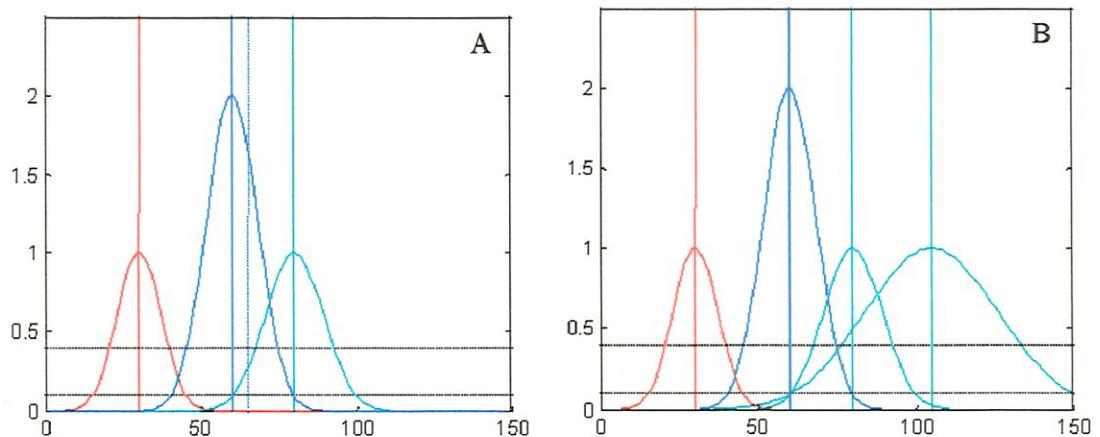


FIGURA 52 - Montagem da camada intermediária à partir do algoritmo DDA: (A) Fortalecimento do protótipo azul; (B) Acréscimo de um novo protótipo verde.

O ajuste dos valores dos *thresholds* influencia o desempenho do algoritmo. O ajuste de θ^+ altera o número de protótipos definidos para uma mesma classe. O ajuste de θ^- altera a interferência de uma classe sobre as demais. Quanto maior este parâmetro, maiores são os raios ajustados e conseqüentemente maior é a interação entre as classes.

Uma vez definida a camada intermediária, segue um treinamento supervisionado dos pesos da camada de saída (EQUAÇÃO 4.5).

4.7. Seleção supervisionada de centros e raios

Os quatro algoritmos propostos anteriormente baseavam-se no treinamento não-supervisionado da camada intermediária seguido do treinamento supervisionado dos pesos da camada de saída. Uma abordagem mais genérica propõe o treinamento supervisionado de todos os parâmetros livres da rede: centros, raios e pesos. Parte-se da definição do somatório dos erros quadráticos (SSE):

$$SSE = \sum_{i=1}^{np} e_i^2 \quad (4.17)$$

$$e_i = Y^i - f(x_i) \quad (4.18)$$

$$e_i = Y^i - \sum_{j=1}^{nj} w_j \cdot K\left(\frac{\|x_i - \mu_j\|}{\sigma_j^2}\right) \quad (4.19)$$

sendo e_i o erro do padrão x_i ; Y^i a saída desejada deste padrão; K , μ_j e σ_j a função ativação, o centro e o raio da unidade escondida j ; w_j é o vetor de pesos entre esta unidade e as unidades da camada de saída.

Considerando uma função de ativação gaussiana, HAYKIN (1994) reescreve a equação (4.19) da seguinte forma:

$$K\left(\frac{\|x_i - \mu_j\|}{\sigma_j^2}\right) = \exp\left(\frac{\|x_i - \mu_j\|^2}{2\sigma_j^2}\right) \quad (4.20)$$

$$e_i = Y^i - \sum_{j=1}^{nj} w_j \cdot \exp\left(\frac{\|x_i - \mu_j\|^2}{2\sigma_j^2}\right) \quad (4.21)$$

$$e_i = Y^i - \sum_{j=1}^{nj} w_j \cdot \exp\left[(x_i - \mu_j)^T \cdot \Lambda \cdot (x_i - \mu_j)\right] \quad (4.22)$$

sendo Λ a matriz de covariância definida em função dos raios σ_j .

O objetivo do treinamento é determinar os parâmetros livres da rede de forma a minimizar o valor do SSE. Esta determinação segue um processo iterativo, regido pelas seguintes equações:

1. Treinamento dos pesos da camada intermediária.

$$\frac{\partial SSE(n)}{\partial w_j(n)} = \sum_{i=1}^{np} e_i(n) \cdot \exp\left(\|x_i - \mu_j(n)\|^T \cdot \Lambda(n) \cdot \|x_i - \mu_j(n)\|\right) \quad (4.23)$$

$$w_j(n+1) = w_j(n) - \eta_1 \frac{\partial SSE(n)}{\partial w_j(n)} \quad j=1,2,\dots,nj \quad (4.24)$$

2. Treinamento dos centros.

$$\frac{\partial SSE(n)}{\partial \mu_j(n)} = 2w_j(n) \cdot \sum_{i=1}^{np} e_i(n) \cdot \frac{\partial}{\partial \mu_j(n)} \left[\exp\left(\|x_i - \mu_j(n)\|^T \cdot \Lambda(n) \cdot \|x_i - \mu_j(n)\|\right) \cdot \Lambda(n) \cdot [x_i - \mu_j(n)] \right] \quad (4.25)$$

$$\mu_j(n+1) = \mu_j(n) - \eta_2 \frac{\partial \text{SSE}(n)}{\partial \mu_j(n)} \quad j=1,2,\dots,n_j \quad (4.26)$$

3. Treinamento da matriz de covariância.

$$\frac{\partial \text{SSE}(n)}{\partial \Lambda(n)} = -w_j(n) \cdot \sum_{i=1}^{np} e_i(n) \cdot \frac{\partial}{\partial \Lambda(n)} \left[\exp \left(\|x_i - \mu_j(n)\|^T \cdot \Lambda(n) \cdot \|x_i - \mu_j(n)\| \right) \right] \cdot Q_{ij}(n) \quad (4.27)$$

$$Q_{ij}(n) = [x_i - \mu_j(n)] \cdot [x_i - \mu_j(n)]^T \quad (4.28)$$

$$\Lambda(n+1) = \Lambda(n) - \eta_3 \frac{\partial \text{SSE}(n)}{\partial \Lambda(n)} \quad (4.29)$$

Este treinamento difere profundamente do método *back-propagation*, uma vez que não envolve a retropropagação do erro. As três etapas de treinamento (pesos, centros e raios) utilizam simultaneamente o mesmo valor $e_i(n)$, porém com taxas de aprendizado η_1 , η_2 e η_3 diferentes.

4.8. Comparação entre os diversos algoritmos de treinamento

Nesta seção, os diversos algoritmos de treinamento das redes RBF são comparados com o algoritmo *back-propagation* (detalhado no APÊNDICE 1). Como tarefa, propõe-se definir os três pontos de contato da garra, utilizando o polígono aproximado pela rede Competitiva de Hopfield como entrada. Para o processo de treinamento, é utilizado o conjunto de padrões montados a partir de quinze classes: um quadrado, quatro retângulos, sete triângulos, um círculo e duas elipses (os padrões são apresentados no APÊNDICE 2). Cada classe é formada por um polígono regular e três variações deste, definidas através da aplicação de um erro gaussiano de 1% ao polígono regular. Os pontos alvos foram obtidos através de um algoritmo de busca pelos melhores pontos de fixação (ANEXO 1).

Seis modelos distintos de redes neurais foram implementados:

- duas redes MLP com tamanhos diferentes, treinadas pelo algoritmo *back-propagation* com taxa de treinamento adaptativa.
- quatro redes RBF, treinadas por algoritmos diferentes: *Forward Selection* (FS), *Global Ridge Regression* (GRR), *Local Ridge Regression* (LRR) e *Dynamic Decay Algorithm* (DDA).

Inicialmente, foi proposto o teste de recuperação de padrões. Através deste teste, pretendia-se avaliar a capacidade de cada algoritmo de recuperar as informações previamente treinadas, considerando a ação de uma escala crescente de ruído gaussiano. Assim, para cada algoritmo, foram apresentados cem padrões arbitrários, escolhidos aleatoriamente dentre os polígonos regulares das quinze classes de treinamento. Uma escala crescente de ruído gaussiano (0 a 4%) foi aplicada a cada um destes conjuntos de teste e a TABELA 3 apresenta as taxas de recuperação correta obtidas por cada um dos algoritmo de treinamento. Um padrão era considerado corretamente recuperado quando o valor SSE era menor que um limite estipulado (0,003).

TABELA 3. Teste de recuperação de padrões, apresentando a taxa de recuperação correta e o tempo de treinamento para cada algoritmo (Pentium II 233MHz). Os valores entre parênteses indicam a média do SSE.

Algoritmo (unidades por camada)	Ruído gaussiano					Tempo de treinamento (tempo de CPU)
	0% (SSE)	1% (SSE)	2% (SSE)	3% (SSE)	4% (SSE)	
MLP-25 (20x25x6)	100% (3,04e-4)	99% (5,59e-4)	97% (0,0012)	78% (0,0019)	56% (0,0038)	39min01seg (53808 épocas)
MLP-50 (20x50x6)	100% (3,60e-4)	100% (6,16e-4)	95% (0,0013)	83% (0,0022)	60% (0,0039)	44min48seg (44620 épocas)
RBF-FS (20x35x6)	100% (1,76e-4)	100% (3,99e-4)	97% (9,83e-4)	54% (0,0038)	23% (0,0090)	2,03seg
RBF-GRR (20x60x6)	100% (5,26e-7)	100% (2,56e-5)	100% (1,49e-4)	98% (6,70e-4)	74% (0,0018)	6,01seg
RBF-LRR (20x53x6)	100% (6,06e-4)	91% (0,0009)	80% (0,0017)	74% (0,0019)	62% (0,0033)	20,65seg
RBF-DDA (20x15x6)	100% (6.51e-6)	89% (0.0024)	11% (0.0292)	-	-	2,31seg

Analisando a TABELA 3, nota-se que o melhor desempenho na tarefa de recuperação de padrões foi apresentado pelo algoritmo *Global Ridge Regression*. Este obteve a melhor porcentagem de recuperação correta de padrões dentre os algoritmos testados, destacando-se principalmente nos casos de ruído elevado (FIGURA 53C). Os algoritmos *Forward Selection* e *Back-propagation* obtiveram desempenho intermediário. Contudo, o primeiro

apresentou uma considerável queda de desempenho para taxas mais elevadas de ruído (FIGURA 53D). Já as redes MLP, apesar de terem obtido resultados ainda satisfatórios nestas condições de ruído (FIGURA 53E), exigiram um tempo de treinamento incompatível com *aplicações on-line*. Finalmente, o algoritmo DDA demonstrou ser um método muito sensível a ruídos (FIGURA 53F).

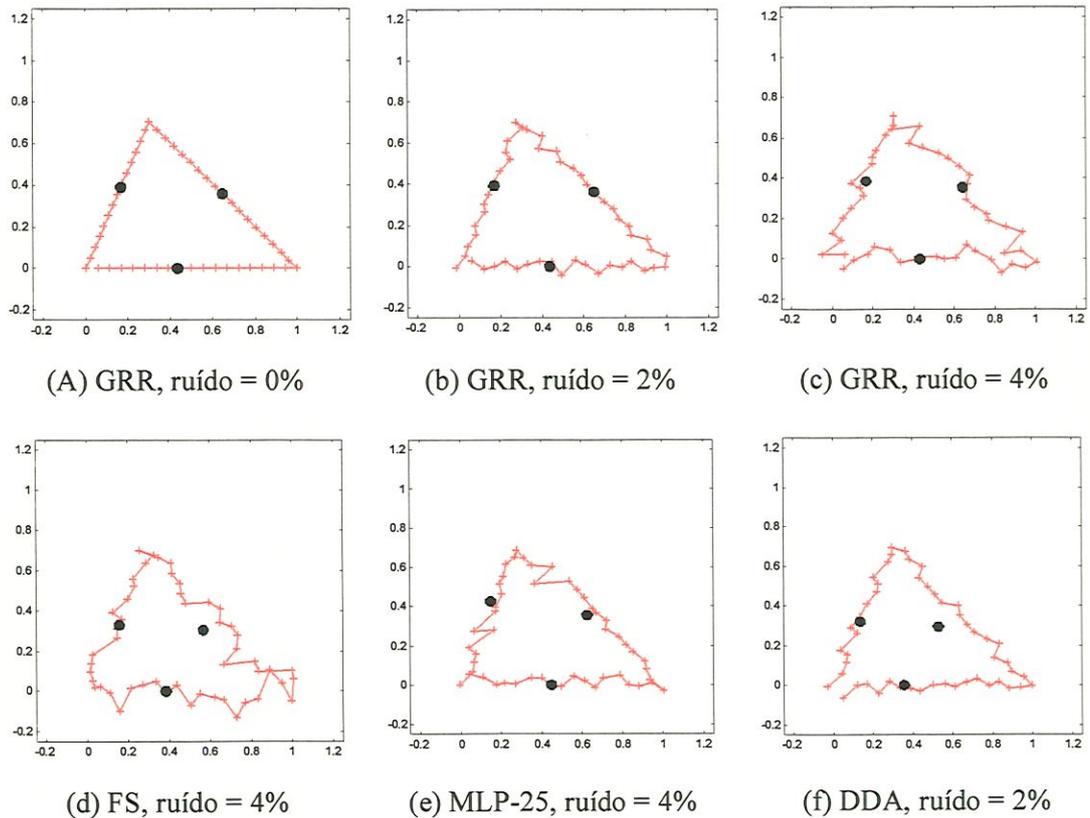


FIGURA 53: Resultados típicos dos diversos algoritmos durante teste de recuperação de padrões, considerando uma taxa crescente de ruído gaussiano. Os pontos pretos representam os pontos de contato calculados por cada algoritmo.

Após o teste de recuperação de padrões, foi proposto um teste típico de generalização, utilizando dois objetos trapezoidais e ruidosos (FIGURA 54). Estes objetos representam polígonos com algumas características semelhantes aos retângulos e triângulos utilizados como padrões de treinamento. O primeiro objeto assemelha-se a uma forma quadrada (teste A), enquanto o segundo tem uma aparência mais próxima de um triângulo (teste B). Os valores SSE, calculados para cada caso, são apresentados na TABELA 4.

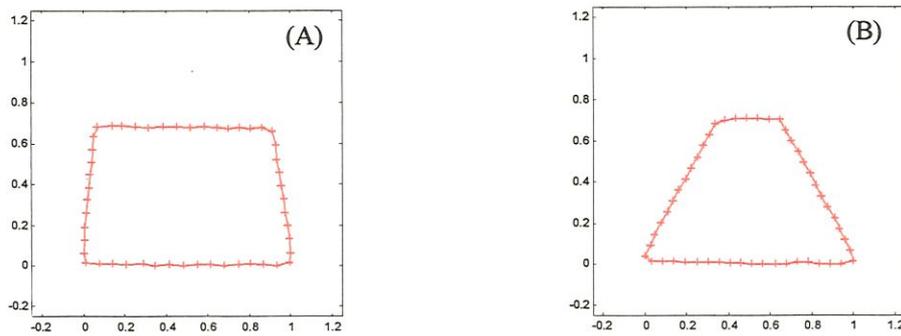


FIGURA 54: Exemplos de padrões não treinados utilizados para o teste de generalização: (A) forma trapezoidal mais próxima de um quadrado, (B) forma trapezoidal mais próxima de um triângulo.

TABELA 4: Resultados do teste de generalização, apresentando o SSE para diferentes configurações de rede e diferentes algoritmos de treinamento.

Algoritmo (unidades por camada)	SSE (teste A)	SSE (teste B)
MLP-25 (20x25x6)	0.0542	0.0657
MLP-50 (20x50x6)	0.0637	0.0841
RBF-FS (20x35x6)	0.0518	0.0736
RBF-GRR (20x60x6)	0.0377	0.0697
RBF – LRR (20x53x6)	0.0425	0.0941
RBF-DDA (20x15x6)	0.0822	0.1172

Analisando a TABELA 4, nota-se que o melhor desempenho foi novamente obtido pela rede RBF treinada pelo algoritmo *Global Ridge Regression*. Em ambos os casos, este conjunto apresentou os menores valores de SSE, demonstrando uma excelente capacidade de generalização do aprendizado adquirido. O desempenho relativo dos demais algoritmos também se manteve (FIGURA 55). As redes MLP-25, MLP-50, RBF-FS e RBF-LRR apresentaram resultados um pouco inferiores aos apresentados pela rede RBF-GRR, enquanto a rede RBF-DDA novamente apresentou um resultado insatisfatório.

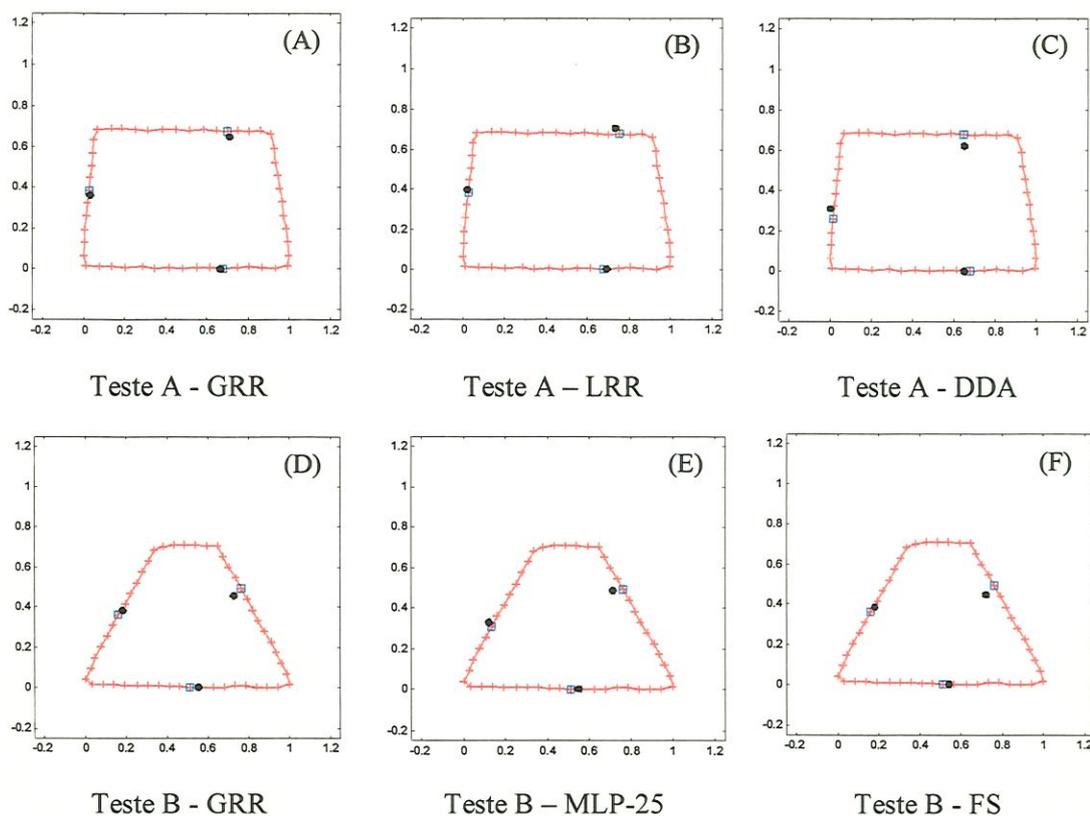


FIGURA 55 – Resultados típicos do teste de generalização. Para cada caso, são definidos os pontos de contato (pontos preto) calculados pelas redes neurais (MLP ou RBF), bem como os pontos ótimos (quadrados azuis) calculados pelo processo de busca iterativa.

Na figura acima, os pontos alvos de contato foram computados através de um processo de busca por soluções ótimas. Os critérios para analisar a qualidade de fixação, bem como outros detalhes do processo, serão melhor detalhados no ANEXO 1.

Com base nos resultados dos testes apresentados nesta seção, conclui-se que a rede RBF, treinada pelo algoritmo *Global Ridge Regression*, representa o melhor conjunto para a execução da tarefa proposta: determinar três pontos estáveis de contato para a fixação de um objeto arbitrário por uma garra de robô. Outras respostas do algoritmo de treinamento escolhido serão apresentadas na SEÇÃO 5.4.3 e 5.6.

4.9. Comparação de performance entre RBF e MLP

Em suma, HASSOUN (1995) caracteriza as redes RBF por um treinamento muito mais rápido quando comparado às redes MLP (TABELA 3), embora normalmente necessitem de um conjunto de treinamento muito maior para atingir a mesma acuracidade. Esta acuracidade pode ser aumentada se o treinamento das funções de base radial for

realizado pelo método supervisionado (SEÇÃO 4.7) em detrimento à redução da velocidade do processo.

A maior velocidade de treinamento se explica pela adoção da etapa de aprendizado não-supervisionado da camada intermediária. No algoritmo *Forward Selection*, um rápido processo heurístico é utilizado para definição dos centros das funções de base radial. Nos algoritmos *Global e Local Ridge Regression*, somente os parâmetros reguladores são otimizados. No algoritmo DDA, o ajuste dinâmico dos raios é atingido após poucas iterações. Apenas o processo de seleção supervisionada de centros e raios é mais demorado. O resultado depende da convergência de três equações independentes (4.24, 4.26 e 4.29). Porém, este processo ainda é bem mais rápido do que o algoritmo *back-propagation* utilizado no treinamento das redes MLP, já que este último depende do ajuste de todos os pesos da rede através da retropropagação do erro.

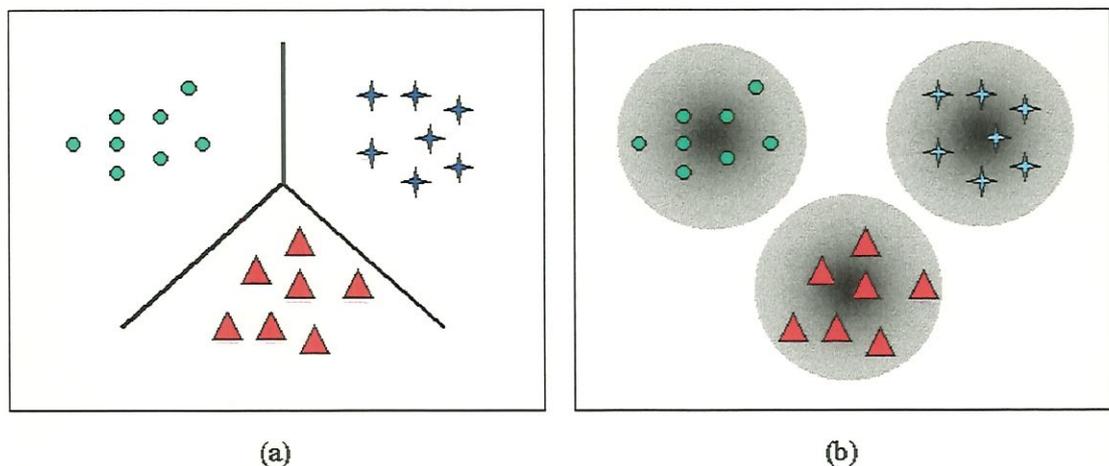


FIGURA 56 - Formas de divisão do espaço de entrada: (A) a rede MLP divide o espaço em hiperplanos; (B) a rede RBF define regiões de atração local.

As variações acima acontecem devido ao processamento diferenciado das unidades da camada intermediária. Na RBF, existe uma única camada intermediária formada por funções de base radial, que apresentam um funcionamento local e respondem apenas aos vetores de entrada situados próximo ao seu centro. Na MLP, a característica sigmoideal de uma ou mais camadas intermediárias possibilita respostas significativas mesmo para vetores de entrada distantes dos vetores de treinamento. Assim, as redes apresentam características próprias quando realizam a tarefa de classificação de padrões. No exemplo da FIGURA 56, os dados de entrada estão dispostos em três classes. As redes MLP dividem o espaço de entrada através de hiperplanos, enquanto as redes RBF utilizam as funções de base radial para definir regiões de atração local.

Ainda segundo HASSOUN, a utilização de uma ou outra rede depende da tarefa a ser realizada. As redes MLP são vantajosas nos casos onde existam poucos padrões de treinamento. As redes RBF são indicadas nos casos onde seja fácil a geração de padrões de treinamento ou nos casos onde seja necessário um treinamento on-line.

5. METODOLOGIA E RESULTADOS

5.1. Sistema de fixação

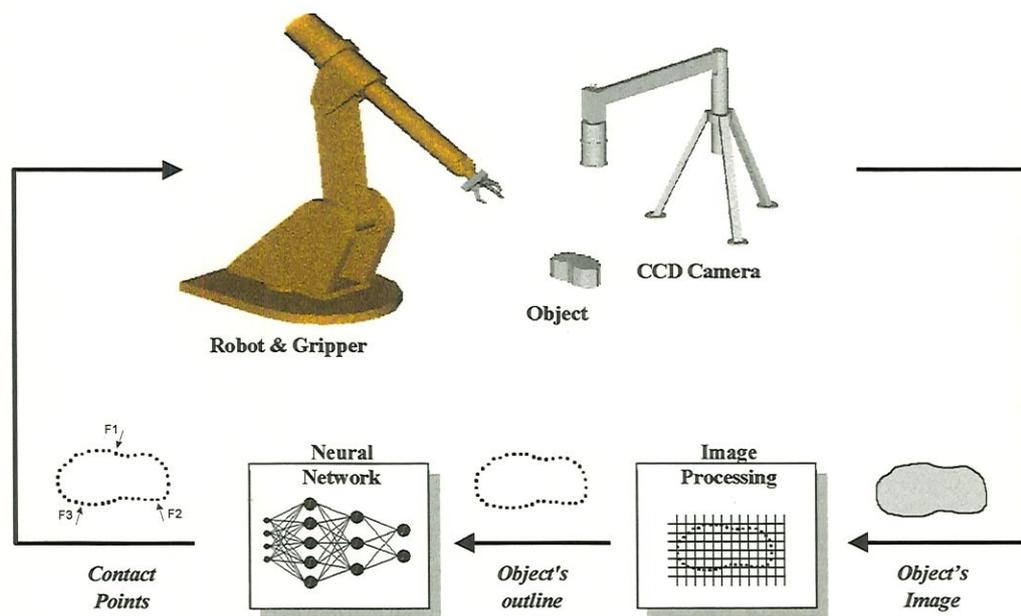


FIGURA 57 – Diagrama mostrando a sequência de operação do sistema de fixação. A imagem do objeto é capturada pela câmera CCD. Esta imagem é processada em um conjunto finito de pontos que definem o contorno do objeto. Este conjunto é fornecido ao sistema neural, que calcula os três pontos de fixação da garra.

A FIGURA 57 mostra o sistema proposto, que é capaz de utilizar uma garra, montada sobre um manipulador robótico, para capturar objetos dentro do seu espaço de trabalho. A única fonte de informação é uma câmera CCD, que fornece a imagem de topo do objeto. Esta imagem é processada em um conjunto finito de pontos que definem o contorno da peça. Em seguida, uma etapa neural utiliza este conjunto de pontos para determinar os locais de fixação dos dedos da garra. Com a utilização de uma garra de três dedos, o sistema neural deve ser capaz de escolher três pontos de contato que garantam uma fixação estável, independente do tamanho, forma, material, posicionamento ou orientação da peça. Finalmente, são estimadas forças a serem aplicadas em cada contato de forma a evitar escorregamentos. A FIGURA 58 mostra o sistema final implementado.

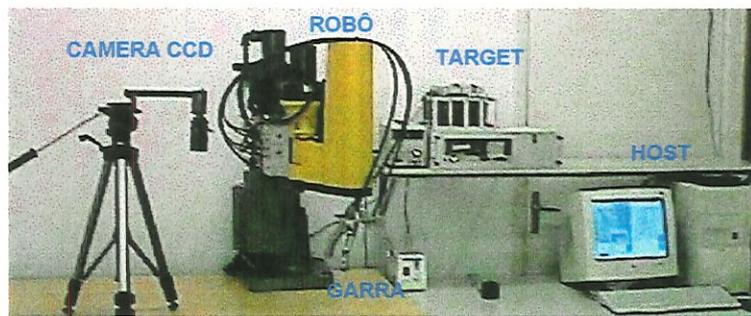


FIGURA 58 – Foto do sistema de fixação implementado.

A seguir, serão detalhadas cada uma destas partes: o projeto da garra, o sistema de processamento de imagem, a sistema neural formado pelas redes Competitiva de Hopfield e RBF e o sistema para estimativa das forças a serem aplicadas.

5.2. Projeto mecânico da garra

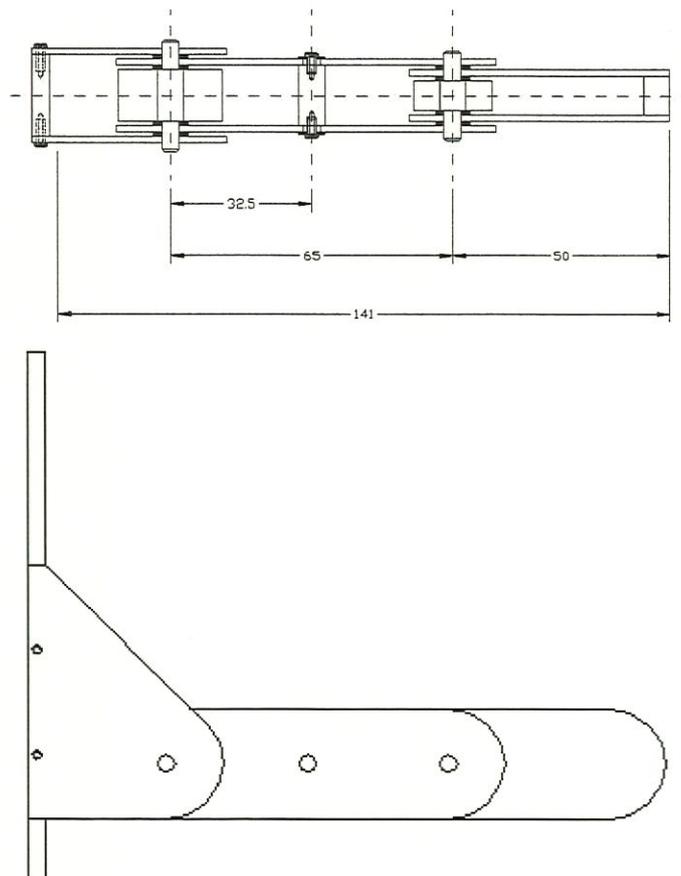


FIGURA 59 – Projeto mecânico do dedo, composto por uma base e duas falanges. Em cada junta, são montadas polias para condução do sistema de tendões antagonísticos.

A garra foi projetada numa configuração de três dedos e dois graus de liberdade por dedo. Cada um é composto basicamente por três partes (FIGURA 59): uma base de fixação e duas falanges. Todas estas partes são construídas a partir de pares de chapas de alumínio com 1,5 mm de espessura. Para reduzir o atrito nas junções, foram montados anéis de bronze entre as paredes das falanges, com o objetivo de evitar o contato direto destas durante seus movimentos relativos. A estrutura projetada é leve, fácil de montar e torna livre o caminho dos tendões. A FIGURA 60B apresenta o primeiro protótipo construído para a garra.

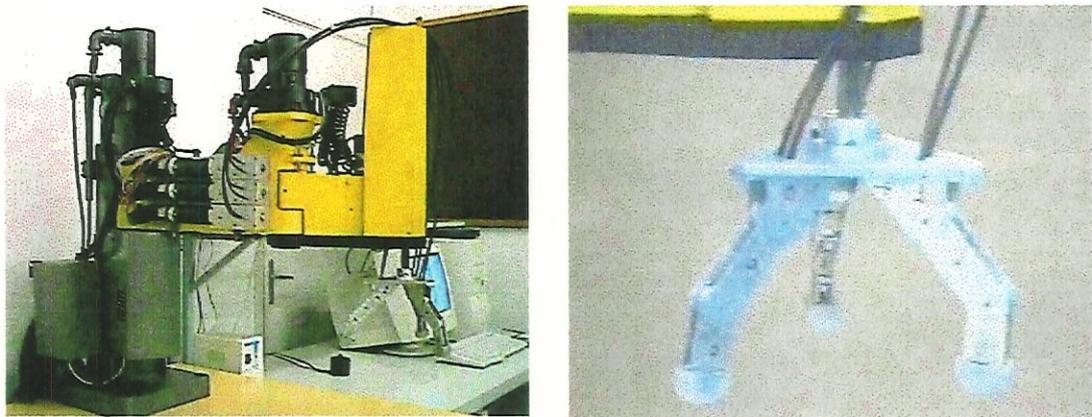


FIGURA 60 – (A) Sistema de acionamento da garra; (B) protótipo da garra.

Esta garra é acionada por três motores DC (FIGURA 60A). O sistema utiliza apenas um motor para cada dedo, estando acoplado o acionamento dos dois graus de liberdade. O movimento é transmitido por dois tendões antagônicos (FIGURA 61), sendo um responsável pela abertura e outro pelo fechamento da garra. Um sistema de polias é responsável pela condução dos tendões através do dedo, de forma que uma das pontas do tendão é presa ao topo do dedo e a outra é enrolada em uma bobina acoplada ao motor (FIGURA 62). Os tendões são enrolados em direções opostas, de forma que o giro da bobina tensiona um enquanto libera o outro. As polias não transmitem nenhum torque ao eixo, uma vez que são montadas livres sobre este. O movimento é gerado pelo encurtamento dos tendões, o que provoca a deflexão do dedo.

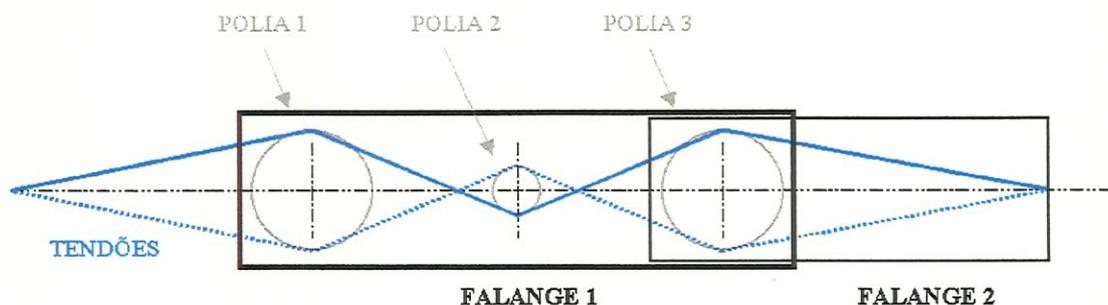


FIGURA 61 – Configuração dos tendões e das polias (vista lateral simplificada). O movimento é gerado pelo encurtamento de um dos tendões, provocando a deflexão do dedo.

Foi também construído um dispositivo, em alumínio, para acoplamento do motor e transmissão do movimento para os tendões (FIGURAS 62 e 63). Os fios vermelhos representam os tendões vindos do dedo e guiados por tubos (pretos). Estes tubos permitem manter as tensões dos tendões durante o transporte destes até a bobina. Existe ainda um mecanismo de ajuste (parafusos azuis) que permite regular estas tensões. Os tendões são enrolados na bobina, um no sentido horário e o outro no sentido anti-horário. A bobina ($\phi = 20$ mm) é usinada diretamente no eixo ($\phi = 12$ mm), sendo este bi-apoiado em rolamentos e unido diretamente ao eixo do redutor através de uma bucha interferente. Para o acionamento, foi utilizado um servo-motor MAXON (ref. internet) modelo RE036-072-07EAB200A, com 70W de potência e caracterizado por um torque de 0.085 Nm a uma rotação de 1560 RPM. O redutor, modelo GP042A103-0156C1A00A, é um planetário de três estágios e relação de transmissão de 156 x 1. Utiliza-se ainda um encoder HP (ref. internet), modelo HEDS 5540, acoplado ao motor. Com três canais e 500 pulsos por minuto, este sensor permite a realimentação do sistema.

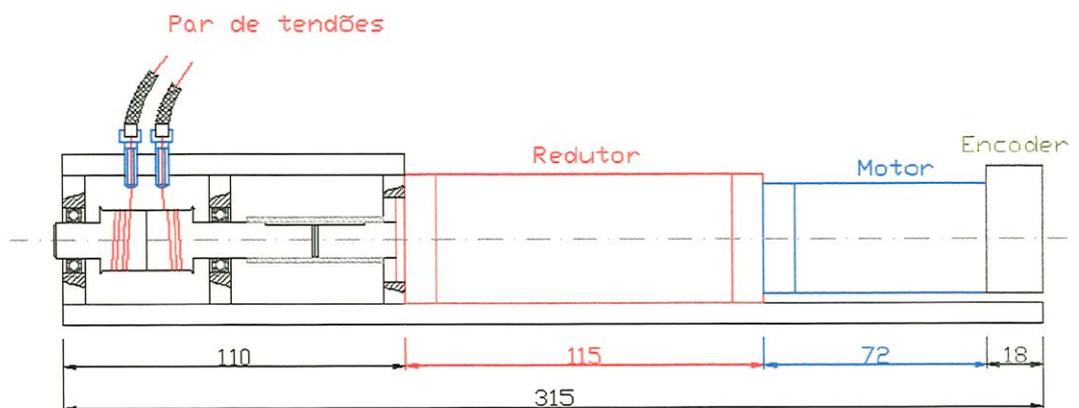


FIGURA 62 – Projeto mecânico do dispositivo de acionamento da garra.

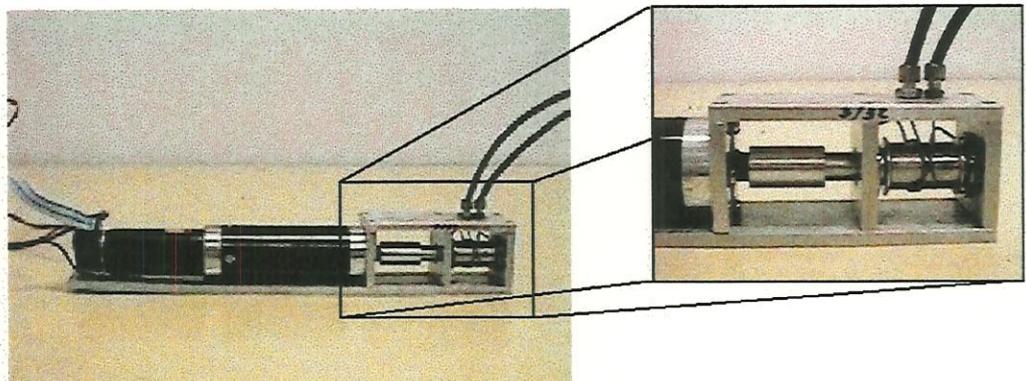


FIGURA 63 – Dispositivo de acionamento da garra. Em destaque, é apresentada a bobina responsável pelo tracionamento dos cabos.

Os *drivers* dos motores DC e dos encoders são conectados a um sistema dedicado VME. Desta forma, todos os processos (controle de motores, leitura de sensores, dentre outros) são executados na placa MOTOROLA MVME 162 através do ambiente operacional de tempo real XOBBERON (REISER & WIRTH, 1992) Contudo, todos os *softwares* de controle são desenvolvidos num microcomputador PC (*host*).



FIGURA 64 – Sistema dedicado VME, destacando a placa MOTOROLA MVME 162.

No momento, a garra está funcionando em caráter experimental. Para finalizar o projeto, propõe-se acoplar sensores de força aos dedos, o que possibilitará a medição e o controle das forças aplicadas pela garra.

5.3. Sistema de processamento de imagem

Para o processamento da imagem do objeto a ser capturado, foi utilizada uma câmera CCD preto/branco modelo KP-M1, comercializado pela HITASHI DENSHI LTD, trabalhando em conjunto com uma placa de aquisição de imagens em tempo real Video Blaster RT300, comercializada pela CREATIVE LABS. Este conjunto permite aquisição de imagens com uma resolução de 320x240 pixels, a uma taxa de 30 *frames* por segundo.

Para implementação do algoritmo de detecção de borda, foi utilizado o ambiente de programação visual DELPHI 3 (CORNELL, 1995). Além de facilitar a criação de uma interface gráfica amigável, o DELPHI apresenta um componente específico para captura de imagens: TSCAP32 (TÜEFE). Este recurso estabelece automaticamente a comunicação com a placa de aquisição, o que permite capturar e apresentar a imagem na tela.

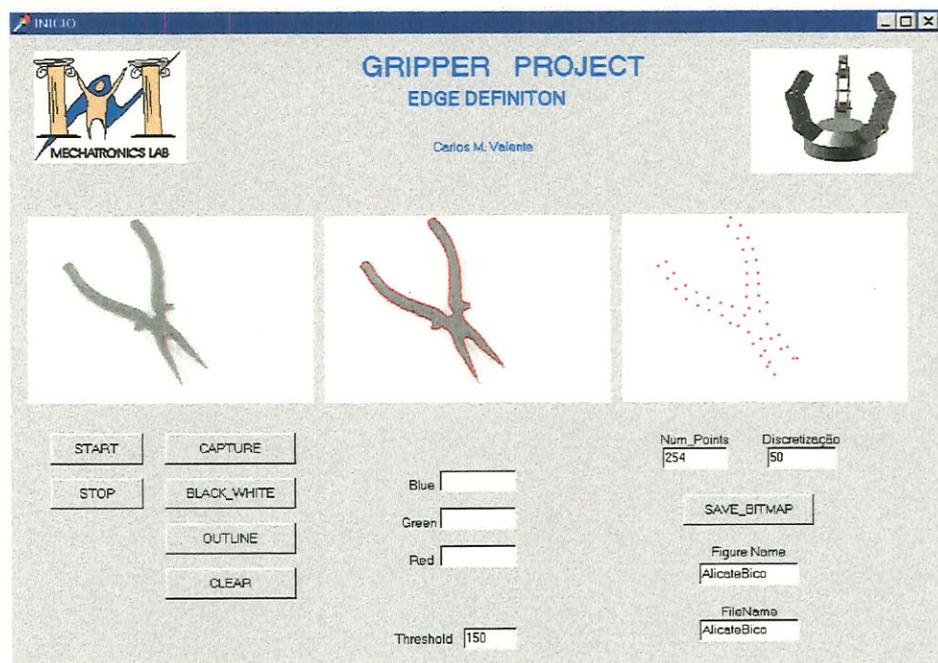


FIGURA 65 – Tela de controle do sistema de captura e processamento da imagem. Nesta tela, é permitida uma visualização *on-line* da cena capturada pela placa de aquisição (imagem à esquerda), bem como o contorno do objeto (imagem central) e sua discretização em cinquenta pontos (imagem à esquerda).

Nesta tela, destacam-se três imagens: à esquerda aparece a imagem original capturada pelo componente TSCAP32, ao centro aparece a imagem processada, com o contorno do objeto destacado em vermelho, e à direita aparece este contorno discretizado em um número pré-determinado de pontos. Os botões são utilizados para controlar os diversos eventos de processamento.

Os botões *START* e *STOP* determinam respectivamente o início e o término do processo de visualização *on-line* da cena, projetada na imagem1. Esta imagem funciona apenas como um visor da câmera CCD. O botão *CAPTURE* executa a captura de um *frame* desta visualização, gerando uma imagem2 no padrão *bitmap*. Esta imagem permite a manipulação direta de cada um dos seus pixels. Através do botão *BLACK_WHITE*, aplica-se uma binarização à imagem2, tornando preto todos os pixels inferiores ao *threshold* e branco todos os demais. O valor deste *threshold* pode ser ajustado na própria tela. Finalmente, sob o botão *OUTLINE*, está implementado o algoritmo do vizinho mais próximo para a detecção do contorno do objeto. Devido às características das imagens (presença de um objeto em destaque sobre o fundo da cena), este algoritmo tornou-se o mais adequado por gerar bons resultados em um reduzido espaço de tempo. A rapidez do processamento é importante para o funcionamento da garra em tempo real.

O contorno é destacado em vermelho na própria imagem2 (FIGURA 66A e 66C). Para o caso do alicate da FIGURA 65, este contorno foi definido através de 254 pontos. Porém, o sistema neural não necessita de toda esta definição para identificar o formato do objeto. Para acelerar o processamento neural, são escolhidos somente cinquenta pontos equidistantes, que representam o contorno da peça de uma forma mais simplificada (FIGURA 66B e 66D).

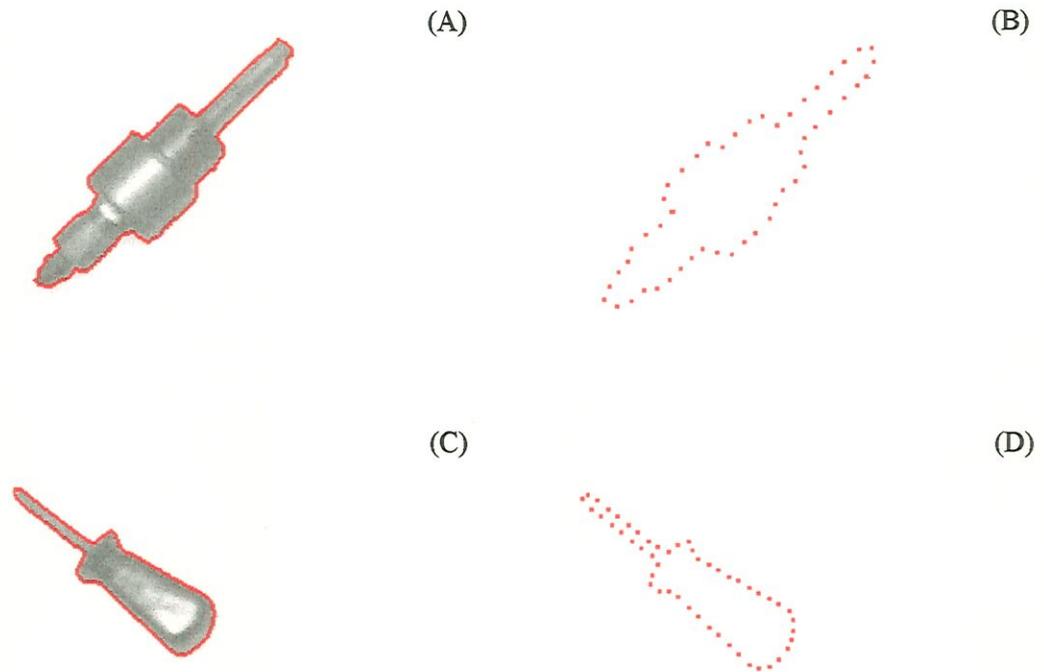


FIGURA 66 – Resultados do processamento de imagem: (A) e (C) apresentam os objetos e os contornos destacados em vermelho. (B) e (D) apresentam estes contornos definidos a partir de cinquenta pontos igualmente espaçados.

As coordenadas (x,y) destes cinquenta pontos do contorno do objeto são armazenadas em um arquivo, de forma a deixá-las à disposição da próxima etapa de processamento.

5.4. Sistema de definição dos pontos de contato

Todas as etapas para detecção dos três pontos de fixação da garra foram implementadas no Matlab versão 5.2, comercializado pela MathWorks[®]. Além de facilitar a programação de operações matemáticas, principalmente operações matriciais, o software também apresenta algumas ferramentas (*toolboxes*) específicas para implementação das redes MLP e RBF.

As diversas etapas são detalhadas a seguir, destacando cada um dos resultados parciais. Para isto, são utilizados os mesmos objetos da FIGURA 66.

5.4.1. Aproximação Poligonal (Rede Competitiva de Hopfield)

O primeiro passo é a leitura do arquivo com as coordenadas (x,y) dos cinquenta pontos que definem o contorno da peça. A partir destes dados, é iniciado o processo de aproximação poligonal, utilizando uma Rede Competitiva de Hopfield na configuração matricial com cinquenta linhas ($n = 50$) e dez colunas ($m = 10$). Isto significa que, sobre os pontos do contorno, será aproximado um polígono de dez lados. Os dados de entrada são utilizados para o cálculo da matriz de conexão (EQUAÇÃO 3.4). Segue o processo iterativo até a convergência para uma posição estável da rede. Esta posição indica os pontos definidos para vértice do polígono.

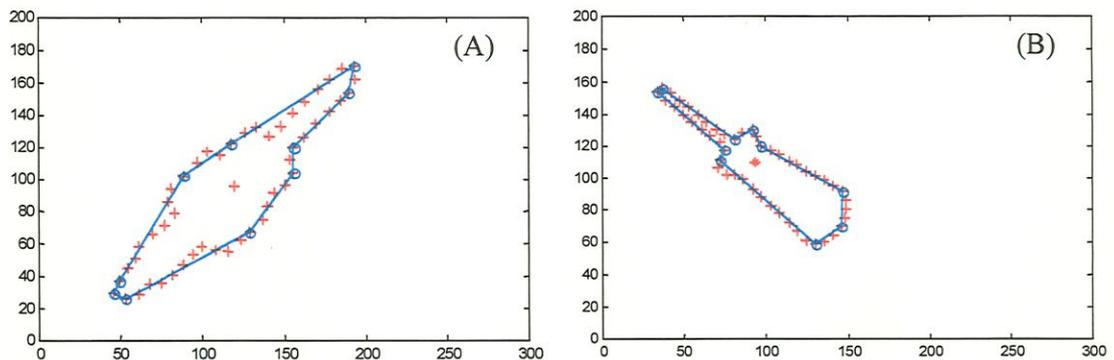


FIGURA 67 – Processo de aproximação poligonal, com a definição de um polígono de dez lados sobre os pontos do contorno da peça: (A) polígono de aproximação para o eixo escalonado; (B) polígono de aproximação para a chave de fenda.

5.4.2. Rotação e normalização da figura

O polígono de aproximação é rotacionado e normalizado, de forma a posicionar o maior lado na base da figura (FIGURAS 68A e 68C). Estes dois processamentos intermediários facilitam o aprendizado da rede RBF, uma vez que eliminam a necessidade de treinamento de formas de diferentes tamanhos e orientações. O resultado é a considerável redução do tamanho do conjunto de treinamento e, por conseqüência, do tempo deste processo.

Por fim, calcula-se o CG dos pontos do objeto rotacionado e normalizado. O objetivo é orientar as figuras sempre para um mesmo lado. Assim, para os caso em que o CG for maior que 0,5, o objeto é ainda verticalmente invertido. As FIGURAS 68B e 68D exemplificam este processo.

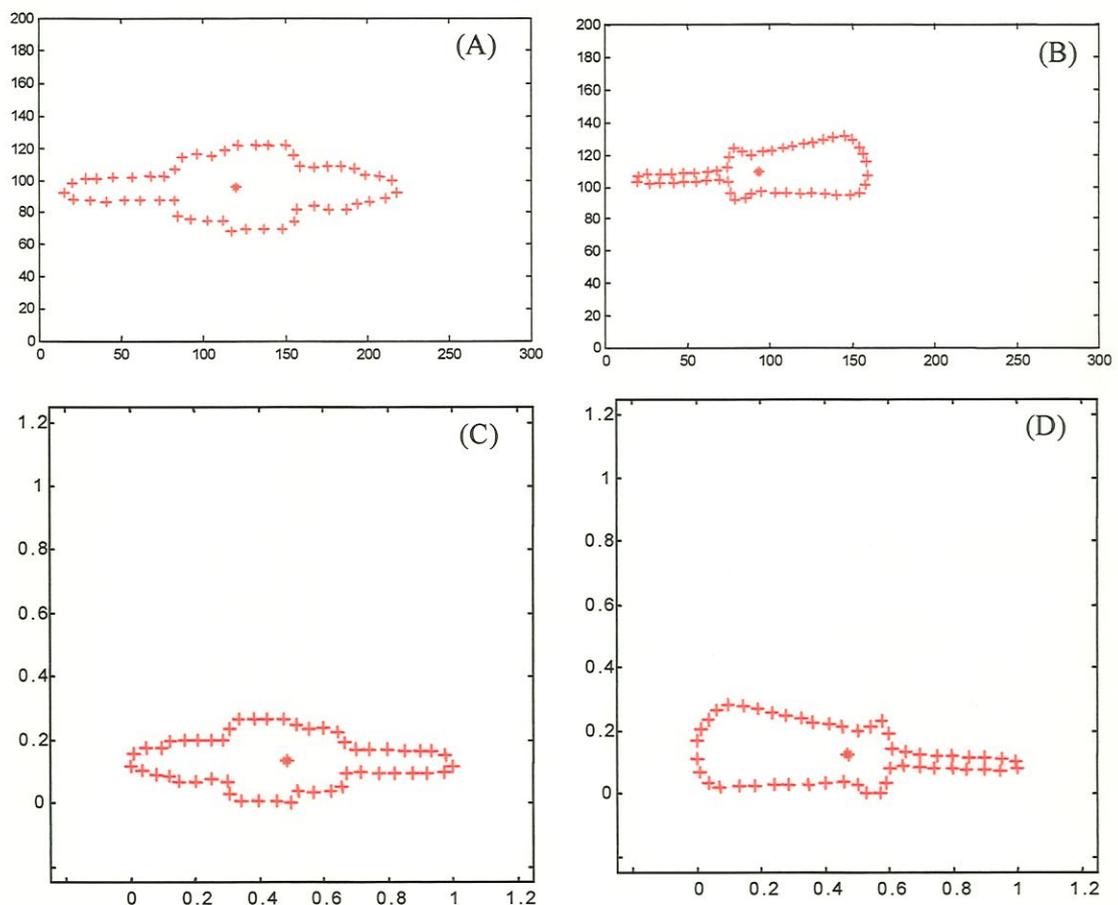


FIGURA 68 – Rotação e normalização das figuras, a fim de reduzir o conjunto de treinamento da rede RBF. A FIGURA 68A mostra o eixo rotacionado, de forma a posicionar o maior lado do polígono de aproximação na horizontal. O resultado é então normalizado (FIGURA 68C). A chave de fenda é rotacionada na FIGURA 68B e normalizada na FIGURA 68D. Nota-se que ocorre ainda uma inversão desta figura, uma vez que o CG calculado era maior que 0,5.

5.4.3. Definição dos pontos de Contato (Rede de Funções de Base Radial)

Os pontos do contorno, devidamente rotacionados e normalizados, são utilizados como dados de entrada para a rede RBF. Esta rede foi treinada pelo algoritmo *Global Ridge Regression* (SEÇÃO 4.4), apontado como o mais adequado após uma série de comparações (SEÇÃO 4.8) com os demais algoritmos de treinamento, incluindo MLP – *Back-propagation*. Todos os padrões de treinamento foram definidos como funções de base radial, assumindo raios iguais a 1,8.

As FIGURAS 68C e 68D foram apresentadas à rede. As respostas (FIGURA 69) representam as coordenadas (x,y) dos três pontos indicados para fixação da garra.

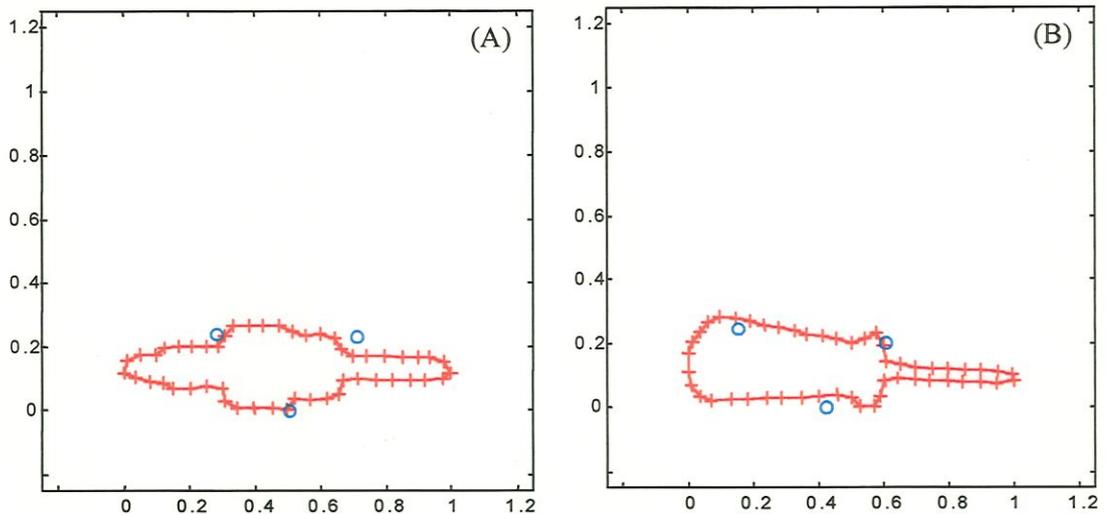


FIGURA 69 – Definição dos três pontos de fixação pela rede RBF.

Como mostra a figura acima, nada garante que os pontos de contato sugeridos pela RBF estejam exatamente sobre o contorno da peça. Para isto, foi implementado um pós-processamento dos pontos, a fim de aproximá-los dos pontos reais do contorno do objeto.

5.4.4. Aproximação do contato para pontos do contorno do objeto

A fim de se determinar os pontos efetivos de contato, deve-se relacionar os resultados da rede RBF com pontos pertencentes ao contorno do objeto. Esta relação é efetuada seguindo as linhas de ação dos dedos da garra, que unem os pontos gerados pela RBF ao ponto central da fixação. Estas linhas representam as trajetórias dos dedos durante a operação de abertura ou fechamento da garra. Uma vez estipuladas as linhas de ação, definem-se como pontos efetivos de contato aqueles que mais se aproximarem destas linhas.

Contudo, o ponto central da fixação ainda é desconhecido. Seu cálculo é essencial para equacionar as trajetórias dos dedos. Este ponto representa o lugar geométrico onde as linhas de ação se cruzam a 120° . O processo de cálculo envolve a solução de um sistema não-linear, que tem como incógnitas: os coeficientes angulares das linhas de ação (m_1 , m_2 e m_3) e as coordenadas (X_C , Y_C) do ponto central. O APÊNDICE 3 demonstra todo o processo. Os resultados da aproximação são apresentados na figura abaixo.

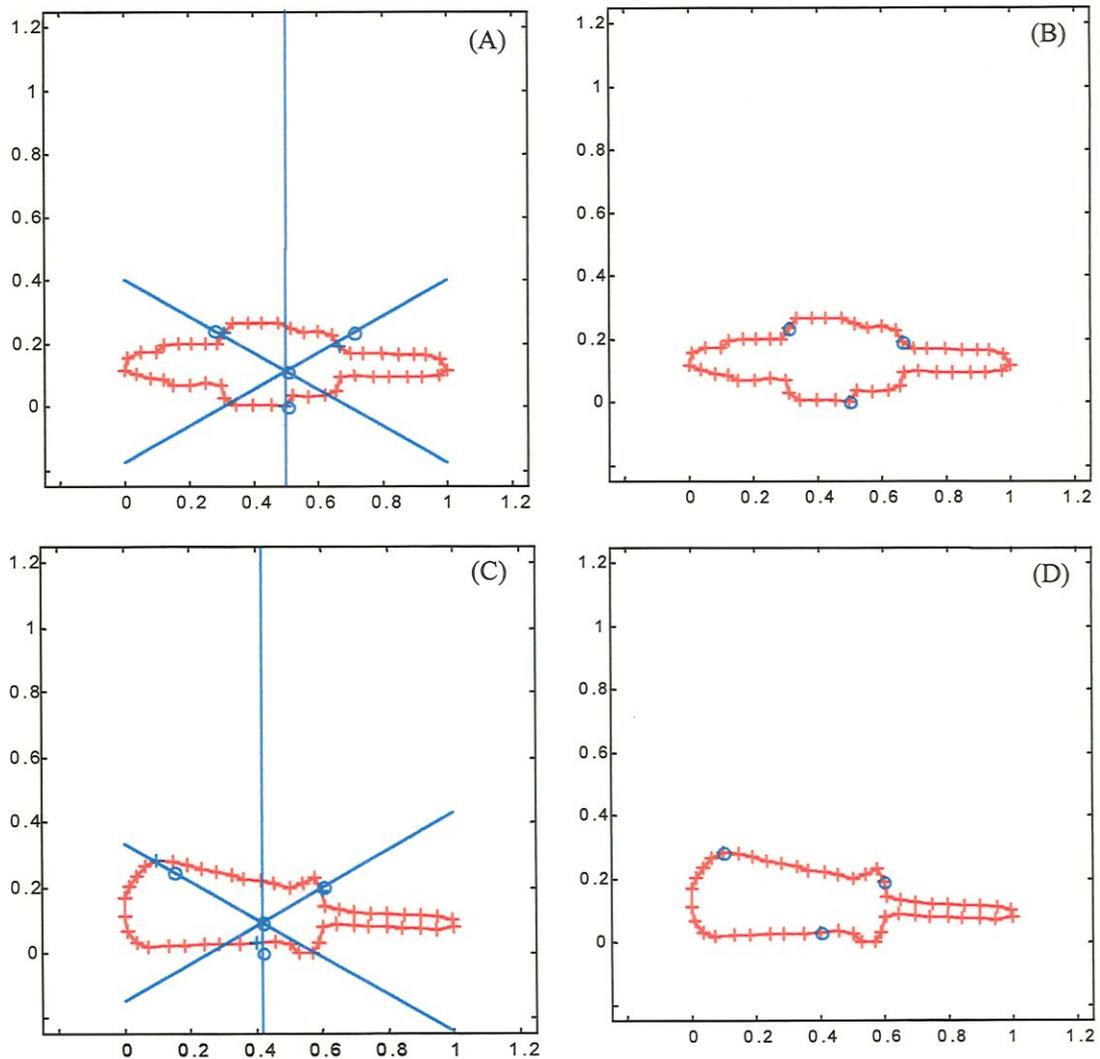


FIGURA 70 – Determinação dos pontos efetivos de contato. Inicialmente são determinados o ponto central de fixação e os coeficientes angulares das linhas de ação. Isto permite equacionar e traçar as linhas azuis (FIGURAS 69A e 69C). Os pontos efetivos de contato são definidos como aqueles que mais se aproximam destas linhas (FIGURA 69B e 69D).

5.5. Cálculo das forças estimadas de contato

Uma vez determinados os pontos efetivos de contato, resta estimar as forças a serem aplicadas por cada um dos dedos da garra. Para isto, será utilizada a teoria clássica de equilíbrio de forças. Através do modelo físico da fixação (FIGURA 71), é definido o seguinte sistema de equações lineares:

$$\begin{cases} F_1 \cdot \cos(\theta_1) + F_2 \cdot \cos(\theta_2) + F_3 \cdot \cos(\theta_3) = 0 \\ F_1 \cdot \sin(\theta_1) + F_2 \cdot \sin(\theta_2) + F_3 \cdot \sin(\theta_3) = 0 \\ F_{N1} \cdot \mu + F_{N2} \cdot \mu + F_{N3} \cdot \mu = P_C \end{cases} \quad (5.1)$$

onde F_i representa o módulo das forças aplicadas, F_{Ni} representa o módulo das componentes normais ao contato, P_C representa o peso do objeto e μ representa o coeficiente de atrito. O somatório dos momentos será automaticamente zero, uma vez que todas as forças convergem para um único ponto (ponto central da fixação).

Conforme figura abaixo, o contato dedo-peça foi modelado fisicamente como uma junta rotacional, eliminando a possibilidade de escorregamento. Esta simplificação é possível uma vez que foram considerados dedos rígidos e peça fixada ao solo.

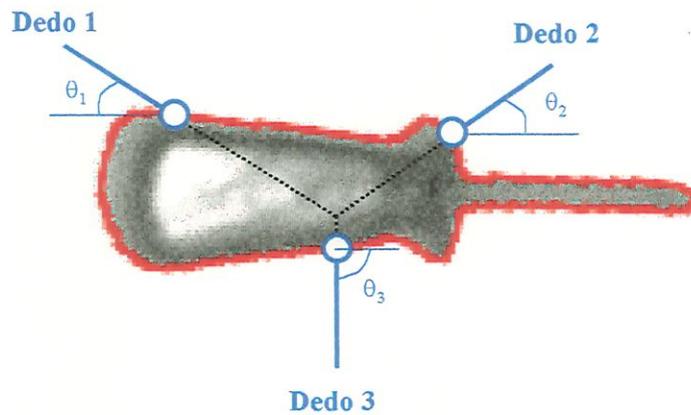


FIGURA 71- Modelo físico da fixação, no qual o contato dedo-peça foi aproximado por uma junta rotacional. São também indicados os ângulos θ_i entre o dedo e a horizontal.

Utilizando esta metodologia de cálculo, a FIGURA 72 apresenta as forças estimadas, considerando os mesmos objetos apresentados anteriormente. Nesta figura, os objetos também foram reescalados, assumindo tamanho e orientação originais.

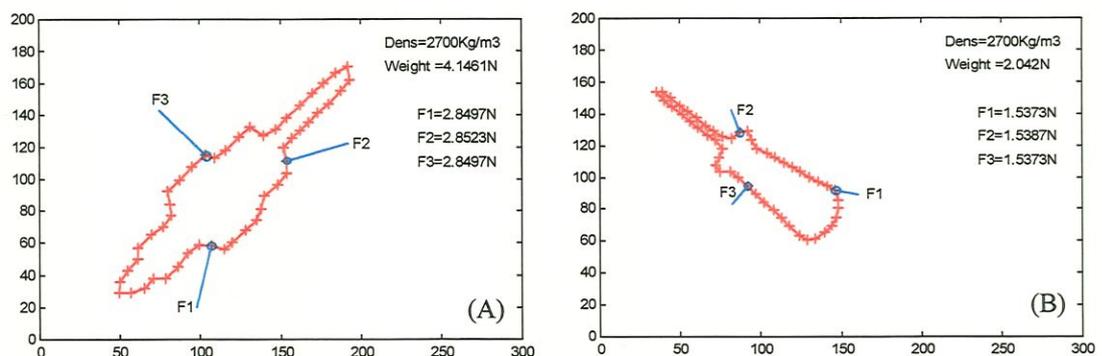
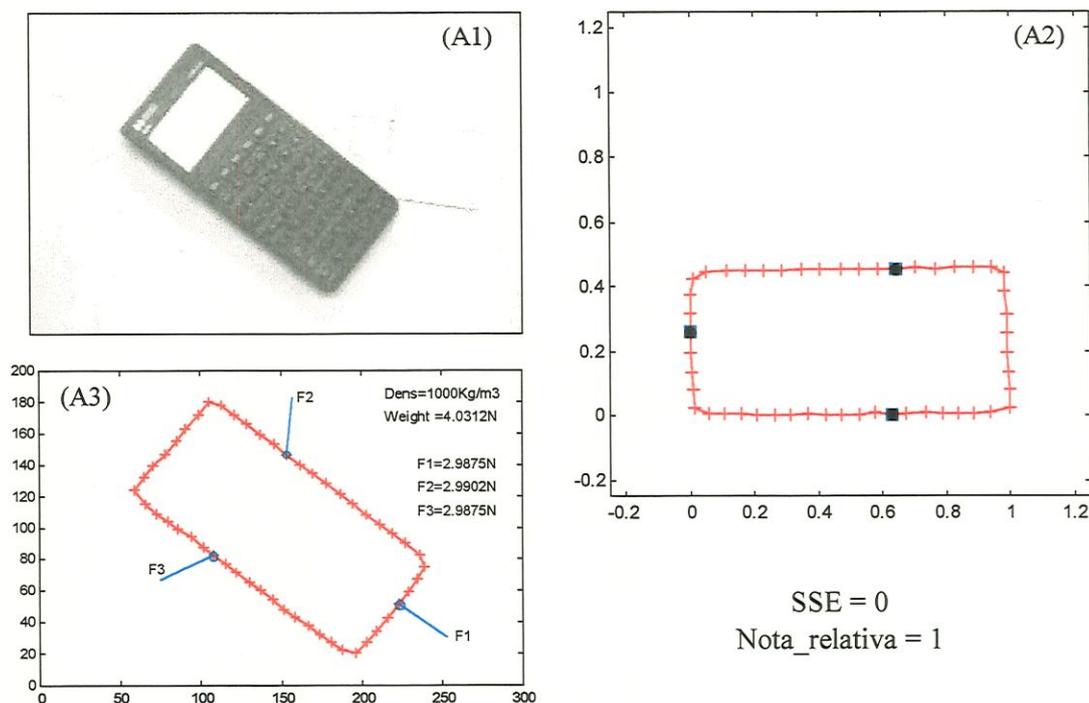


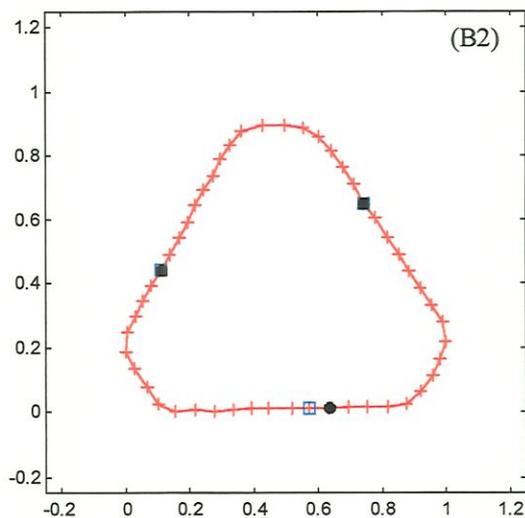
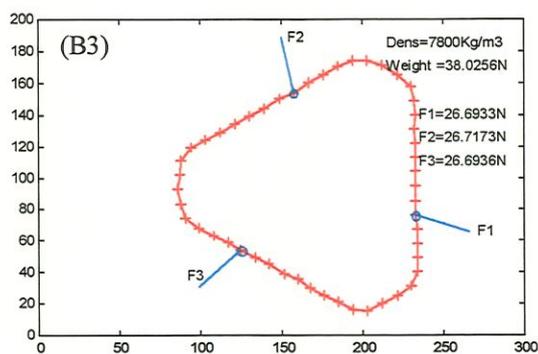
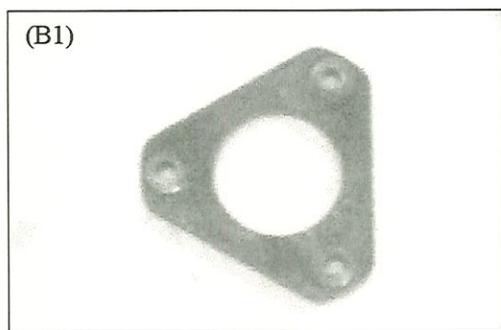
FIGURA 72 – Definição das forças estimadas de contato para cada um dos dedos da garra. Neste cálculo, assume-se que objetos são de alumínio e possuem altura constante de 0,03 m (objetos prismáticos).

Para estimar as forças de fixação acima, todos os objetos foram considerados homogêneos e constituídos de alumínio. Contudo, o material da peça também pode ser fornecido ao sistema, como um dado de entrada. Foi considerado também que os objetos apresentavam altura constante = 0,03 m (outro dado de entrada do sistema). Esta aproximação é resultado da limitação do sistema de visão, que utiliza apenas uma câmera para capturar imagens de topo do objeto. Com esta configuração, o sistema se restringe a uma análise bidimensional da cena, tratando os objetos como peças prismáticas. Variações na altura do objeto poderiam ser analisadas com a instalação de uma câmera adicional, que seria responsável pela captura de imagens num plano perpendicular ao primeiro. Desta forma, tornar-se-ia possível montar um mapa tridimensional do objeto. Esta configuração fica como proposta para trabalhos futuros.

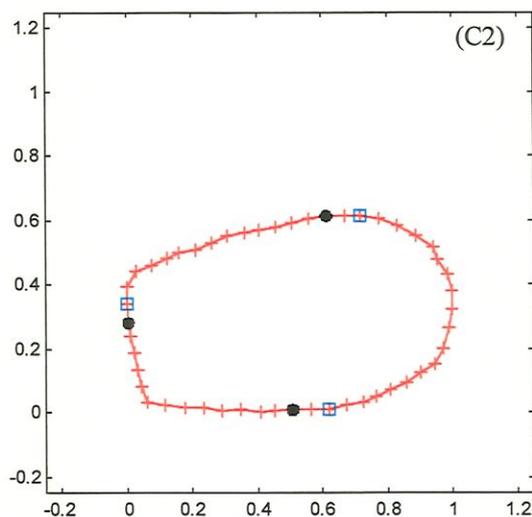
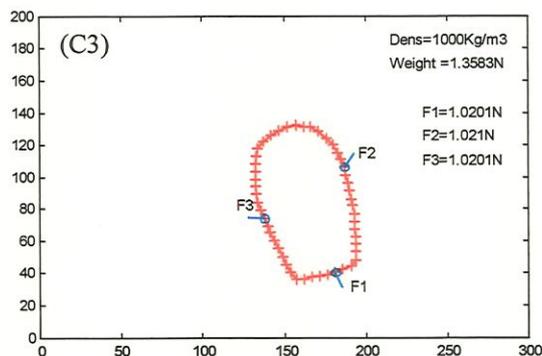
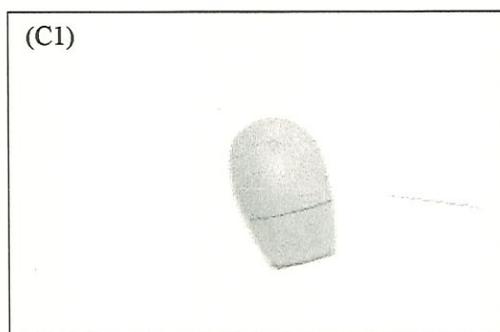
5.6. Resposta do sistema para objetos de formato desconhecido

Segue uma lista dos resultados obtidos para diferentes objeto (FIGURA 73). Todas estes objetos apresentam algumas variações com relação aos padrões treinados: variação de forma, ruído do sistema de visão e presença de sombras. Para cada caso, são apresentados: a figura original capturada pela câmera CCD, os pontos de contato calculados pela rede RBF, os pontos de contato ótimos, o valor SSE da fixação proposta, a nota relativa da fixação e as forças estimadas para cada dedo da garra.

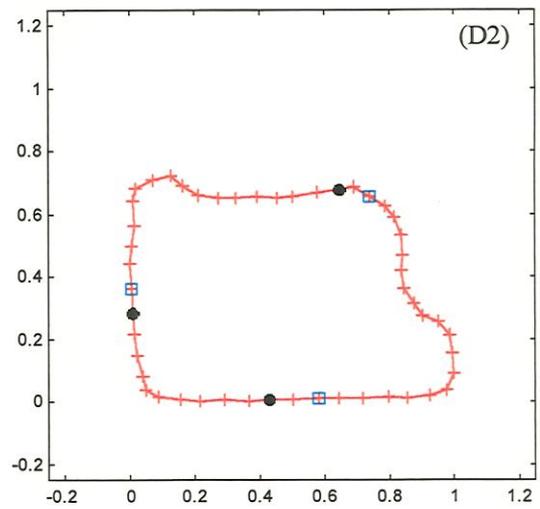
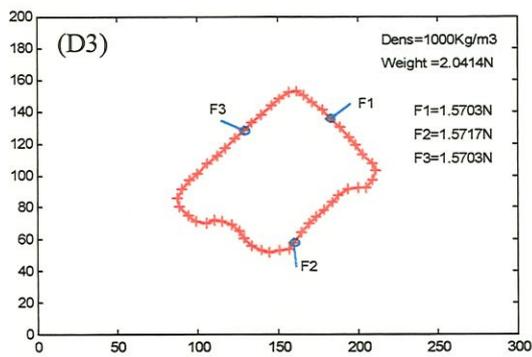
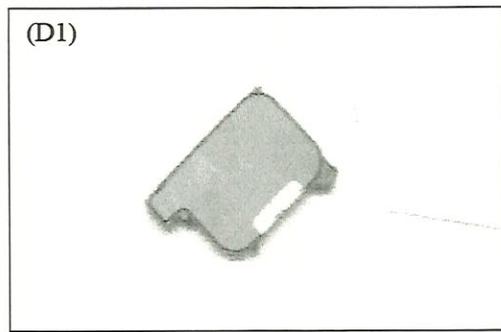




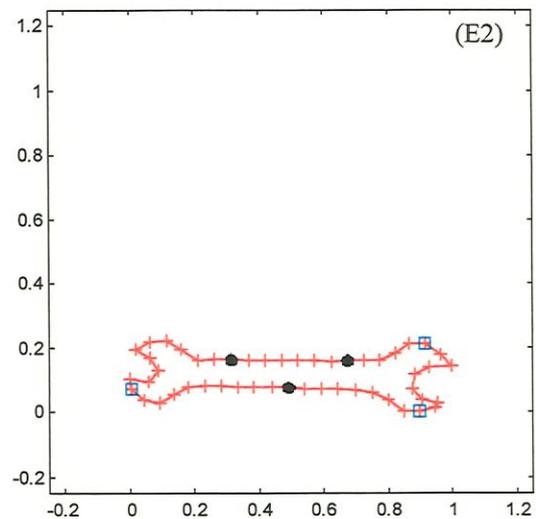
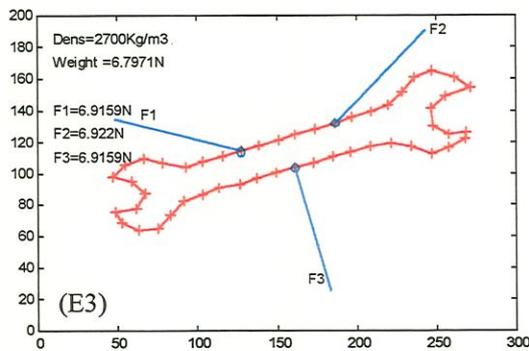
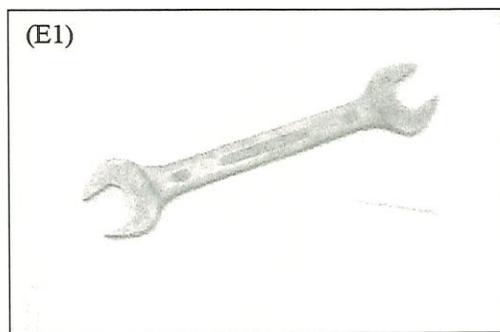
SSE = 0.0632
Nota_relativa = 1.0105



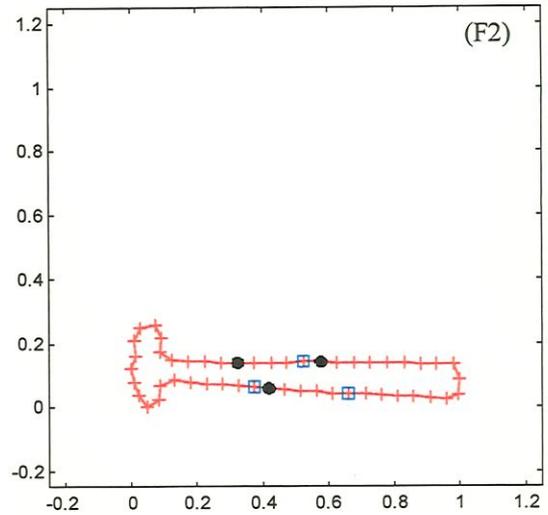
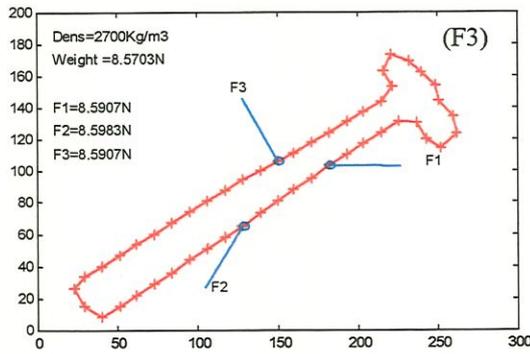
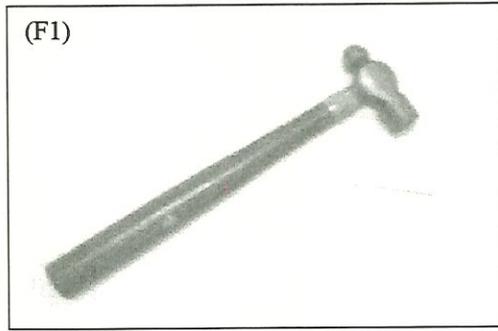
SSE = 0.1629
Nota_relativa = 1.1689



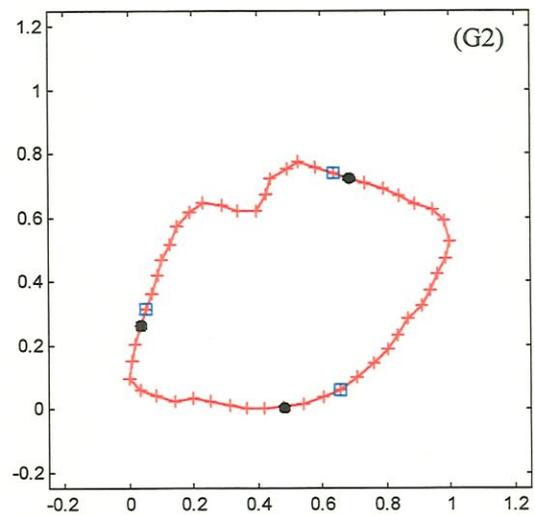
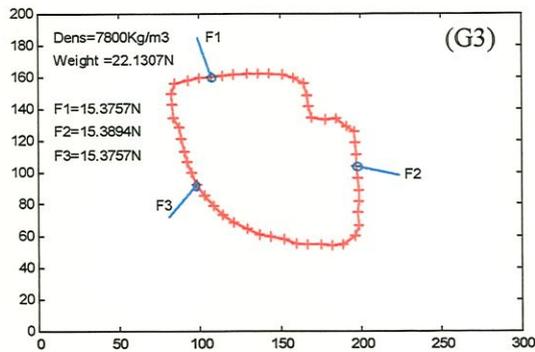
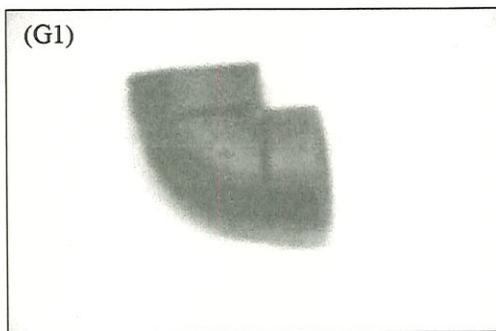
SSE = 0.1942
Nota_relativa = 1.2639



SSE = 0.5774
Nota_relativa = 5.5670



SSE = 0.2081
Nota_relativa = 1.5845



SSE = 0.1973
Nota_relativa = 1.1465

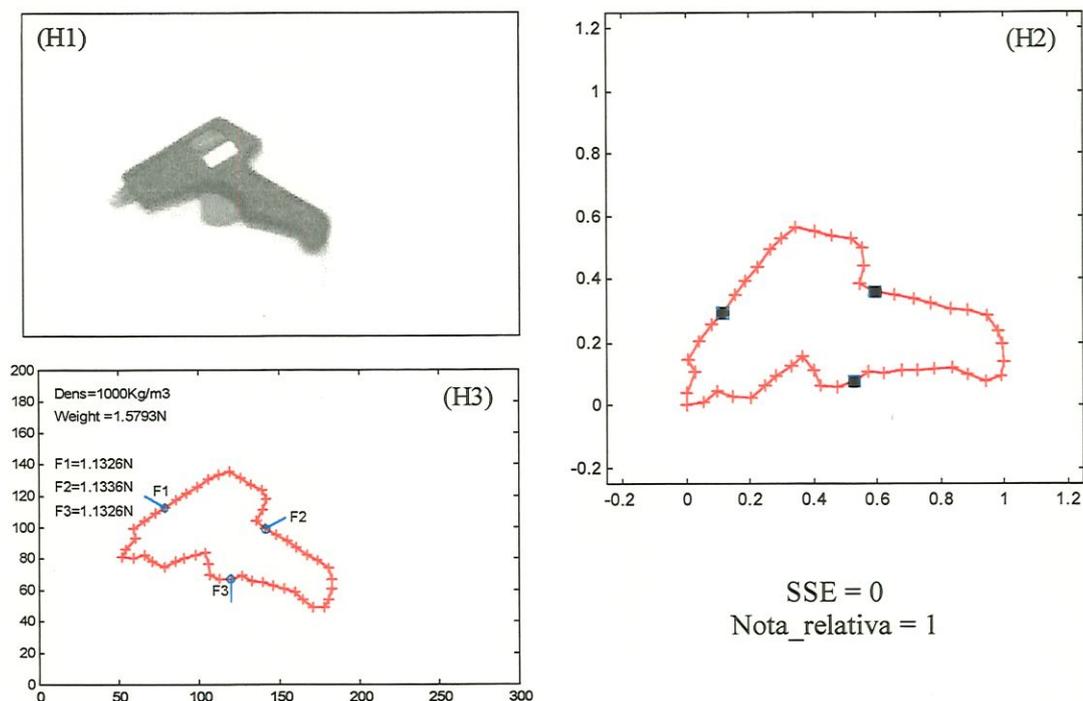


FIGURA 73 – Resposta do sistema para objetos desconhecidos. Para cada caso, são apresentados a imagem original do objeto, os pontos de contato calculados (preto), os pontos de contato ótimos (azul), as forças estimadas, o erro SSE e a nota da fixação. Os seguintes objetos foram analisados: (A) calculadora; (B) flange triangular; (C) mouse; (D) cartucho de impressão; (E) chave de boca; (F) martelo; (G) luva em L; (H) pistola de solda.

A grandeza “Nota_relativa” é utilizada para avaliar a qualidade do contato proposto pelo sistema. A metodologia utilizada para definir esta relação é a seguinte:

- após determinar os pontos de contato através do sistema neural, executa-se o algoritmo de busca pelas soluções ótimas de fixação (ANEXO 1). Este é um método de otimização global, que analisa todas as possíveis soluções, apontando uma nota absoluta para cada caso. Como resultado, é definido um conjunto de soluções otimizadas, formado por todas as configurações de fixação que apresentam nota absoluta dentro de um intervalo de 5% da menor.
- as soluções otimizadas são comparadas com a resposta definida pelo sistema neural, calculando-se o SSE para cada caso.
- o menor valor SSE é apontado como o erro da fixação.
- a solução responsável pela definição do menor SSE é apontada como solução ótima para o problema.

- a razão entre a nota da fixação definida pelo sistema neural e a nota da solução ótima resulta na nota relativa apresentada na FIGURA 73. Quanto menor a nota relativa, melhor é a fixação proposta.

Contudo, os casos (E) e (F) da figura acima merecem destaque. Analisando suas notas relativas, estas fixações seriam consideradas de baixa qualidade. Contudo, estes casos isolados merecem uma explicação. Durante o processamento neural, tais objetos (chave de boca e martelo) foram relacionados com a classe de treinamento Retângulo4 (APÊNDICE 2). Para todas as classes, com exceção da classe acima, foi utilizado o processo de otimização a fim de calcular os valores alvos de contato. Contudo, para a classe destacada, os pontos alvos foram alterados para pontos considerados mais seguros, que estão bem próximos daqueles definidos pelo sistema neural (FIGURA 73-F2 e 73-E2). A FIGURA 74 apresenta a classe alterada e os novos valores alvos.

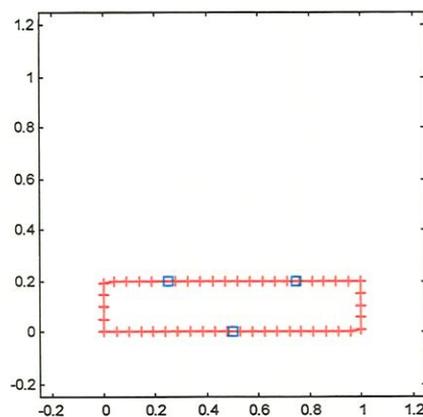


FIGURA 74 – Padrão regular da classe Retângulo4, apresentando os pontos alvos modificados (azul).

Esta alteração inviabiliza a comparação entre os pontos calculados e os ótimos, de tal forma a desconsiderar a nota apresentada anteriormente. Isto acontece porque o algoritmo de otimização determinou pontos alvos de contato do tipo lateral, enquanto a rede foi treinada com os pontos alvos da figura acima. Esta situação é melhor detalhada no APÊNDICE 2.

6. CONCLUSÕES E PROPOSTAS FUTURAS

Nesta dissertação, foi apresentado um sistema de planejamento da fixação a ser adotada por uma garra de robô na captura de objetos arbitrários. Este sistema é capaz de determinar os nf pontos de contato da garra dentre um conjunto de n candidatos pertencentes ao contorno do objeto. Contudo, este é restrito a uma análise bidimensional da cena, utilizando uma imagem de topo capturada por uma câmera CCD.

A dissertação foi iniciada com uma revisão das principais garras já desenvolvidas, seguida da análise dos principais métodos de definição dos pontos de fixação. Alguns destes métodos propõem processos analíticos para determinação do contato. Estes processos apresentam soluções bastante elegantes, porém, pecam pela lentidão de processamento, o que dificulta uma implementação *on-line*. Outros métodos baseiam-se no comportamento da mão humana, tentando prever o tipo de fixação a ser utilizada em tarefas cotidianas. Porém, estes métodos dependem da quantificação de parâmetros abstratos: precisão, segurança, agilidade e sensibilidade.

Neste contexto, abre-se uma aplicação para redes neurais, onde exemplos dos parâmetros de fixação são apresentadas às redes. SCHERRER propôs um algoritmo neural baseado na associação de duas redes: uma rede clássica de Hopfield e uma rede MLP. Apesar de lidar muito bem com objetos de formato próximo aos padrões treinados, o método de SCHERRER não apresenta boa generalização para formatos diferentes destes. Outro ponto negativo é o lento tempo de treinamento da rede MLP (aproximadamente vinte horas), o que impossibilita uma implementação em tempo real.

Face aos problemas dos métodos anteriores, foi proposto um sistema de fixação capaz de lidar com objetos de tamanhos e formatos arbitrários, além de processar todas as informações em tempo real. Para isto, foi implementado um primeiro módulo de processamento de imagem, seguido de um segundo módulo baseado em redes neurais artificiais. Para cada módulo, diversos algoritmos foram comparados a fim de apontar o mais adequado para a tarefa em questão.

O módulo de processamento de imagem é responsável pela extração do contorno do objeto a ser fixado. Foram implementados diversos operadores de detecção de borda, sendo o

algoritmo do vizinho mais próximo aquele que apresentou o melhor desempenho. Este método é bastante rápido, embora apresente uma maior sensibilidade aos efeitos de iluminação. Contudo, os demais métodos são muito mais lentos e inviabilizam uma implementação *on-line*.

O módulo neural é composto pela associação de duas redes neurais. A primeira, uma rede Competitiva de Hopfield, é responsável pela aproximação de um polígono de dez lados sobre os cinquenta pontos do contorno do objeto. Este polígono permite uma simplificação da análise, uma vez que reduz o número de pontos processáveis.

A segunda rede utiliza este polígono aproximado para gerar os três pontos efetivos de contato da garra. Para esta tarefa, foram testados seis modelos supervisionados de redes neurais: uma rede MLP (100x25x6), uma rede MLP (100x50x6) e quatro redes RBF treinadas pelos algoritmos *Forward Selection* (FS), *Global Ridge Regression* (GRR), *Local Ridge Regression* (LRR) e *Dynamic Decay Algorithm* (DDA). Dentre estas alternativas, destacou-se o conjunto RBF-GRR, que apresentou os menores valores SSE, além de um tempo de processamento adequado para aplicação *on-line*. Por outro lado, o algoritmo DDA apresentou o pior resultado e se mostrou bastante sensível a ruídos. Uma explicação pode estar no tamanho da rede montada, a menor dentre os modelos testados. As redes MLP apresentaram desempenho intermediário, contudo os tempos de treinamento condenam a utilização destas.

Testes finais foram realizados utilizando objetos de manipulação do dia-a-dia. O sistema, utilizando o algoritmo GRR, demonstrou boa capacidade de generalização, mesmo considerando padrões geométricos simples de treinamento. A fim de calcular o erro SSE para cada fixação resultante, foram definidas respostas ótimas através de um algoritmo de otimização (ANEXO 1) que busca minimizar as forças de contato. Cada fixação recebeu ainda uma nota, que busca identificar a qualidade da mesma. Todos estes testes apresentaram baixos valores de SSE e fixações de boa qualidade, demonstrando que o sistema proposto sempre determina uma fixação estável, mesmo para os casos em que os objetos são muito diferentes do padrões treinados.

A grande contribuição do sistema está na sua capacidade de processar informações variadas em um curto tempo de processamento (aproximadamente cinco segundos). Estas habilidades são resultado das características do sistema neural implementado, como generalização, tolerância a ruído e processamento paralelo. Assim, o sistema neural é mais rápido que os métodos analíticos e mantém a qualidade da resposta.

Contudo, o sistema apresenta alguns pontos complementares, a serem desenvolvidos para aperfeiçoar o seu funcionamento. Estes pontos ficam como proposta para futuros trabalhos e são listados abaixo:

- implementar o sistema de controle de força da garra, que permitirá correções nas forças de contato estimadas e evitar possíveis escorregamentos da peça capturada. Para isto, será necessário uma análise do tipo de sensor de força mais adequado à tarefa em questão e a forma como estes sensores serão montados no dedo. A estrutura do dedo foi previamente projetada para facilitar esta montagem.
- expandir o sistema para análises tridimensionais. Este procedimento irá requerer a montagem de uma câmera CCD adicional, captando imagens em um plano perpendicular ao primeiro. Assim, estas imagens poderiam ser superpostas e definiriam um mapa tridimensional do objeto.
- integrar as etapas de processamento de imagem e processamento neural. Neste trabalho, todo processamento de imagem foi implementado no ambiente visual Delphi 3.0, enquanto as redes neurais e demais processos intermediárias foram implementados no ambiente Matlab 5.2. Portanto, seria mais interessante reprogramar as redes neurais no Delphi, a fim de utilizar todas as ferramentas visuais deste programa.

ANEXO 1

ALGORITMO DE BUSCA PELOS PONTOS ÓTIMOS DE CONTATO

Este anexo apresenta um algoritmo para definir as configurações ótimas de contato. Através de uma análise de todas as possibilidades do problema, é determinado um conjunto de soluções otimizadas que garante uma fixação de boa qualidade. Para avaliar esta qualidade, calcula-se uma nota para cada caso, que é função das magnitudes das forças de contato, da diferença entre estas magnitudes e da simetria da fixação. Segue uma descrição mais detalhada deste processo.

O algoritmo inicia-se com a definição de um processamento iterativo, que é responsável para análise de todas as possíveis combinações entre os pontos P1, P2 e P3. Para evitar que uma mesma combinação seja analisada várias vezes, utilizam-se as equações apresentadas no primeiro bloco da FIGURA 75.

Para cada combinação, são calculadas as distâncias entre os pontos, o centro da fixação (APÊNDICE 2) e as respectivas distâncias pontos-centro. Estas últimas determinam o curso de fechamento ou abertura da garra. Todos estes dados serão utilizados para verificar a viabilidade dos pontos de contato sugeridos, que devem pertencer à área de trabalho da garra. Em caso afirmativo, o processo continua. Caso contrário, novos pontos de contato são sugeridos.

Na continuação do algoritmo, são calculados os ângulos formados por cada um dos pontos P_i e os seus pontos vizinhos P_{i-1} e P_{i+1} . Estes ângulos serão utilizados para evitar a definição de pontos de contato sobre um vértice do contorno do objeto. Estes locais não representam regiões seguras para fixação dos dedos.

Caso os pontos sugeridos não sejam vértices, são calculadas as forças de contato para cada um dos dedos. Utiliza-se um sistema de equações lineares, onde as linhas representam o equilíbrio das forças nas direções coordenadas. Contudo, serão considerados apenas os casos em que todas as forças são de compressão, ou seja, casos em que todas as forças são positivas.

Uma vez aprovados por todas as condições anteriores, os pontos P_1 , P_2 e P_3 são considerados válidos. Neste caso, calcula-se uma nota representando a qualidade da fixação. A melhor fixação será aquela que apresentar a menor nota. Para computar esta nota, três aspectos importantes são ponderados:

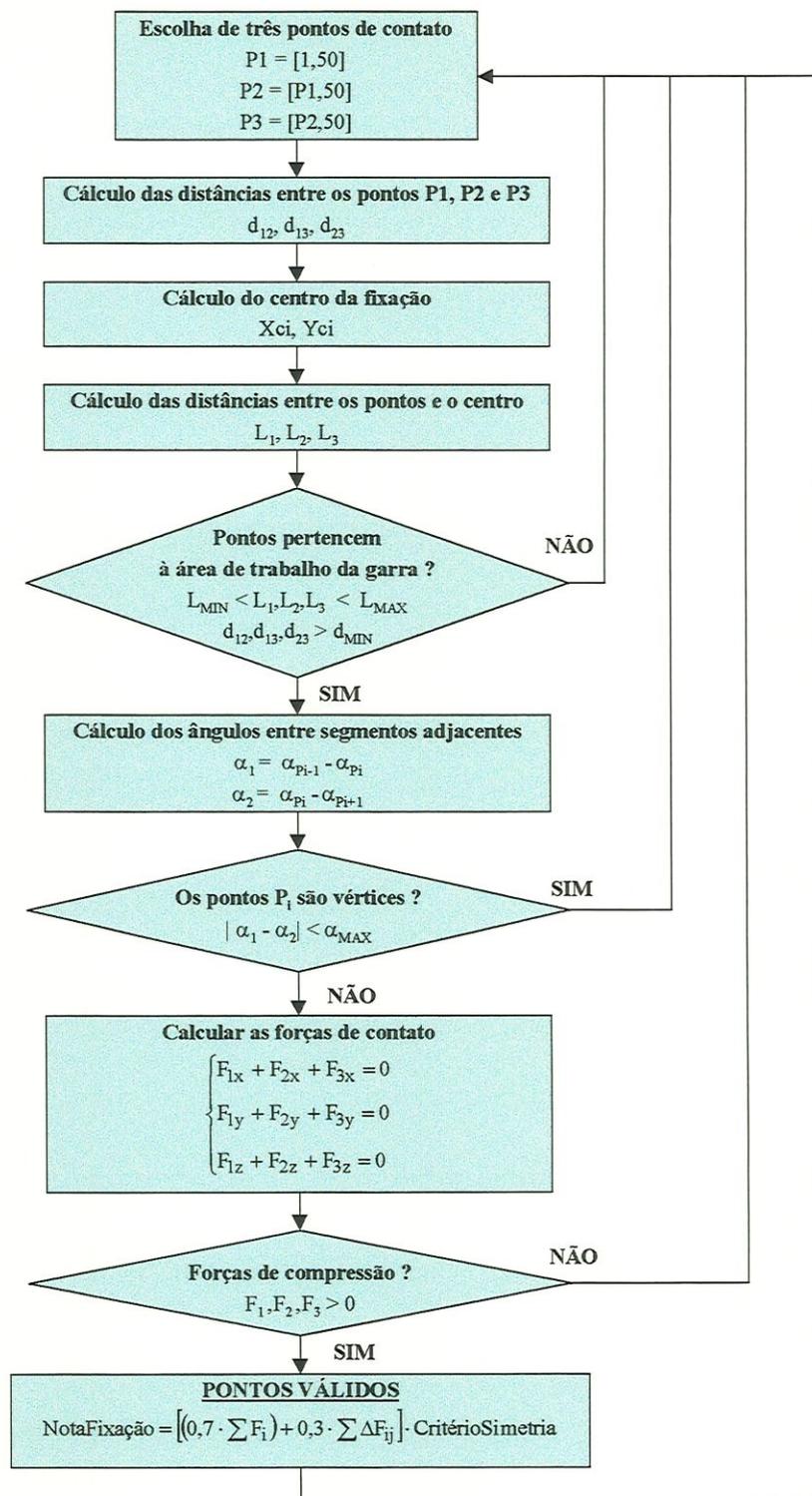


FIGURA 75 – Algoritmo de busca pelos pontos ótimos de contato. Todas as possíveis combinações de P_1 , P_2 e P_3 são analisadas. Assumindo algumas condições para a viabilidade do contato, o algoritmo seleciona as soluções válidas e aponta uma nota para a fixação.

1. somatório das forças de contato, visando minimizar a potência requerida nos atuadores, bem como reduzir as deformações da peça e do dedo.
2. somatório das diferenças entre as forças de contato, visando balancear a ação dos dedos sobre a peça.
3. critério de simetria da fixação, calculado a partir da distância entre o centro geométrico da peça e o centro da fixação. Quanto mais simétrica a fixação, menores serão os momentos gerados por eventuais escorregamentos ou deformações dos dedos. A equação de simetria é apresentada abaixo.

$$\text{CritérioSimetria} = 1 + \text{abs}(P_{(X_{cg}, Y_{cg})} - P_{(X_{ci}, Y_{ci})}) \quad (\text{AN.1})$$

Por fim, o algoritmo prossegue com a análise de novas combinações de pontos de contato. Após analisar todas as possíveis soluções do problema, o processo resulta numa matriz contendo todas as fixações aprovadas e suas respectivas notas. Analisando estes resultados, percebe-se que existem diversas soluções com notas bastante próximas do mínimo, de forma que torna-se difícil apontar a melhor opção e descartar as demais. Assim, foi proposta a definição de um conjunto de soluções otimizadas, formado pela melhor solução e todas as demais que superam a nota mínimo em 5%.

REFERÊNCIAS BIBLIOGRÁFICAS

- AMADEUS PROJECT HOMEPAGE. <http://www.cee.hw.ac.uk/oceans/projects/amadeus/amadeus.htm> (07 dezembro 1998).
- ARAÚJO, A. F.; TANAKA, J. S. (1995). Variação de um modelo de rede neural competitiva de Hopfield para aproximação poligonal. In: SIMPÓSIO BRASILEIRO DE REDES NEURAIS, 2., São Carlos, 1995. Anais. p.193-198.
- BEALE, R; JACKSON, T. (1990) Neural Computing: An Introduction. Bristol, Adam Hilger, Reino Unido.
- BENTES, P.C.L. (1997) *Deteção de formas circulares utilizando a transformada de rough, desenvolvida no ambiente Windows*. São Carlos. 183p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.
- BERTHOLD, M. R.; DIAMOND, J. (1995) Boosting Performance of RBF Networks with Dynamic Decay Adjustment. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. Anais. v. 7, p.352-359.
- BISHOP, C.M. (1995) Neural networks for pattern recognition, Oxford University Press Inc., New York, EUA.
- CHUNG, P.C. et al. (1994). Polygonal approximation using a competitive neural network. *Pattern Recognition*, v.27, n.11, p. 1505-1512.
- COELHO JR, J. A.; GRUPEN, R. A. (1993) Sensor-Based Contact Geometry Optimization for Multifingered Robot Hands. In: INTERNATIONAL WORKSHOP ON MECHATRONICAL COMPUTER SYSTEMS FOR PERCEPTION AND ACTION, Halmstad, Suécia. Anais. p.147-156.
- COELHO JR, J. A.; GRUPEN, R. A. (1994) Control Pre-Imaging for Multifingered Grasp Synthesis. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, San Diego, EUA. Anais. p. 3117-3122.
- COELHO JR, J. A.; GRUPEN, R. A. (1997) A Control Basis for Learning Multifingered Grasps. *Journal of Robotics Systems*, v.14, n.7, p.545-557.
- CORNELL, G. (1995) Delphi: segredos e soluções. São Paulo, MAKRON Books.

- CREATIVE LABS. <http://www.creaf.com>. (07 dezembro 1998).
- CRISMAN, J. D. et. al. (1996). Grasp: a flexible, easily controllable robotic hand. *IEEE Robotic & Automation Magazine*, New York, v.3, n.2, p.32-38.
- CUTKOSKY, M. R. (1989) On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks. *IEEE Transactions on Robotics and Automation*, v.5, n.3, p.269-279.
- DLR ARTICULATED HAND. http://www.robotic.dlr.de/MECHATRONICS/DFG/ver_apr97.html (07 de dezembro 1998)
- DOWLING, J. E. (1992) *Neurons and network, an introduction to neuroscience*. Cambridge, Harvard University Press, EUA.
- ETHZ – INSTITUTE OF ROBOTICS. <http://www.ifr.mavt.ethz.ch>. (07 dezembro 1998).
- FERRARI, C.; CANNY, J.; (1992) Planning optimal Grasps. In IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, Nice, França. Anais. p.2290-2295.
- GONZAGA, A.; FRANÇA, C.A. (1996) *Algoritmos de detecção de bordas em imagens digitais*. <http://sel.eesc.sc.usp.br/sel/lavi/tec-report/TR-SEL-LAVI-00296/tr-sel-lavi-00296.html>. (15 Abril 1998).
- GRASPAR HAND. <http://rvsl.ece.neu.edu/projects/graspar/index.html> (07 dezembro 1998)
- HASSOUN, M.H. (1995). *Fundamentals of artificial neural networks*. Cambridge, MIT Press.
- HAYKIN, S. (1994). *Neural Networks, a comprehensive foundation*. New York, IEEE PRESS.
- HEBB, D. O. (1949) *The organization of behavior: A neuropsychological theory*. New York, John Wiley, EUA.
- HERTZ, J; KROGH, A; PALMER, R. G. (1991) *Introduction to the theory of neural computation*. Redwood City, Addison-Wesley Publishing Company, EUA.
- HIRZINGER G., et. al. (1993) Rotex, the first space robot technology experiment. In: INTERN. SYMPOSIUM ON ROBOTICS RESEARCH, 6., Hidden-Valley, PA, EUA.
- HOPFIELD, J. J., & TANK, D. W. (1985) Neural computation of decision in optimization problems. *Biological Cybernetics*, v.52, p.141-152.

- HP ENCODER HOME PAGE. <http://www.hp.com/HP-COMP/motion/hed1550x.html> (07 dezembro 1998).
- IBERALL, T. (1997) Human Prehension and Dexterous Robot Hands. *The International Journal of Robotics Research*, v.16, n.3, p.285-299.
- JACOBSEN, S.C. et. al. (1987). Design of the Utah/MIT Dexterous Hand. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORK, 1., San Diego, Anais. p.57-63.
- JAIN, R. et al. (1995) Machine Vision. New York, McGraw-Hill, Inc., EUA.
- JOHNSON, R.P. (1990) Contrast Based Edge Detection. *Pattern Recognition*, v.23, n.3, p.311-318.
- JONGKIND, W. (1993) Control and fault tolerance of a dextrous gripper in a hazardous environment operating. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, Le Touquet, France.
- KAMON, I.; FLASH, T.; EDELMAN, S. (1994) *Learning to Grasp Using Visual Information*. <http://www.wisdom.weizmann.ac.il/Papers/trs/CS94-04/abstract.html> (07 de dezembro 1998)
- MACHADO, A. B. M. (1974) Neuro-anatomia funcional. Rio de Janeiro, Editora Atheneu, 2ª Edição, Brasil.
- MAXON MOTOR. <http://www.mpm.maxonmotor.com/default3.cfm> (07 dezembro 1998).
- MENZEL, R. (1994) *Konstruktion und regelung einer hydraulischen hand*, PhD Thesis No 451, VDI - TUM, Munique, Alemanha.
- MONTANA, D.J. (1988). The kinematics of contact and grasp. *The International Journal of Robotics Research*, v.7, n.3, p.203-217.
- MOTOROLA MVME HOME PAGE. <http://www.mcg.mot.com/WebOS/omf/GSS/MCG/products> (07 dezembro 1998).
- MOUSSA, M. A.; KAMEL, M. S. (1995) A connectionist Model of Human Grasps and its Application to Robot Grasping. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORK, Perth, Austrália. Anais. p.2555-2559.

- NGUYEN, V.D. (1987). The synthesis of stable grasps in the plane. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORK, 1., San Diego. Anais. p.151-159.
- NGUYEN, V.D. (1988). Constructing force-closure grasp. *The International Journal of Robotics Research*, v.7, n.3, p.157-171.
- O'BRIEN, D. J.; LANE, D. M. (1998) *Force and Slip Sensing for a Dextrous Underwater Gripper*. <http://www.cee.hw.ac.uk/oceans> (07 dezembro 1998).
- ORR, M. J. (1996). Introduction to radial basis function networks. Centre for Cognitive Science, University of Edinburgh, Buccleuch Place. (Report EH8 9LW).
- ORR, M. J. (1998) Optimising the widths of Radial Basis Functions. In: SIMPÓSIO BRASILEIRO DE REDES NEURAIAS, 5., Belo Horizonte, 1998. Anais. v.1, p.26-29.
- OVERDIJK, R. (1997) *Tactile sensors give robot sense of touch – Gripper feels slip*. Delft University of Technology. (Delft Outlook 97.2).
- PFEIFFER, F. (1996) Grasping with Hydraulic Fingers – An Example of Mechatronics. IEEE/ASME Transaction on Mechatronics, v.1, n.2, p.158-167.
- PODHORODESKI, R.P. et. al. (1989). The feasibility and optimization of grasp contact forces for robotic applications. In: CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA, 10., Rio de Janeiro, 1987. Anais. Rio de Janeiro, p.43-50.
- PONCE, J.; FAVERJON, B; (1995) On Computing Three-Finger Force-Closure Grasps of Polygonal Objects. *IEEE Transaction on Robotics and Automation*, v.11, n.6, p.868-881.
- PROTZEL, P.W. et. al. (1993) Performance and fault-tolerance of neural networks for optimization., *IEEE Transaction on Neural Network*, Jul., v.4, n.4, p.600-614.
- RAWLINGS, J. O. (1988). Applied regression analysis, Pacific Grove, Wadsworth & Brooks/Cole, CA.
- REISER, M.; WIRTH, N. (1992) Programming in Oberon: Step Beyond Pascal and Modula. New York, ACM Press.
- RUMELHART, D. E; McCLELLAND, J. L. (1986) Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations. Cambridge, MIT Press, EUA.

- SALISBURY, J. K. (1982) *Kinematic and force analysis of articulated hands*. PhD Thesis, Stanford University, Stanford, CA, May.
- SCHALKOFF, R.J. (1989) *Digital Image Processing and Computer Vision*. New York, John Wiley & Sons, Inc, EUA.
- SCHERRER J. K., (1993) *Sensorhand für intelligentes Greifen mit Robotern*. PhD Thesis No 10249, ETH Zürich, Suíça.
- SCHUNK PRECISION WORKHOLDING SYSTEMS. <http://www.schunk.de/en/index.html> (07 dezembro 1998)
- SOUTHAMPTON ARTIFICIAL HAND. <http://www.soton.ac.uk/~eewww/Arthand.htm> (07 dezembro 1998).
- STANFORD/JPL HAND. *Smart Hands Homepage*. <http://tommy.jsc.nasa.gov/~li/SmartHand.html>. (07 dezembro 1998).
- TELEMAN PROJECT. <http://www.teleman.org>. (07 dezembro 1998).
- TOWNSEND, W. (1999) Robots To Give a Big Hand To Humans. *Mass High Tech*, Feb.
- TÜEFE, T. *TsTECH Main Page*. <http://migr.et.fh-stralsund.de/~tuefe> (03 Maio 1998).
- TUNG, C. P.; KAK, A. C.; (1996) Fast Construction of Force-Closure Grasps, *IEEE Transaction on Robotics and Automation*, v.12, n.4, p.615-626.
- UTAH/MIT DEXTROUS HAND. <http://www-robotics.cs.umass.edu/p50/utah-mit-hand.html> (07 dezembro 1998).
- WOELFL, K.; PFEIFFER, F. (1994) Grasp Strategies for a Dextrous Robotic Hand. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, Munique, Alemanha, Anais. p.366-373.
- WOELFL, K. *Planung von Manipulationsvorgängen einer Roboterhand*. PhD Thesis No 455, VDI-TUM, Munique.
- XOBERON HOME PAGE. <http://enterprise.ethz.ch/research/xoberon>. (07 dezembro 1998).

APÊNDICE 1

INTRODUÇÃO ÀS REDES NEURAIS ARTIFICIAIS

A1. INTRODUÇÃO ÀS REDES NEURAIS ARTIFICIAIS

A1.1. A estrutura do cérebro humano

O cérebro é o órgão mais complexo do corpo humano. A grande maioria das questões ligadas ao seu funcionamento ainda permanece oculta. Porém, anos de pesquisa possibilitaram o levantamento de algumas características fundamentais. Segundo DOWLING (1992), o cérebro humano contém de 10^{11} a 10^{13} unidades básicas chamadas neurônios. Cada neurônio é uma unidade analógica de processamento lógico. Alguns detalhes básicos deste funcionamento já foram compreendidos. Uma representação genérica destes detalhes é mostrada abaixo:

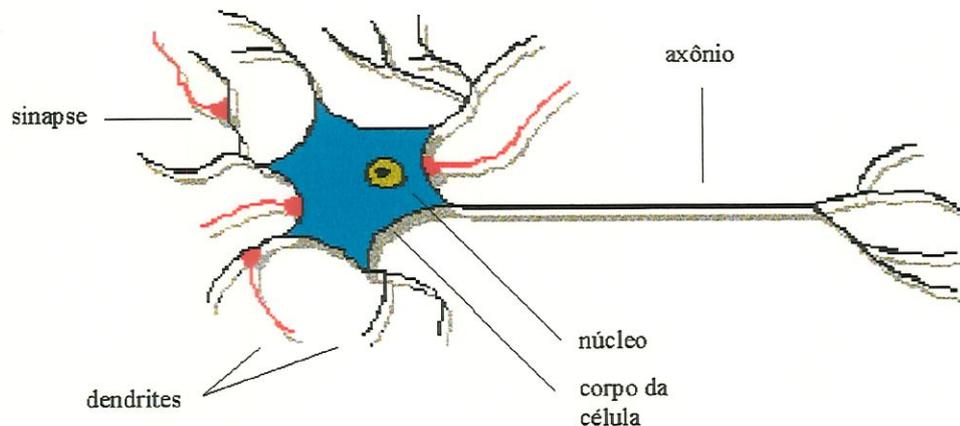


FIGURA A1.1 – Representação genérica de um neurônio biológico, mostrando as três partes principais: dendritos, corpo ou soma da célula e axônio.

Milhares de fibras nervosas chamadas dendritos são conectadas ao corpo ou soma do neurônio. Elas funcionam como conexões de entrada do neurônio, cada uma com menos de um micron de diâmetro. O soma é capaz de executar operações complexas sobre os sinais de entrada, porém, estas operações podem ser aproximadas por um somatório simples destes.

A extensão do soma é uma longa e única fibra nervosa chamada axônio. Esta estrutura recebe e analisa o valor do somatório anterior. Se este somatório for superior a um limite

mínimo, o axônio produz uma voltagem de saída (10^{-3} V), também denominada potencial de ação, que é conduzida até sua extremidade.

A extremidade do axônio se ramifica em diversas terminações nervosas (telodendros). Estas terminações se conectam aos dendritos ou ao soma das unidades vizinhas através de complexas estruturas chamadas sinapses (FIGURAS A1.2 e A1.3). Segundo MACHADO (1974), não ocorre o contato direto entre telodendro e dendrito, mas sim uma conexão química temporária (processo de transmissão sináptica), que provoca a excitação, inibição ou modulação da unidade seguinte.

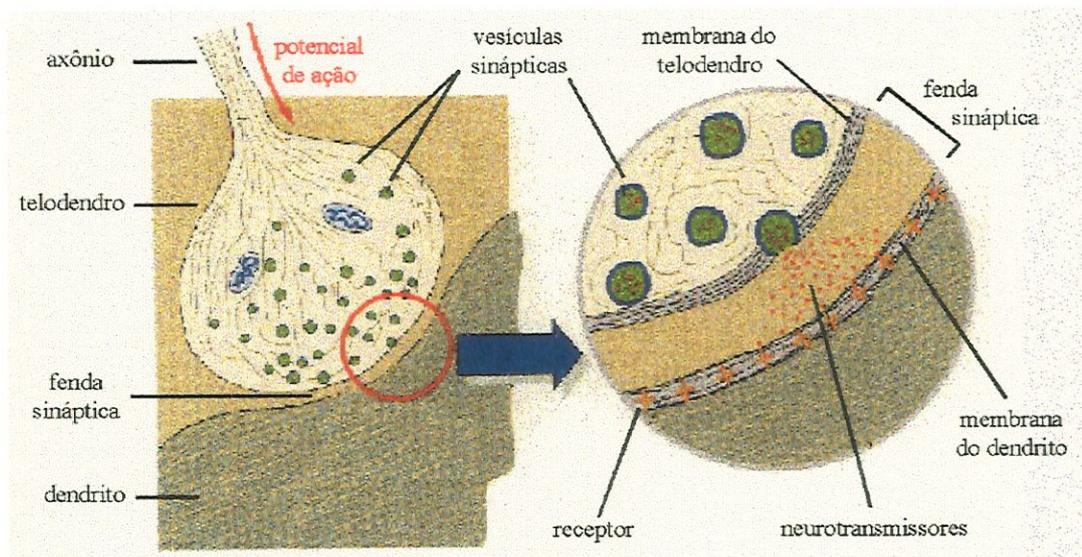


FIGURA A1.2 – Processo de transmissão sináptica. Quando o potencial de ação atinge a membrana da telodendro, são liberados neurotransmissores. Estas substâncias atravessam a fenda sináptica e se conectam a receptores específicos, localizados na membrana do dendrito associado.

No telodendro, existem vesículas com neurotransmissores responsáveis pela condução do potencial de ação entre os neurônios. Quando o potencial de ação atinge a membrana do telodendro, abrem-se canais provocados pela fusão da membrana do mesmo com as membranas das vesículas. Esta fusão permite a saída dos neurotransmissores, que se difundem pela fenda sináptica e se ligam aos receptores da membrana do dendrito. Esta ligação desestabiliza a membrana, permite a entrada de íons e possibilita a condução do potencial de ação para outro neurônio.

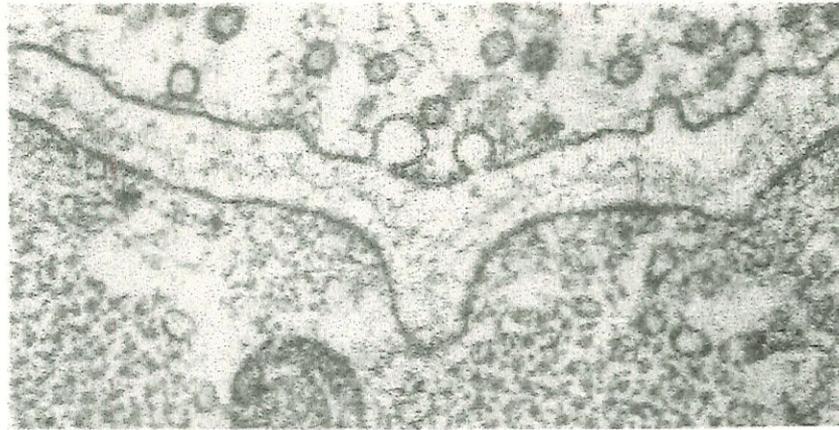


FIGURA A1.3 – Foto do processo real de transmissão sináptica. Torna-se visível a ligação entre as vesículas e a membrana do telodendro.

A1.2. Aprendizado em sistemas biológicos

A principal característica dos sistemas neurais biológicos é a sua capacidade de aprendizado, abstraindo informações fornecidas pelo meio ambiente. O processo de aprendizado biológico é ainda pouco conhecido, porém alguns postulados básicos já foram levantados. HEBB (1949) propôs a mais antiga e ainda famosa regra de aprendizado:

“Quando o axônio de uma célula A está próximo o suficiente para excitar uma célula B, e repetidamente toma parte na emissão do potencial de ação desta célula, algum processo de crescimento ou mudança metabólica acontece em uma ou ambas as células de forma que a eficiência de A para ativar a célula B é aumentada.”

Mesmo sem conhecer o processo, HEBB já definia algumas mudanças metabólicas como base do aprendizado. Hoje, estas mudanças são basicamente compreendidas. Elas provocam o fortalecimento de conexões específicas entre neurônios, o que leva ao aparecimento de novos circuitos neurais. Para o fortalecimento de uma conexão sináptica, o mecanismo de liberação de neurotransmissores é estimulado, resultando numa maior transmissão do potencial de ação. Assim, a base do aprendizado é a criação de conexões e caminhos mais favoráveis.

A1.3. O neurônio artificial

O modelo de um neurônio artificial (FIGURA A1.4) foi inicialmente proposto por McCulloch e Pitts (1943) e posteriormente aprimorado por Rosenblatt (1958), que o

denominou perceptron. Este modelo representa uma aproximação extremamente simplificada do ponto de vista neurofisiológico, porém conserva algumas características básicas do neurônio biológico e viabiliza a sua implementação computacional.

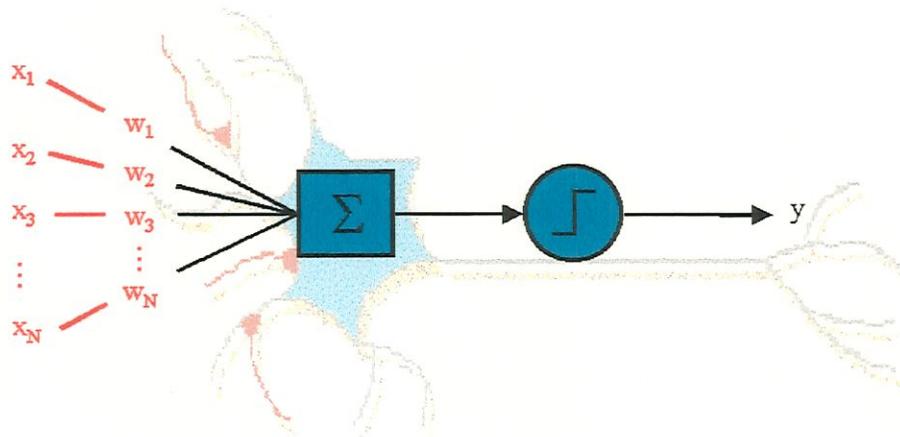


FIGURA A1.4 – Modelo matemático básico do perceptron. Ocorre o somatório das entradas x_i , ponderado pelos pesos w_i . O resultado é aplicado a uma função degrau.

O perceptron realiza o somatório ponderado dos sinais binários de entrada ($x_i = [0,1]$), provindos das sinapses de outros neurônios. O peso w_i representa a força da sinapse i , podendo ser positivo ou negativo correspondendo a uma conexão excitatória ou inibitória. O valor $w_i = 0$ indica que não há conexão entre os neurônios. O valor do somatório é aplicado a uma função degrau f_h e resulta numa saída binária ($y = [0,1]$), correspondente a um somatório inferior ou superior ao *threshold* T . Sendo N o número de entradas, a representação matemática de um perceptron é expressa por:

$$y = f_h \left[\sum_{i=1}^N w_i x_i - T \right] \quad (\text{A1.1})$$

sendo

$$f_h = \begin{cases} 1 & \text{se } \sum_{i=1}^N w_i x_i \geq T \\ 0 & \text{se } \sum_{i=1}^N w_i x_i < T \end{cases} \quad (\text{A1.2})$$

A1.4. Aprendizado Artificial

A definição do processo artificial parte, mais uma vez, da observação do mecanismo biológico de treinamento, no qual o conhecimento é absorvido por meio de alterações no

processo de transmissão sináptica. A Lei de HEBB demonstra uma tendência dos sistemas biológicos de aprender um comportamento recompensado e de esquecer um comportamento penalizado.

O processo de aprendizado artificial também busca alterar a eficiência das conexões neuronais, através do ajuste ou treinamento dos pesos w_i . Assim, o treinamento é um processo iterativo de ajuste dos pesos de forma a aproximar a saída do neurônio de um valor de resposta desejada. O paradigma de treinamento pode ser sintetizado nos seguintes passos:

1. Considerar valores iniciais dos pesos w_i e do *threshold* T ;
2. Apresentar um entrada $x = [x_1, \dots, x_N]$ correspondente a um padrão p ;
3. Calcular a resposta y correspondente ao padrão entrado;
4. Alterar pesos se a resposta não for igual a um valor desejado \hat{Y} .

Os algoritmos de treinamento aplicados ao perceptron foram evoluindo até que, em 1960, Widrow e Hoff propuseram o modelo *Adaptive Linear Neurons* – ADALINE (BEALE & JACKSON, 1990). Este algoritmo propõe um ajuste dos pesos de forma proporcional à diferença entre a saída calculada e a saída desejada. Esta diferença representa o erro ε da unidade, descrito pela seguinte equação:

$$\varepsilon = (y - \hat{Y})^2 \quad (\text{A1.3})$$

Sendo um processo iterativo, o ajuste dos pesos é realizado em função da variação deste erro:

$$\frac{\partial \varepsilon}{\partial w_i} = \varepsilon \cdot \frac{\partial}{\partial w_i} (y - \hat{Y}) = \varepsilon \cdot x_i \quad (\text{A1.4})$$

Calculada a variação do erro, define-se a regra de Hidrow-Hoff ou regra delta ou regra *least-mean-square* (LMS):

$$w_i(t+1) = w_i(t) + \eta \cdot \frac{\partial \varepsilon(t)}{\partial w_i(t)} \quad (\text{A1.5})$$

$$w_i(t+1) = w_i(t) + \eta \cdot \varepsilon(t) \cdot x_i(t) \quad (\text{A1.6})$$

sendo η ($0 \leq \eta \leq 1$) a fator de aprendizado responsável pelo controle da taxa de ajuste.

A presença de uma saída desejada \hat{Y} caracteriza este treinamento como um processo supervisionado, onde existe uma resposta alvo a ser atingida pela rede. Existem porém outras metodologias de aprendizado que não envolvem a presença de uma resposta alvo. Estas novas metodologias serão descritas mais adiante.

A1.5. Redes de múltiplas camadas (*Multilayer Perceptron* – MLP)

Mais uma vez tomando como base o cérebro humano, os neurônios são unidades de processamento capazes também de cooperarem entre si. Cada neurônio se conecta a aproximadamente 10^4 outros, formando uma rede maciça de unidades fortemente conectadas.

Esta rede biológica pode ser modelada por uma rede neural artificial, onde as complexas conexões interneuronais são representadas por uma simplificada rede de perceptrons. Assim, foi inicialmente proposta uma configuração de perceptrons dispostos em camadas, sendo que cada unidade de uma camada se conectava com todas as unidades da camada seguinte. Não existe conexão entre unidades da mesma camada. Esta rede foi denominada rede de múltiplas camadas (*Multilayer Perceptron* – MLP).

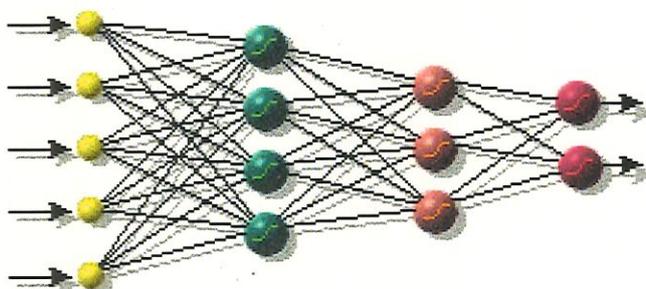


FIGURA A1.5 – Arquitetura de uma rede *Multilayer Perceptron* (MLP), construída a partir de uma camada de entrada, duas camadas intermediárias e uma camada de saída.

As redes MLP seguem uma configuração *feed-forward*, apresentando uma camada específica para a entrada do sinal, uma ou mais camadas intermediárias para o processamento da informação e outra camada específica para a saída da rede. Porém, a camada de entrada apenas conduz o sinal, não realizando nenhum processamento. A informação é fornecida à rede e segue um caminho unidirecional até a geração da resposta. Outras arquiteturas de rede apresentam uma realimentação do sinal. Estas redes são denominadas redes recorrentes e são melhor detalhadas no CAPÍTULO 3, através do modelo de HOPFIELD.

Contudo, BEALE & JACKSON (1990) descrevem que a utilização de camadas sucessivas de perceptrons impossibilita o treinamento da rede. Com a aplicação de funções de ativação *threshold*, a informação é processada por uma camada e repassada para a camada seguinte na forma binária. Devido a esta representação de saída, parte da informação é perdida. Assim, a próxima camada tem acesso a um sinal mascarado, o que impossibilita um correto ajuste das suas conexões. À medida que caminha pela rede, a informação vai se enfraquecendo ainda mais, inviabilizando o processo de treinamento.

A solução é alterar o modelo de perceptron proposto por Rosenblatt. A suavização da ativação *threshold* permite a determinação de estados intermediários entre o ativado e o desativado, o que possibilita à informação chegar mais detalhada a todas as camadas da rede. Novas funções de ativação foram propostas, sendo a ativação sigmoideal (EQUAÇÃO A1.7) a mais utilizada:

$$y = \frac{1}{1 + \exp \left[-k \cdot \left(\sum_{i=1}^N w_i x_i - T_i \right) \right]} \quad (\text{A1.7})$$

sendo k a constante positiva que controla a inclinação da função. Quanto menor o k , mais suave é a transição entre o estado ativado e o estado desativado (FIGURA A1.6). Em contrapartida, um valor infinito de k transforma a função sigmoideal em uma função *threshold*.

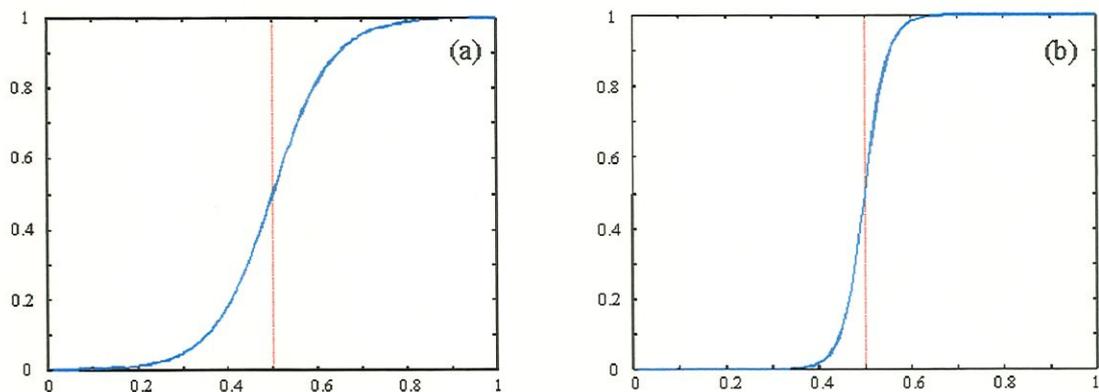


FIGURA A1.6 – Funções de ativação para *threshold* $T_i = 0,50$: (a) função sigmoideal com $k = 0,15$; (b) função sigmoideal com $k = 0,40$.

Porém, continuava não existindo um algoritmo eficiente para treinamento das redes MLP. Somente após a introdução do algoritmo *back-propagation* por RUMELHART & McCLELLAND (1986) é que elas se tornaram amplamente utilizadas.

A1.6. Treinamento das redes Multilayer Perceptron (algoritmo *back-propagation*)

O algoritmo *back-propagation* é um método de treinamento supervisionado que funciona em duas etapas distintas. Na etapa *forward*, um padrão de entrada x^P , pertencente à classe P , é fornecido à rede e sua resposta y^P é calculada. Na etapa *backward*, o erro ε^P entre a resposta calculada e a resposta desejada \hat{Y}^P é retropropagado pela rede de forma a ajustar os pesos entre uma camada e a camada anterior.

Quando x^P é apresentado à rede, cada unidade j da camada de saída tem uma resposta y_j^P e uma resposta desejada \hat{Y}_j^P . As funções erro ε^P para a classe P e a função erro total são definidas abaixo:

$$\varepsilon^P = \frac{1}{2} \cdot \sum_j (\hat{Y}_j^P - y_j^P)^2 \quad (\text{A1.8})$$

$$E_t = \sum_{p=1}^{np} \varepsilon^P \quad (\text{A1.9})$$

sendo np o número total de padrões. A saída da unidade j para o padrão p é dada por:

$$\text{net}_j^P = \sum_i w_{ij} y_i^P \quad (\text{A1.10})$$

$$y_j^P = f_j(\text{net}_j^P) \quad (\text{A1.11})$$

sendo f_j a função ativação da unidade j e w_{ij} o peso da conexão entre esta unidade e cada unidade i da camada anterior. O objetivo do algoritmo é determinar o ponto de mínimo da função erro ou função energia ε^P através do ajuste dos pesos w_{ij} . Assim, representa-se a variação de ε^P em função dos pesos através da regra da cadeia:

$$\frac{\partial \varepsilon^P}{\partial w_{ij}} = \frac{\partial \varepsilon^P}{\partial \text{net}_j^P} \cdot \frac{\partial \text{net}_j^P}{\partial w_{ij}} \quad (\text{A1.12})$$

Derivando-se a equação (A1.10) em função dos pesos, obtém-se seguinte resultado:

$$\frac{\partial \text{net}_j^P}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_k w_{kj} y_k^P \right) = \sum_k \frac{\partial w_{kj}}{\partial w_{ij}} y_k^P \quad (\text{A1.13})$$

Como $\frac{\partial w_{kj}}{\partial w_{ij}} = 0$ para $k \neq i$ e $\frac{\partial w_{kj}}{\partial w_{ij}} = 1$ para $k=i$, a equação (A1.13) pode ser

simplificada:

$$\frac{\partial \text{net}_j^P}{\partial w_{ij}} = y_i^P \quad (\text{A1.14})$$

Definindo-se ainda a derivada do erro ε^P em função da ativação net_j^P da unidade j :

$$-\frac{\partial \varepsilon^P}{\partial \text{net}_j^P} = \delta_j^P \quad (\text{A1.15})$$

e substituindo na equação (A1.12), obtém-se:

$$-\frac{\partial \varepsilon^P}{\partial w_{ij}} = \delta_j^P y_i^P \quad (\text{A1.16})$$

A equação (A1.16) indica que decrescer o erro ε^P significa variar o peso proporcionalmente a $\delta_j^P x_i^P$.

$$\Delta^P w_{ij} = \eta \delta_j^P y_i^P \quad (\text{A1.17})$$

sendo η a taxa de aprendizado e δ_j^P a parcela de erro para cada unidade j . A minimização de ε^P depende da correta determinação de δ_j^P . Assim, aplica-se a regra da cadeia à equação (A1.15).

$$\delta_j^P = -\frac{\partial \varepsilon^P}{\partial \text{net}_j^P} = -\frac{\partial \varepsilon^P}{\partial y_j^P} \frac{\partial y_j^P}{\partial \text{net}_j^P} \quad (\text{A1.18})$$

Considerando o segundo termo e as equações (A1.8) e (A1.11):

$$\frac{\partial \varepsilon^P}{\partial y_j^P} = -(\hat{Y}_j^P - y_j^P) \quad (\text{A1.19})$$

$$\frac{\partial y_j^P}{\partial \text{net}_j^P} = f_j'(\text{net}_j^P) \quad (\text{A1.20})$$

Substituindo as equações (A1.19) e (A1.20) em (A1.18):

$$\delta_j^P = f_j'(\text{net}_j^P) \cdot (\hat{Y}_j^P - y_j^P) \quad (\text{A1.21})$$

Contudo, a equação (A1.21) é indicada apenas para as unidades da camada de saída. Para o treinamento das unidades das camadas intermediárias, uma nova equação deverá ser desenvolvida:

$$\frac{\partial \varepsilon^P}{\partial y_j^P} = \sum_k \frac{\partial \varepsilon^P}{\partial \text{net}_k^P} \cdot \frac{\partial \text{net}_k^P}{\partial y_j^P}$$

$$\frac{\partial \varepsilon^P}{\partial y_j^P} = \sum_k \frac{\partial \varepsilon^P}{\partial \text{net}_k^P} \cdot \frac{\partial}{\partial y_j^P} \left(\sum_i w_{ik} y_i^P \right) \quad (\text{A1.22})$$

$$\frac{\partial \varepsilon^P}{\partial y_j^P} = -\sum_k \delta_k^P w_{jk} \quad (\text{A1.23})$$

Substituindo a equação (A1.23) na equação (A1.18), obtém-se finalmente:

$$\delta_j^P = f_j'(\text{net}_j^P) \cdot \sum_k \delta_k^P w_{jk} \quad (\text{A1.24})$$

Esta equação representa a retropropagação do erro δ_k^P para as unidades de uma camada j anterior à camada k .

O uso de redes MLP treinadas pelo algoritmo *back-propagation* representa um grande percentual dos trabalhos envolvendo redes neurais artificiais. Isto se deve principalmente

pela simplicidade de implementação tanto da rede como do algoritmo. Porém, o estágio de treinamento exige um grande esforço computacional para a retropropagação do erro. A consequência imediata é um longo período de processamento, o que dificulta a utilização *on-line* do algoritmo.

A1.7. Algoritmo *back-propagation* com taxa de aprendizado adaptativa

O equação (A1.24) propõe um ajuste dos pesos w_{ij} de forma a minimizar a função energia ε^P . Porém, este processo pode convergir para uma solução estável que não seja exatamente a solução procurada. Neste caso, diz-se que a rede convergiu para um ponto de mínimo local, diferente do ponto desejado que corresponde o mínimo global da função energia (FIGURA A1.7). Uma vez atingido, o algoritmo não consegue escapar do mínimo local porque qualquer ajuste dos pesos provocará um aumento do nível energético.

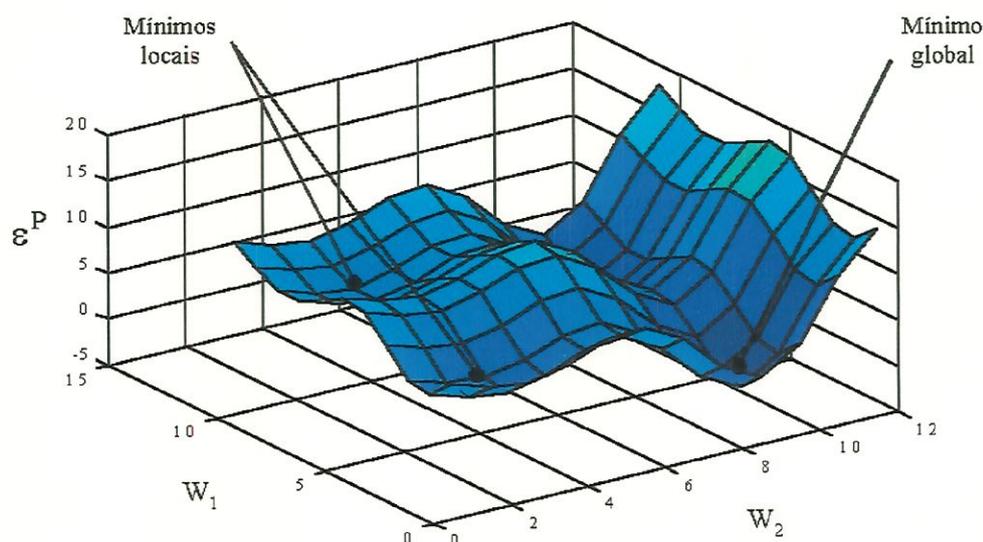


FIGURA A1.7 – Função energia bidimensional definida a partir da variação de apenas dois pesos. Esta simplificação é utilizada somente para critério de visualização do processo. Na realidade, o ajuste simultâneo de todos os pesos resulta numa função energia multidimensional.

Algumas estratégias (HAYKIN, 1994) visam evitar os mínimos locais através da manipulação dos parâmetros da rede. Uma alternativa é a aplicação do algoritmo *back-propagation* com taxa de aprendizado η adaptativa. Adota-se inicialmente um η alto, que vai decrescendo progressivamente. No começo, os pesos são alterados através de passos largos, o que possibilita ultrapassar os mínimos locais iniciais. Com a redução do passo, o algoritmo finalmente converge para um ponto que apresenta maior probabilidade de ser um mínimo

global. Este procedimento apresenta a desvantagem de aumentar o tempo de convergência da rede.

A1.8. Algoritmo *back-propagation* com termo de momento

O algoritmo *back-propagation* é muito sensível a pequenas variações da função energia. A FIGURA A1.8 mostra que ocorrerá uma inversão brusca na direção do ajuste dos pesos (EQUAÇÃO A1.24), caso a função energia apresente uma ligeira elevação local. A inversão do gradiente resultará na convergência para um mínimo local.

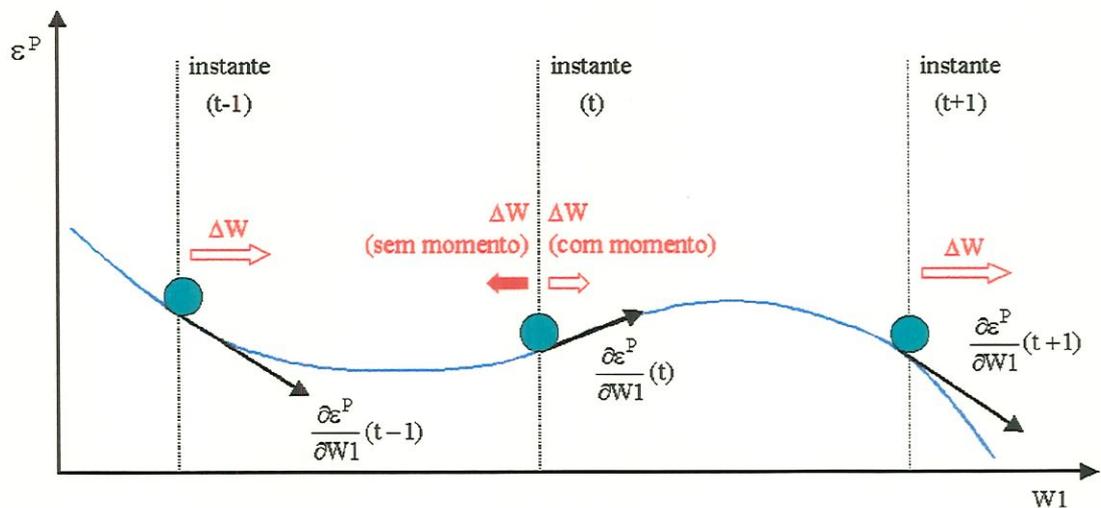


FIGURA A1.8 – Representação unidimensional da função energia como variação do peso $W1$. No instante t , um ligeira elevação da função resultaria na inversão do ajuste ΔW e consequentemente na convergência para o mínimo local. Com o termo de momento, a tendência do ajuste é mantida e o mínimo local é evitado.

É proposto, então, acrescentar um termo que introduza “inércia” ao processo, ou seja, que leve em consideração o ajuste dos pesos no tempo imediatamente anterior. Para o gradiente mudar de direção, este termo, denominado termo de momento, deverá ser vencido. A equação (A1.24) é então reescrita, sendo $0 < \alpha < 1$ o fator de momento.

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \delta_j^P \cdot y_j^P + \alpha \cdot (w_{ij}(t) - w_{ij}(t-1)) \tag{A1.25}$$

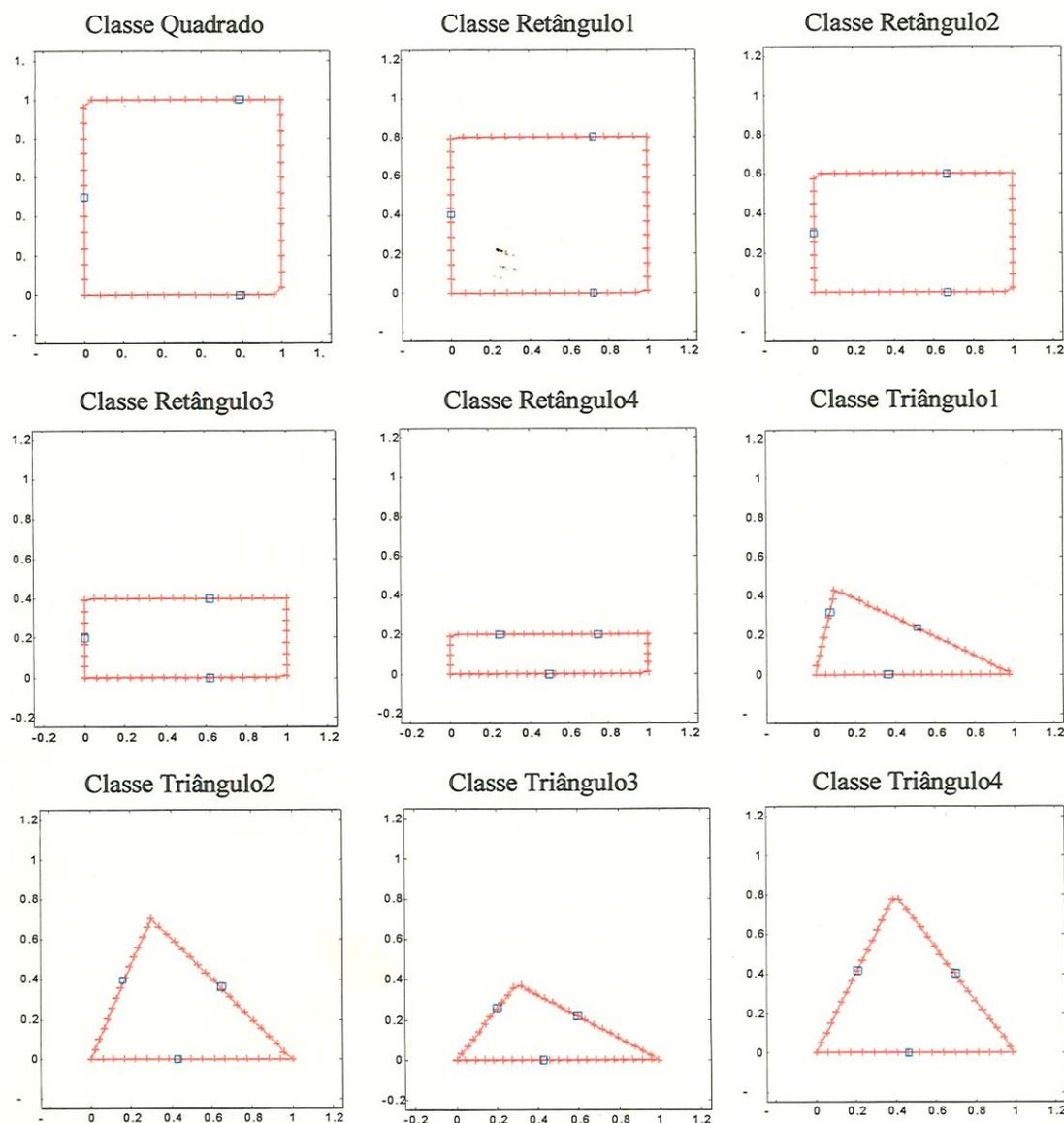
O termo de momento torna as redes menos sensíveis a mudanças locais e mais sensíveis a grandes tendências.

APÊNDICE 2

Apresentação dos padrões de Treinamento da Rede de Funções Base Radial

Neste APÊNDICE 2, é apresentado o conjunto de treinamento da rede de Funções de Base Radial. Este conjunto está dividido em quinze classes: um quadrado, quatro retângulos, sete triângulos, um círculo e duas elipses. Cada classe é formada por um polígono regular e por outros três polígonos gerados a partir da aplicação de um ruído gaussiano de 1% sobre o primeiro, totalizando sessenta padrões de treinamento.

A FIGURA A2.1 apresenta os quinze polígonos regulares geradores das classes. Todos são normalizados e posicionados de forma que os maiores lados definam a suas bases. A figura mostra ainda os pontos alvos de contato gerados pelo algoritmo de otimização proposto no ANEXO 1.



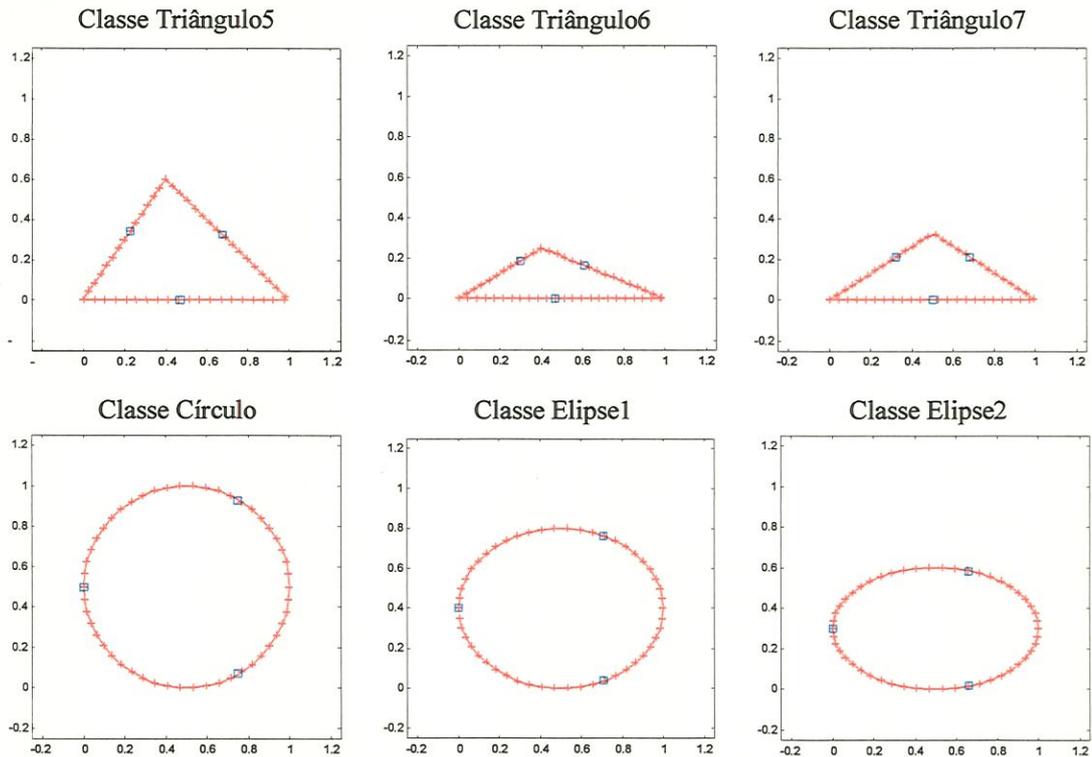


FIGURA A2.1 – Apresentação de todas as classes utilizadas para treinamento da rede de Funções de Base Radial. Cada classe é formada pelo respectivo polígono regular apresentado nesta figura, mais três outros polígonos gerados a partir da aplicação de um ruído gaussiano. São mostrados ainda os pontos alvos de contato (azul) gerados pelo método de otimização.

A FIGURA A2.2 apresenta exemplos de padrões ruidosos formados, respectivamente, à partir das classes: Retângulo1, Triângulo1 e Elipse1.

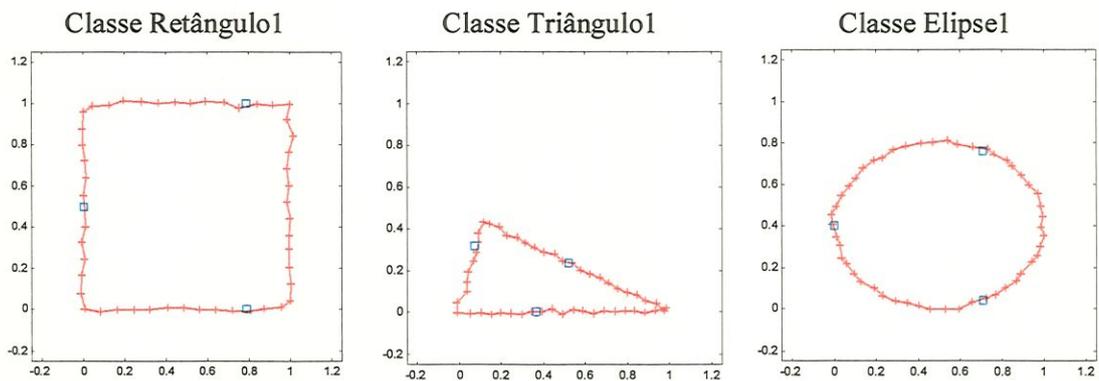


Figura A2.2 – Exemplo de padrões ruidosos definidos através da aplicação de um ruído gaussiano de 1% sobre os polígonos regulares apresentados na FIGURA A2.1. Os pontos alvos de contato são apresentados em azul.

Analisando os padrões de treinamento (FIGURA A2.1), nota-se uma diferença na configuração da fixação alvo proposta para as três primeiras classes de retângulo com relação à fixação proposta para a classe Retângulo4. Enquanto nos primeiros casos, os

padrões são fixados através de um contato na sua face lateral, a classe em destaque apresenta contatos apenas nas faces superior e inferior. O contato lateral foi proposto pelo algoritmo de busca pela solução ótima (ANEXO 1) e representa uma configuração de pontos de contato que minimiza a função objetivo. A partir desta função, o algoritmo busca minimizar as forças de contato. Porém, este não possui critérios bem definidos para analisar a segurança da fixação. Desta forma, o contato lateral definido para a classe Retângulo4, conforme demonstrado na FIGURA A2.3A, foi considerado perigoso em virtude da pequena área de contato. Esta situação se agravaria nos casos em que objetos ainda mais pontiagudos (FIGURA A2.3B) fosse apresentada à rede RBF treinada através do contato lateral. Estes objetos seriam relacionados com a classe Retângulo4 e a fixação lateral resultante seria completamente imprópria. Sendo assim, os pontos alvos de contato desta classe foram intuitivamente alterados para aqueles indicados na FIGURA A2.1. A FIGURA A2.4 ilustra a nova situação conseguida.

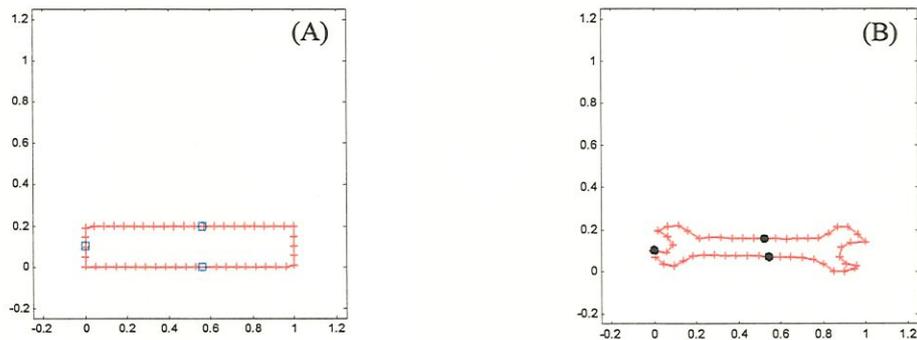


FIGURA A2.3 – Definição imprópria dos pontos de fixação para objetos da classe Retângulo4: (A) pontos de fixação calculados para um dos padrão de treinamento desta classe (quadrados azuis), resultado do algoritmo de otimização; (B) pontos de fixação calculados para um objeto pontiagudo (pontos pretos), resultado da rede RBF.

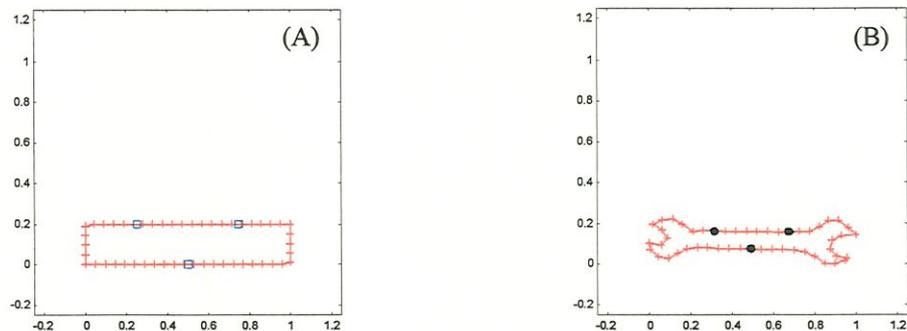


FIGURA A2.4 - Definição correta dos pontos de fixação para objetos da classe Retângulo4: (A) pontos de fixação definidos intuitivamente para esta classe; (B) pontos de fixação calculados para um objeto pontiagudo (pontos pretos), resultado da rede RBF.

APÊNDICE 3

**Definição do ponto central da fixação
através da solução de um sistema não linear.**

Neste apêndice, é apresentado o cálculo das coordenadas do ponto central da fixação (X_C, Y_C), de forma que as retas que unem este aos três pontos de contato (X_1, Y_1), (X_2, Y_2) e (X_3, Y_3) formem um ângulo de 120° entre si.

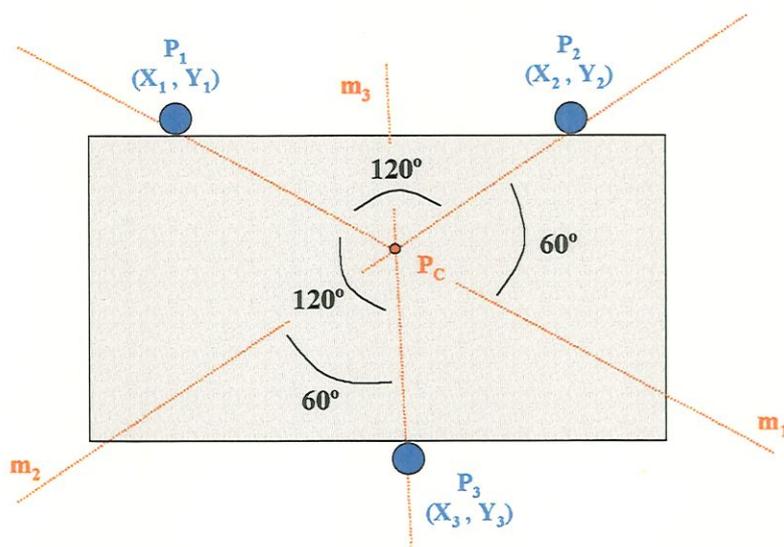


FIGURA A3.1 – Definição do ponto central de fixação como sendo o lugar geométrico onde as linhas de ação se cruzam a 120° .

A partir da equação resumida das retas que ligam o ponto central aos três pontos de contato:

$$Y_1 - Y_C = m_1(X_1 - X_C) \quad (\text{A3.1})$$

$$Y_2 - Y_C = m_2(X_2 - X_C) \quad (\text{A3.2})$$

$$Y_3 - Y_C = m_3(X_3 - X_C) \quad (\text{A3.3})$$

e da equação que define o ângulo agudo entre as retas r_1 e r_2 e entre r_2 e r_3 :

$$\text{tg}\theta_1 = \left| \frac{m_2 - m_1}{1 + m_1 \cdot m_2} \right| \quad (\text{A3.4})$$

$$\text{tg}\theta_2 = \left| \frac{m_3 - m_2}{1 + m_2 \cdot m_3} \right| \quad (\text{A3.5})$$

pode-se equacionar o problema acima. Considerando θ_1 e θ_2 iguais a 60° (ângulo suplementar de 120°), obtém-se o seguinte sistema de equações:

$$\begin{cases} X_1 m_1 - m_1 X_C + Y_C = Y_1 \\ X_2 m_2 - m_2 X_C + Y_C = Y_2 \\ X_3 m_3 - m_3 X_C + Y_C = Y_3 \\ -m_1 + m_2 - \sqrt{3} \cdot m_1 m_2 = \sqrt{3} \\ -m_2 + m_3 - \sqrt{3} \cdot m_2 m_3 = \sqrt{3} \end{cases} \quad (A3.6)$$

Este sistema é não linear, definido por cinco equações e cinco incógnitas. Para solucioná-lo, será utilizado o método de Newton para sistemas não lineares. Assim, cada equação anterior é expressa como função dos coeficientes angulares das retas (m_1 , m_2 e m_3) e das coordenadas do ponto central (X_C e Y_C):

$$\begin{aligned} F(m_1, m_2, m_3, X_C, Y_C) &= X_1 m_1 - m_1 X_C + Y_C - Y_1 = 0 \\ G(m_1, m_2, m_3, X_C, Y_C) &= X_2 m_2 - m_2 X_C + Y_C - Y_2 = 0 \\ H(m_1, m_2, m_3, X_C, Y_C) &= X_3 m_3 - m_3 X_C + Y_C - Y_3 = 0 \\ K(m_1, m_2, m_3, X_C, Y_C) &= -m_1 + m_2 - \sqrt{3} \cdot m_1 m_2 - \sqrt{3} = 0 \\ L(m_1, m_2, m_3, X_C, Y_C) &= -m_2 + m_3 - \sqrt{3} \cdot m_2 m_3 - \sqrt{3} = 0 \end{aligned} \quad (A3.7)$$

Considerando (m_{10} , m_{20} , m_{30} , X_{C0} e Y_{C0}) uma aproximação inicial para a solução do sistema, expande-se as funções utilizando séries de Taylor para cinco variáveis:

$$\begin{cases} F(m_1, m_2, m_3, X_C, Y_C) = F(m_{10}, m_{20}, m_{30}, X_{C0}, Y_{C0}) + F_{m_1}(m_1 - m_{10}) + \\ F_{m_2}(m_2 - m_{20}) + F_{m_3}(m_3 - m_{30}) + F_{X_C}(X_C - X_{C0}) + F_{Y_C}(Y_C - Y_{C0}) + \dots \\ G(m_1, m_2, m_3, X_C, Y_C) = G(m_{10}, m_{20}, m_{30}, X_{C0}, Y_{C0}) + G_{m_1}(m_1 - m_{10}) + \\ G_{m_2}(m_2 - m_{20}) + G_{m_3}(m_3 - m_{30}) + G_{X_C}(X_C - X_{C0}) + G_{Y_C}(Y_C - Y_{C0}) + \dots \\ H(m_1, m_2, m_3, X_C, Y_C) = H(m_{10}, m_{20}, m_{30}, X_{C0}, Y_{C0}) + H_{m_1}(m_1 - m_{10}) + \\ H_{m_2}(m_2 - m_{20}) + H_{m_3}(m_3 - m_{30}) + H_{X_C}(X_C - X_{C0}) + H_{Y_C}(Y_C - Y_{C0}) + \dots \\ K(m_1, m_2, m_3, X_C, Y_C) = K(m_{10}, m_{20}, m_{30}, X_{C0}, Y_{C0}) + K_{m_1}(m_1 - m_{10}) + \\ K_{m_2}(m_2 - m_{20}) + K_{m_3}(m_3 - m_{30}) + K_{X_C}(X_C - X_{C0}) + K_{Y_C}(Y_C - Y_{C0}) + \dots \\ L(m_1, m_2, m_3, X_C, Y_C) = L(m_{10}, m_{20}, m_{30}, X_{C0}, Y_{C0}) + L_{m_1}(m_1 - m_{10}) + \\ L_{m_2}(m_2 - m_{20}) + L_{m_3}(m_3 - m_{30}) + L_{X_C}(X_C - X_{C0}) + L_{Y_C}(Y_C - Y_{C0}) + \dots \end{cases} \quad (A3.8)$$

onde F_{m_1} , F_{m_2} , F_{m_3} , F_{X_C} e F_{Y_C} são as derivadas parciais da função F com relação às cinco respectivas variáveis (m_1 , m_2 , m_3 , X_C , Y_C), todas calculadas no ponto inicial. As derivadas das demais funções seguem a mesma definição.

Admitindo que o ponto inicial esteja próximo da solução do sistema, pode-se abandonar os termos de maior potência. Analisando as equações (A3.7) e (A3.8), define-se o novo sistema:

$$\begin{cases} -F = F_{m_1}(m_1 - m_{10}) + F_{m_2}(m_2 - m_{20}) + F_{m_3}(m_3 - m_{30}) + F_{X_C}(X_C - X_{C0}) + F_{Y_C}(Y_C - Y_{C0}) \\ -G = G_{m_1}(m_1 - m_{10}) + G_{m_2}(m_2 - m_{20}) + G_{m_3}(m_3 - m_{30}) + G_{X_C}(X_C - X_{C0}) + G_{Y_C}(Y_C - Y_{C0}) \\ -H = H_{m_1}(m_1 - m_{10}) + H_{m_2}(m_2 - m_{20}) + H_{m_3}(m_3 - m_{30}) + H_{X_C}(X_C - X_{C0}) + H_{Y_C}(Y_C - Y_{C0}) \\ -K = K_{m_1}(m_1 - m_{10}) + K_{m_2}(m_2 - m_{20}) + K_{m_3}(m_3 - m_{30}) + K_{X_C}(X_C - X_{C0}) + K_{Y_C}(Y_C - Y_{C0}) \\ -L = L_{m_1}(m_1 - m_{10}) + L_{m_2}(m_2 - m_{20}) + L_{m_3}(m_3 - m_{30}) + L_{X_C}(X_C - X_{C0}) + L_{Y_C}(Y_C - Y_{C0}) \end{cases} \quad (A3.9)$$

onde está compreendido que todas as funções e derivadas são calculadas no ponto inicial (m_{10} , m_{20} , m_{30} , X_{C0} e Y_{C0}). O sistema acima pode ser representado na forma matricial:

$$\begin{bmatrix} F_{m_1} & F_{m_2} & F_{m_3} & F_{X_C} & F_{Y_C} \\ G_{m_1} & G_{m_2} & G_{m_3} & G_{X_C} & G_{Y_C} \\ H_{m_1} & H_{m_2} & H_{m_3} & H_{X_C} & H_{Y_C} \\ K_{m_1} & K_{m_2} & K_{m_3} & K_{X_C} & K_{Y_C} \\ L_{m_1} & L_{m_2} & L_{m_3} & L_{X_C} & L_{Y_C} \end{bmatrix} \cdot \begin{bmatrix} m_1 - m_{10} \\ m_2 - m_{20} \\ m_3 - m_{30} \\ X_C - X_{C0} \\ Y_C - Y_{C0} \end{bmatrix} = \begin{bmatrix} -F \\ -G \\ -H \\ -K \\ -L \end{bmatrix} \quad (A3.10)$$

A solução deste determina novas variáveis (m_1 , m_2 , m_3 , X_C e Y_C) mais próximas da solução real. Estas variáveis são utilizadas como condição inicial para uma nova aproximação. A repetição deste processo iterativo pode ser equacionada pela seguintes expressão:

$$J_{(m_{1i}, m_{2i}, m_{3i}, X_{Ci}, Y_{Ci})} \cdot (A_{i+1} - A_i) = B_{(m_{1i}, m_{2i}, m_{3i}, X_{Ci}, Y_{Ci})} \quad (A3.11)$$

$$A_{i+1} = A_i + J_{(m_{1i}, m_{2i}, m_{3i}, X_{Ci}, Y_{Ci})}^{-1} \cdot B_{(m_{1i}, m_{2i}, m_{3i}, X_{Ci}, Y_{Ci})} \quad (A3.12)$$

sendo J a matriz Jacobiana incorporando todos as derivadas parciais, A o vetor de variáveis e B o vetor de funções. Todos estes termos são calculados a partir do valor das variáveis no instante i e $i+1$.

Com a convergência do processo, são determinadas as coordenadas do ponto central P_C e os coeficientes angulares das retas.